

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT COMPUTERWETENSCHAPPEN
AFDELING NUMERIEKE ANALYSE EN
TOEGEPASTE WISKUNDE
Celestijnenlaan 200 A — B-3001 Leuven

Multigrid algorithms for electromagnetic field computations

Promotoren :
Prof. Dr. ir. S. VANDEWALLE
Prof. Dr.-Ing. K. HAMEYER

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Tim BOONEN

February 2008

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT COMPUTERWETENSCHAPPEN
AFDELING NUMERIEKE ANALYSE EN
TOEGEPASTE WISKUNDE
Celestijnenlaan 200 A — B-3001 Leuven

Multigrid algorithms for electromagnetic field computations

Jury :
Prof. Dr. ir. D. Vandermeulen, voorzitter
Prof. Dr. ir. S. Vandewalle, promotor
Prof. Dr.-Ing. K. Hameyer, promotor (RWTH Aachen)
Prof. Dr. ir. H. De Gersem
Prof. Dr. ir. J. Driesen
Prof. Dr. ir. C. Oosterlee (CWI, TU Delft)
Prof. Dr. S. Poedts
Prof. Dr. ir. D. Roose

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Tim BOONEN

U.D.C. 519.63

February 2008

©Katholieke Universiteit Leuven – Faculteit Ingenieurswetenschappen
Arenbergkasteel, B-3001 Heverlee (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotocopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the publisher.

D/2008/7515/9
ISBN 978-90-5682-900-1

Dedicated to my son Arthur
and to the memory of his grandmother Christiane Goossens, my mother,
who never got to know each other.

Preface

This thesis is the result of four years of research. I would like to thank all people who have supported me during this research.

I would like to express my gratitude to my promotors Prof. Stefan Vandewalle and Prof. Kay Hameyer for assuring the funding and the scientific support that made this work possible. I acknowledge the Fund for Scientific Research - Flanders (F.W.O.-Vlaanderen) and the university (Katholieke Universiteit Leuven) for funding my research. I thank Prof. Herbert De Gersem and Prof. Thomas Weiland for giving me the opportunity to come over to the research group Theorie Elektromagnetischer Felder (TEMF) of the Technical University of Darmstadt for a study stay. I thank Dr. Barry Lee for agreeing to be my supervisor during my internship at the Lawrence Livermore National Laboratory (LLNL).

I thank all people that contributed to my knowledge of numerical analysis and electromagnetic computations. In particular, I appreciated very much the high quality lectures of my promotor Prof. Stefan Vandewalle and the time he took to discuss my research. A very important thank you goes to Prof. Herbert De Gersem, for the numerous things he explained to me and for sharing his knowledge in private discussions. I also learned a lot from discussions with Jan Van lent, Geoffrey Deliége, Bert Seynaeve, Eveline Rosseel, Hans Vande Sande, Barry Lee, Jim Jones and Stefan Reitzinger.

Prof. Stefan Vandewalle provided an invaluable help in correcting the manuscripts of my publications and of the present thesis text, and in improving their quality.

I thank all members of the jury for agreeing to take part in my defense.

An important revision of the present text was made when I had already left the university. This would not have been possible without the support of my dear wife Elisabeth and her parents, who gave me the opportunity to work on this text in the evening and during weekends. A warm thanks also for my parents, who supported me during my study career.

Tim Boonen
February 2008

Abstract

Electromagnetic problems give rise to large systems of linear equations, which are difficult to solve. In this PhD, we consider multigrid solvers for low-frequency electromagnetic problems which can be described by the so-called div-grad and curl-curl equations.

Fourier analysis is often used to study the efficiency of multigrid methods. However, classical Fourier techniques cannot be used to analyze curl-curl problems. The first contribution of this PhD is the development of a new Fourier technique for curl-curl problems. This technique is based on the decoupling of edges with different orientations.

Next, a new class of edge prolongators for algebraic multigrid methods for curl-curl problems is presented. These edge prolongators are obtained by the application of the analytical formulas of Whitney elements to the coarse nodal elements represented by a nodal prolongator. In a similar way, a face and volume prolongator can be derived as well. These prolongators transfer the mathematical structure of the original problem to all levels of the multigrid hierarchy. Numerical experiments will show that the use of these prolongators increases the efficiency of algebraic multigrid methods significantly.

In the last part of this PhD, we consider time-dependent problems. Typically, classical multigrid method cannot be combined with advanced time discretization techniques. In a previous PhD in the research group Scientific Computing, a technique was developed which allows to combine advanced time discretization methods and so-called geometric multigrid solvers for div-grad problems. This technique will be extended to algebraic multigrid and to curl-curl problems. Numerical experiments will be shown to demonstrate the efficiency of this technique.

Multigrid algoritmes voor elektromagnetische veldberekeningen

Nederlandse samenvatting

1 Inleiding

In dit doctoraat werd onderzoek verricht naar multigrid methodes voor elektromagnetische problemen die geformuleerd kunnen worden als een div-grad of curl-curl probleem:

$$-\operatorname{div}(\alpha \operatorname{grad}(V)) + \beta V = \rho, \tag{1}$$
$$\operatorname{curl}(\alpha \operatorname{curl}(\vec{A})) + \alpha \vec{A} = \vec{J}. \tag{2}$$

Eerst werd een literatuurstudie uitgevoerd van bestaande multigrid methodes. De nieuwe bijdragen van dit doctoraat zijn:

- lokale Fourier analyse van multigrid voor curl-curl problemen,
- ontwikkeling van een nieuwe familie van prolongatoren voor AMG methodes, toegepast in AMG voor curl-curl problemen,
- ontwikkeling van AMG methodes voor hogere orde tijdsdiscretizaties van tijdsafhankelijke div-grad en curl-curl problemen.

2 Numerieke methodes voor elektromagnetische veldberekeningen

2.1 Wiskundige formulering van elektromagnetische problemen

Elektromagnetische problemen worden beschreven door de wetten van Maxwell:

$$\operatorname{div}(\vec{D}) = \rho \qquad \operatorname{curl}(\vec{E}) = -\frac{\partial \vec{B}}{\partial t}$$
$$\operatorname{div}(\vec{B}) = 0 \qquad \operatorname{curl}(\vec{H}) = \vec{J} + \frac{\partial \vec{D}}{\partial t}$$

en de materiaalwetten:

$$\vec{D} = \epsilon \vec{E}, \qquad \vec{B} = \mu \vec{H}, \qquad \vec{J} = \vec{J}_0 + \sigma \vec{E}.$$

Deze wetten kunnen geherformuleerd worden door gebruik te maken van de magnetische vectorpotentiaal $\vec{A}$ en de scalaire elektrische potentiaal V. Deze potentialen voldoen aan:

$$\begin{aligned} \vec{B} &= \operatorname{curl}(\vec{A}), \\ \vec{E} + \frac{\partial \vec{A}}{\partial t} &= -\operatorname{grad}(V). \end{aligned}$$

Voor tijdsonafhankelijke problemen geeft dit:

$$\begin{aligned} -\operatorname{div}(\epsilon \operatorname{grad}(V)) &= \rho, \\ \operatorname{curl}(\frac{1}{\mu} \operatorname{curl}(\vec{A})) &= \vec{J}. \end{aligned}$$

Van de quasi-magnetostatische benadering voor tijdsafhankelijke problemen, waarin de term $\frac{\partial \vec{D}}{\partial t}$ verwaarloosd wordt, kunnen de volgende curl-curl formuleringen afgeleid worden:

$$\begin{aligned} \operatorname{curl}(\frac{1}{\mu} \operatorname{curl}(\vec{A})) + \sigma \frac{\partial \vec{A}}{\partial t} &= \vec{J}_0 - \sigma \operatorname{grad}(V), \\ \operatorname{curl}(\frac{1}{\mu} \operatorname{curl}(\vec{E})) + \sigma \frac{\partial \vec{E}}{\partial t} &= -\frac{\partial \vec{J}_0}{\partial t}. \end{aligned}$$

2.2 Een functioneel raamwerk voor elektromagnetische problemen

De continuïteitsvoorwaarden waaraan elektromagnetische velden voldoen, kunnen afgeleid worden van de integraalformulering van de wetten van Maxwell. Deze wetten laten toe dat de continuïteit van elektromagnetische velden verstoord wordt aan materiaalovergangen. Meer bepaald, voor $\vec{H}$ en $\vec{E}$ dient enkel de tangentiële component continu te zijn. Voor $\vec{B}$, $\vec{D}$ en $\vec{J}$ dient enkel de normale component continu te zijn. De functieruimtes waarin deze vormen van continuïteit gelden, zijn gedefinieerd als volgt:

$$\begin{array}{lcl} V & \in & H(\operatorname{grad}, \Omega) := \{x \in L^2(\Omega) \mid \operatorname{grad}(\vec{x}) \in \vec{L}^2(\Omega)\} := H^1(\Omega), \\ \vec{H}, \vec{E} & \in & H(\operatorname{curl}, \Omega) := \{\vec{x} \in \vec{L}^2(\Omega) \mid \operatorname{curl}(\vec{x}) \in \vec{L}^2(\Omega)\}, \\ \vec{B}, \vec{D}, \vec{J} & \in & H(\operatorname{div}, \Omega) := \{\vec{x} \in \vec{L}^2(\Omega) \mid \operatorname{div}(\vec{x}) \in L^2(\Omega)\}, \\ \rho & \in & L^2(\Omega). \end{array}$$

Als gevolg van de continuïteitskenmerken van deze functieruimtes moet de zwakke vorm van de differentiaaloperatoren gradiënt, rotor en divergentie gebruikt worden in plaats van de klassieke, puntsgewijze definities.
De zwakke differentiaaloperatoren vormen samen met de functieruimtes $H^1(\Omega)$, $H(\operatorname{curl}, \Omega)$, $H(\operatorname{div}, \Omega)$ en $L^2(\Omega)$ een zogenaamd exact de Rham complex. Dit complex wordt als volgt voorgesteld:

$$H^1(\Omega) \xrightarrow{\text{grad}} H(\operatorname{curl}, \Omega) \xrightarrow{\text{curl}} H(\operatorname{div}, \Omega) \xrightarrow{\text{div}} L^2(\Omega).$$

Het drukt de volgende verbanden uit tussen het domein en het bereik van de differentiaaloperatoren:

$$\begin{aligned} \mathrm{grad}(H^1(\Omega)) &= \ker(\mathrm{curl}) \subset H(\mathrm{curl},\Omega), \\ \mathrm{curl}(H(\mathrm{curl},\Omega)) &= \ker(\mathrm{div}) \subset H(\mathrm{div},\Omega), \\ \mathrm{div}(H(\mathrm{div},\Omega)) &= L^2(\Omega). \end{aligned}$$

De rotor operator en de divergentie operator induceren de volgende orthogonale splitsingen van $H(\mathrm{curl},\Omega)$ en $H(\mathrm{div},\Omega)$:

$$\begin{aligned} H(\mathrm{curl},\Omega) &= \ker(\mathrm{curl}) \oplus \ker^{\perp}(\mathrm{curl}), \\ H(\mathrm{div},\Omega) &= \ker(\mathrm{div}) \oplus \ker^{\perp}(\mathrm{div}). \end{aligned}$$

Deze splitsingen worden Helmholtz splitsingen genoemd. Ze zijn van groot belang voor discretizatie-schema's en solvers voor problemen in $H(\mathrm{curl},\Omega)$ en $H(\mathrm{div},\Omega)$.

2.3 Discretizatie

In dit doctoraat beschouwen we eindige elementen discretizaties van de div-grad- en curl-curl-vergelijkingen. De vormfuncties die hiervoor meestal gebruikt worden zijn de Whitney elementen van de eerste soort. Deze vormfuncties worden vaak nodale, zijde, vlak en volume elementen genoemd. Ze zullen genoteerd worden als N_i, $\vec{E}_i$, $\vec{F}_i$ en V_i respectievelijk, en ze worden gebruikt om velden te benaderen die behoren tot $H^1(\Omega)$, $H(\mathrm{curl},\Omega)$, $H(\mathrm{div},\Omega)$ en $L^2(\Omega)$ respectievelijk. Ze voldoen aan de volgende eigenschappen:

1. Compatibiliteit: ze behoren tot dezelfde functieruimte als de velden die ze moeten benaderen:

$$N_i \in H^1(\Omega), \quad \vec{E}_i \in H(\mathrm{curl},\Omega), \quad \vec{F}_i \in H(\mathrm{div},\Omega), \quad V_i \in L^2(\Omega).$$

2. Normalizatie: de punt evaluatie, lijn-, oppervlakte-, of volume-integraal van N_i, $\vec{E}_i$, $\vec{F}_i$ en V_i is 1 in het punt n_i, zijde e_i, vlak f_i of volume v_i waarmee ze geassocieerd is, en 0 in alle andere punten, zijdes, vlakken of volumes van de vermazing:

$$N_i(n_j) = \int_{e_j} \vec{E}_i \cdot d\vec{l} = \int_{f_j} \vec{F}_i \cdot d\vec{\Omega} = \int_{v_j} V_i \, dV = \begin{cases} 1 & \text{if} \quad i = j, \\ 0 & \text{if} \quad i \neq j. \end{cases} \tag{3}$$

3. Ze zijn stuksgewijs polynomiaal en hebben een lokaal bereik.

4. De ruimtes $\mathcal{S}_N$, $\mathcal{S}_E$, $\mathcal{S}_F$ en $\mathcal{S}_V$, opgespannen door alle nodale, zijde, vlak en volume elementen, worden op elkaar afgebeeld door de differentiaaloperatoren:

$$\mathrm{grad}(\mathcal{S}_N) \subset \mathcal{S}_E, \quad \mathrm{curl}(\mathcal{S}_E) \subset \mathcal{S}_F, \quad \mathrm{div}(\mathcal{S}_F) \subset \mathcal{S}_V. \tag{4}$$

Uit de eigenschappen (4) volgt dat de toepassingen van de gradiënt, rotor en divergentie operatoren op $\mathcal{S}_N$, $\mathcal{S}_E$, $\mathcal{S}_F$ kan voorgesteld worden door transformatie matrices. Deze discrete differentiaaloperatoren worden genoteerd als G, C en D, en ze voldoen aan:

$$G(e_j, n_i) = \begin{cases} 1 & \text{if} \quad n_i = \text{end}(e_j), \\ -1 & \text{if} \quad n_i = \text{start}(e_j), \\ 0 & \text{else}, \end{cases} \tag{5}$$

$$C(f_j, e_i) = \begin{cases} 1 & \text{if} \quad f_i \in \partial v_j, \\ -1 & \text{if} \quad -f_i \in \partial v_j, \\ 0 & \text{else}, \end{cases} \tag{6}$$

$$D(v_j, f_i) = \begin{cases} 1 & \text{if} \quad f_i \in \partial v_j, \\ -1 & \text{if} \quad -f_i \in \partial v_j, \\ 0 & \text{else}. \end{cases} \tag{7}$$

De vectorruimtes van de coefficiënten van de lineaire combinaties van nodale, zijde, vlak en volume elementen worden genoteerd als $\mathcal{N}$, $\mathcal{E}$, $\mathcal{F}$ en $\mathcal{V}$. Samen met de discrete differentiaaloperatoren vormen deze vectorruimtes een exact de Rham complex:

$$\mathbb{R} \xrightarrow{\mathbf{1}} \mathcal{N} \xrightarrow{G} \mathcal{E} \xrightarrow{C} \mathcal{F} \xrightarrow{D} \mathcal{V}, \tag{8}$$

met

$$\text{ran}(\mathbf{1}) = \ker(G), \qquad \text{ran}(G) = \ker(C), \qquad \text{ran}(C) = \ker(D). \tag{9}$$

Voor tetrahedrale vermazingen zijn de nodale elementen de klassieke tent functies. De zijde, vlak en volume elementen worden hiervan afgeleid volgens de formules:

$$\vec{E}_{ij} = N_i \text{ grad}(N_j) - N_j \text{ grad}(N_i), \tag{10}$$

$$\vec{F}_{ijk} = N_i \,\text{curl}(\vec{E}_{jk}) + N_j \,\text{curl}(\vec{E}_{ki}) + N_k \,\text{curl}(\vec{E}_{ij}), \tag{11}$$

$$V_{ijkl} = N_i \text{ div}(\vec{F}_{jkl}) + N_j \text{ div}(\vec{F}_{kli}) + N_k \text{ div}(\vec{F}_{lij}) + N_l \text{ div}(\vec{F}_{ijk}). \tag{12}$$

3 Introductie tot multigrid

3.1 Iteraties gebaseerd op een matrix splitsing

Een belangrijke klasse van iteratieve methodes voor de oplossing van een stelsel lineare vergelijkingen $Kx = b$ zijn iteraties die gebaseerd zijn op een splitsing $K = K^+ + K^-$ van de systeem matrix:

$$x^{[\nu+1]} = \left(K^+\right)^{-1} \left(b - K^- x^{[\nu]}\right). \tag{13}$$

De superindex ν duidt de iteratie-index aan. Het convergentiegedrag van de iteratie (13) wordt bepaald door het spectrum van de iteratiematrix

$$L = -\left(K^+\right)^{-1} K^-. \tag{14}$$

Meer bepaald, de iteratie zal convergeren indien $\rho(K) < 1$. De spectraalradius $\rho(K)$ wordt in dit geval ook wel de convergentiefactor genoemd. Hoe kleiner dit getal, hoe sneller de convergentie.
In elke stap van de iteratie (13) moet een stelsel opgelost worden met als matrix K^+. Er bestaan verschillende types van matrixsplitsingen waarvoor dit bijzonder goedkoop is. Bijvoorbeeld, voor Jacobi iteraties is K^+ gelijk aan het diagonaaldeel van K, en voor Gauss-Seidel iteraties is K^+ het boven- of onderdriehoeksdeel van K. Typisch convergeren zulke iteraties die gebaseerd zijn op eenvoudige splitsingen van de matrix echter bijzonder traag.

3.2 Multigrid

De klassieke iteratieve methodes van Jacobi en Gauss-Seidel convergeren typisch traag voor discretizaties van elliptische en parabolische partiële differentiaalvergelijkingen. Deze iteratieve methodes worden vaak smoothers genoemd, omdat ze de hoogfrequente componenten in de fout typisch veel sneller reduceren dan de laagfrequente componenten, waardoor de fout een zacht verloop krijgt.
In multigrid methodes worden berekeningen op grove roosters gebruikt om de convergentie van smoothers te versnellen. Daartoe wordt een grove voorstelling $K_c x_c = b_c$ van het oorspronkelijke probleem opgelost. Het rechterlid $b_c = R(b - Kx)$ wordt verkregen door restrictie van het residu. De benaderende oplossing van het oorspronkelijke probleem wordt gecorrigeerd door prolongatie van de oplossing x_c van dit grove probleem. Dit levert het volgende schema op:

1. μ_{pre} iteraties van een smoother op $Kx = b$,
2. restrictie van het residu: $b_c = R(b - Kx)$,
3. (benaderende) oplossing van $K_c x_c = b_c$,
4. prolongatie: $x \leftarrow x + Px_c$,
5. μ_{post} iteraties van een smoother op $Kx = b$.

Dit scheme kan recursief toegepast worden in stap 3. In algebraische multigrid methodes wordt $R = P^T$ gekozen en wordt K_c geconstrueerd als het Galerkin product $K_c = P^T K P$.
Het convergentiegedrag van een multigrid iteratie wordt bepaald door het spectrum van haar iteratiematrix. Indien het grove probleem in stap 3 exact opgelost wordt, wordt de iteratiematrix gegeven door:

$$M \quad = \quad S_{post}^{\mu_{post}} \left(I - PK_c^{-1}RK\right) S_{pre}^{\mu_{pre}}. \tag{15}$$

$S_{pre}^{\mu_{pre}}$ en $S_{post}^{\mu_{post}}$ zijn de iteratiematrices van de pre- en postsmoother respectievelijk.

3.3 Lokale Fourier analyse

Voor problemen met constante materiaalparameters kan de convergentie van iteratieve methoden voor stelsels afkomstig van de discretizatie van partiële differentiaalvergelijkingen op regelmatige roosters, geanalyseerd worden met behulp van Fourier modes

$$\varphi(\theta) \quad := \quad e^{i\theta\cdot\mathbf{x}/h}. \tag{16}$$

Deze Fourier modes laten toe laag-dimensionale eigenruimtes te identificeren voor de verschillende operatoren in (15). Hierdoor wordt de spectrale analyse van de iteratiematrix, die nodig is om het convergentiegedrag te bestuderen, sterk vereenvoudigd.

4 Algebraische multigrid voor de div-grad vergelijking

4.1 Smoothing

In AMG methodes voor de div-grad vergelijking (1) worden meestal puntsgewijze smoothers, zoals Jacobi of Gauss-Seidel iteraties, of polynomiale smoothers gebruikt. Wanneer de convergentie van een smoother stagneert, wordt de fout algebraisch zachtverlopend genoemd met betrekking tot deze smoother. De toepassing van een smoother op een elliptisch probleem zorgt typisch voor:

- een fout met een geometrisch zacht verloop in de richtingen van sterke connectie,
- een reductie van de energienorm $\|e\|_K$ van de fout.

4.2 Coarsening

Onder coarsening wordt de identificatie verstaan van de variabelen voor het grove probleem. Een grove variabele kan geassocieerd zijn met:

- een knoop van de fijne vermazing (C/F-splitsing),
- een groep knopen van de fijne vermazing,
- een groep elementen van de fijne vermazing.

4.3 Prolongatie

De kwaliteit van een prolongator hangt af van de mate waarin algebraisch zachtverlopende fouten behoren tot zijn bereik. Daarnaast dient een prolongator voor een div-grad probleem typisch te voldoen aan de eenheidspartitie-eigenschap. Dit wil zeggen dat zijn rijsommen 1 moeten zijn. In dit doctoraat werden de volgende prolongatieschema's voor AMG methodes voor div-grad problemen bestudeerd:

- ideale prolongatie: een Schur complement benadering.
- klassieke AMG: gewogen interpolatie gebaseerd op een C/F-splitsing.
- smoothed aggregatie: smoothing van een stuksgewijs constante prolongator.
- expliciete energie minimizatie van een prolongator met behoud van de spaarsheidsstructuur en van de eenheidspartitie-eigenschap.
- AMGe: gebaseerd op het Schur complement van deelproblemen.
- spectrale AMGe: gebaseerd op de eigenwaarde ontbinding van deelproblemen.
- element agglomeratie AMGe: constructie van grove vormfuncties voor de deelproblemen.

5 Multigrid voor de curl-curl vergelijking

5.1 Smoothing

De multigrid smoothers die typisch gebruikt worden voor div-grad problemen (1) leveren geen goede resultaten voor curl-curl problemen (2). Dit wordt veroorzaakt door het feit dat de discrete curl-curl operator zich fundamenteel verschillend gedraagt in het gradiënt en solenoidale deel van de discrete Helmholtz splitsing:

$$\mathcal{E} \quad = \quad \mathrm{ran}(G) + \mathrm{ran}(C^T). \tag{17}$$

Om dit fenomeen in rekening te brengen, dienen complexere smoothers gebruikt te worden. In dit werk wordt voornamelijk de zogenaamde hybride smoother gebruikt. Deze smoother is formeel identiek aan een AMG cyclus met twee niveaus, waarbij de discrete gradiënt G als prolongator gebruikt wordt:

1. klassieke smoothing op $Kx = b$,
2. restrictie: $b_n = G^T(b - Kx)$,
3. klassieke smoothing op $G^T K G x_n = b_n$,
4. prolongatie: $x \leftarrow x + Gx_c$,
5. klassieke smoothing op $Kx = b$.

Als alternatief kan een overlappende bloksmoother gebruikt worden. Deze smoother bestaat uit de oplossing van lokale stelsels naar alle zijdes E_n die een knoop n bevatten:

$$x(E_n) \quad \leftarrow \quad x(E_n) + K(E_n, E_n)^{-1} \left(b(E_n) - K(E_n, :)x\right). \tag{18}$$

Hierin is E_n gedefinieerd als volgt:

$$G(e, n) \neq 0 \quad \Longleftrightarrow \quad e \in E_n.$$

5.2 Prolongatie

Prolongatieschema's voor curl-curl problemen dienen rekening te houden met de oriëntatie van de zijden waarmee de vrijheidsgraden van curl-curl problemen geassocieerd zijn. Klassieke prolongatieschema's voor div-grad problemen kunnen daarom niet toegepast worden.

Een zijde prolongator P_e wordt typisch op zo'n manier geconstrueerd dat hij commuteert met een nodale prolongator P_n die voldoet aan de eenheidspartitie-eigenschap:

$$P_e G_c = G_f P_n. \tag{19}$$

Voor elke stukgewijs constante P_n kan er een stukgewijs constante P_e geconstrueerd worden die hieraan voldoet. De kwaliteit van P_e kan vervolgens verbeterd worden door:

- smoothed aggregatie: smoothing van P_e met (benaderend) behoud van (19).
- beperking van de spaarsheidsstructuur van P_e zodat (19) rij per rij opgelost kan worden naar P_e.
- energieminimizatie van P_e met een vooraf bepaalde spaarsheidsstructuur.

Alternatieve methodes zijn gebaseerd op:

- een herformulering als een singulier bloksysteem.
- compatible ijking: herformulering als een regulier bloksysteem.
- splitsing in componenten: een herformulering van het curl-curl probleem als gekoppelde div-grad problemen.

6 Fourier analyse van multigrid in $H(\mathrm{curl})$

Voor problemen in $H(\mathrm{curl})$ zijn de onbekenden verbonden met de zijdes van een vermazing. De oriëntatie van de zijden waarmee de onbekenden relateerd zijn, dient in dit geval in rekening gebracht te worden in een Fourier analyse. Dit is niet mogelijk met klassieke Fourier technieken. In dit hoofdstuk beschrijven we een manier om Fourier technieken toe te passen op problemen in $H(\mathrm{curl})$. Daartoe beschouwen we het voorbeeld van de eindige elementen discretizatie van de twee-dimensionale curl-curl vergelijking (2) op een regelmatig rooster.

6.1 Fourier modes

Voor elke frequentie $\theta = (\theta_1, \theta_2)$ construeren we een twee-dimensionale ruimte $F_e(\theta)$ die invariant is voor translatie-invariante operatoren. De basisvectoren van $F_e(\theta)$ bestaan uit de evaluaties van de Fourier mode $\varphi(\theta)$ in de centra van

de horizontale zijdes E_1 en de verticale zijdes E_2 respectievelijk. Deze vectoren worden als volgt genoteerd:

$$\phi_1(\theta)_E := \begin{bmatrix} \varphi(\theta)_{E_1} \\ 0_{E_2} \end{bmatrix}, \qquad \phi_2(\theta)_E := \begin{bmatrix} 0_{E_1} \\ \varphi(\theta)_{E_2} \end{bmatrix}.$$

Dit komt neer op een ontkoppeling van de horizontale en verticale zijdes. Dit laat toe om een functie $\phi_E \in F_e(\theta)$ compact voor te stellen als:

$$\phi_E \quad = \quad \Phi(\theta) \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}, \qquad \text{met} \qquad \Phi(\theta) := \begin{bmatrix} \phi_1(\theta)_E & \phi_2(\theta)_E \end{bmatrix}.$$

$F_e(\theta)$ is invariant voor translatie-invariante operatoren L. Dit wordt compact genoteerd als volgt:

$$L\Phi(\theta) \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} \quad = \quad \Phi(\theta) \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix},$$

met

$$\begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} \quad = \quad L_\Phi(\theta) \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}, \qquad L_\Phi(\theta) \in \mathbb{C}^{2\times 2}.$$

De twee-dimensionale ruimtes $F_e(\theta)$ vormen de basis van de Fourier analyse, en het spectrum van L_Φ bepaalt het gedrag van L voor de frequentie θ.

6.2 Helmholtz splitsing in de Fourier ruimte

De ruimtes $F_e(\theta)$ kunnen orthogonaal gesplitst worden als

$$F_e(\theta) \quad = \quad \psi_g(\theta)_E \oplus \psi_s(\theta)_E,$$

met

$$\psi_g(\theta)_E \in \mathrm{ran}(G) \qquad \text{en} \qquad \psi_s(\theta)_E \in \mathrm{ran}(C^T).$$

Dit komt overeen met de Helmholtz splitsing (17) van de vectorruimte $\mathcal{E}$, die een belangrijke rol speelt in de multigrid theorie voor problemen in $H(\mathrm{curl})$. Deze vectoren worden gegenereerd door de toepassing van de discrete gradiënt G en de duale discrete curl C^T op de evaluatie van de Fourier mode in de knopen N en de centra F van de vlakken:

$$\psi_g(\theta)_E = G\varphi(\theta)_N \qquad \text{en} \qquad \psi_s(\theta)_E = C^T\varphi(\theta)_F.$$

Deze vectoren vormen de basis van een alternatieve parametrizatie van $F_e(\theta)$:

$$\Psi(\theta) \quad := \quad \begin{bmatrix} \psi_g(\theta)_E & \psi_s(\theta)_E \end{bmatrix}$$

Deze parametrizatie laat toe om verscheidene eigenschappen van (multigrid voor) curl-curl problemen te identificeren in het frequentiedomein. Beide basissen zijn gerelateerd als $\Psi(\theta) = 2i\Phi(\theta)H(\theta)$, met:

$$H(\theta) := \begin{cases} \begin{bmatrix} \sin(\frac{\theta_1}{2}) & \sin(\frac{\theta_2}{2}) \\ \sin(\frac{\theta_2}{2}) & -\sin(\frac{\theta_1}{2}) \end{bmatrix} & \text{if} \quad \theta \neq (0,0), \\ \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} & \text{if} \quad \theta = (0,0). \end{cases}$$

6.3 Lokale Fourier analyse

De curl-curl operator en de hybride smoother zijn invariant voor $F_e(\theta)$, en kunnen bijgevolg in het frequentiedomein voorgesteld worden door matrices behorende tot $\mathbb{C}^{2\times 2}$. Voor de overlappende bloksmoother, de prolongatie en de restrictie dienen telkens 4 frequenties tesamen bestudeerd te worden. Dit levert frequentiedomein voorstellingen op in $\mathbb{C}^{8\times 8}$, $\mathbb{C}^{8\times 2}$ en $\mathbb{C}^{2\times 8}$ respectievelijk.

De voorstelling in het frequentiedomein van een multigrid cyclus met twee niveaus wordt bekomen door de vermenigvuldiging van de frequentievoorstellingen van haar samenstellende componenten, zoals in (15). Dit levert een matrix op behorende tot $\mathbb{C}^{8\times 8}$.

Het frequentiegedrag van deze operatoren wordt bepaald door het spectrum van hun voorstellingen in het frequentiedomein.

6.4 Veralgemening

De voorgestelde techniek kan veralgemeend worden voor alle problemen die beschreven worden door een set van translatie invariante stencils.

7 Een nieuwe familie prolongatie operators

In dit hoofdstuk leiden we een zijde prolongator P_e, een vlak prolongator P_f en een volume prolongator P_v af van een nodale prolongator P_n.

7.1 Prolongatoren afgeleid van een nodale prolongator

We veronderstellen dat de rijsommen van P_n gelijk zijn aan 1 (eenheidspartitie). Elke kolom van P_n kan geïnterpreteerd worden als een nodaal element N_I voor het grove rooster:

$$N_I = \sum_j P_n(j,I) N_j.$$

De toepassing van de formule (10) voor zijde elementen op tetrahedrale roosters op deze grove nodale elementen levert grove zijde elementen $\vec{E}_{IJ}$ op. Deze grove

zijde elementen kunnen geschreven worden als lineaire combinaties van de zijde elementen op het fijne rooster:

$$\begin{aligned}\vec{E}_{IJ} &= N_I \ \mathrm{grad}(N_J) - N_J \ \mathrm{grad}(N_I) \\ &= \sum_{k<l} P_e(kl, IJ)\vec{E}_{kl}.\end{aligned}$$

Dit laat toe om de elementen van de zijde prolongator te identificeren als de determinanten van 2×2 deelmatrices van P_n:

$$P_e(kl, IJ) = \begin{vmatrix} P_n(k, I) & P_n(k, J) \\ P_n(l, I) & P_n(l, J) \end{vmatrix}.$$

Op een gelijkaardige manier worden grove vlak elementen $\vec{F}_{IJK}$ en grove volume elementen V_{IJKL} bekomen door de toepassing van (11) en (12). Dit laat toe om de elementen van P_f en P_v te identificeren als de determinanten van 3×3 en 4×4 deelmatrices van P_n:

$$P_f(rst, IJK) = \begin{vmatrix} P_n(r, I) & P_n(r, J) & P_n(r, K) \\ P_n(s, I) & P_n(s, J) & P_n(s, K) \\ P_n(t, I) & P_n(t, J) & P_n(t, K) \end{vmatrix},$$

$$P_v(rstu, IJKL) = \begin{vmatrix} P_n(r, I) & P_n(r, J) & P_n(r, K) & P_n(r, L) \\ P_n(s, I) & P_n(s, J) & P_n(s, K) & P_n(s, L) \\ P_n(t, I) & P_n(t, J) & P_n(t, K) & P_n(t, L) \\ P_n(u, I) & P_n(u, J) & P_n(u, K) & P_n(u, L) \end{vmatrix}.$$

7.2 Geïmpliceerde topologie op het grove rooster

Indien een grof zijde element $\vec{E}_{IJ}$ verschillend is van 0, dan stellen we dat de grove knopen n_I en n_J verbonden zijn door een grove zijde e_{IJ}. De oriëntatie van e_{IJ} kan arbitrair gekozen worden. Op een gelijkaardige manier worden grove vlakken f_{IJK} en grove volumes v_{IJKL} gedefiniëerd. Net als op het fijne rooster laat dit toe om een discrete gradiënt G_c, rotor C_c en divergentie D_c operator te definiëren voor het grove rooster:

$$G_c(e_{IJ}, n_K) := \begin{cases} 1 & \text{if} \quad n_K = \mathrm{end}(e_{IJ}), \\ -1 & \text{if} \quad n_K = \mathrm{start}(e_{IJ}), \\ 0 & \text{else}, \end{cases} \tag{20}$$

$$C_c(f_{IJK}, e_{RS}) := \begin{cases} 1 & \text{if} \quad e_{RS} \in \partial f_{IJK}, \\ -1 & \text{if} \quad -e_{RS} \in \partial f_{IJK}, \\ 0 & \text{else}, \end{cases} \tag{21}$$

$$D_c(v_{IJKL}, f_{RST}) := \begin{cases} 1 & \text{if} \quad f_{RST} \in \partial v_{IJKL}, \\ -1 & \text{if} \quad -f_{RST} \in \partial v_{IJKL}, \\ 0 & \text{else}. \end{cases} \tag{22}$$

De detectie van grove zijdes, vlakken en volumes kan gebeuren op basis van de volgende criteria:

$$\begin{aligned}
\vec{E}_{IJ} \not\equiv 0 &\iff \exists \pm e_{rs} \ : \ P_n(r,I), P_n(s,J) \neq 0. \\
\vec{F}_{IJK} \not\equiv 0 &\iff \exists \pm f_{rst} \ : \ P_n(r,I), P_n(s,J), P_n(t,K) \neq 0. \\
V_{IJK} \not\equiv 0 &\iff \exists \pm v_{rstu} \ : \ P_n(r,I), P_n(s,J), P_n(t,K), P_n(u,L) \neq 0.
\end{aligned}$$

Tesamen vormen de grove zijden, vlakken en volumes een grove topologie. In het algemene geval bevat deze topologie kruisende zijden en vlakken, en overlappende elementen. We tonen aan dat dit fenomeen zorgt voor extra geheugengebruik in AMG methodes. De grove topologie kan evenwel gecontroleerd worden door bijkomende beperkingen op te leggen voor de spaarsheidsstructuur van P_n.

In tegenstelling tot de Whitney elementen op een gewone vermazing, zijn de grove zijde, vlak en volume elementen niet genormalizeerd.

7.3 De Rham complex op het grove rooster

De prolongatoren P_n, P_e, P_f en P_v, en de grove en fijne discrete differentiaaloperatoren zijn gerelateerd door de volgende commutatie-eigenschappen:

$$P_e G_c = G_f P_n, \qquad P_f C_c = C_f P_e, \qquad P_v D_c = D_f P_f. \tag{23}$$

Deze eigenschappen zijn van groot belang in de context van multigrid algoritmes. Uit deze eigenschappen kan afgeleid worden dat de grove differentiaaloperatoren voldoen aan de volgende eigenschappen:

$$\ker(G_c) = \mathrm{ran}(\mathbf{1}_c), \qquad \ker(C_c) = \mathrm{ran}(G_c), \qquad \ker(D_c) = \mathrm{ran}(C_c),$$

en dat de prolongatoren P_n, P_e, P_f en P_v de structuur van het de Rham complex (8) overbrengen naar het grove niveau:

$$\begin{array}{ccccccccc}
\mathbb{R} & \xrightarrow{\mathbf{1}_f} & \mathcal{N}_f & \xrightarrow{G_f} & \mathcal{E}_f & \xrightarrow{C_f} & \mathcal{F}_f & \xrightarrow{D_f} & \mathcal{V}_f \\
 & & \uparrow P_n & & \uparrow P_e & & \uparrow P_f & & \uparrow P_v \\
\mathbb{R} & \xrightarrow{\mathbf{1}_c} & \mathcal{N}_c & \xrightarrow{\mathbf{G}_c} & \mathcal{E}_c & \xrightarrow{\mathbf{C}_c} & \mathcal{F}_c & \xrightarrow{\mathbf{D}_c} & \mathcal{V}_c.
\end{array} \tag{24}$$

Als gevolg hiervan wordt ook de structuur van de div-grad, curl-curl, en grad-div operatoren overgebracht naar de grove niveaus van een multigrid hiërarchie. Dit is essentieel voor de recursieve toepassing van AMG algoritmes.

7.4 Numerieke resultaten

Het zijde prolongatieschema (20) werd geïmplementeerd binnen een AMG methode voor de curl-curl vergelijking. Numerieke experimenten tonen aan dat deze methode zeer efficiënt is in vergelijking met alternatieve AMG prolongatieschema's, zowel wat convergentiesnelheid, rekentijd als geheugengebruik betreft.

8 Algebraische multigrid voor hoge orde tijdsdiscretizatie

Multigrid methodes voor de div-grad vergelijking (1) en de curl-curl vergelijking (2) zijn bijzonder efficiënt voor achterwaartse Euler (BE) tijdsdiscretizaties van tijdsafhankelijke div-grad en curl-curl problemen

$$-\operatorname{div}(\alpha \operatorname{grad}(V)) + \beta \frac{\partial V}{\partial t} = \rho, \tag{25}$$

$$\operatorname{curl}(\alpha \operatorname{curl}(\vec{A})) + \beta \frac{\partial \vec{A}}{\partial t} = \vec{J}. \tag{26}$$

Ze kunnen echter niet onmiddellijk toegepast worden voor volledig impliciete Runge-Kutta (IRK) tijdsdiscretizaties of randelementen tijdsdiscretizaties (BVM). Deze tijdsdiscretizatie methodes zijn vaak gewenst omwille van hun goede stabiliteit en hoge nauwkeurigheid.

8.1 Hogere orde tijdsdiscretizatie

IRK en BVM tijdsdiscretizaties van (25) en (26) vereisen de oplossing van een tensor lineair probleem dat geschreven kan worden in de volgende vorm:

$$MX + \Delta t K X A_{irk}^T = B. \tag{27}$$

Met n het aantal knopen respectievelijk zijden van de vermazing, zijn $M, K \in \mathbb{R}^{n \times n}$ dezelfde massa- en stijfheidsmatrices waaruit ook de discretizaties van de problemen (1) en (2) opgebouwd zijn. $A_{irk} \in \mathbb{R}^{s \times s}$ is een matrix van coëfficiënten die afhangen van de gebruikte tijdsdiscretizatie methode. Het aantal tussenstappen van de tijdsdiscretizatie methode wordt voorgesteld door s. De onbekenden X en de excitaties B zijn geordend als $n \times s$ reële matrices.

8.2 Een blok algebraisch multigrid algoritme

We ontwikkelen een AMG algoritme voor (27) dat gebruik maakt van bloksmoothing, blokprolongatie en blokrestrictie. In de bloksmoother worden alle onbekenden die gerelateerd zijn aan dezelfde knoop of zijde, tesamen opgelost. Voor de knoop of zijde die hoort bij de $i$de rij van X, kan dit geschreven worden als volgt:

$$\begin{aligned} &(M_{ii} I_s + \Delta t K_{ii} A_{irk})\, X(i,:)^T \\ &= B(i,:)^T - \sum_{j \neq i} (M_{ij} I_s + \Delta t K_{ij} A_{irk})\, X(j,:)^T. \end{aligned} \tag{28}$$

De blokprolongatie en -restrictie maakt gebruik van een prolongator P die bekomen is met een klassiek AMG algoritme voor de div-grad vergelijking (1) of de curl-curl vergelijking (2). Meer bepaald, de prolongatie en restrictie gebeuren voor elke tussenstap (kolom) afzonderlijk:

- blok restrictie: $B_c = P^T \left(B - MX - \Delta t K X A_{irk}^T\right)$,
- blok prolongatie: $X \leftarrow X + PX_c$.

Het systeem op het grove niveau van de multigrid hiërarchie wordt bekomen door een tensor Galerkin product:

$$\left(P^T M P\right) X + \Delta t \left(P^T K P\right) X A_{irk}^T \quad = \quad B_c. \tag{29}$$

Voor curl-curl problemen wordt een blokvariante van de hybride smoother gebruikt:

1. bloksmoothing op $MX + \Delta t K X A_{irk}^T = B$,
2. blokrestrictie: $B_n = G^T \left(B - MX - \Delta t K X A_{irk}^T\right)$,
3. bloksmoothing op $\left(P^T M P\right) X_n + \Delta t \left(P^T K P\right) X_n A_{irk}^T = B_n$,
4. blokprolongatie: $X \leftarrow X + GX_n$,
5. bloksmoothing op $MX + \Delta t K X A_{irk}^T = B$.

8.3 Convergentie-analyse

De iteratiematrix van een AMG algoritme voor de BE discretizatie met tijdstap Δt van (25) en (26) kan geschreven worden als een matrix functie $T(\Delta t)$. Als gebruik gemaakt wordt van dezelfde prolongatoren, dan wordt de iteratiematrix van het blok AMG algoritme voor de hogere orde tijdsdiscretizatie met coëfficiënten A_{irk} van hetzelfde probleem, beschreven door de tensor matrix functie $T(\Delta t A_{irk})$. De eigenwaarde ontbinding van A_{irk} laat toe de asymptotische convergentiefactor van het blok AMG algoritme te berekenen op basis van de asymptotische convergentiefactoren van het gewone AMG algoritme met een gescaleerde tijdstap:

$$\rho\left(T(\Delta t A_{irk})\right) \quad = \quad \max_{\lambda \in \sigma(A_{irk})} \rho\left(T(\Delta t \lambda)\right).$$

8.4 Implementatie en numerieke experimenten

Een naieve implementatie van dit blok AMG algoritme is duur. We stellen verscheidene maatregelen voor om de kost van dit algoritme te reduceren.
Als de tensor systeem matrices volledig opgeslagen worden, dan is het geheugengebruik gelijk aan $s^2 \cdot \mathrm{mem}(K + M)$. Hier is $\mathrm{mem}(A)$ het geheugen nodig om de matrix A op te slagen. Door gebruik te maken van de compacte formulering (27), wordt het geheugengebruik voor de tensor systeem matrices echter gereduceerd tot $\mathrm{mem}(K) + \mathrm{mem}(M)$.
Het aantal floating point operaties van het blok AMG algoritme scaleert in principe als s^3. Deze kubische scalering kan gereduceerd worden tot een kwadratische door de $s \times s$ matrices van de basistap (28) van de bloksmoother te berekenen, te inverteren en op te slagen in de setup faze. Verder kan de snelheid van de berekeningen sterk opgedreven worden door:

- gebruik te maken van ad hoc geoptimaliseerde inlined routines voor de basis stap (28) van de bloksmoother, in plaats van generieke BLAS / LAPACK routines,
- de multivectoren X en B rij per rij op te slagen in plaats van kolom per kolom, wat zorgt voor een verbetering van het cache gebruik,
- gebruik te maken van template code in functie van het aantal tussenstappen s, wat de compiler toelaat bijkomende optimizaties (in het bijzonder loop unrolling) uit te voeren.

Numerieke experimenten met dit blok AMG algoritme tonen aan dat de convergentie vergelijkbaar of beter is dan de convergentie van het overeenkomstige gewone AMG algoritme. Voor een beperkt aantal tussenstappen s is bovendien de kost per iteratie zeer gunstig.

9 Suggesties voor verder onderzoek

Het onderzoek van dit doctoraat kan bijvoorbeeld op de volgende manieren voortgezet worden:

- De nieuwe algoritmes ontwikkeld in dit doctoraat kunnen getest worden op industriële problemen.
- Het algoritme om de grove topologie van een nodale prolongator te controleren, werkt niet optimaal voor drie-dimensionale problemen. Dit kan verbeterd worden.
- Er kan onderzocht worden of ingewikkelde rand- of koppelingsvoorwaarden met behulp van commutatie naar de grove niveaus van een multigrid hiërarchie kunnen overgebracht worden.
- Het energie minimizatie algoritme voor nodale en zijde prolongatoren, ontwikkeld in de appendix van dit doctoraat, kan uitgebreider getest worden en kan vergeleken worden met andere AMG algoritmes.

List of definitions and symbols

Functional analysis

Ω	computational domain
d	spatial dimension of the Ω
$(\cdot, \cdot)$	inner product
$L^2(\Omega)$	square integrable scalar functions on Ω
$\vec{L}^2(\Omega)$	square integrable vector functions on Ω
$C^m(\Omega)$	m-fold continuously differentiable scalar functions on Ω
$C_0^m(\Omega)$	m-fold continuously differentiable scalar functions on Ω with compact support
$\vec{C}^m(\Omega)$	$:= (C^m(\Omega))^d$
$\vec{C}_0^m(\Omega)$	$:= (C_0^m(\Omega))^d$
$H^1(\Omega)$	$:= \{x \in L^2(\Omega) \mid \text{grad}(x) \in \vec{L}^2\}$
$H(\Omega, \text{grad})$	$:= H^1(\Omega)$
$H(\Omega, \text{curl})$	$:= \{\vec{x} \in \vec{L}^2(\Omega) \mid \text{curl}(\vec{x}) \in \vec{L}^2\}$
$H(\Omega, \text{div})$	$:= \{\vec{x} \in \vec{L}^2(\Omega) \mid \text{div}(\vec{x}) \in L^2\}$
∂	boundary operator
$\ker(f)$	kernel of an operator f
$\text{ran}(f)$	range of an operator f
$\text{supp}(f)$	support of an operator f, i.e. the domain in which f is not equal to 0.

Mesh

n, e, f, v	number of nodes, edges, faces, volumes of a mesh
s_i	simplex (i.e. node, edge, face or volume) with index i
n_i, e_i, f_i, v_i	node, edge, face, volume with index i
e_{ij}	edge from n_i to n_j

f_{ijk}	triangular face connecting nodes n_i, n_j and n_k
v_{ijkl}	tetrahedral volume connecting nodes n_i, n_j, n_k, n_l
N, E, F, V	set of all nodes, edges, faces, volumes
$\mathcal{N}, \mathcal{E}, \mathcal{F}, \mathcal{V}$	vector space of degrees of freedom associated with N, E, F and V
area(f)	area of f
vol(v)	volume of v
τ	mesh
$\tilde{\tau}$	dual mesh
$\tilde{.}$	refers to the dual mesh
T_i	element with index i of a mesh τ

Vector analysis

S_i	shape functions associated with simplex s_i
N_i	nodal element associated with n_i
$\vec{E}_i$, $\vec{E}_{ij}$	edge element associated with e_i, e_{ij}
$\vec{F}_i$, $\vec{F}_{ijk}$	face element associated with f_i, f_{ijk}
V_i, V_{ijkl}	volume element associated with v_i, v_{ijkl}
$\mathcal{S}_N$, $\mathcal{S}_E$, $\mathcal{S}_F$, $\mathcal{S}_V$	range space of all nodal, edge, face, volume elements
$\mathcal{N}$, $\mathcal{E}$, $\mathcal{F}$, $\mathcal{V}$	set of the degrees of freedom of all nodal, edge, face, volume elements
Π_N, Π_E, Π_F, Π_V	nodal, edge, face, volume interpolation operators
Σ_N, Σ_E, Σ_F, Σ_V	nodal, edge, face, volume sampling operators
ϵ_x, ϵ_y, ϵ_z	unit vector in X-, Y-, Z-directions
$.^{\perp}$	refers to the perpendicular translation-invariant mode for 2D problems
$.^{\parallel}$	refers to the parallel translation-invariant mode for 2D problems

Differential operators

cst(r)	constant function mapping to r
grad(u)	$\left(\frac{\partial u}{dx}, \frac{\partial u}{dy}, \frac{\partial u}{dz}\right)$
curl$(\vec{u})$	$\left(\frac{\partial u_z}{dy} - \frac{\partial u_y}{dz}, \frac{\partial u_x}{dz} - \frac{\partial u_z}{dx}, \frac{\partial u_y}{dx} - \frac{\partial u_x}{dy}\right)$
div$(\vec{u})$	$\frac{\partial u_x}{dx} + \frac{\partial u_y}{dy} + \frac{\partial u_z}{dz}$
$\mathbf{1}$	constant vector with value 1
G	discrete gradient operator
C	discrete curl operator
D	discrete divergence operator
$\tilde{G}, \tilde{C}, \tilde{D}$	dual discrete gradient, curl, divergence operator
${}^{w}.$	refers to the weak differential operators

$\cdot_g$, $\cdot_s$	refers to gradient and solenoidal vectors, i.e. vectors belonging to ran(G) and ran(C^T) respectively

Electromagnetism

$\vec{E}$	electric field strength	[V/m]
$\vec{D}$	electric displacement density	[C/m^2]
$\vec{B}$	magnetic flux density	[T]
$\vec{H}$	magnetic field strength	[A/m]
V	electric scalar potential	[V]
$\vec{A}$	magnetic vector potential	[Tm]
I	current	[A]
$\vec{J}$	current density	[A/m^2]
ρ	charge density	[C/m^3]
ϵ	permittivity	[F/m]
μ	permeability	[H/m]
σ	conductivity	[S/m]

Matrices

K_α^{dg}	discretization of div(α grad($\cdot$)) using nodal elements
K_α^{cc}	discretization of curl(α curl($\cdot$)) using edge elements
K_α^{gd}	discretization of grad(α div($\cdot$)) using face elements
M_β^n	mass matrix, discretization of $\beta\cdot$ using nodal elements
M_β^e	mass matrix, discretization of $\beta\cdot$ using edge elements
M_β^f	mass matrix, discretization of $\beta\cdot$ using face elements
M_β^v	mass matrix, discretization of $\beta\cdot$ using volume elements

Multigrid

$\cdot_c$	refers to the coarse grid
$\cdot^\nu$	refers to a value after ν iterations
$\tilde{\cdot}$	refers to the updated value

Local Fourier analysis

h	grid size

β_h	$:= \beta h^2$
$\mathcal{G}_h$	regular quadrilateral grid with mesh size h
c_i	$\cos(\theta_i/2)$
s_i	$\sin(\theta_i/2)$
$\tilde{c}_i$	$\cos(\theta_i)$
$\tilde{s}_i$	$\sin(\theta_i)$
$\mathcal{ST}_L$	stencil of the operator L
S	iteration matrix of a smoother
d	damping factor for a matrix-splitting iterative method
$\|S\|$	number of elements of a set S
Θ	$:= [-\pi/2, 3\pi/2)^2$
Θ^{low}	$:= [-\pi/2, \pi/2)^2$
Θ^{high}	$:= \Theta \backslash \Theta^{\text{low}}$

MATLAB notation

Throughout this thesis, the MATLAB [51] notational convention will be used to extract a subvector of a vector x or a submatrix of a $n \times m$ matrix A. More precisely, with r and c denoting vectors of indices, we assign the following meaning to the assignments $y = x(r)$ and $B = A(r, c)$:

$$\begin{array}{lcll} y = x(r) & \longrightarrow & y(i) := x(r(i)), & \forall 0 < i \leq \text{size}(r), \\ B = A(r,c) & \longrightarrow & B(i,j) := A(r(i), c(j)), & \forall 0 < i \leq \text{size}(r),\ 0 < j \leq \text{size}(c). \end{array}$$

With the colon ":" we shall indicate the entire row or column range. For example:

$$\begin{array}{lcll} B = A(r,:) & \longrightarrow & B(i,j) := A(r(i), j), & \forall 0 < i \leq \text{size}(r),\ 0 < j \leq m, \\ B = A(:,c) & \longrightarrow & B(i,j) := A(i, c(j)), & \forall 0 < i \leq n,\ 0 < j \leq \text{size}(c). \end{array}$$

List of acronyms

AMG	algebraic multigrid
BE	backward Euler time discretization
BiCGstab	bi-conjugate gradients stabilized
BVM	boundary value method time discretization
CG	conjugate gradients
DIRK	diagonally implicit Runge-Kutta time discretization
DoF	degree of freedom
FE	finite element discretization, or forward Euler discretization in time
GAM	generalized Adam time discretization
GMG	geometric multigrid
GMRES	generalized minimal residual
GS	Gauss-Seidel iteration
ILU	incomplete LU factorization
IRK	fully implicit Runge-Kutta time discretization
LFA	local Fourier analysis
LMM	linear multistep time discretization
MG	multigrid
ODE	ordinary differential equation
PDE	partial differential equation

Contents

Preface iii

Abstract v

Nederlandse samenvatting vii

List of definitions and symbols xxiii

List of acronyms xxvii

1 Introduction 1
- 1.1 Overview of the research domain and contributions of this work . 1
  - 1.1.1 Multigrid . . . 1
  - 1.1.2 High order time discretization . . . 2
- 1.2 Outline . . . 3

2 A review of computational electromagnetics 5
- 2.1 Introduction . . . 5
- 2.2 Governing laws . . . 6
  - 2.2.1 Maxwell's laws . . . 6
  - 2.2.2 Constitutive laws . . . 6
  - 2.2.3 Transmission conditions . . . 6
- 2.3 Second order formulations . . . 7
  - 2.3.1 Static problems . . . 8
  - 2.3.2 Time-dependent problems . . . 8
- 2.4 A functional framework for electromagnetism . . . 9
  - 2.4.1 Weak differential operators . . . 9
  - 2.4.2 De Rham sequence . . . 10
  - 2.4.3 Function spaces for electromagnetic fields . . . 11
  - 2.4.4 Mathematical problem setting . . . 11
- 2.5 The discrete setting . . . 13
  - 2.5.1 Shape functions . . . 14
  - 2.5.2 Discrete differential operators . . . 22
  - 2.5.3 Discrete de Rham sequence . . . 23
  - 2.5.4 Finite element discretization . . . 26

2.6 Conclusion . . . 30

3 Introduction to multigrid 31
3.1 Matrix splitting iterations . . . 31
3.2 The multigrid principle . . . 32
3.3 Local Fourier analysis for multigrid . . . 35
3.3.1 Definitions and notations . . . 35
3.3.2 Fourier modes . . . 36
3.3.3 Smoother . . . 39
3.3.4 Intergrid transfer operators . . . 40
3.3.5 Multigrid cycle . . . 41
3.4 Concluding remarks . . . 42

4 Overview of algebraic multigrid for the div-grad equation 45
4.1 Smoothing . . . 45
4.1.1 Pointwise and polynomial smoothing methods . . . 46
4.1.2 Algebraically smooth error . . . 46
4.1.3 Reduction of the energy norm of the error . . . 49
4.2 Coarsening . . . 49
4.2.1 C/F-splitting . . . 49
4.2.2 Node aggregation . . . 51
4.2.3 Element agglomeration . . . 51
4.3 Prolongation . . . 52
4.3.1 Constraints . . . 52
4.3.2 Ideal prolongator . . . 55
4.3.3 Classical AMG . . . 56
4.3.4 Smoothed aggregation . . . 56
4.3.5 Energy minimization of a prolongator . . . 57
4.3.6 Element-based AMG methods . . . 58
4.3.7 Adaptive AMG . . . 61
4.4 Galerkin coarse operator . . . 62
4.5 Concluding remarks . . . 63

5 Overview of multigrid for the curl-curl equation 65
5.1 Geometric multigrid . . . 65
5.1.1 Prolongation . . . 66
5.1.2 Smoothing . . . 68
5.2 AMG methods satisfying the commutation property . . . 73
5.2.1 Piecewise constant edge prolongation . . . 74
5.2.2 Smoothed aggregation . . . 76
5.2.3 Constrained solution of the commutation equality . . . 77
5.2.4 Energy minimization . . . 77
5.2.5 Parallel multigrid hierarchy . . . 78
5.3 AMG methods with decoupled parallel hierarchy . . . 79
5.3.1 Reformulation as a block system . . . 79
5.3.2 Multiple coarsening AMG . . . 80

5.4 Concluding remarks 84

6 Fourier analysis of multigrid in $H(curl)$ 85
6.1 Introduction 85
6.2 Definitions and notations 86
6.3 Fourier modes 87
6.4 The Helmholtz splitting in Fourier space 91
6.5 Fourier domain representation of the operator 93
6.5.1 Stiffness part 93
6.5.2 Mass part 94
6.6 Fourier domain representation of the smoother 94
6.6.1 Classical smoothers 94
6.6.2 Hybrid smoother 97
6.6.3 Overlapping block smoother 100
6.7 Fourier domain representation of the intergrid transfer operators 103
6.7.1 Coarse grid 103
6.7.2 Restriction 103
6.7.3 Prolongation 105
6.8 Multigrid cycle 107
6.9 Remarks about the numerical tests 108
6.10 Generalization 110
6.11 Conclusion 112

7 A new family of prolongation operators 113
7.1 Introduction 113
7.2 Prolongators derived from a nodal prolongator 114
7.2.1 Edge prolongator 114
7.2.2 Face prolongator 116
7.2.3 Volume prolongator 119
7.2.4 Piecewise constant prolongators 120
7.2.5 Remark on determinants in finite element discretizations . 121
7.3 Implied coarse topology 121
7.3.1 Existence criteria for coarse simplices 122
7.3.2 Coarse differential operators 125
7.3.3 Properties of the implied coarse topology 125
7.3.4 Impact of the coarse topology on the memory requirements of AMG methods 129
7.4 Normalization 130
7.5 Coarse de Rham complex 134
7.6 Structure of the coarse system matrices 139
7.7 Extension to quadrilateral and hexahedral meshes 141
7.7.1 Application of the formulae for quadrilateral and hexahedral meshes 141
7.7.2 Application of the formulae for simplicial meshes 142
7.8 Implementation 143
7.8.1 Construction of the underlying nodal prolongator 144

7.8.2 Controlling the implied coarse topology 145
7.9 Numerical results 145
7.9.1 Two-dimensional problems 146
7.9.2 Three-dimensional problems 152
7.10 Concluding remarks 152

8 AMG for higher order time discretization 155
8.1 Introduction 155
8.2 Discretization in time 156
8.2.1 Time discretization methods 156
8.2.2 Application to a semi-discretized PDE 159
8.3 A block algebraic multigrid algorithm 161
8.3.1 Prolongation and restriction 161
8.3.2 Construction of the coarse system 162
8.3.3 Definition of the smoother 162
8.4 Convergence analysis 163
8.4.1 A formula for the asymptotic convergence factor 163
8.4.2 Discussion 165
8.5 Implementation issues 166
8.5.1 Storage of the multivectors 168
8.5.2 Storage of the tensor system matrices 168
8.5.3 Blocksmoother 169
8.5.4 Loop unrolling 169
8.6 Numerical experiments 170
8.7 Conclusion 172

9 Conclusions 175
9.1 Overview of the central results 175
9.1.1 LFA of multigrid for curl-curl problems 175
9.1.2 AMG for curl-curl problems 175
9.1.3 AMG for high order time discretizations 176
9.2 Some suggestions for further research 177
9.2.1 Application to industrial problems 177
9.2.2 Coarse topology control 177
9.2.3 Commutation properties 177
9.2.4 AMG based on energy minimization 178

A Energy minimization of multigrid prolongators 179
A.1 Energy minimization of a nodal prolongator 179
A.2 Energy minimization of an edge prolongator 181
A.3 The importance of prolongator normalization 183

Chapter 1

Introduction

1.1 Overview of the research domain and contributions of this work

In this thesis, we will study multigrid methods for solving linear systems obtained from the discretization of the time-dependent div-grad and curl-curl equations

$$\begin{aligned} -\operatorname{div}(\alpha \operatorname{grad}(U)) + \beta \frac{\partial U}{\partial t} &= b \\ \operatorname{curl}(\alpha \operatorname{curl}(\vec{U})) + \beta \frac{\partial \vec{U}}{\partial t} &= \vec{b}. \end{aligned}$$

These partial differential equations (PDE) can be used to model approximations of Maxwell's equations.

1.1.1 Multigrid

The discretization of these PDEs gives rise to large sparse linear systems, which are difficult to solve. Multigrid methods are among the most efficient solvers for this type of linear systems. They use a hierarchy of linear systems of decreasing size, which can be considered representations of the original problem on different grids. On each grid, a simple iterative solver is applied, which is called a smoother. The efficiency of multigrid methods depends on the complementarity of the smoothing operators and the strategy used to generate the coarse grid problems. Two main families of multigrid methods exist. In geometric multigrid methods (GMG), the smoothers are adapted to the coarse grids, which are predefined. In algebraic multigrid methods (AMG), the smoothers are chosen in advance, and the construction of the coarse grids is adapted to them.

The div-grad equation is the model problem for geometric multigrid methods. This problem has been studied for several decades. It is well understood

and supported by a large amount of theory. Several handbooks are available on this topic [63, 18, 68].

AMG methods for the div-grad equation have been introduced in the early eighties of the previous century. One of the first AMG methods was developed in [59]. This method is often described as classical AMG. A large variety of AMG methods for div-grad problems has been developed since, such as AMG based on energy minimization [48, 67], smoothed aggregation AMG [65], element-based AMG methods [15, 21, 42] and adaptive AMG [17, 16]. Several efficient implementations are available and in widespread use [28, 49, 61].

The smoothing and coarsening techniques used in multigrid for the div-grad equation cannot be applied immediately for the curl-curl equation. When applied to curl-curl problems, classical smoothers perform poorly, because they cannot deal with the large kernel of the curl-curl operator. Also, the direction and orientation of the edges, to which the degrees of freedom of curl-curl problems are related, cannot be taken into account appropriately by node-based coarsening and prolongation principles.

Geometric multigrid for curl-curl problems was pioneered in [35]. For this purpose, a specialized smoother, the so-called hybrid smoother, was used. In [3], an overlapping blocksmoother was developed which achieves similar results. Both smoothers are used in AMG methods as well. In the theory of geometric multigrid for curl-curl problems, there still are some issues that need to be resolved. One of these issues, the development of a Fourier analysis, will be addressed in chapter 6 of this thesis.

The first successful AMG method for curl-curl problems was developed in [56]. Here, the so-called node-edge commutation property was identified as a key principle to successful AMG coarsening. This commutation property serves as a guideline for deriving the coarsening for a curl-curl problem from the coarsening used for a related div-grad problem. It has been combined with smoothed aggregation techniques [5, 39] and energy minimization [53]. Other coarsening techniques are based on a reformulation of the curl-curl problem as a block system [41], a transformation of the edge-based unknowns to node-based unknowns [4, 38] or gauging techniques [6]. Typically, existing AMG methods for curl-curl problems don't achieve the same convergence rates as their counterparts for div-grad problems. There is room for improvement here, and there is substantial research going on in this area. Chapter 7 of this thesis contributes to this research. In this chapter, a new AMG method for curl-curl problems will be developed.

1.1.2 High order time discretization

Multigrid methods are typically developed for stationary problems or backward Euler (BE) time discretizations of time-dependent problems. These multigrid techniques can not be applied to all types of high order time discretization schemes. For fully implicit Runge-Kutta (IRK) methods and Boundary Value Methods (BVM), which are often preferred because of their superior stability and accuracy properties, this is for instance not possible.

Currently, two techniques exist to apply multigrid techniques to IRK and BVM discretizations. In [50], a preconditioner is used in which the time discretization scheme is approximated. This approximation is constructed such that standard multigrid techniques for BE time discretizations can be reused.

In [64], no additional approximations are introduced. Instead, knowledge about the underlying time discretization scheme is used inside the multigrid method. For this purpose, both the smoothers and the coarsening strategy are changed. This algorithm has been developed in the context of geometric multigrid for the time-dependent div-grad equation. In chapter 8, it will be extended to algebraic multigrid for the time-dependent div-grad and curl-curl equations.

1.2 Outline

In chapter 2, we will introduce the terminology, concepts and techniques from computational electromagnetics which will be needed throughout this thesis. The discretization techniques presented in this chapter will be the guideline for the AMG techniques of chapter 7. Chapter 2 is largely based on the monographs [13, 52, 37], and can easily be skipped by readers familiar with computational electromagnetics.

In chapter 3, we will introduce the basic principles of multigrid methods. We will also present the essential concepts of local Fourier analysis (LFA), which is an important tool for the analysis of multigrid methods. This will serve as a reference for the development of the local Fourier analysis for curl-curl problems in chapter 6. Chapter 3 is largely based on the monograph [63], and can easily be skipped by readers familiar with multigrid.

In chapters 4 and 5, we will present an overview of AMG methods for div-grad and curl-curl problems problems. These chapters are the result of a literature study, and introduce many essential concepts for the remainder of this work.

Chapters 6, 7 and 8 contain the three central contributions of this work. In chapter 6, we will develop a local Fourier analysis for multigrid for the curl-curl equation. This is a new contribution to the field, since existing Fourier techniques cannot be applied immediately to this problem and need to be extended. We will show that these extended Fourier techniques can be applied to other problems as well. The Fourier analysis results have been published in [11].

In chapter 7, we will develop a new AMG algorithm for the curl-curl equation. This algorithm is based on the application of the discretization techniques of chapter 2 on the coarse levels. This principle can be extended to other problem types, such as the grad-div equation. Several key properties of the discretization will be recovered on the coarse levels as well. Numerical results will be presented, showing the efficiency of this AMG algorithm. This chapter is based on our paper [8].

In chapter 8, we will focus on AMG for high order time discretization schemes. We will extend the algorithm of [64] to AMG for the div-grad and

curl-curl equations. The convergence of the resulting methods will be analyzed theoretically, and their efficiency will be demonstrated by numerical experiments. The results of this chapter have been published in [9] and [10]. Similar ideas are also at the basis of the paper [57].

In chapter 9, we will present some concluding remarks and outline some ideas for further research.

Finally, some unpublished work has been collected into an appendix. It describes a new energy minimization procedure, which can be applied to the prolongation operators for AMG methods for the div-grad and curl-curl equations.

Chapter 2

A review of computational electromagnetics

2.1 Introduction

In this chapter, we will introduce the terminology, concepts and techniques from computational electromagnetics which will be needed throughout this thesis. This will serve two purposes. First, it allows to clearly define the physical and mathematical setting. Second, many of these concepts will be reused in the analysis of the multigrid solvers developed in this thesis.

This chapter is organized as follows. In section 2.2, the governing laws of electromagnetics are described. Also, the continuity characteristics of electromagnetic field quantities will be studied. In this thesis, we will focus on low-frequency problems, and we will consider reformulations of the governing laws as second order partial differential equations (PDEs). The approximations and reformulations used for this purpose will be clarified in section 2.3.

In section 2.4, we will introduce the classical functional framework for electromagnetic problems. This framework is essentially induced by the specific continuity characteristics of electromagnetic fields, and is based on the use of weak differential operators. These operators are related by a so-called de Rham sequence. The properties of this de Rham sequence will be studied.

In the last section 2.5, we will focus on spatial discretization using the so-called Whitney finite elements. Whitney elements are a natural choice for discretizations in the functional framework we are operating in. This is essentially because they transfer the properties, represented by the de Rham sequence, to the discrete level. This is important for the stability of the discretization.

2.2 Governing laws

In this section, we will present the governing laws of electromagnetism. They consist of the Maxwell's laws and the so-called constitutive laws, which describe the material properties. Also, the specific continuity characteristics of electromagnetic field quantities, which are implied by Maxwell's laws, will be derived.

2.2.1 Maxwell's laws

Maxwell's laws form the basis for electromagnetism. In differential form, they read:

$$\begin{aligned} \mathrm{div}(\vec{D}) &= \rho, && (2.1)\\ \mathrm{div}(\vec{B}) &= 0, && (2.2)\\ \mathrm{curl}(\vec{E}) &= -\frac{\partial \vec{B}}{\partial t}, && (2.3)\\ \mathrm{curl}(\vec{H}) &= \vec{J} + \frac{\partial \vec{D}}{\partial t}. && (2.4) \end{aligned}$$

Here, $\vec{D}$ denotes the electric flux density, $\vec{B}$ the magnetic flux density, $\vec{E}$ the electric field, $\vec{H}$ the magnetic field, ρ the charge density and $\vec{J}$ the current density. Together, equations (2.1) and (2.4) imply conservation of electric charge:

$$\mathrm{div}\left(\mathrm{curl}(\vec{H})\right) = \mathrm{div}\left(\vec{J} + \frac{\partial \vec{D}}{\partial t}\right) = 0 \implies \mathrm{div}(\vec{J}) + \frac{\partial \rho}{\partial t} = 0. \qquad (2.5)$$

2.2.2 Constitutive laws

The electric quantities $\vec{E}$, $\vec{D}$ and $\vec{J}$, and the magnetic quantities $\vec{H}$ and $\vec{B}$ are related by material properties. In general these relations are non-linear tensor relations. In this thesis, we will only consider linear materials, which satisfy:

$$\begin{aligned} \vec{D} &= \epsilon \vec{E}, && (2.6)\\ \vec{B} &= \mu \vec{H}, && (2.7)\\ \vec{J} &= \vec{J}_0 + \sigma \vec{E}. && (2.8) \end{aligned}$$

$\vec{J}_0$ denotes the current due to a current source. The permittivity ϵ, permeability μ and conductivity σ are material properties.

2.2.3 Transmission conditions

The direct application of Maxwell's laws in differential form (2.1)-(2.2)-(2.3)-(2.4) assumes that all components of the vector fields $\vec{D}$, $\vec{B}$, $\vec{H}$ and $\vec{E}$ are differentiable. At material interfaces, however, these vector fields are not differentiable, not even continuous. Here, very specific transmission conditions are to be satisfied.

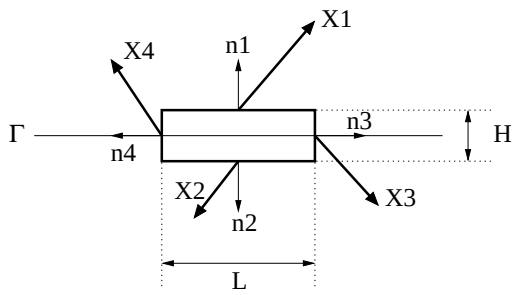


Figure 2.1: Contour around a point belonging to an interface Γ between two different materials. The vectors n_i denote the unit outward normal vectors.

For simplicity reasons, we will derive these transmission conditions in the 2D setting of Figure 2.1. Consider an infinitesimal quadrilateral, with $H, L, \frac{H}{L} \to 0$, around a point belonging to an interface Γ between two different materials, see figure 2.1. The integration over this quadrilateral of the equation

$$\frac{1}{L}\left(\mathrm{div}(\vec{x}) - y\right) \quad = \quad 0,$$

which is generic for (2.1), (2.2) and (2.5), yields:

$$\lim_{H,L,\frac{H}{L}\to 0} \frac{1}{L}\left(L\vec{x}_1 \cdot \vec{n}_1 + L\vec{x}_2 \cdot \vec{n}_2 + H\vec{x}_3 \cdot \vec{n}_3 + H\vec{x}_4 \cdot \vec{n}_4 - LHy\right) \quad = \quad 0.$$

This leads to:

$$\vec{x}_1 \cdot \vec{n}_1 + \vec{x}_2 \cdot \vec{n}_2 \quad = \quad 0. \tag{2.9}$$

This shows that *only the normal components* of $\vec{D}$, $\vec{B}$ and $\vec{J}$ need to be continuous at an interface. In a similar way, the integration over the same infinitesimal quadrilateral of the equation

$$\frac{1}{L}\left(\mathrm{curl}(\vec{x}) - \vec{y}\right) \quad = \quad 0,$$

which is generic for (2.3) and (2.4), yields:

$$\vec{n}_1 \times (\vec{n}_1 \times \vec{x}_1) + \vec{n}_2 \times (\vec{n}_2 \times \vec{x}_2) \quad = \quad \vec{0}. \tag{2.10}$$

This shows that *only the tangential components* of $\vec{H}$ and $\vec{E}$ need to be continuous at an interface.

2.3 Second order formulations

In this thesis, we will focus on the solution of specific types of second order partial differential equations (PDE), which arise from reformulations of Maxwell's equations. In this section, these reformulations will be derived.

2.3.1 Static problems

For static problems, the coupling between the electric and magnetic problem by the time-derivatives in (2.3) and (2.4) disappears. This allows both problems to be solved separately.

Electrostatics

For static problems, (2.3) shows that $\vec{E}$ is curl-free. Consequently, (2.31) allows $\vec{E}$ to be derived from a scalar potential V:

$$\operatorname{curl}(\vec{E}) = 0 \quad \Longrightarrow \quad \exists V \ : \ \vec{E} = -\operatorname{grad}(V). \tag{2.11}$$

Using V, which is called the electric scalar potential, (2.1) reads:

$$-\operatorname{div}(\epsilon \operatorname{grad}(V)) = \rho. \tag{2.12}$$

Magnetostatics

(2.2) allows $\vec{B}$ to be derived from a vector potential $\vec{A}$:

$$\operatorname{div}(\vec{B}) = 0 \quad \Longrightarrow \quad \exists \vec{A} \ : \ \vec{B} = \operatorname{curl}(\vec{A}). \tag{2.13}$$

Using $\vec{A}$, which is called the magnetic vector potential, (2.4) reads:

$$\operatorname{curl}\left(\frac{1}{\mu}\operatorname{curl}(\vec{A})\right) = \vec{J}. \tag{2.14}$$

2.3.2 Time-dependent problems

For time-dependent problems, we will consider the so-called magneto-quasi-static or eddy current approximation (see [25]). This approximation neglects the displacement current density $\frac{\partial \vec{D}}{\partial t}$ in (2.4). This can be done if $\frac{\partial \vec{D}}{\partial t}$ is small with respect to the conductive current density $\vec{J}$. This is typically the case for low frequency applications, such as the simulation of electrical machines.

Note that for the magneto-quasi-static approximation, the charge conservation equation (2.5) reduces to:

$$\operatorname{div}(\vec{J}) = 0 \quad \Longleftrightarrow \quad \frac{\partial \rho}{\partial t} = 0. \tag{2.15}$$

Eddy current problem

Using the magnetic vector potential (2.13), equation (2.3) can be enforced by the introduction of an electric scalar potential V:

$$\begin{aligned} \operatorname{curl}(\vec{E}) = -\frac{\partial \vec{B}}{\partial t} \quad &\Longrightarrow \quad \operatorname{curl}\left(\vec{E} + \frac{\partial \vec{A}}{\partial t}\right) = 0 \\ &\Longrightarrow \quad \exists V \in H^1 \ : \ \vec{E} + \frac{\partial \vec{A}}{\partial t} = -\operatorname{grad}(V). \end{aligned} \tag{2.16}$$

Using Ohm's law (2.8), the electric scalar potential (2.16) and the magnetic vector potential (2.13), (2.4) becomes for the magneto-quasi-static approximation:

$$\operatorname{curl}\left(\frac{1}{\mu}\operatorname{curl}(\vec{A})\right)+\sigma\frac{\partial\vec{A}}{\partial t}=\vec{J}_0-\sigma\operatorname{grad}(V). \tag{2.17}$$

The scalar electric potential V is to be calculated in advance, for instance using the techniques presented in [29, 26].

Curl-curl equation for $\vec{E}$

A curl-curl equation for $\vec{E}$ can be derived as follows:

$$\operatorname{curl}(\vec{E})=-\frac{\partial\vec{B}}{\partial t}\quad\Longrightarrow\quad\operatorname{curl}\left(\frac{1}{\mu}\operatorname{curl}(\vec{E})\right)=-\operatorname{curl}\left(\frac{1}{\mu}\frac{\partial\vec{B}}{\partial t}\right).$$

Since the order of time and spatial derivatives can be switched, we have, using subsequently (2.7), (2.4) and (2.8):

$$\operatorname{curl}\left(\frac{1}{\mu}\operatorname{curl}(\vec{E})\right)+\sigma\frac{\partial\vec{E}}{\partial t}=-\frac{\partial\vec{J}_0}{\partial t}. \tag{2.18}$$

2.4 A functional framework for electromagnetism

Since electromagnetic fields are not continuous at material interfaces (see section 2.2.3), the classical definitions of the differential operators gradient, curl and divergence, cannot be used. Instead, a weak form of the differential operators will be needed. In this section, the weak differential operators will be introduced, and their properties will be studied.

The following symbols will be used to denote common function spaces, with Ω denoting the computational domain, and d the spatial dimension:

$$\begin{aligned}
L^2(\Omega) &:= \text{square integrable scalar functions on } \Omega,\\
\vec{L}^2(\Omega) &:= \left(L^2(\Omega)\right)^d,\\
C^m(\Omega) &:= \text{m-fold continuously differentiable scalar functions on } \Omega,\\
C_0^m(\Omega) &:= \text{m-fold continuously differentiable scalar functions on } \Omega\\
&\quad\ \text{with compact support},\\
\vec{C}^m(\Omega) &:= \left(C^m(\Omega)\right)^d,\\
\vec{C}_0^m(\Omega) &:= \left(C_0^m(\Omega)\right)^d.
\end{aligned}$$

2.4.1 Weak differential operators

In a weak formulation, a function equation $f=g$ is not required to hold absolutely, i.e. in each point, but only with respect to a certain set $\mathcal{T}$ of test

functions:

$$ {}^{w}f = {}^{w}g \iff \int_\Omega f\,\varphi = \int_\Omega g\,\varphi, \quad \forall \varphi \in \mathcal{T}. $$

Here, the superscript ${}^{w}\cdot$ refers to the weak form. The weak form of the gradient, curl and divergence operators is defined as follows (see [13]):

$$ {}^{w}\mathrm{grad}(x) = {}^{w}\vec{y} \iff \int_\Omega \mathrm{grad}(x)\cdot\vec{\varphi} = \int_\Omega \vec{y}\cdot\vec{\varphi}, \quad \forall \vec{\varphi} \in \vec{C}_0^1(\Omega), \tag{2.19} $$

$$ {}^{w}\mathrm{curl}(\vec{x}) = {}^{w}\vec{y} \iff \int_\Omega \mathrm{curl}(\vec{x})\cdot\vec{\varphi} = \int_\Omega \vec{y}\cdot\vec{\varphi}, \quad \forall \vec{\varphi} \in \vec{C}_0^1(\Omega), \tag{2.20} $$

$$ {}^{w}\mathrm{div}(\vec{x}) = {}^{w}y \iff \int_\Omega \mathrm{div}(\vec{x})\,\varphi = \int_\Omega y\,\varphi, \quad \forall \varphi \in C_0^1(\Omega). \tag{2.21} $$

The domains of the weak differential operators are respectively (see [13]):

$$ H(\mathrm{grad},\Omega) := \{x \in L^2(\Omega) \mid \mathrm{grad}(x) \in \vec{L}^2\}, \tag{2.22} $$

$$ H(\mathrm{curl},\Omega) := \{\vec{x} \in \vec{L}^2(\Omega) \mid \mathrm{curl}(\vec{x}) \in \vec{L}^2\}, \tag{2.23} $$

$$ H(\mathrm{div},\Omega) := \{\vec{x} \in \vec{L}^2(\Omega) \mid \mathrm{div}(\vec{x}) \in L^2\}. \tag{2.24} $$

$H(\mathrm{grad},\Omega)$ is often denoted as:

$$ H^1(\Omega) := H(\mathrm{grad},\Omega). \tag{2.25} $$

It can be proved (see [13]) that vector functions with discontinuities satisfying the transmission conditions (2.9) and (2.10) are contained in $H(\mathrm{div})$ and $H(\mathrm{curl})$ respectively.

2.4.2 De Rham sequence

The weak differential operators satisfy similar properties as their classical counterparts. In particular, the Poincaré lemma holds for the weak differential operators as well. The Poincaré lemma asserts that in a simply connected domain Ω with a connected boundary $\partial\Omega$, smooth curl-free and divergence-free fields are gradients and curls respectively (see for instance [13]). This lemma reads, with $\mathrm{ran}(f)$ and $\ker(f)$ denoting the range and kernel spaces of f:

$$ \mathrm{ran}(\mathrm{grad}) = \ker(\mathrm{curl}), \qquad \mathrm{ran}(\mathrm{curl}) = \ker(\mathrm{div}). \tag{2.26} $$

This property is often represented by means of the following operator sequence:

$$ C^\infty(\bar{\Omega}) \xrightarrow{\mathrm{grad}} \vec{C}^\infty(\bar{\Omega}) \xrightarrow{\mathrm{curl}} \vec{C}^\infty(\bar{\Omega}) \xrightarrow{\mathrm{div}} C^\infty(\bar{\Omega}). \tag{2.27} $$

A sequence of this form, satisfying (2.26) is called an *exact de Rham sequence*.

It can be proved (see for instance [13]) that the weak differential operators form an exact de Rham sequence as well. On a bounded domain Ω, this sequence

can be extended to include the operator cst, which maps a real number onto the corresponding constant function, i.e.

$$\text{cst} \ : \ \mathbb{R} \mapsto H^1(\Omega) \ : \ r \to f_r \quad \text{with} \quad f_r(\mathbf{x}) = r, \ \forall \mathbf{x} \in \Omega. \tag{2.28}$$

The exact de Rham sequence for the weak differential operators reads:

$$\mathbb{R} \ \xrightarrow{\text{cst}} \ H^1(\Omega) \ \xrightarrow{^w\text{grad}} \ H(\text{curl}, \Omega) \ \xrightarrow{^w\text{curl}} \ H(\text{div}, \Omega) \ \xrightarrow{^w\text{div}} \ L^2(\Omega), \tag{2.29}$$

with

$$\begin{aligned}
\text{cst}(\mathbb{R}) &= \ker(^w\text{grad}) \subset H^1(\Omega), && (2.30)\\
^w\text{grad}(H^1(\Omega)) &= \ker(^w\text{curl}) \subset H(\text{curl}, \Omega), && (2.31)\\
^w\text{curl}(H(\text{curl}, \Omega)) &= \ker(^w\text{div}) \subset H(\text{div}, \Omega), && (2.32)\\
^w\text{div}(H(\text{div}, \Omega)) &= L^2(\Omega). && (2.33)
\end{aligned}$$

The kernels of the gradient, curl and divergence operators induce the following orthogonal splittings of $H^1(\Omega)$, $H(\text{curl}, \Omega)$ and $H(\text{div}, \Omega)$:

$$\begin{aligned}
H^1(\Omega) &= \ker(^w\text{grad}) \oplus \ker^\perp(^w\text{grad}), && (2.34)\\
H(^w\text{curl}, \Omega) &= \ker(^w\text{curl}) \oplus \ker^\perp(^w\text{curl}), && (2.35)\\
H(^w\text{div}, \Omega) &= \ker(^w\text{div}) \oplus \ker^\perp(^w\text{div}). && (2.36)
\end{aligned}$$

The splittings (2.35) and (2.36), which are commonly referred to as Helmholtz splittings, play a fundamental role in discretization schemes and solvers for problems in $H(\text{curl}, \Omega)$ and $H(\text{div}, \Omega)$.

In the rest of this thesis, we will only consider the weak differential operators. To simplify the notation, they will be denoted without the superscript $^w\cdot$.

2.4.3 Function spaces for electromagnetic fields

Since electromagnetic fields satisfy very specific transmission conditions (see section 2.2.3), they are in general not smooth, i.e. they don't belong to $C^1(\Omega)$ or $\vec{C}^1(\Omega)$. The weak differential operators introduced in section 2.4.1 and the exact de Rham sequence introduced in section 2.4.2 allow the functions spaces to be identified to which specific electromagnetic fields belong:

$$\begin{aligned}
V &\in H^1(\Omega),\\
\vec{E}, \vec{H}, \vec{A} &\in H(\text{curl}, \Omega),\\
\vec{D}, \vec{B}, \vec{J} &\in H(\text{div}, \Omega),\\
\rho &\in L^2(\Omega).
\end{aligned}$$

2.4.4 Mathematical problem setting

Next, we will present the mathematical setting in which the equations, derived in sections 2.3.1 and 2.3.2, will be studied. First, we will consider the general three-dimensional setting. Next, we will study the two-dimensional setting of translation-invariant problems. Both the two- and the three-dimensional setting will be used further in this thesis.

General problem setting

We will consider the div-grad, curl-curl and grad-div equations

$$-\operatorname{div}(\alpha \operatorname{grad}(U)) + \beta U = f, \qquad U \in H^1(\Omega),\ f \in L^2(\Omega), \tag{2.37}$$

$$\operatorname{curl}(\alpha \operatorname{curl}(\vec{U})) + \beta \vec{U} = \vec{f}, \qquad \vec{U} \in H(\operatorname{curl}, \Omega),\ \vec{f} \in H(\operatorname{div}, \Omega), \tag{2.38}$$

$$-\operatorname{grad}(\alpha \operatorname{div}(\vec{U})) + \beta \vec{U} = \vec{f}, \qquad \vec{U} \in H(\operatorname{div}, \Omega),\ \vec{f} \in H(\operatorname{curl}, \Omega), \tag{2.39}$$

with $\alpha > 0$ and $\beta \geq 0$. The div-grad and curl-curl equations are obtained directly from (2.12) and (2.14) and from (2.17) and (2.18) after discretization in time. The grad-div equation will be considered as well, for completeness reasons.

The transmission conditions for electromagnetic fields, as described in section 2.2.3, require the weak differential operators (see section 2.4.1) to be used. Consequently, the function spaces to which the solution U or $\vec{U}$ and the excitation f or $\vec{f}$ of (2.37), (2.38) and (2.39) belong, are the domain and range spaces of the weak differential operators.

The linear system resulting from the discretization will be denoted

$$(K + M)x \quad = \quad b, \tag{2.40}$$

with K and M denoting the contribution of the stiffness part and the mass part respectively.

Mathematical problem setting in 2D

Three-dimensional problems give rise to very large linear systems. For translation invariant problems, however, it is often sufficient to consider only a cross-section of the computational domain. In this case, the problem size is reduced significantly.

Consider a model which is translation invariant in the direction of the Z-axis, i.e. the unknown, the excitation, the boundary conditions and the material parameters don't depend on the Z-coordinate. For problems with a vector-valued unknown $\vec{U}$ and excitation $\vec{f}$, i.e. problems in $H(\operatorname{curl}, \Omega)$ and $H(\operatorname{div}, \Omega)$, there are two important types of translation invariance. Either the X- and Y-component or the Z-component of $\vec{U}$ and $\vec{f}$ can be identical to zero, i.e.:

$$\vec{U}^{\perp} = (0, 0, U_z(x, y)) \quad \text{and} \quad \vec{f}^{\perp} = (0, 0, f_z(x, y)), \tag{2.41}$$

$$\vec{U}^{\parallel} = (U_x(x, y), U_y(x, y), 0) \quad \text{and} \quad \vec{f}^{\parallel} = (f_x(x, y), f_y(x, y), 0). \tag{2.42}$$

(2.41) and (2.42) will be referred to as the perpendicular mode and the parallel mode respectively. Typically, the translation invariant eddy current equation (2.17) satisfies (2.41), and the translation invariant curl-curl equation (2.18) for $\vec{E}$ satisfies (2.42).

The specific form of translation invariance introduces relations between the differential operators. For the perpendicular mode (2.41), the curl operator is

equivalent to a gradient operator:

$$\operatorname{curl}(\vec{U}^{\perp}) \;=\; \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ 0 & 0 & U_z \end{vmatrix} \;=\; \operatorname{grad}(\vec{U}^{\perp}\cdot\vec{\epsilon}_z)\times\vec{\epsilon}_z. \tag{2.43}$$

Here, $\vec{\epsilon}_z$ denotes the unit vector in the Z-direction, i.e. $(0,0,1)$. The dot product of a vector $\vec{U}$ with $\vec{\epsilon}_z$ amounts to the selection of the Z-component of $\vec{U}$,

$$\vec{U}\cdot\vec{\epsilon}_z \;=\; U_z, \tag{2.44}$$

and the vector product of $\vec{U}$ with $\vec{\epsilon}_z$ equals:

$$\vec{U}\times\vec{\epsilon}_z \;=\; (U_y,-U_x,0). \tag{2.45}$$

For the parallel mode (2.42), the divergence operator is equivalent to a curl operator:

$$\begin{aligned} \operatorname{curl}(\vec{U}^{\parallel}) \;=\; \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ U_x & U_y & 0 \end{vmatrix} \;&=\; (0,0,\tfrac{\partial U_y}{\partial x}-\tfrac{\partial U_x}{\partial y}) \;=\; \operatorname{div}(\vec{U}^{\parallel}\times\vec{\epsilon}_z)\;\vec{\epsilon}_z \\ \Longleftrightarrow \qquad \operatorname{curl}(\vec{U}^{\parallel}\times\vec{\epsilon}_z) \;&=\; -\operatorname{div}(\vec{U}^{\parallel})\;\vec{\epsilon}_z. \end{aligned} \tag{2.46}$$

Consequently, the curl-curl operator is equivalent to a div-grad operator in the perpendicular mode:

$$\begin{aligned} \operatorname{curl}(\alpha\operatorname{curl}(\vec{U}^{\perp})) \;&=\; \operatorname{curl}\Big(\alpha\operatorname{grad}(\vec{U}^{\perp}\cdot\vec{\epsilon}_z)\times\vec{\epsilon}_z\Big) \\ &=\; -\operatorname{div}\big(\alpha\operatorname{grad}(U^{\perp}\cdot\vec{\epsilon}_z)\big)\;\vec{\epsilon}_z. \end{aligned} \tag{2.47}$$

In the last step, we have taken into account that $\alpha\operatorname{grad}(\vec{U}^{\perp}\cdot\vec{\epsilon}_z)$ satisfies (2.42). This shows that a curl-curl problem in perpendicular mode can be solved as a div-grad problem (see for instance [45, 44]).

Similarly, the following identity shows that a grad-div problem in the parallel mode can be solved as a curl-curl problem:

$$\begin{aligned} \operatorname{curl}(\alpha\operatorname{curl}(\vec{U}^{\parallel})) \;&=\; \operatorname{curl}\Big(\alpha\operatorname{div}(\vec{U}^{\parallel}\times\vec{\epsilon}_z)\;\vec{\epsilon}_z\Big) \\ &=\; \operatorname{grad}\Big(\alpha\operatorname{div}(\vec{U}^{\parallel}\times\vec{\epsilon}_z)\Big)\times\vec{\epsilon}_z. \end{aligned} \tag{2.48}$$

2.5 The discrete setting

The multigrid methods, developed in this thesis, mainly focus on the solution of finite element discretizations of the div-grad, curl-curl and grad-div equations. In this section, we will present a brief review of finite element discretizations using Whitney finite elements. Several concepts which arise in the context of finite element discretizations, such as discrete differential operators and the transfer of the de Rham sequence from the continuous to the discrete level, will have an analogon in the context of the multigrid methods developed in this thesis. For this purpose, these concepts will be studied in some detail.

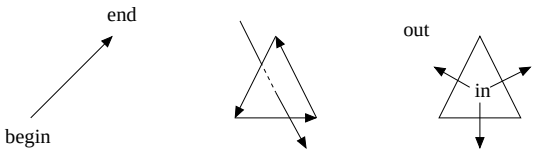


Figure 2.2: Orientation of an edge, a face and its boundary, and a volume and its boundary of a 2D triangular mesh.

Mesh

We will consider discretizations on a bounded and simply connected domain $\Omega \in \mathbb{R}^d,\ d = 2, 3$, with a connected, smooth or polygonal boundary $\partial\Omega$. The discretization will be based on a mesh μ for Ω. A mesh is composed of elements T, which form a partitioning of Ω:

$$\overline{\Omega} = \bigcup_{T \in \mu} \overline{T} \quad \text{and} \quad T_i \cap T_j = \phi \ \text{ if } \ T_i \neq T_j. \tag{2.49}$$

Here $\overline{T}$ denotes the closure of T.

A simplex or n-simplex is the n-dimensional analogon of a triangle. In particular, points, edges, triangles and tetrahedra are simplices. If all elements of a mesh are simplices, the mesh is called *simplicial*.

For each edge, face or volume, an *orientation* needs to be fixed. This orientation can be chosen arbitrarily (see Figure 2.2). The orientation of an edge e depends on which of its end nodes is chosen as start and end node. The orientation of a face f is determined by choosing a crossing direction. The orientation of a volume is determined by choosing what is its interior and exterior.

The *boundary operator* will be denoted ∂. The orientation of the boundary ∂f of a face f, which is a closed loop of edges, corresponds to the positive rotation with respect to this crossing direction according to the 'corkscrew' rule. The orientation of the boundary ∂v of a volume, which is a closed surface of faces, is prescribed by the crossing direction from its interior to its exterior.

A dual mesh $\tilde{\tau}$ for τ is defined as follows (see Figure 2.3):

Definition 2.5.1 *Each node $\tilde{n}_i$, edge $\tilde{e}_i$, face $\tilde{f}_i$ or volume $\tilde{v}_i$ of a dual mesh $\tilde{\tau}$ for τ corresponds to a volume v_i, face f_i, edge e_i or node n_i of τ respectively, such that:*

$$\tilde{n}_i \in v_i, \qquad \tilde{e}_i \cap f_i \neq \phi, \qquad \tilde{f}_i \cap e_i \neq \phi, \qquad n_i \in \tilde{v}_i. \tag{2.50}$$

The orientation of a dual edge $\tilde{e}_i$ corresponds to the crossing direction of f_i. The crossing direction of a dual face $\tilde{f}_i$ corresponds to the direction of e_i.

2.5.1 Shape functions

In the discretization methods considered in this thesis, an approximate solution $\tilde{u}(\mathbf{x})$ of (2.37), (2.38) or (2.39) is constructed as a linear combination of a finite

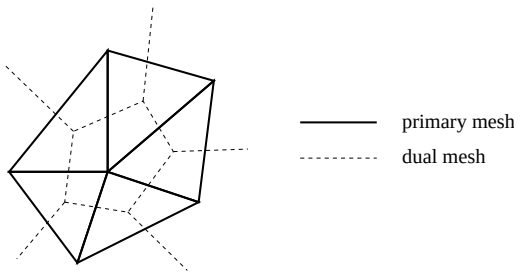


Figure 2.3: 2D example of a dual mesh.

number of shape functions $S_i(\mathbf{x})$, i.e.:

$$\tilde{u}(\mathbf{x}) \quad \simeq \quad \sum_{i=1}^{n} s_i S_i(\mathbf{x}). \tag{2.51}$$

The coefficients s_i in (2.51) are called the *degrees of freedom* (DoF). Let S and $\mathcal{S}$ denote the solution space of the PDE problem and the space spanned by all shape functions respectively:

$$\mathcal{S} \quad = \quad \text{ran}\{S_i(\mathbf{x}) \mid 1 \leq i \leq n\}. \tag{2.52}$$

The discretization is called *compatible* if $\mathcal{S}$ is a subspace of S. Compatibility is important with respect to the stability of the discretization. We refer to [13, 2, 52, 7]) for more details.

Whitney elements of the first kind

We will consider lowest order *Whitney elements of the first kind* (see [12, 54, 52, 13, 37]). These shape functions can be considered generalizations of classical lowest order nodal elements, which are hat functions on simplicial meshes and bilinear nodal elements on quadrilateral or hexahedral meshes. Whitney elements satisfy the following properties:

1. They are *compatible*. The lowest order Whitney elements of the first kind in $H^1(\Omega)$, $H(\text{curl}, \Omega)$, $H(\text{div}, \Omega)$ and $L^2(\Omega)$, which are often called nodal, edge, face and volume elements, will be denoted N_i, $\vec{E}_i$, $\vec{F}_i$ and V_i:

$$\forall i \quad : \quad \begin{cases} N_i & \in & H^1(\Omega), \\ \vec{E}_i & \in & H(\text{curl}, \Omega), \\ \vec{F}_i & \in & H(\text{div}, \Omega), \\ V_i & \in & L^2(\Omega). \end{cases} \tag{2.53}$$

   The spaces spanned by all nodal, edge, face and volume elements will be denoted $\mathcal{S}_N$, $\mathcal{S}_E$, $\mathcal{S}_F$ and $\mathcal{S}_V$ respectively.

2. Each nodal, edge, face or volume element is *associated with one vertex, edge, face or volume* of the mesh respectively. They are *normalized* as follows, with the index of N_i, $\vec{E}_i$, $\vec{F}_i$ or V_i referring to its associated node n_i, edge e_i, face f_i or volume v_i:

$$N_i(n_j) = \int_{e_j} \vec{E}_i \cdot d\vec{l} = \int_{f_j} \vec{F}_i \cdot d\vec{\Omega} = \int_{v_j} V_i \, dV = \begin{cases} 1 & \text{if} \quad i = j, \\ 0 & \text{if} \quad i \neq j. \end{cases} \tag{2.54}$$

Consequently, each degree of freedom of a linear combination (2.51) of nodal, edge, face or volume elements equals the point evaluation, edge integral, face integral or volume integral of the approximate solution $\tilde{u}$ over the corresponding mesh node, edge, face or volume:

$$\tilde{u} = \sum_i \alpha_i N_i \quad \Longrightarrow \quad \tilde{u}(\mathbf{x}_j) = \alpha_j, \tag{2.55}$$

$$\vec{\tilde{u}} = \sum_i \alpha_i \vec{E}_i \quad \Longrightarrow \quad \int_{e_j} \vec{\tilde{u}} \cdot d\vec{l} = \alpha_j, \tag{2.56}$$

$$\vec{\tilde{u}} = \sum_i \alpha_i \vec{F}_i \quad \Longrightarrow \quad \int_{f_j} \vec{\tilde{u}} \cdot d\vec{\Omega} = \alpha_j, \tag{2.57}$$

$$\tilde{u} = \sum_i \alpha_i V_i \quad \Longrightarrow \quad \int_{v_j} \tilde{u} \, dV = \alpha_j. \tag{2.58}$$

3. They are *piecewise polynomial* and have a *local support*. More precisely, they are polynomials within each element of the mesh, and the support of the Whitney element associated with a node, edge, face or volume i is restricted to all elements of the mesh whose closure contains i.

4. The cst (2.28) operator and the weak gradient (2.19), curl (2.20) and divergence (2.21) map $\mathbb{R}$, $\mathcal{S}_N$, $\mathcal{S}_E$ and $\mathcal{S}_F$ on $\mathcal{S}_N$, $\mathcal{S}_E$, $\mathcal{S}_F$ and $\mathcal{S}_V$ respectively:

$$\begin{aligned} \text{cst}(\mathbb{R}) &\subset \mathcal{S}_N, & (2.59) \\ \text{grad}(\mathcal{S}_N) &\subset \mathcal{S}_E, & (2.60) \\ \text{curl}(\mathcal{S}_E) &\subset \mathcal{S}_F, & (2.61) \\ \text{div}(\mathcal{S}_F) &\subset \mathcal{S}_V. & (2.62) \end{aligned}$$

Note that the classical *partition of unity property* for the nodal elements

$$\sum_i N_i(\mathbf{x}) = 1 \tag{2.63}$$

immediately follows from (2.59) and (2.55).

Next, we will study these Whitney elements on the mesh types which will be used further in thesis.

Whitney elements on tetrahedral meshes

On tetrahedral meshes, the lowest order Whitney elements of the first kind N_i in $H^1(\Omega)$ are the classical piecewise linear nodal elements, which are often called *hat functions*. The edge, face and volume elements are derived from these hat functions (see [13, 52, 37]).

The edge element $\vec{E}_{ij}$ for the edge e_{ij} with start node n_i and end node n_j, the face element $\vec{F}_{ijk}$ for the face f_{ijk} with oriented boundary $n_i \rightarrow n_j \rightarrow n_k$ and the volume element $\vec{V}_{ijkl}$ for the tetrahedron v_{ijkl} connecting the nodes n_i, n_j, n_k and n_l are obtained as:

$$\vec{E}_{ij} = N_i \operatorname{grad}(N_j) - N_j \operatorname{grad}(N_i), \tag{2.64}$$

$$\vec{F}_{ijk} = N_i \operatorname{curl}(\vec{E}_{jk}) + N_j \operatorname{curl}(\vec{E}_{ki}) + N_k \operatorname{curl}(\vec{E}_{ij}), \tag{2.65}$$

$$V_{ijkl} = N_i \operatorname{div}(\vec{F}_{jkl}) + N_j \operatorname{div}(\vec{F}_{kli}) + N_k \operatorname{div}(\vec{F}_{lij}) + N_l \operatorname{div}(\vec{F}_{ijk}). \tag{2.66}$$

It can be checked that the volume elements (2.66) are piecewise constant, satisfying:

$$V_{ijkl} = \frac{1}{\operatorname{vol}(v_{ijkl})}.$$

It is obvious that the nodal elements are normalized. The normalization of the edge elements follows from the normalization of the nodal elements and from the partition of unity property. Consider the edge e_{ij} with start node n_i and end node n_j. Since in all points belonging to e_{ij}, the partition of unity property (2.63) reduces to:

$$\forall \mathbf{x} \in e_{ij} \quad : \quad N_i(\mathbf{x}) + N_j(\mathbf{x}) = 1,$$

the line integral of $\vec{E}_{ij}$ along e_{ij} is identical to 1:

$$\begin{aligned} \int_{e_{ij}} \vec{E}_{ij} \cdot d\vec{l} &= \int_{e_{ij}} (N_i \operatorname{grad}(1 - N_i) - (1 - N_i) \operatorname{grad}(N_i)) \cdot d\vec{l} \\ &= -\int_{e_{ij}} \operatorname{grad}(N_i) \cdot d\vec{l} \\ &= N_i(n_i) - N_i(n_j) = 1. \end{aligned}$$

Next, we need to prove that the line integral of $\vec{E}_{ij}$ along any other edge e_{kl} is 0. This immediately follows from the fact that $\vec{E}_{ij}$ is identical to 0 in all points belonging to $e_{kl} \neq \pm e_{ij}$.

The normalization of the face and volume elements can be proved in a similar way. For this purpose, the following vector identities will be needed as well:

$$\begin{aligned} \operatorname{curl}(\vec{E}_{ij}) &= 2 \operatorname{grad}(N_i) \times \operatorname{grad}(N_j), \\ \operatorname{div}(\vec{F}_{ijk}) &= 6 \operatorname{grad}(N_i) \cdot (\operatorname{grad}(N_j) \times \operatorname{grad}(N_k)). \end{aligned}$$

Note that these identities only hold in a weak sense, since we are using the weak differential operators.

Next, we will prove the properties (2.59), (2.60), (2.61) and (2.62). (2.59) immediately follows from the fact that the set of all hat functions constitutes a partition of unity. Relation (2.60) follows from the fact that the gradient of any nodal element can be written as a linear combination of edge elements:

$$\operatorname{grad}(N_j) = \left(\sum_i N_i\right) \operatorname{grad}(N_j) - N_j \operatorname{grad}\left(\sum_i N_i\right) = \sum_i \vec{E}_{ij}. \tag{2.67}$$

Here, we have used subsequently the partition of unity property for the nodal elements and the identity $\operatorname{grad} \circ \operatorname{cst} \equiv 0$ for the weak differential operators. Similarly, (2.61) follows from the fact that the curl of any edge element can be written as a linear combination of face elements:

$$\begin{aligned} \operatorname{curl}(\vec{E}_{jk}) &= \left(\sum_i N_i\right) \operatorname{curl}(\vec{E}_{jk}) + N_j \operatorname{curl}(-\operatorname{grad}(N_k)) + N_k \operatorname{curl}(\operatorname{grad}(N_j)) \\ &= \sum_i \Big(N_i \operatorname{curl}(\vec{E}_{jk}) + N_j \operatorname{curl}(\vec{E}_{ki}) + N_k \operatorname{curl}(\vec{E}_{ij})\Big) \\ &= \sum_i \vec{F}_{ijk}. \end{aligned} \tag{2.68}$$

Here, we have taken into account subsequently the identity $\operatorname{curl} \circ \operatorname{grad} \equiv 0$ for the weak differential operators, the partition of unity property for the nodal elements and (2.67). Finally, (2.62) follows from the fact that the divergence of any face element can be written as

$$\operatorname{div}(\vec{F}_{jkl}) = \sum_i V_{ijkl}, \tag{2.69}$$

which can be proved in a similar way.

Whitney elements on triangular meshes

On triangular meshes, nodal and volume elements are the classical hat functions and piecewise constant functions, similarly as on tetrahedral meshes. For the edge and face elements, a difference must be made for the perpendicular mode (2.41) and the parallel mode (2.42).

First, we consider the shape functions for $H(\operatorname{curl}, \Omega)$. In the parallel mode (2.42), we need to model vectors parallel to the computational domain Ω. In this case, the shape functions can be associated with the edges of the 2D mesh. They are obtained by the same formula (2.64) as on tetrahedral meshes:

$$\vec{E}_{ij}^{\parallel} = N_i \operatorname{grad}(N_j) - N_j \operatorname{grad}(N_i). \tag{2.70}$$

In the perpendicular mode, we need to model vectors perpendicular to the computational domain Ω. Clearly, the shape functions to be used for this purpose

are to be associated with edges which are perpendicular to Ω as well. Since each node of the mesh can be considered a degenerate representation of an edge perpendicular to Ω, the following shape functions can be used:

$$\forall \text{ node } n_i \quad : \quad \vec{E}_i^{\perp} \quad = \quad (0,0,N_i) \quad = \quad N_i \; \vec{\epsilon}_z. \tag{2.71}$$

Since $\vec{E}_i^{\perp}$ is continuous, it clearly belongs to $H(\mathrm{curl},\Omega)$. Note that the normalization (2.54) for edge elements does not apply here, and is replaced by a point-based normalization, similar as for nodal elements:

$$\vec{E}_i^{\perp}(n_j) \quad = \quad \begin{cases} (0,0,1) & \text{if } i=j, \\ (0,0,0) & \text{else.} \end{cases}$$

Next, we consider the shape functions for $H(\mathrm{div},\Omega)$. In the perpendicular mode, the face elements are associated with the triangles of the mesh. They are derived from the edge elements (2.70) by the same formula as on tetrahedral meshes:

$$\vec{F}_{ijk}^{\perp} = N_i \,\mathrm{curl}(\vec{E}_{jk}^{\parallel}) + N_j \,\mathrm{curl}(\vec{E}_{ki}^{\parallel}) + N_k \,\mathrm{curl}(\vec{E}_{ij}^{\parallel}). \tag{2.72}$$

This formula can be simplified as follows. Since the edge elements occurring in (2.72) are piecewise linear and satisfy (2.42), their curl is piecewise constant and satisfies (2.41). This allows the curls figuring in (2.72) to be determined as:

$$\begin{aligned} \int_{f_{ijk}} \mathrm{curl}(\vec{E}_{ij}) \cdot d\vec{\Omega} \quad &= \quad \int_{\partial f_{ijk}} \vec{E}_{ij} \cdot d\vec{l} \quad = \quad 1 \\ \Longrightarrow \qquad & \Big(\mathrm{curl}(\vec{E}_{ij})\Big)_{|f_{ijk}} = (0,0,1/\,\mathrm{area}(f_{ijk})). \end{aligned}$$

Consequently, (2.72) is equivalent to:

$$\vec{F}_{ijk}^{\perp}(\mathbf{x}) \quad = \quad \begin{cases} (0,0,1/\,\mathrm{area}(f_{ijk})) & \text{if } \mathbf{x} \in f_{ijk} \\ (0,0,0) & \text{else.} \end{cases}$$

In other words, the face elements in the perpendicular mode are piecewise constant vector functions. The normalization of $\vec{F}_{ijk}^{\perp}$ is self-evident.

The vectors to be modeled in the parallel mode are to be associated with faces perpendicular to Ω. For this purpose, each edge of the mesh can be regarded as a degenerate representative of such a face. The related shape functions are obtained as, using (2.43):

$$\begin{aligned} \vec{F}_{ij}^{\parallel} \quad &= \quad N_i \,\mathrm{curl}(\vec{E}_j^{\perp}) - N_j \,\mathrm{curl}(\vec{E}_i^{\perp}) \\ &= \quad N_i \,\mathrm{curl}(0,0,N_j) - N_j \,\mathrm{curl}(0,0,N_i) \\ &= \quad N_i \,\mathrm{grad}(N_j) \times \vec{\epsilon}_z - N_j \,\mathrm{grad}(N_i) \times \vec{\epsilon}_z \\ &= \quad \vec{E}_{ij}^{\parallel} \times \vec{\epsilon}_z. \end{aligned} \tag{2.73}$$

The normal continuity of $\vec{F}_{ij}^{\parallel}$ across an interface immediately follows from the tangential continuity of $\vec{E}_{ij}^{\parallel}$ along that same interface. With $\vec{n}_{ij}$ denoting the

oriented unit normal vector to the edge e_{ij}, the normalization of $\vec{F}_{ij}^{\parallel}$ reads:

$$\begin{aligned}\int_{e_{kl}} \vec{F}_{ij}^{\parallel} \cdot d\vec{n}_{kl} &= \int_{e_{kl}} \vec{E}_{ij}^{\parallel} \times \vec{\epsilon}_z \cdot d\vec{n}_{kl} \\ &= \int_{e_{kl}} \vec{E}_{ij}^{\parallel} \cdot d\vec{l} \;=\; \begin{cases} \pm 1 & \text{if } e_{kl} = \pm e_{ij} \\ 0 & \text{else.} \end{cases}\end{aligned}$$

The properties (2.59), (2.60) and (2.61) can be proved in a similar way as for the Whitney elements on tetrahedral meshes.

Figure 2.4 shows an edge and face element on a triangular mesh for the parallel modes. They are only different from 0 in triangles containing the edge to which they are associated, i.e. the edge indicated in bold. Note that they are not continuous at the edges. More precisely, *only* the tangential component of the edge element and the normal component of the face element are continuous, as required by the transmission conditions (2.10) and (2.9) respectively.

Whitney elements on quadrilateral and hexahedral meshes

Next, we will present the formulas of some Whitney elements on quadrilateral and hexahedral meshes, without studying their properties in detail. The lowest order Whitney nodal elements of the first kind correspond to classical bilinear nodal elements. For the unit square, for instance, they read (see Figure 2.5):

$$\begin{aligned} N_0 &= (1-x)(1-y), & N_1 &= x(1-y), \\ N_2 &= (1-x)y, & N_3 &= xy. \end{aligned}$$

The corresponding edge and face elements are (see Figure 2.5):

$$\begin{aligned} \vec{E}_{01} &= (1-y, 0), & \vec{F}_{01} &= (0, 1-y), \\ \vec{E}_{13} &= (0, x), & \vec{F}_{13} &= (x, 0), \\ \vec{E}_{23} &= (y, 0), & \vec{F}_{23} &= (0, y), \\ \vec{E}_{02} &= (0, 1-x), & \vec{F}_{02} &= (1-x, 0). \end{aligned}$$

With $F_{i\overline{j}}$ denoting the set of all faces containing node i and *not* containing node j, the analytical formula for edge elements on quadrilateral and hexahedral meshes reads (see [27]):

$$\vec{E}_{ij} \;=\; N_j \sum_{r \in F_{\overline{ij}}} \operatorname{grad}(N_r) - N_i \sum_{r \in F_{i\overline{j}}} \operatorname{grad}(N_r). \tag{2.74}$$

Other types of Whitney elements

For Whitney elements of the first kind on other element types, we refer to [32, 36]. For higher order Whitney elements, we refer to [52]. For Whitney elements of the second kind, we refer to [55]).

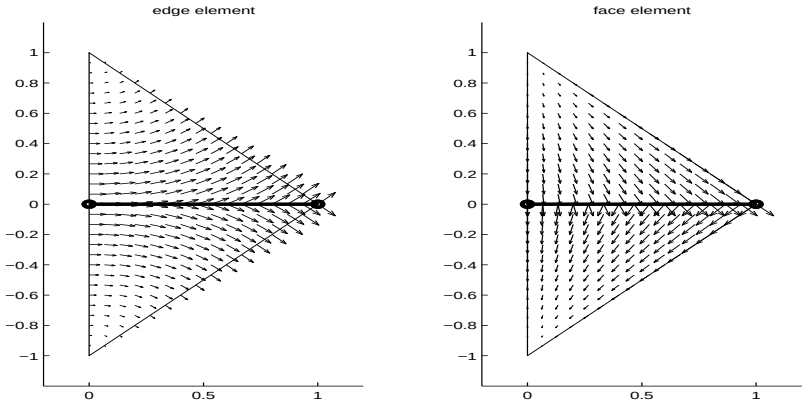


Figure 2.4: Edge and face element on a triangular mesh for the edge indicated in bold.

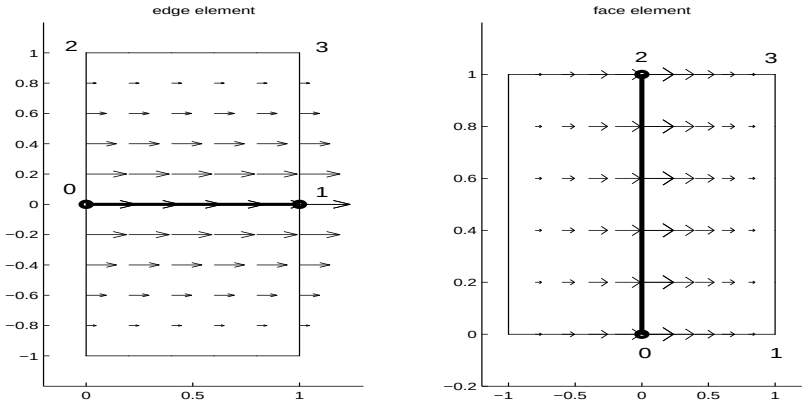


Figure 2.5: Edge and face element on a quadrilateral mesh for the edge indicated in bold.

2.5.2 Discrete differential operators

In this section, we will show that the relations (2.59), (2.60), (2.61) and (2.62) allow the action on the discrete level of the cst operator (2.28) and the weak differential operators gradient (2.19), curl (2.20) and divergence (2.21) to be represented by a transformation matrix.

Let n, e, f and v denote the number of nodes, edges, faces and volumes of the mesh respectively. First, we consider the cst operator. Formula (2.59) asserts that

$$\forall \alpha \in \mathbb{R},\ \exists \beta \in \mathbb{R}^n \quad : \quad \mathrm{cst}(\alpha) = \sum_i \beta_i N_i.$$

Consequently, the entries of the vector β can be determined by using (2.55):

$$\beta_i = \alpha \quad \Longleftrightarrow \quad \beta = \begin{bmatrix} \vdots \\ 1 \\ \vdots \end{bmatrix} \alpha. \tag{2.75}$$

In other words, the discrete representation of the cst operator is the unit vector, which will be denoted $\mathbf{1}$.

The discrete representation of the gradient operator is derived from formula (2.60), which asserts that

$$\forall \alpha \in \mathbb{R}^n,\ \exists \beta \in \mathbb{R}^e \quad : \quad \mathrm{grad}\left(\sum_i \alpha_i N_i\right) = \sum_j \beta_j \vec{E}_j.$$

Formula (2.56) allows the entries of the vector β to be determined as:

$$\beta_j = \int_{e_j} \mathrm{grad}\left(\sum_i \alpha_i N_i\right) \cdot d\vec{l} \quad \Longleftrightarrow \quad \beta = G\alpha,$$

with the matrix $G \in \mathbb{R}^{e \times n}$ defined as follows:

$$\begin{aligned} G(e_j, n_i) &= \int_{e_j} \mathrm{grad}(N_i) \cdot d\vec{l} \;=\; N_i(end(e_j)) - N_i(start(e_j)) \\ &= \begin{cases} 1 & \text{if} \quad n_i = \mathrm{end}(e_j), \\ -1 & \text{if} \quad n_i = \mathrm{start}(e_j), \\ 0 & \text{else.} \end{cases} \end{aligned} \tag{2.76}$$

The matrix G will be called the discrete gradient operator. In an analogous way, a discrete curl operator $C \in \mathbb{R}^{f \times e}$ and a discrete divergence operator $D \in \mathbb{R}^{v \times f}$

can be derived:

$$\begin{aligned} C(f_j, e_i) &= \int_{f_j} \mathrm{curl}(\vec{E}_i) \cdot d\vec{\Omega} = \int_{\partial f_j} \vec{E}_i \cdot d\vec{l} \\ &= \begin{cases} 1 & \text{if} \quad e_i \in \partial f_j, \\ -1 & \text{if} \quad -e_i \in \partial f_j, \\ 0 & \text{else,} \end{cases} \end{aligned} \tag{2.77}$$

$$\begin{aligned} D(v_j, f_i) &= \int_{v_j} \mathrm{div}(\vec{F}_i)\, dV = \int_{\partial v_j} \vec{F}_i \cdot d\vec{\Omega} \\ &= \begin{cases} 1 & \text{if} \quad f_i \in \partial v_j, \\ -1 & \text{if} \quad -f_i \in \partial v_j, \\ 0 & \text{else.} \end{cases} \end{aligned} \tag{2.78}$$

Note that the integrals figuring in the derivation of (2.76), (2.77) and (2.78), exist thanks to the fact that the nodal, edge and face elements are conforming in respectively $H^1(\Omega)$, $H(\mathrm{curl}, \Omega)$ and $H(\mathrm{div}, \Omega)$ (see section 2.4). Also, the formulas (2.76), (2.77) and (2.78) imply the following relations between the nodal, edge, face and volume elements:

$$\mathrm{grad}(N_n) = \sum_{n=end(e)} \vec{E}_e, \tag{2.79}$$

$$\mathrm{curl}(\vec{E}_e) = \sum_{e \in \partial f} \vec{F}_f, \tag{2.80}$$

$$\mathrm{div}(\vec{F}_f) = \sum_{f \in \partial v} V_v. \tag{2.81}$$

These relations correspond to (2.67), (2.68) and (2.69). It can be proved (see [24]) that the discrete differential operators G, C and D for a mesh τ and the discrete differential operators $\tilde{G}$, $\tilde{C}$ and $\tilde{D}$ on a dual mesh $\tilde{\tau}$ for τ are related as follows:

$$\tilde{D} = -G^T, \qquad \tilde{C} = C^T, \qquad \tilde{G} = -D^T. \tag{2.82}$$

2.5.3 Discrete de Rham sequence

In this section, we will show that the range spaces of the Whitney elements and the discrete differential operators are related by an exact de Rham sequence. This is crucial for the stability of a discretization (see [2, 52, 13, 37]). We will study this discrete de Rham structure in detail for the 3D case.

First, we will introduce some additional notation. The vector spaces of the degrees of freedom in (2.51) in $\mathcal{S}_N$, $\mathcal{S}_E$, $\mathcal{S}_F$ and $\mathcal{S}_V$ will be denoted $\mathcal{N}$, $\mathcal{E}$, $\mathcal{F}$

and $\mathcal{V}$. They are mapped onto $\mathcal{S}_N$, $\mathcal{S}_E$, $\mathcal{S}_F$ and $\mathcal{S}_V$ by the following operators:

$$\Pi_N \quad : \quad \mathcal{N} \mapsto \mathcal{S}_N \quad : \quad \alpha \to \tilde{u}(\mathbf{x}) = \sum_i \alpha_i N_i(\mathbf{x}), \tag{2.83}$$

$$\Pi_E \quad : \quad \mathcal{E} \mapsto \mathcal{S}_E \quad : \quad \alpha \to \tilde{u}(\mathbf{x}) = \sum_i \alpha_i \vec{E}_i(\mathbf{x}), \tag{2.84}$$

$$\Pi_F \quad : \quad \mathcal{F} \mapsto \mathcal{S}_F \quad : \quad \alpha \to \tilde{u}(\mathbf{x}) = \sum_i \alpha_i \vec{F}_i(\mathbf{x}), \tag{2.85}$$

$$\Pi_V \quad : \quad \mathcal{V} \mapsto \mathcal{S}_V \quad : \quad \alpha \to \tilde{u}(\mathbf{x}) = \sum_i \alpha_i V_i(\mathbf{x}). \tag{2.86}$$

The action of these operators can be interpreted as a linear interpolation, and they will be referred to as *interpolation operators*. Also, we define the following *sampling operators*, which calculate the point evaluations and line, face and volume integrals of a function for the nodes, edges, faces and volumes of the mesh:

$$\Sigma_N \quad : \quad \mathcal{S}_N \mapsto \mathcal{N} \quad : \quad f \to [\ \dots\ f(\mathbf{x}_i)\ \dots\ ]^T, \tag{2.87}$$

$$\Sigma_E \quad : \quad \mathcal{S}_E \mapsto \mathcal{E} \quad : \quad \vec{f} \to [\ \dots \int_{e_i} \vec{f} \cdot d\vec{l}\ \dots\ ]^T, \tag{2.88}$$

$$\Sigma_F \quad : \quad \mathcal{S}_F \mapsto \mathcal{F} \quad : \quad \vec{f} \to [\ \dots \int_{f_i} \vec{f} \cdot d\vec{\Omega}\ \dots\ ]^T, \tag{2.89}$$

$$\Sigma_V \quad : \quad \mathcal{S}_V \mapsto \mathcal{V} \quad : \quad f \to [\ \dots \int_{v_i} f\, dV\ \dots\ ]^T. \tag{2.90}$$

Taking into account the normalization properties (2.55), (2.56), (2.57) and (2.58) of the Whitney elements, the interpolation and sampling operators satisfy:

$$\forall \alpha \in \mathcal{N} : \Sigma_N\left(\Pi_N(\alpha)\right) = \alpha, \quad \text{and} \quad \forall f \in \mathcal{S}_N : \Pi_N(\Sigma_N(f)) = f, \tag{2.91}$$

$$\forall \alpha \in \mathcal{E} : \Sigma_E\left(\Pi_E(\alpha)\right) = \alpha, \quad \text{and} \quad \forall \vec{f} \in \mathcal{S}_E : \Pi_E(\Sigma_E(\vec{f})) = \vec{f}, \tag{2.92}$$

$$\forall \alpha \in \mathcal{F} : \Sigma_F\left(\Pi_F(\alpha)\right) = \alpha, \quad \text{and} \quad \forall \vec{f} \in \mathcal{S}_F : \Pi_F(\Sigma_F(\vec{f})) = \vec{f}, \tag{2.93}$$

$$\forall \alpha \in \mathcal{V} : \Sigma_V\left(\Pi_V(\alpha)\right) = \alpha, \quad \text{and} \quad \forall f \in \mathcal{S}_V : \Pi_V(\Sigma_V(f)) = f. \tag{2.94}$$

Theorem 2.5.1 *The interpolation operators and the continuous and discrete differential operators satisfy the commutation properties:*

$$\text{cst} \quad = \quad \Pi_N \circ \mathbf{1}, \tag{2.95}$$

$$\text{grad} \circ \Pi_N \quad = \quad \Pi_E \circ G, \tag{2.96}$$

$$\text{curl} \circ \Pi_E \quad = \quad \Pi_F \circ C, \tag{2.97}$$

$$\text{div} \circ \Pi_F \quad = \quad \Pi_V \circ D. \tag{2.98}$$

Proof. (2.95) is self-evident. Using (2.83), (2.88) and (2.76), we have, for

each $\alpha \in \mathcal{N}$:

$$\begin{aligned}(\Sigma_E \circ \operatorname{grad} \circ \Pi_N)\,\alpha &= \Sigma_E \left(\sum_i \alpha_i \operatorname{grad}(N_i) \right) \\ &= \left[\ldots \int_{e_j} \sum_i \alpha_i \operatorname{grad}(N_i) \cdot d\vec{l} \ \ldots \right]^T = G\alpha.\end{aligned}$$

According to (2.60), $\operatorname{grad}(\Pi_N(\alpha))$ belongs to $\mathcal{S}_E$. Consequently, application of (2.92) yields:

$$(\Pi_E \circ \Sigma_E \circ \operatorname{grad} \circ \Pi_N)(\alpha) = (\operatorname{grad} \circ \Pi_N)(\alpha) = (\Pi_E \circ G)(\alpha).$$

This concludes the proof for (2.96). The proof for (2.97) and (2.98) is analogous. □

The relations represented by the continuous and discrete differential operators and by the interpolation operators can be summarized in the following diagram:

$$\begin{array}{ccccccccc} \mathbb{R} & \xrightarrow{\text{cst}} & H^1(\Omega) & \xrightarrow{\text{grad}} & H(\text{curl},\Omega) & \xrightarrow{\text{curl}} & H(\text{div},\Omega) & \xrightarrow{\text{div}} & L^2(\Omega) \\ & & \uparrow \Pi_N & & \uparrow \Pi_E & & \uparrow \Pi_F & & \uparrow \Pi_V \\ \mathbb{R} & \xrightarrow{\mathbf{1}} & \mathcal{N} & \xrightarrow{G} & \mathcal{E} & \xrightarrow{C} & \mathcal{F} & \xrightarrow{D} & \mathcal{V}. \end{array} \tag{2.99}$$

Next, we will prove that the discrete level of (2.99) is an exact de Rham sequence.

Theorem 2.5.2 *On a simply connected domain Ω, the spaces $\mathcal{N}$, $\mathcal{E}$, $\mathcal{F}$ and $\mathcal{V}$ and the discrete differential operators $\mathbf{1}$, G, C and D form an exact de Rham sequence:*

$$\operatorname{ran}(\mathbf{1}) = \ker(G) \subset \mathcal{N}, \tag{2.100}$$
$$\operatorname{ran}(G) = \ker(C) \subset \mathcal{E}, \tag{2.101}$$
$$\operatorname{ran}(C) = \ker(D) \subset \mathcal{F}. \tag{2.102}$$

Proof. We will only prove (2.101). The proof for (2.100) and (2.102) is analogous. Using subsequently the commutation properties (2.96) and (2.97), we have:

$$\begin{aligned}\operatorname{curl} \circ \operatorname{grad} \circ \Pi_N \equiv 0 &\iff \operatorname{curl} \circ \Pi_E \circ G \equiv 0 \\ &\iff \Pi_F \circ C \circ G \equiv 0 \\ &\iff \operatorname{ran}(G) \subset \ker(C).\end{aligned}$$

In order to prove $\ker(C) \subset \operatorname{ran}(G)$, let us assume that there is an $\alpha \notin \operatorname{ran}(G)$ such that $C\alpha = 0$. Consequently,

$$\exists \alpha \notin \operatorname{ran}(G) \quad : \quad \Pi_F(C\alpha) = \operatorname{curl}(\Pi_E(\alpha)) = 0.$$

Since Ω is simply connected, $\Pi_E(\alpha)$ must belong to ran(grad):

$$\exists f \ : \ \Pi_E(\alpha) = \text{grad}(f) \quad \Longrightarrow \quad \exists f \ : \ \alpha = \Sigma_E(\text{grad}(f)).$$

Since the entries of α don't change if f is replaced by $\tilde{f} := \Pi_N(\Sigma_N(f))$:

$$\alpha_{ij} \quad = \quad \int_{e_{ij}} \text{grad}(f) \cdot d\vec{l} \quad = \quad f(\mathbf{x}_j) - f(\mathbf{x}_i) \quad = \quad \int_{e_{ij}} \text{grad}(\tilde{f}) \cdot d\vec{l},$$

we have, using (2.96) and (2.92):

$$\begin{aligned} \alpha = \left(\Sigma_E \circ \text{grad} \circ \Pi_N \circ \Sigma_N\right)(f) \quad &= \quad \left(\Sigma_E \circ \Pi_E \circ G \circ \Sigma_N\right)(f) \\ &= \quad G\left(\Sigma_N(f)\right). \end{aligned}$$

Since this contradicts the assumption, this assumption is incorrect, which concludes the proof.
□

The exact sequence properties (2.100), (2.101) and(2.102) imply the following orthogonal splittings of $\mathcal{N}$, $\mathcal{E}$ and $\mathcal{F}$:

$$\mathcal{N} \quad = \quad \ker(G) \oplus \ker^{\perp}(G) \quad = \quad \text{ran}(\mathbf{1}) \oplus \text{ran}(G^T), \tag{2.103}$$

$$\mathcal{E} \quad = \quad \ker(C) \oplus \ker^{\perp}(C) \quad = \quad \text{ran}(G) \oplus \text{ran}(C^T), \tag{2.104}$$

$$\mathcal{F} \quad = \quad \ker(D) \oplus \ker^{\perp}(D) \quad = \quad \text{ran}(C) \oplus \text{ran}(D^T). \tag{2.105}$$

The splittings (2.104) and (2.105) are the discrete analogons of the Helmholtz splittings (2.35) and (2.36) and will be referred to as the *discrete Helmholtz splittings*. They show that each vector of $\mathcal{E}$ and $\mathcal{F}$ can be written as a sum of two orthogonal components:

$$\begin{aligned} &\forall e \in \mathcal{E}, \ \exists p_n \in \mathcal{N}, \ \exists p_f \in \mathcal{F} \quad : \quad e = Gp_n + C^T p_f, \ (Gp_n)^T (C^T p_f) = 0 \\ &\forall f \in \mathcal{F}, \ \exists p_e \in \mathcal{E}, \ \exists p_v \in \mathcal{V} \quad : \quad f = Cp_e + D^T p_v, \ (Cp_e)^T (D^T p_v) = 0. \end{aligned}$$

These sums are referred to as *Helmholtz decompositions*, and the auxiliary vectors p_n, p_e, p_f and p_v are called *potentials*. Vectors belonging to ran(G) and ran(C^T) will be called gradient and solenoidal vectors respectively, and will be denoted using the subscripts $\cdot_g$ and $\cdot_s$. Knowledge of the Helmholtz splitting (2.104) of $\mathcal{E}$ will prove to be crucial for the understanding of multigrid methods for the curl-curl problem, see chapter 5.

2.5.4 Finite element discretization

The Whitney elements, described in the previous sections, can be used to construct finite element discretizations. In a finite element discretization, the solution of a partial differential equation $L(U) = f$ is approximated as a linear combination of shape functions $u(\mathbf{x}) = \sum_i s_i S_i(\mathbf{x})$ (2.51). The coefficients s_i are calculated by a variational principle, i.e. by requiring the residual to be L^2-orthogonal to the range space $\mathcal{S}$ of the shape functions:

$$f - L(u) \perp \mathcal{S} \quad \Longleftrightarrow \quad \forall i : \int_{\Omega} S_i(\mathbf{x}) \cdot (f - L \sum_j s_j S_j(\mathbf{x})) \, dV = 0. \tag{2.106}$$

Discretization formulae

Since this thesis mainly focuses on the curl-curl equation

$$\operatorname{curl}(\alpha \operatorname{curl}(\vec{U})) + \beta \vec{U} \quad = \quad \vec{f},$$

we will derive the finite element discretization of this equation in detail. For this purpose, edge elements $\vec{E}_i$ will be used. The application of the general finite element discretization formula (2.106) reads in this case, with $\vec{u} := \sum_j c_j \vec{E}_j$ the finite element approximation to the solution $\vec{U}$:

$$\int_\Omega \operatorname{curl}(\alpha \operatorname{curl}(\vec{u})) \cdot \vec{E}_i \, dV + \int_\Omega \beta \, \vec{u} \cdot \vec{E}_i \, dV = \int_\Omega \vec{f} \cdot \vec{E}_i \, dV.$$

The first term, i.e. the stiffness part, can be simplified as follows, by using the linearity of differential and integral operators, the vector identity

$$\operatorname{div}(\vec{x} \times \vec{y}) \quad = \quad \operatorname{curl}(\vec{x}) \cdot \vec{y} - \vec{x} \cdot \operatorname{curl}(\vec{y})$$

and the theorem of Gauss:

$$\begin{aligned}
& \int_\Omega \operatorname{curl}(\alpha \operatorname{curl}(\vec{u})) \cdot \vec{E}_i \, dV \\
= \quad & \int_\Omega \alpha \operatorname{curl}(\vec{u}) \cdot \operatorname{curl}(\vec{E}_i) \, dV + \int_\Omega \operatorname{div}\left(\alpha \operatorname{curl}(\vec{u}) \times \vec{E}_i\right) dV \\
= \quad & \int_\Omega \alpha \operatorname{curl}(\vec{u}) \cdot \operatorname{curl}(\vec{E}_i) \, dV + \int_{\partial\Omega} \alpha \operatorname{curl}(\vec{u}) \times \vec{E}_i \cdot d\vec{\Omega} \\
= \quad & \sum_j c_j \int_\Omega \alpha \operatorname{curl}(\vec{E}_j) \cdot \operatorname{curl}(\vec{E}_i) \, dV + \int_{\partial\Omega} \alpha \operatorname{curl}(\vec{u}) \times \vec{E}_i \cdot d\vec{\Omega}. \quad (2.107)
\end{aligned}$$

Application of the vector identity $\vec{a} \cdot (\vec{b} \times \vec{c}) = \vec{b} \cdot (\vec{c} \times \vec{a})$ shows that the second term of (2.107) vanishes if $\alpha \operatorname{curl}(\vec{u})$ is perpendicular to the boundary $\partial\Omega$. This term comes only into play at Neumann boundaries, i.e. for boundaries where $\alpha \operatorname{curl}(\vec{U})$ is imposed. In matrix notation, the discretization of the curl-curl equation using edge elements can be summarized as follows:

$$\left(K_\alpha^{cc} + M_\beta^e\right) x \quad = \quad b, \qquad (2.108)$$

with x the vector of unknown coefficients c_j, and with

$$K_\alpha^{cc}(i,j) \quad = \quad \int_\Omega \alpha \operatorname{curl}(\vec{E}_i) \cdot \operatorname{curl}(\vec{E}_j) \, dV, \qquad (2.109)$$

$$M_\beta^e(i,j) \quad = \quad \int_\Omega \beta \vec{E}_i \cdot \vec{E}_j \, dV, \qquad (2.110)$$

$$b(i) \quad = \quad \int_\Omega \vec{f} \cdot \vec{E}_i \, dV + \int_{\partial\Omega} \vec{E}_i \times \left(\alpha \operatorname{curl}(\vec{U})\right) \cdot d\vec{\Omega}. \qquad (2.111)$$

The finite element discretization of the div-grad equation

$$-\operatorname{div}(\alpha \operatorname{grad}(U)) + \beta U \quad = \quad f$$

using nodal elements is obtained by a similar derivation, and reads in matrix notation:

$$\left(K_\alpha^{dg} + M_\beta^n\right) x \quad = \quad b, \tag{2.112}$$

with

$$K_\alpha^{dg}(i,j) \quad = \quad \int_\Omega \alpha \,\mathrm{grad}(N_i) \cdot \mathrm{grad}(N_j)\; dV, \tag{2.113}$$

$$M_\beta^n(i,j) \quad = \quad \int_\Omega \beta N_i N_j \, dV, \tag{2.114}$$

$$b(i) \quad = \quad \int_\Omega f N_i \, dV + \int_{\partial\Omega} \alpha N_i \,\mathrm{grad}(U) \cdot d\vec{\Omega}. \tag{2.115}$$

Similarly, the finite element discretization of the grad-div equation

$$-\,\mathrm{grad}(\alpha \,\mathrm{div}(\vec{U})) + \beta\vec{U} \quad = \quad \vec{f}$$

using face elements reads:

$$\left(K_\alpha^{gd} + M_\beta^f\right) x \quad = \quad b, \tag{2.116}$$

with

$$K_\alpha^{gd}(i,j) \quad = \quad \int_\Omega \alpha \,\mathrm{div}(\vec{F}_i) \cdot \mathrm{div}(\vec{F}_j)\; dV, \tag{2.117}$$

$$M_\beta^f(i,j) \quad = \quad \int_\Omega \beta \vec{F}_i \cdot \vec{F}_j \, dV, \tag{2.118}$$

$$b(i) \quad = \quad \int_\Omega \vec{f} \cdot \vec{F}_i \, dV + \int_{\partial\Omega} \alpha \,\mathrm{div}(\vec{U}) \vec{F}_i \cdot d\vec{\Omega}. \tag{2.119}$$

Typically, a finite element discretization matrix is constructed by calculating the contributions of each element T_k of the mesh separately. This amounts to replacing the integration domain Ω by T_k in (2.106). The contribution of an element T_k to the system matrix is often referred to as an *element matrix*, and will be denoted K_{T_k}. For instance, the element matrix for the element T_k for the div-grad equation reads:

$$K_{T_k}(i,j) \quad = \quad \int_{T_k} \alpha \,\mathrm{grad}(N_i) \cdot \mathrm{grad}(N_j) \, dV + \int_{T_k} \beta N_i N_j \, dV. \tag{2.120}$$

Dirichlet boundary conditions

If the values of some unknowns are imposed, these unknowns are said to satisfy a Dirichlet boundary condition. It is a standard practice to first assemble the discretization matrix K *without* taking into account the Dirichlet boundary conditions. In a next step, the Dirichlet boundary conditions are imposed. This

is achieved by replacing the equation for a Dirichlet unknown x_i by a trivial equation of the form:

$$x_i \quad = \quad v_i.$$

If the matrix of the original system $Kx = b$, the symmetry can be restored by eliminating the columns of Dirichlet unknowns. The resulting system reads, with d, $\overline{d}$ and v_d denoting the indices of the Dirichlet and the non-Dirichlet unknowns and the Dirichlet values respectively:

$$\begin{bmatrix} I & 0 \\ 0 & K(\overline{d},\overline{d}) \end{bmatrix} \begin{bmatrix} x(d) \\ x(\overline{d}) \end{bmatrix} \quad = \quad \begin{bmatrix} v_d \\ b(\overline{d}) - K(\overline{d},d)v_d \end{bmatrix}. \tag{2.121}$$

Factorization

Next, we will prove that the stiffness matrices have a specific structure, reflecting the structure of the underlying differential operator. This will prove to be important for the derivation of the multigrid methods developed in chapter 7.

Lemma 2.5.1 *The stiffness matrices (2.113), (2.109) and (2.117) can be written as a Galerkin product of an edge, face or volume mass matrix with the discrete gradient G, discrete curl C or discrete divergence D respectively:*

$$K_\alpha^{dg} \quad = \quad G^T M_\alpha^e G, \tag{2.122}$$
$$K_\alpha^{cc} \quad = \quad C^T M_\alpha^f C, \tag{2.123}$$
$$K_\alpha^{gd} \quad = \quad D^T M_\alpha^v D. \tag{2.124}$$

Proof. Using (2.79) and (2.76), we have for K_α^{dg}:

$$\begin{aligned} K_\alpha^{dg}(i,j) \quad &= \quad \int_\Omega \alpha \operatorname{grad}(N_i) \cdot \operatorname{grad}(N_j)\, dV \\ &= \quad \int_\Omega \alpha \sum_{end(k)=i} \vec{E}_k \cdot \sum_{end(l)=j} \vec{E}_l \, dV \quad = \quad G^T(i,:) M_\alpha^e G(:,j). \end{aligned}$$

The proof for K_α^{cc} and K_α^{gd} uses the properties (2.80)-(2.77) and (2.81)-(2.78) respectively, and is analogous.
□

When applying the definitions of the dual discrete differential operators (2.82), is becomes clear that these factorizations reflect the structure of the underlying differential operator:

$$\begin{aligned} -\operatorname{div}(\alpha \operatorname{grad}(\cdot)) \quad &\simeq \quad K_\alpha^{dg} = -\tilde{D} M_\alpha^e G, \\ \operatorname{curl}(\alpha \operatorname{curl}(\cdot)) \quad &\simeq \quad K_\alpha^{cc} = \tilde{C} M_\alpha^f C, \\ -\operatorname{grad}(\alpha \operatorname{div}(\cdot)) \quad &\simeq \quad K_\alpha^{gd} = -\tilde{G} M_\alpha^v D. \end{aligned}$$

2.6 Conclusion

We presented the main ingredients for finite element discretizations of the div-grad, curl-curl and grad-div equations using Whitney finite elements. We showed that for each of the solution spaces $H^1(\Omega)$, $H(\mathrm{curl}, \Omega)$, $H(\mathrm{div}, \Omega)$ and $L^2(\Omega)$, there is a type of Whitney elements which is compatible with it. Moreover, Whitney elements are normalized, have local support, and transfer important properties, represented by the de Rham sequence, to the discrete level. This makes them the natural choice for the shape functions to be used in a finite element discretization. The test problems for the multigrid methods developed further in this thesis, will be constructed as finite element discretizations using Whitney elements.

Many of the concepts introduced in this chapter, will be reused for the analysis of the multigrid methods developed in chapter 7. In fact, the prolongation operators developed in chapter 7 can be considered the multilevel equivalent of Whitney elements.

Chapter 3

Introduction to multigrid

The discretization of a partial differential equation typically leads to very large sparse systems of equations. The size of these linear systems usually rules out the use of direct linear solvers. Multigrid solvers are very well suited for this type of problems.

In this chapter, the basic multigrid principles will be explained. This chapter can easily be skipped by readers that are familiar with multigrid. This chapter is organized as follows. In section 3.1, matrix-splitting iterations will be introduced. In section 3.2 the multigrid principle and the basic structure of a multigrid solver will be explained. Multigrid solvers are typically analyzed by means of a Fourier analysis. In section 3.3, the principles of local Fourier analysis will be introduced.

3.1 Matrix splitting iterations

Consider a linear system $Kx = b$. Many iterative solvers can be written as an iteration based on a splitting of the system matrix as $K = K^+ + K^-$. With ν denoting the iteration index, the corresponding scheme reads:

$$K^+ x^{[\nu+1]} = b - K^- x^{[\nu]} \quad \Longleftrightarrow \quad x^{[\nu+1]} = x^{[\nu]} + \left(K^+\right)^{-1} (b - K x^{[\nu]}). \tag{3.1}$$

For instance, for Richardson iteration, K^+ equals the unit matrix I. For Jacobi iteration, K^+ equals the diagonal part of K. For Gauss-Seidel iteration, K^+ equals the lower triangular part or the upper triangular part of K. For polynomial smoothers, $(K^+)^{-1}$ equals $q(K)$, with $q(x)$ a polynomial with $q(0) = 1$. For ILU iteration, K^+ equals LU, with the factors L and U chosen in such a way that $K \simeq LU$. The matrices L and U are respectively lower and upper triangular, with a specified or controlled sparsity structure. All these splittings have in common that K^+ can be inverted cheaply (see [58]).

Let $e^{[\nu]}$ denote the error after ν iterations, applied to an initial iterate $x^{[0]}$:

$$e^{[\nu]} \quad := \quad x - x^{[\nu]}. \tag{3.2}$$

The error equation corresponding to (3.1) reads:

$$K^+ e^{[\nu+1]} = -K^- e^{[\nu]}, \quad \text{or} \quad e^{[\nu+1]} = -\left(K^+\right)^{-1} K^- e^{[\nu]}. \tag{3.3}$$

Formula (3.3) shows that the convergence behavior of the iteration (3.1) is determined by the spectrum of the iteration matrix $S := -\left(K^+\right)^{-1} K^-$. In particular, the iteration will converge, for all righthand sides and all starting vectors, if the spectral radius of S, denoted as $\rho(S)$, is smaller than 1. The spectral radius is defined as:

$$\rho(S) \quad := \quad \max\{|\lambda| \ : \ \lambda \in \sigma(S)\}. \tag{3.4}$$

It corresponds to the asymptotic convergence factor:

$$\rho(S) = \lim_{\nu \to \infty} \rho^{[\nu]}, \quad \text{with} \quad \rho^{[\nu]} := \frac{\|b - Kx^{[\nu]}\|}{\|b - Kx^{[\nu-1]}\|}.$$

That is the factor by which the norm of the error and residual is multiplied from one iteration to the next, asymptotically. A more practical measure is the geometric mean convergence factor:

$$\bar{\rho} \quad := \quad \sqrt[\nu]{\frac{\|b - Kx^{[\nu]}\|}{\|b - Kx^{[0]}\|}} \quad = \quad \sqrt[\nu]{\Pi_{i=1}^{\nu} \rho^{[i]}}. \tag{3.5}$$

3.2 The multigrid principle

The original target problem class for multigrid solvers are discretizations of elliptic PDEs. The prototype of an elliptic PDE is the Poisson equation

$$-\operatorname{div}(\operatorname{grad}(u)) \quad = \quad b. \tag{3.6}$$

Let the linear system $Kx = b$, with $K \in \mathbb{R}^{n \times n}$, be the discretization of an elliptic PDE on a certain grid or mesh. In general, iterative solvers based on a simple matrix splitting, such as Jacobi or Gauss-Seidel iterations, are not efficient for this problem, see Table 3.1. However, these simple iterative schemes typically reduce the error in a very specific way, which can be exploited to construct an efficient solver. More precisely, they quickly reduce the *high-frequency* components in the error, while the *low-frequency* error components remain essentially untouched (see Figure 3.1). Although the magnitude of the error remains large, the error after a few iterations typically varies smoothly. For this reason, these simple iterative schemes are called *smoothers*.

In order to obtain an efficient solver, an additional solver component is required, which is able to reduce the low-frequency error components efficiently. Since at this point, only the low-frequency behavior is important, it is possible to resort to a linear system that can be considered an approximation of $Kx = b$ on a *coarser* mesh or grid. Thanks to its smaller size, this coarse system will

iteration ν	$\|b - Kx^{[\nu]}\|/\|b\|$	$\rho^{[\nu]}$
1	0.91719	0.91719
2	0.89143	0.97192
3	0.87938	0.98648
4	0.87186	0.99145
5	0.86647	0.99381
6	0.86221	0.99508
7	0.85861	0.99583
8	0.85544	0.99631
9	0.85256	0.99664
10	0.84990	0.99688

Table 3.1: Convergence history of Gauss-Seidel iterations for the finite difference discretization of the 1D Poisson equation with 100 unknowns. The righthand side b is a random vector and the initial guess for the solution is equal to 0.

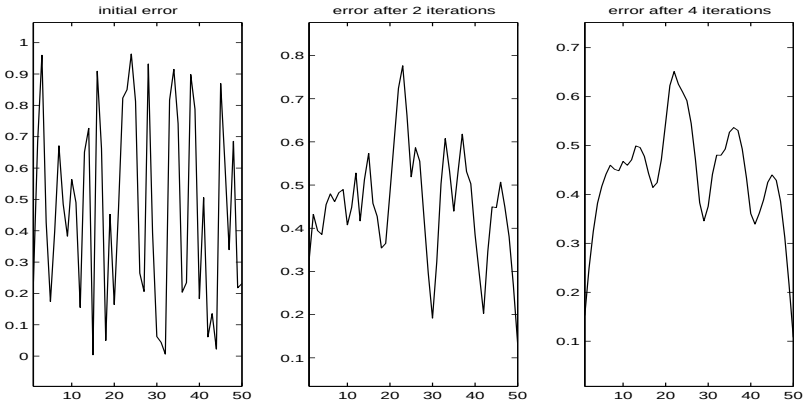


Figure 3.1: Reduction of the error of the finite difference discretization of the 1D Poisson equation with 50 unknowns by Gauss-Seidel iterations. A random vector is chosen as the initial guess for the solution.

be cheaper to solve. For this purpose, multigrid methods typically use a coarse approximation of the error equation of $Kx = b$:

$$K_c x_c = b_c \quad \approx \quad Ke = r. \tag{3.7}$$

Here, r and e denote the residual $b - K\tilde{x}$ and the error $x - \tilde{x}$, with $\tilde{x}$ the approximation of the solution x obtained after smoothing. The subscript $\cdot_c$ refers to the coarse grid. The coarse operator $K_c \in \mathbb{R}^{n_c \times n_c}$, with $n_c < n$, should be a coarse representation of K. The coarse representation b_c of the residual r is built by multiplying r with a *restriction* matrix $R \in \mathbb{R}^{n_c \times n}$:

$$b_c \quad = \quad Rr. \tag{3.8}$$

This typically amounts to a weighed redistribution of the entries of r over the vector b_c.

If b_c and K_c are suitable coarse representations of r and K, then the solution x_c of the coarse system can be expected to be a good representation of the fine grid error as well. This will allow an approximation of e to be constructed from x_c. This is done by multiplying x_c with a *prolongation* matrix $P \in \mathbb{R}^{n \times n_c}$, which typically amounts to an interpolation procedure. This allows the approximate solution $\tilde{x}$ to be updated as follows:

$$\left.\begin{array}{rcl} x & = & \tilde{x} + e \\ e & \simeq & Px_c \end{array}\right\} \quad \Longrightarrow \quad x \simeq \tilde{x} + Px_c. \tag{3.9}$$

Typically, some high-frequency error components are reinjected by the coarse grid correction (3.9), due to the interpolatory character of the prolongation operation. These high-frequency error components can be eliminated by applying the smoother again.

The above procedure is the so-called *two-level cycle*. It can be summarized as follows:

1. μ_{pre} presmoothing steps on $Kx = b$,
2. restriction of the residual: $b_c = R(b - Kx)$,
3. (approximate) coarse solve: $x_c \simeq K_c^{-1} b_c$,
4. prolongation of the correction: $x \leftarrow x + Px_c$,
5. μ_{post} postsmoothing steps on $Kx = b$.

The error equation of this two-level cycle reads, with S_{pre} and S_{post} the iteration matrices of the pre- and postsmoother:

$$e^{[\nu+1]} \quad = \quad S_{pre}^{\mu_{pre}} \left(I - PK_c^{-1}RK\right) S_{post}^{\mu_{post}} e^{[\nu]}. \tag{3.10}$$

A *multigrid cycle* is obtained by applying the two-level cycle recursively. More precisely, the approximate solution x_c of the coarse system in step 3 is obtained by applying one or more multigrid cycles to the coarse system.

The multigrid cycle can be used as a standalone iterative solver or as a preconditioner for a Krylov solver (see for instance [58]). The latter practice is usually more efficient and robust.

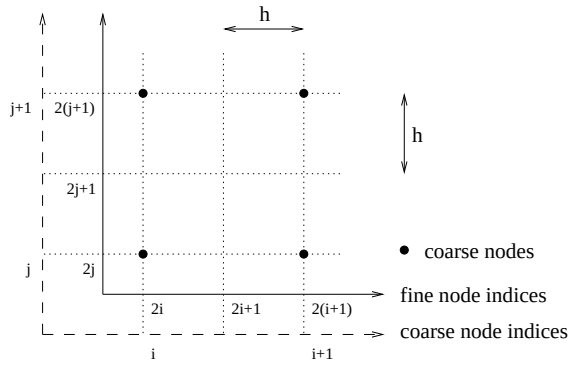


Figure 3.2: Fine and coarse quadrilateral grids. The coarse nodes are the fine nodes with even indices.

3.3 Local Fourier analysis for multigrid

The convergence of iterative methods for linear systems that arise from the discretization of a partial differential equation (PDE) can often be studied in a model problem setting by means of Fourier modes. In a so-called *local Fourier analysis (LFA)* or local mode analysis, an infinite grid is considered and boundary conditions are not taken into account. This type of analysis was introduced in [14]. It has been applied successfully to iterative solvers for several types of PDEs, such as the diffusion equation, the convection-diffusion equation, the Helmholtz equation, the biharmonic equation, the Stokes equations, the Oseen equations and elasticity. For comprehensive surveys, we refer the reader to [63, 69, 68].

Further on, in chapter 6, we will apply LFA to the curl-curl equation. Here, we would like to explain the basic methodology of LFA, and introduce some essential terminology and notations. As an example, multigrid for the 2D diffusion equation with constant material parameters α and β

$$-\operatorname{div}(\alpha \operatorname{grad}(V)) + \beta V \quad = \quad \rho, \qquad \alpha \neq 0, \tag{3.11}$$

will be analyzed using LFA. With an appropriate scaling, only the case $\alpha = 1$ needs to be studied.

3.3.1 Definitions and notations

We consider an infinite, two-dimensional regular quadrilateral grid $\mathcal{G}_h$ with grid size h (see Figure 3.2):

$$\mathcal{G}_h \quad := \quad \{\mathbf{x} = h\mathbf{k} \mid \mathbf{k} \in \mathbb{Z}^2\}. \tag{3.12}$$

A node $\mathbf{x} = (k_1, k_2)h \in \mathcal{G}_h$ will be referred to as (k_1, k_2). The set of the indices of all nodes will be denoted N and the vector space of the degrees of freedom

related to the nodes of N will be denoted $\mathcal{N}$. We define the following evaluation operator in a point $\mathbf{k} = (k_1, k_2)$:

$$\cdot_{\mathbf{k}} : f \mapsto f_{\mathbf{k}} := f(hk_1, hk_2). \tag{3.13}$$

The vector of the evaluations of a function f in the nodes of N will be denoted f_N. This allows a vector belonging to $\mathcal{N}$ to be represented by means of a function f as f_N. Such vectors will be referred to as *grid functions*. The evaluation operator $\cdot_{\mathbf{k}}$ will be applied to grid functions and vectors as well. In this case, it selects the entry corresponding to $\mathbf{k}$. For example, $(f_N)_{\mathbf{k}} = f_{\mathbf{k}}$, if $\mathbf{k} \in N$.

We consider discrete node-based operators L characterized by a translation-invariant *difference stencil* with constant coefficients. The application of L to a grid function w_N is defined as:

$$(Lw_N)_{\mathbf{k}} \quad := \quad \sum_{\mathbf{n} \in N} s_{\mathbf{n}} w((\mathbf{k}+\mathbf{n})h), \quad \forall \mathbf{k} \in N. \tag{3.14}$$

The coefficients of (3.14) can be represented by means of a *stencil* $\mathcal{ST}_L$ as:

$$\mathcal{ST}_L \quad := \quad \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & s_{-1,1} & s_{0,1} & s_{1,1} & \cdots \\ \cdots & s_{-1,0} & s_{0,0} & s_{1,0} & \cdots \\ \cdots & s_{-1,-1} & s_{0,-1} & s_{1,-1} & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix}. \tag{3.15}$$

The stencils for the discretization of the 2D diffusion equation (3.11) read, using finite differences:

$$\frac{1}{h^2} \begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix} + \beta \begin{bmatrix} & 0 & \\ 0 & 1 & 0 \\ & 0 & \end{bmatrix}, \tag{3.16}$$

and using bilinear finite elements:

$$\frac{1}{3h^2} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} + \frac{\beta}{36} \begin{bmatrix} 1 & 4 & 1 \\ 4 & 16 & 4 \\ 1 & 4 & 1 \end{bmatrix}. \tag{3.17}$$

In order to simplify notations, the following abbreviations will be used in the rest of this thesis:

$$\begin{array}{llll} s_i = \sin(\frac{\theta_i}{2}), & \tilde{s}_i = \sin(\theta_i), & s_{i,a} = \sin(\frac{\theta_i + a\pi}{2}), & \tilde{s}_{i,a} = \sin(\theta_i + a\pi), \\ c_i = \cos(\frac{\theta_i}{2}), & \tilde{c}_i = \cos(\theta_i), & c_{i,a} = \cos(\frac{\theta_i + a\pi}{2}), & \tilde{c}_{i,a} = \cos(\theta_i + a\pi). \end{array}$$

3.3.2 Fourier modes

The properties of an operator L of the form (3.14) can be studied for each frequency $\theta = (\theta_1, \theta_2)$ separately. This is because the grid functions corresponding

to the Fourier modes

$$\varphi(\theta) := e^{i\theta\cdot\mathbf{x}/h} := e^{i\frac{\theta_1 x_1+\theta_2 x_2}{h}} \qquad \text{with } \theta, \mathbf{x} \in \mathbb{R}^2 \tag{3.18}$$

are the eigenfunctions of L. Let $F_n(\theta)$ denote the space spanned by $\varphi(\theta)_N$:

$$F_n(\theta) \quad := \quad \text{ran}\{\varphi(\theta)_N\}. \tag{3.19}$$

For future use, we formulate this classical result in a theorem.

Theorem 3.3.1 *[Invariant subspace] For all $\theta \in \mathbb{R}^2$, $F_n(\theta)$ is invariant for a discrete operator L of the form (3.14).*

Proof. The application of L to $\varphi(\theta)$ yields in $\mathbf{k}$:

$$(L\varphi(\theta)_N)_{\mathbf{k}} = \varphi(\theta)_{\mathbf{k}} \sum_{\mathbf{n}\in N} s_{\mathbf{n}} e^{i\theta\cdot\mathbf{n}}.$$

Hence, the global result reads:

$$L\varphi(\theta)_N = \varphi(\theta)_N L(\theta), \quad \text{with} \quad L(\theta) = \sum_{\mathbf{n}\in N} s_{\mathbf{n}} e^{i\theta\cdot\mathbf{n}}. \tag{3.20}$$

□

Formula (3.20) can be used to represent the operator L in $F_n(\theta)$ by a single number $L(\theta) \in \mathbb{C}$. For instance, the frequency domain representation of the Poisson operator corresponding to the stencil (3.16) reads:

$$\begin{aligned} L(\theta) &= \frac{1}{h^2}\left(4 - e^{i\theta_1} - e^{-i\theta_1} - e^{i\theta_2} - e^{-i\theta_2}\right) + \beta \\ &= \frac{1}{h^2}(4 - 2\tilde{c}_1 - 2\tilde{c}_2) + \beta \quad = \quad \frac{4}{h^2}(s_1^2 + s_2^2) + \beta. \end{aligned} \tag{3.21}$$

The discrete frequency domain representation of L is not unique for all frequencies. This classical aliasing phenomenon is summarized in the following theorem.

Theorem 3.3.2 *[Frequency aliasing] All spaces $F_n(\theta_1 + 2\pi a, \theta_2 + 2\pi b)$ with $a, b \in \mathbb{Z}$, are identical, for any $\theta_1, \theta_2 \in \mathbb{R}$.*

Proof. It is sufficient to note that the basis vectors for $F_n(\theta_1 + 2\pi a, \theta_2 + 2\pi b)$ and $F_n(\theta_1, \theta_2)$ are identical, which immediately follows from:

$$\forall k_1, k_2 \in \mathbb{Z} \quad : \quad e^{i((\theta_1+2\pi a)k_1 + (\theta_2+2\pi b)k_2)} = e^{i(\theta_1 k_1 + \theta_2 k_2)}.$$

□

Theorem 3.3.2 shows that it is sufficient to consider any frequency interval of length 2π for θ_1 and θ_2. We will study the two-dimensional frequency interval

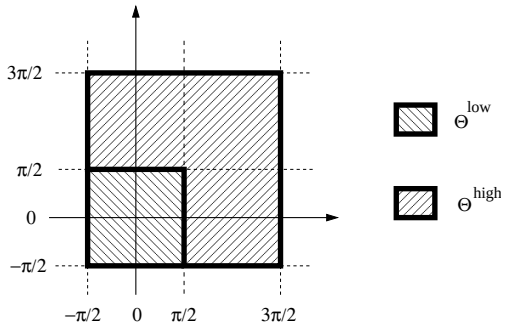


Figure 3.3: The low- and high-frequency regions in the two-dimensional frequency interval considered in the local Fourier analysis.

$\Theta := [-\pi/2, 3\pi/2)^2$. Θ is different from the frequency interval $[-\pi, \pi)^2$ which is usually considered in a LFA (see for instance [63]). This choice is motivated by an important simplification of the LFA for edge-based problems in chapter 6. The low and high frequency interval will be denoted (see Figure 3.3)

$$\Theta^{\text{low}} := [-\pi/2, \pi/2)^2 \qquad \Theta^{\text{high}} := \Theta \setminus \Theta^{\text{low}}.$$

We define the harmonics of a low frequency $\theta = (\theta_1, \theta_2) \in \Theta^{\text{low}}$ as:

$$\theta^{(a,b)} = (\theta_1 + a\pi, \theta_2 + b\pi), \qquad a, b \in \{0, 1\}. \tag{3.22}$$

The following classical lemma is the key to the study of the intergrid transfer operators.

Lemma 3.3.1 *A grid function satisfying*

$$\psi(\theta)_{\mathbf{k}} \quad = \quad \begin{cases} \alpha_{00}\varphi(\theta)_{\mathbf{k}}, \ \alpha_{00} \in \mathbb{C} & \text{if } k_1 \text{ and } k_2 \text{ even}, \\ \alpha_{10}\varphi(\theta)_{\mathbf{k}}, \ \alpha_{10} \in \mathbb{C} & \text{if } k_1 \text{ odd and } k_2 \text{ even}, \\ \alpha_{01}\varphi(\theta)_{\mathbf{k}}, \ \alpha_{01} \in \mathbb{C} & \text{if } k_1 \text{ even and } k_2 \text{ odd}, \\ \alpha_{11}\varphi(\theta)_{\mathbf{k}}, \ \alpha_{11} \in \mathbb{C} & \text{if } k_1 \text{ and } k_2 \text{ odd}. \end{cases}$$

can be written as a linear combination of the grid functions $\varphi(\theta)_N$ for the harmonic frequencies (3.22) of θ:

$$\psi(\theta)_N \quad = \quad \sum_{a,b \in \{0,1\}} \beta^{(a,b)} \varphi(\theta^{(a,b)})_N, \quad \beta^{(a,b)} \in \mathbb{C}. \tag{3.23}$$

Proof. The grid functions $\varphi(\theta)_N$ for the harmonic frequencies of θ are related as:

$$\varphi(\theta^{(a,b)})_{\mathbf{k}} \quad = \quad \varphi(\theta^{(0,0)})_{\mathbf{k}} (-1)^{ak_1 + bk_2}. \tag{3.24}$$

It suffices to note that (3.24) and (3.23) give rise to the following regular linear system:

$$\begin{bmatrix} \alpha_{00} \\ \alpha_{10} \\ \alpha_{01} \\ \alpha_{11} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} \beta^{(0,0)} \\ \beta^{(1,0)} \\ \beta^{(0,1)} \\ \beta^{(1,1)} \end{bmatrix}. \tag{3.25}$$

□

3.3.3 Smoother

Consider a smoother based on a matrix splitting iteration (3.1). Let the iteration matrix be denoted $S = -(K^+)^{-1} K^-$. If the splitting of K corresponds to a splitting of the stencil $\mathcal{ST}_K$ for K, i.e.:

$$\mathcal{ST}_K = \mathcal{ST}_K^+ + \mathcal{ST}_K^- \quad \text{with} \quad \begin{cases} \mathcal{ST}_K^+ = \mathcal{ST}_{K^+} \\ \mathcal{ST}_K^- = \mathcal{ST}_{K^-}, \end{cases} \tag{3.26}$$

then theorem 3.3.1 holds for K^+ and K^- as well. This implies that the grid functions $\varphi(\theta)_N$ are eigenvectors of S. The corresponding eigenvalues are calculated as:

$$S(\theta) = -\frac{K^-(\theta)}{K^+(\theta)}. \tag{3.27}$$

This allows the corresponding error equation for each frequency to be studied separately.

This can be done for most popular multigrid smoothers. Here, we consider the example of damped Jacobi and Gauss-Seidel iterations applied to the finite difference discretization (3.16) of the diffusion equation with $\beta = 0$, i.e. the Poisson equation. The stencil splitting for damped Jacobi iterations reads, with d denoting the damping factor:

$$\frac{1}{h^2} \begin{bmatrix} & 0 & \\ 0 & 4d & 0 \\ & 0 & \end{bmatrix} e^{[\nu+1]} = \frac{1}{h^2} \begin{bmatrix} & 1 & \\ 1 & 4(d-1) & 1 \\ & 1 & \end{bmatrix} e^{[\nu]}. \tag{3.28}$$

This corresponds to the following Fourier domain representation:

$$S(\theta) = 1 - \frac{s_1^2 + s_2^2}{d}. \tag{3.29}$$

The matrix splitting for lexicographic Gauss-Seidel iterations, which reads:

$$\frac{1}{h^2} \begin{bmatrix} & 0 & \\ -1 & 4 & 0 \\ & -1 & \end{bmatrix} e^{[\nu+1]} = \frac{1}{h^2} \begin{bmatrix} & 1 & \\ 0 & 0 & 1 \\ & 0 & \end{bmatrix} e^{[\nu]}, \tag{3.30}$$

corresponds to the following Fourier domain representation:

$$S(\theta) = \frac{e^{i\theta_1} + e^{i\theta_2}}{4 - e^{-i\theta_1} - e^{-i\theta_2}}. \tag{3.31}$$

3.3.4 Intergrid transfer operators

Coarse grid

We consider the coarse grid with mesh size $\tilde{h} = 2h$, depicted in Figure 3.2:

$$G_{\tilde{h}} \quad := \quad \{\mathbf{x} = \mathbf{k}\tilde{h} \mid \mathbf{k} \in \mathbb{Z}^2\}. \tag{3.32}$$

A superscript $\tilde{}$ will be used to refer to the coarse grid. To distinguish the evaluation operators (3.13) with respect to the fine and coarse grid, the grid size will be indicated in superscript as:

$$\cdot^h_{\mathbf{k}} : f \mapsto f^h_{\mathbf{k}} := f(hk_1, hk_2). \tag{3.33}$$

Similarly as (3.18), we define a coarse Fourier mode as:

$$\tilde{\varphi}(\theta) := e^{i\theta\cdot\mathbf{x}/\tilde{h}} \qquad \text{with } \theta \in \mathbb{R}^2. \tag{3.34}$$

and $\tilde{F}_n(\theta)$ as the space spanned by $\tilde{\varphi}(\theta)^{\tilde{h}}_N$:

$$\tilde{F}_n(\theta) \quad := \quad \text{ran}\{\tilde{\varphi}(\theta)^{\tilde{h}}_N\}. \tag{3.35}$$

The fine and coarse evaluation operators and the fine and coarse Fourier modes (3.18) and (3.34) are related as follows, with $\mathbf{r} \in \mathbb{R}^2$:

$$\varphi(\theta)^h_{\mathbf{r}} = \tilde{\varphi}(2\theta)^h_{\mathbf{r}} = \tilde{\varphi}(\theta)^{\tilde{h}}_{\mathbf{r}} = \tilde{\varphi}(\theta)^h_{2\mathbf{r}}. \tag{3.36}$$

Note that the evaluations on the coarse grid of the Fourier modes (3.18) for the harmonics of a frequency θ are identical:

$$\begin{aligned} \varphi(\theta^{(a,b)})^{\tilde{h}}_{\mathbf{k}} &= e^{i\left((\theta_1+a\pi,\theta_2+b\pi)\cdot(k_1,k_2)\tilde{h}/h\right)} \\ &= \varphi(\theta^{(0,0)})^{\tilde{h}}_{\mathbf{k}} e^{2\pi i(ak_1+bk_2)} \\ &= \varphi(\theta^{(0,0)})^{\tilde{h}}_{\mathbf{k}} \quad = \quad \tilde{\varphi}(2\theta^{(0,0)})^{\tilde{h}}_{\mathbf{k}}. \end{aligned} \tag{3.37}$$

Prolongation

We consider the prolongation operator based on bilinear interpolation (see [63]), which can be represented in stencil notation as:

$$\begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix}. \tag{3.38}$$

This yields for the coarse Fourier mode for the frequency $\theta \in \Theta^{\text{low}}$:

$$\left(P\tilde{\varphi}(2\theta)^{\tilde{h}}_N\right)^h_{\mathbf{k}} \quad = \quad \varphi(\theta)^h_{\mathbf{k}} \cdot \begin{cases} 1 & \text{for} \quad k_1, k_2 \text{ even}, \\ \tilde{c}_1 & \text{for} \quad k_1 \text{ odd}, k_2 \text{ even}, \\ \tilde{c}_2 & \text{for} \quad k_1 \text{ even}, k_2 \text{ odd}, \\ \tilde{c}_1\tilde{c}_2 & \text{for} \quad k_1, k_2 \text{ odd}. \end{cases} \tag{3.39}$$

The application of lemma 3.3.1 to (3.39) shows that the prolongation operation for the harmonics of a frequency $\theta \in \Theta^{\text{low}}$ can be studied as a mapping from $\tilde{F}(2\theta^{(0,0)})$ onto $\bigcup_{a,b\in\{0,1\}} F_n(\theta^{(a,b)})$, by means of the following 4×1 matrix:

$$\mathcal{P}(\theta) := \begin{bmatrix} P(\theta^{(0,0)}) \\ P(\theta^{(1,0)}) \\ P(\theta^{(0,1)}) \\ P(\theta^{(1,1)}) \end{bmatrix} \quad \text{with} \quad P(\theta^{(a,b)}) := \frac{(1+\tilde{c}_{1,a})(1+\tilde{c}_{2,b})}{4}. \tag{3.40}$$

Restriction

Corresponding to the AMG practice, we consider the prolongator's transpose as the restriction operator. This yields for the Fourier mode for the frequency $\theta^{(a,b)}$, after some trigonometry and using (3.36) and (3.37):

$$\begin{aligned} \left(R\varphi(\theta^{(a,b)})_N\right)^{\tilde{h}}_{\mathbf{k}} &= \varphi(\theta^{(a,b)})^h_{2\mathbf{k}} \cdot (1 + \tilde{c}_{1,a} + \tilde{c}_{2,b} + \tilde{c}_{1,a}\tilde{c}_{2,b}) \\ &= \tilde{\varphi}(2\theta^{(0,0)})^{\tilde{h}}_{\mathbf{k}} \cdot (1+\tilde{c}_{1,a})(1+\tilde{c}_{2,b}). \end{aligned} \tag{3.41}$$

Formula (3.41) shows that the restriction operation maps the fine spaces $F_n(\theta^{(a,b)})$ for each $a, b \in \{0,1\}$ onto the same coarse space $\tilde{F}(2\theta^{(0,0)})$. This shows that the restriction operation for the harmonics of a frequency $\theta \in \Theta^{\text{low}}$ can be studied by means of the following 1×4 matrix:

$$\mathcal{R}(\theta) = \begin{bmatrix} R(\theta^{(0,0)}) & R(\theta^{(1,0)}) & R(\theta^{(0,1)}) & R(\theta^{(1,1)}) \end{bmatrix}, \tag{3.42}$$

with

$$R(\theta^{(a,b)}) := (1+\tilde{c}_{1,a})(1+\tilde{c}_{2,b}). \tag{3.43}$$

Note that $R(\theta)$ and $P(\theta)$ are related as follows:

$$\mathcal{R}(\theta) = 4\mathcal{P}(\theta)^T. \tag{3.44}$$

3.3.5 Multigrid cycle

We consider a standard two-level multigrid cycle (see section 3.2) in which the coarse system is solved exactly. Corresponding to the AMG practice, the coarse system matrix K_c is built as the Galerkin product RKP. Let $\mathcal{K}(\theta)$, $\mathcal{S}_{pre}(\theta)$ and $\mathcal{S}_{post}(\theta)$ be the representations of the fine level operator and the iteration matrices of the pre- and postsmoother in $\bigcup_{a,b\in\{0,1\}} F_n(\theta^{(a,b)})$, which read, for instance for $\mathcal{K}(\theta)$:

$$\mathcal{K}(\theta) = \begin{bmatrix} K(\theta^{(0,0)}) & 0 & 0 & 0 \\ 0 & K(\theta^{(1,0)}) & 0 & 0 \\ 0 & 0 & K(\theta^{(0,1)}) & 0 \\ 0 & 0 & 0 & K(\theta^{(1,1)}) \end{bmatrix}. \tag{3.45}$$

	Numerical				LFA
$\beta_h \setminus \bar{\rho}$	25×25	51×51	75×75	101×101	
0.00	0.1698	0.1714	0.1708	0.1716	0.1724
0.25	0.1413	0.1429	0.1419	0.1433	0.1439
0.50	0.1174	0.1187	0.1188	0.1188	0.1198
0.75	0.0972	0.0988	0.0990	0.0988	0.0994
1.00	0.0802	0.0815	0.0819	0.0817	0.0823

Table 3.2: Comparison of the asymptotic convergence factor of a $V(1,1)$-cycle using GS-smoothing for the finite element discretization (3.17) of the diffusion equation (3.11) as a function of $\beta_h := \beta h^2$ and of the grid size. The numerical results are obtained as the convergence factor after 100 cycles, with a random initial guess and a zero righthand side.

The Fourier domain representation of the coarse level operator is constructed as:

$$\mathcal{K}_c(\theta) \quad := \quad \mathcal{R}(\theta)\mathcal{K}(\theta)\mathcal{P}(\theta). \tag{3.46}$$

This allows to study the multigrid cycle for the harmonics of a frequency $\theta \in \Theta^{\text{low}}$, by means of the following 4×4 matrix:

$$\mathcal{C}(\theta) \quad = \quad \mathcal{S}_{post}(\theta)^{\mu_{post}} \left(I_4 - \mathcal{P}(\theta)\mathcal{K}_c(\theta)^{-1}\mathcal{R}(\theta)\mathcal{K}(\theta)\right) \mathcal{S}_{pre}(\theta)^{\mu_{pre}}. \tag{3.47}$$

The asymptotic convergence factor can be calculated as:

$$\rho \quad = \quad \max_{\theta \in \Theta^{\text{low}}} \sigma(\mathcal{C}(\theta)), \tag{3.48}$$

with $\sigma(X), X \in \mathbb{R}^{n \times n}$ denoting the spectrum of a square matrix X.

Figure 3.4 shows the asymptotic convergence factor (3.48) of a $V(1,1)$-cycle for the finite element discretization (3.17) of the diffusion equation as a function of $\beta_h = h^2\beta$. Table 3.2 shows that the asymptotic convergence factor, calculated using LFA, is a good approximation of the asymptotic convergence factor obtained from numerical experiments.

3.4 Concluding remarks

The key to efficient multigrid algorithms is the interplay between the smoothing and the coarse grid correction and vice versa. More precisely, errors that cannot be reduced efficiently by smoothing are to be reduced by the coarse grid correction. For simple problems, such as the discretization of the diffusion equation (3.11) with constant material parameters on a regular grid, simple components are sufficient for this purpose, such as Jacobi or Gauss-Seidel smoothing and a prolongation scheme based on linear interpolation. For realistic problems, however, more involved multigrid components are needed.

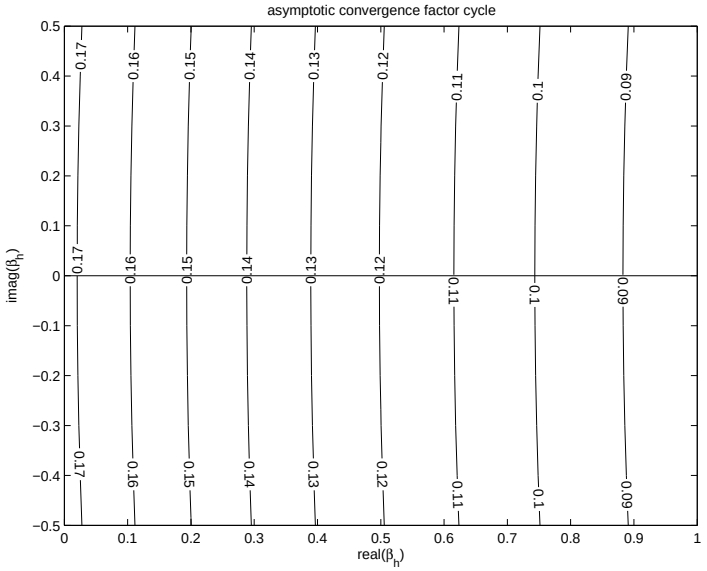


Figure 3.4: Asymptotic convergence factor (3.48) of a $V(1,1)$-cycle using GS-smoothing for the finite element discretization (3.17) of the diffusion equation (3.11) as a function of $\beta_h = \beta h^2$.

Geometric multigrid (GMG) methods typically use a simple prolongation scheme, such as linear interpolation. Such prolongation scheme is only successful if the error after smoothing varies in a geometrically smooth way. This imposes strong requirements on the smoother. Correspondingly, robust smoothers, such as alternating line relaxation or ILU-smoothing, are to be used (cfr. [63]). In geometric multigrid, the coarse level operators are typically constructed by rediscretizing the original problem on a coarse grid. This requires a hierarchy of grids to be available. Also, it requires the multigrid solver and the discretization module to be tightly coupled. This ad-hoc character typically limits the application range of GMG solvers, but it allows them to be highly tailored for specific applications.

On the other hand, *algebraic multigrid* (AMG) methods rely on a simple smoother and try to construct suitable operator-dependent coarsening and prolongation schemes. The coarse level operators are constructed as a Galerkin product. This allows AMG to be used as a black box solver.

In this thesis, we will mainly focus on AMG methods. First, in chapters 4 and 5, we will present an overview of AMG methods for the div-grad and curl-curl equations. Multigrid cycles for the div-grad and curl-curl equations have the same structure, which has been presented in section 3.2. However, both problems types require different smoothing and prolongation methods. These fundamental differences, which will be studied in detail, are closely related to the properties of the discrete de Rham sequences (see section 2.5.3).

Next, in chapter 6, we will present a LFA analysis of the basic multigrid algorithms for the curl-curl equation. This analysis is based on the same principles as the LFA for the div-grad equation, presented in section 3.3. However, the LFA techniques, which are essentially based an the evaluation of Fourier modes in the *nodes* of a grid, need to be translated to the *edge*-based setting of the curl-curl equation. This translation, which is very technical, has not been done before.

In chapter 7, we will develop a new AMG method for the curl-curl equation. This new method is tightly coupled to the Whitney framework, presented in section 2.5. In fact, it can be considered a coarse version of this framework.

Finally, in chapter 8, we will show how AMG methods for the div-grad and curl-curl equations can be combined with higher order time discretization methods. This chapter is an extension of the work presented in [47], which considers geometric multigrid for the div-grad equation.

Chapter 4

Overview of algebraic multigrid for the div-grad equation

In this chapter, we will present an overview of existing AMG methods for the div-grad equation (3.11). In the context of computational electromagnetics, these AMG methods are useful for two reasons. First, they can be used directly to solve electromagnetic problems which can be described by a div-grad equation. This is the case for electrostatic problems (2.12) and for the translation-invariant curl-curl equations (2.14)-(2.17) for the magnetic vector potential $\vec{A}$, which reduce to a div-grad equation (cfr. (2.47)). On the other hand, AMG methods for the curl-curl equation are typically based on or derived from AMG methods for the div-grad equation.

This chapter is organized as follows. In section 4.1, the effect of typical smoothers for discretizations of div-grad problems will be analyzed. Next, the construction of a prolongation operator will be studied. This typically consists of two steps. First, the coarse degrees of freedom are selected. This step, which is often referred to as *coarsening*, is the subject of section 4.2. The second step, in which the nonzero entries of the prolongation matrix are calculated, will be discussed in section 4.3. In the last section 4.4, we will review some properties of the Galerkin coarse system matrix, which is typically used in the context of AMG methods.

4.1 Smoothing

In an algebraic multigrid context, simple smoothers are used. Popular smoothers are point-wise relaxation methods, such as Jacobi or Gauss-Seidel iterations, and polynomial iterations. In section 4.1.1, some characteristics of these smoothers will be investigated. In section 4.1.2, we will explain the concept of algebraically

smooth errors. This concept is essential in the development of many types of AMG methods. Finally, in section 4.1.3, we will show that for positive definite problems, the action of these smoothers corresponds to reducing the energy norm of the error. Energy considerations will be important for the prolongation operators and for the Galerkin coarse system matrices as well.

4.1.1 Pointwise and polynomial smoothing methods

The basic step in pointwise relaxation methods is the solution of the ith equation for the ith unknown:

$$\tilde{x}_i \quad = \quad x_i + \frac{b_i - \sum_j K_{ij} x_j}{K_{ii}} \quad = \quad \frac{b_i - \sum_{j \neq i} K_{ij} x_j}{K_{ii}}. \tag{4.1}$$

In each iteration, (4.1) is performed for each unknown. In the Gauss-Seidel method, the old value of x_i is immediately replaced by the updated value $\tilde{x}_i$. In a Jacobi iteration, x is replaced by $\tilde{x}$ at the end of the iteration. Typically, the Jacobi method is applied with damping, i.e. the denominator of (4.1) is multiplied with a damping factor $d > 1$.

A polynomial smoother (see [1, 28]) is a matrix splitting iteration of the form (3.1), with the approximate inverse $(K^+)^{-1}$ being the application of a polynomial $p(x)$ to K:

$$x^{[\nu+1]} \quad = \quad x^{[\nu]} + p(K) \left(b - K x^{[\nu]} \right). \tag{4.2}$$

The convergence behavior of this iteration is determined by the eigenvalue decomposition of the iteration matrix $q(K) = I - p(K)K$, which reads:

$$K = V \Lambda V^{-1} \quad \Longrightarrow \quad q(K) = V q(\Lambda) V^{-1}. \tag{4.3}$$

More precisely, the error along the eigenvector v_i corresponding to the eigenvalue $\lambda_i \in \sigma(K)$ will be reduced by a factor $q(\lambda_i)$ in each iteration. The error after ν iteration steps reads, for an initial error $e^{[0]} = \sum_i a_i v_i$:

$$e^{[\nu]} \quad = \quad \sum_i a_i q(\lambda_i)^\nu v_i. \tag{4.4}$$

The polynomial $p(x)$ can be chosen such that the corresponding iteration exhibits similar smoothing properties as Jacobi and Gauss-Seidel iterations. For this purpose, $q(x)$ should be small for the large eigenvalues of K. Typically, $q(x)$ is required to be small in the range $[\frac{\lambda_{max}}{2^d},\ \lambda_{max}]$, with d and λ_{max} denoting respectively the geometric dimension of the problem and the spectral radius of K. This can be achieved by choosing $p(x)$ such that $q(x)$ is identical to a scaled and translated Chebyshev polynomial (see Figure 4.1).

4.1.2 Algebraically smooth error

Table 3.1 shows a typical convergence history of a simple iterative solver, such as Jacobi or Gauss-Seidel, applied to the discretization of a PDE. Typically,

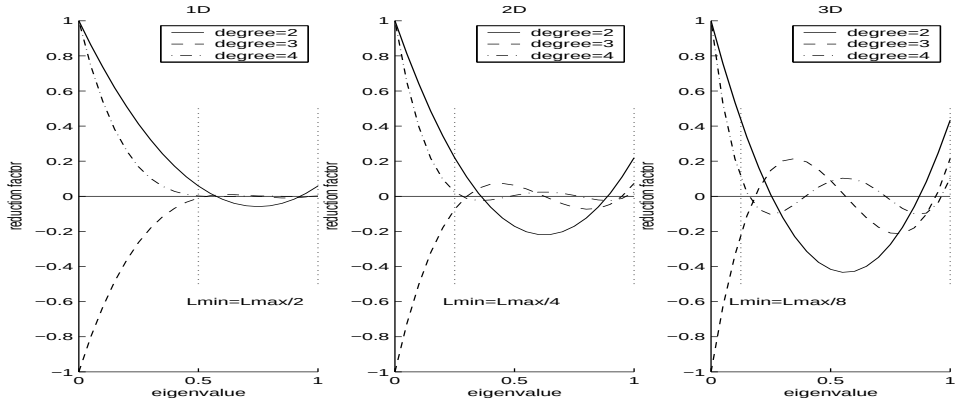


Figure 4.1: Scaled and translated Chebyshev polynomials $q(x)$ representing the iteration matrix of a polynomial smoother, for different polynomial degrees.

the first iterations yield a significant decrease of the norm of the residual. After that, the iteration seems to stagnate. At this point, the error cannot be reduced efficiently any more by this smoother.

Definition 4.1.1 *If the error in an iterative solver is slow to converge, the error is called algebraically smooth with respect to this solver.*

We will now analyze the characteristics of an algebraically smooth error more in detail for the Gauss-Seidel method. In its basic updating step (4.1), the errors e_i and $\tilde{e}_i$, corresponding to x_i and $\tilde{x}_i$, are related as follows, with r_i denoting the residual $b_i - \sum_j K_{ij} x_j$:

$$\tilde{e}_i \quad = \quad e_i - \frac{r_i}{K_{ii}}. \tag{4.5}$$

Consequently, stagnation of the iteration implies that the scaled residual r_i/K_{ii} is small with respect to the ith error component e_i:

$$e_i \simeq \tilde{e}_i \quad \Longrightarrow \quad \frac{|r_i|}{K_{ii}} \ll |e_i|. \tag{4.6}$$

This allows each component of an algebraically smooth error to be approximated as a function of its neighbours:

$$K_{ii} e_i + \sum_{j \neq i} K_{ij} e_j = r_i \simeq 0 \quad \Longrightarrow \quad e_i \simeq - \frac{\sum_{j \neq i} K_{ij} e_j}{K_{ii}}. \tag{4.7}$$

In general, an algebraically smooth error is not smooth in the geometric sense, i.e. varying slowly in space. However, for the application of simple iterative solvers, such as AMG smoothers, to discretizations of elliptic PDEs, there is a connection. In this case, an algebraically smooth error varies slowly *in the*

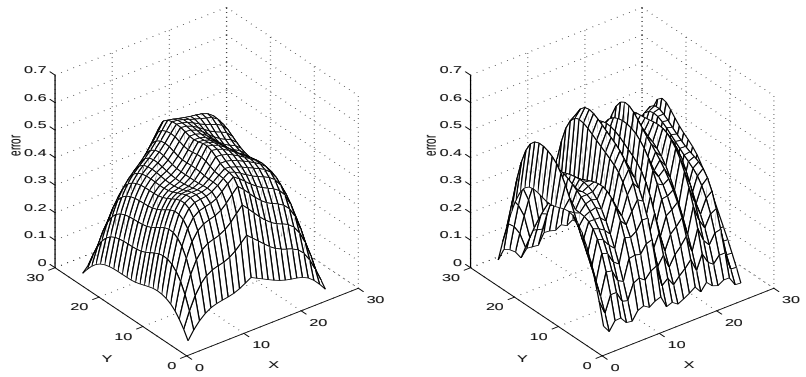


Figure 4.2: Error after 10 Gauss-Seidel iterations for the finite difference discretization of the 2D Poisson equation $\mathrm{div}(\alpha\,\mathrm{grad}(V)) = b$, for a discontinuous α in the left picture ($\alpha = 1$ for $x < 13$ and $\alpha = 1000$ for $x \geq 13$) and an anisotropic $\alpha = [1\ 0;\ 0\ 1000]$ in the right picture.

direction of large negative connections K_{ij} (cfr. [63, 60]). In particular, algebraic smoothness implies geometric smoothness for isotropic problems. The geometric smoothness is disturbed by variations and anisotropies in the diffusion coefficient (see Figure 4.2).

Large negative connections will be called strong. Two definitions of strong connections exist in AMG literature.

Definition 4.1.2 (Strong connection [60]) *A variable i is strongly connected to another variable j if for a fixed $0 < \epsilon_{str} < 1$:*

$$-K_{ij} \geq \epsilon_{str} \max_{K_{ik}<0} |K_{ik}|. \tag{4.8}$$

Definition 4.1.3 (Strong connection [65]) *A variable i is strongly connected to another variable j if for a fixed $0 < \epsilon_{str} < 1$:*

$$-K_{ij} \geq \epsilon_{str} \sqrt{K_{ii} K_{jj}}. \tag{4.9}$$

Definition 4.1.2, which was originally developed in the context of classical AMG (see section 4.3.3), is not symmetric, i.e. a strong connection of i to j does not imply a strong connection of j to i. In definition 4.1.3, symmetry is preserved.

Definition 4.1.4 *The set of nodes with a strong connection to node i will be denoted S_i:*

$$S_i \quad := \quad \{j \neq i \mid i \text{ is strongly connected to } j\}. \tag{4.10}$$

4.1.3 Reduction of the energy norm of the error

Next, we will study the action of the smoothers from the perspective of energy minimization. It is well known that the solution of a linear system $Kx = b$, with K a symmetric positive definite matrix, can be recovered as the solution of the following minimization problem:

$$\arg\min_x \left(F(x)\right) \quad := \quad \arg\min_x \left(\frac{1}{2}x^T K x - x^T b\right). \tag{4.11}$$

Consider the restriction of this minimization problem to the ith unknown. With ϵ_i and x denoting respectively the ith unit vector and the current approximation of the solution, this restricted minimization problem can be formulated as:

$$x \leftarrow x + \alpha_i \epsilon_i \quad \text{with} \quad \alpha_i = \arg\min_{\alpha_i} F(x + \alpha_i \epsilon_i). \tag{4.12}$$

The optimal value for α_i is obtained by differentiating $F(x + \alpha_i \epsilon_i)$ with respect to α_i, which yields:

$$\alpha_i \quad = \quad \frac{b_i - \sum_j K_{ij} x_j}{K_{ii}}. \tag{4.13}$$

The corresponding update $x \leftarrow x + \alpha_i \epsilon_i$ is identical to the basic update step (4.1) of a Jacobi or Gauss-Seidel iteration. Since (4.11) is equivalent to minimizing the energy-norm $\|e\|_K$ of the error, the above shows that Jacobi and Gauss-Seidel iterations can be considered energy minimizing procedures.

Energy is taken into account by the polynomial smoother as well, since by construction, its achieves the greatest error reduction for error components corresponding to the large eigenvalues of K.

4.2 Coarsening

In the coarsening process, the coarse degrees of freedom (DoF) are selected. The coarse DoFs correspond to the columns of the prolongator. Depending on the algorithm used to construct the prolongator, they will be associated with the coarse nodes of a so-called C/F-splitting, with aggregates of nodes or with agglomerates of elements.

4.2.1 C/F-splitting

The coarse DoFs can be associated with a subset of the nodes. These nodes will be called "coarse", and the other nodes will be called "fine". Let C and F denote the sets of the coarse and fine nodes respectively.

In prolongation schemes based on a C/F-splitting, the coarse nodes typically prolongate to themselves exactly. Let n_I be the index of a particular coarse node, and n_i the index of that same node on the fine mesh. The coarse node

n_I is said to be prolongated exactly to n_i if $P(n_I, n_i) = 1$. Moreover, all of the other entries in the n_i'th row of the prolongation matrix P are zero:

$$P(C,:) \quad = \quad I. \tag{4.14}$$

Here, I denotes the identity matrix of size equal to the number of coarse nodes. This allows to conclude that in the prolongation formula $x = Px_c$, x_c can be considered a subsampled version of x. In other words, this type of coarsening can be considered a subsampling procedure. This motivates to select the coarse nodes such that C is a maximal independent set.

Definition 4.2.1 *An independent set S of nodes is a set of nodes such that any two nodes belonging to S are not connected. A maximal independent set is an independent set to which no additional node can be added.*

The coarsening process can be improved by taking into account that ideally, the prolongation step of a multigrid cycle

$$x \leftarrow x + Px_c \quad \Longleftrightarrow \quad e \leftarrow e - Px_c.$$

should correct the errors which cannot be eliminated efficiently by smoothing, i.e. the algebraically smooth errors. In other words, algebraically smooth errors should, in the optimal case, belong to the range of P:

$$\forall \text{ algebraically smooth errors } e, \ \exists x_c \quad : \quad e \simeq Px_c. \tag{4.15}$$

For prolongators satisfying (4.14), this goal can be achieved by requiring the subsampling by P of an algebraically smooth error to be a good coarse representation of that error. This can be achieved by coarsening the nodes only in those directions in which algebraically smooth errors are known to vary smoothly, i.e. in the directions of strong connections (see section 4.1.2). This is often referred to as *semi-coarsening*. A suitable semi-coarsening is typically obtained by a maximally independent set in which *only* strong connections between coarse nodes are ruled out, but weak connections are allowed.

For this purpose, the so-called *standard coarsening* algorithm of [60] can be used. This algorithm is an iterative procedure. Let U denote the set of undecided nodes. U is initialized to include all nodes of the fine mesh. In each iteration, an undecided node i with maximal value for the measure

$$\lambda_i = |S_i^T \cap U| + 2|S_i^T \cap F|, \qquad \text{with } S_i^T := \{j \mid i \in S_j\}, \tag{4.16}$$

is chosen to become a coarse node. This measure favors the choice of nodes from which many nodes strongly depend, i.e. nodes with large $|S_i^T|$. This typically results in a smaller number of coarse nodes. The second term increases the measure for nodes from which many nodes strongly depend, which are already assigned to F. This typically causes the set of decided nodes to grow in a compact way, which yields uniform distributions of C- and F-variables, rather than randomly distributed C/F-patches. Next, all nodes $j \in S_i^T \cap U$ are eliminated

from U and become fine nodes. For problems with a varying or anisotropic coefficient α, this algorithm automatically results in semi-coarsening.

A faster coarsening, i.e., a smaller number of coarse nodes, can be obtained by changing the definition of strong connections. *Aggressive coarsening* of [60] for instance is based on long-range strong connections: a node i has a long-range strong connection to a node j if i and j are connected by paths of strong connections.

4.2.2 Node aggregation

In aggregation-based AMG algorithms, the coarse degrees of freedom are associated with aggregates of nodes. For this purpose, the set of the nodes is split into disjoint subsets of strongly connected nodes, which are called *aggregates*. Since the strength of the connections is taken into account, such splitting automatically has a semi-coarsening character.

For completeness, we present a possible aggregation algorithm below. It is discussed in detail in [65]. Let a denote the number of aggregates, U the set of unknowns which don't belong to an aggregate yet and C_i for $1 \leq i \leq a$ the set of unknowns belonging to the ith aggregate.

1. Initial aggregation:

$$\forall i \in U \quad : \quad S_i \subset U \quad \Longrightarrow \quad \left\{ \begin{array}{lcl} a & \leftarrow & a+1, \\ C_a & \leftarrow & S_i \cup \{i\}, \\ U & \leftarrow & U \setminus C_a. \end{array} \right.$$

2. Adding remaining unknowns to existing aggregates:

   1. $\forall 0 < i < a \ : \ \tilde{C}_i \leftarrow C_i$
   2. $\forall i \in U \ : \ \exists j, \ S_i \cap \tilde{C}_j \neq \phi \quad \Longrightarrow \quad \left\{ \begin{array}{lcl} C_j & \leftarrow & C_j \cup i \\ U & \leftarrow & U \setminus \{i\}. \end{array} \right.$

After step 1, each unknown belonging to U has at least one strongly connected neighbour not belonging to U. Hence, all unknowns of U will be assigned to an existing aggregate in step 2 if the symmetric definition 4.1.3 of strong connections is used.

4.2.3 Element agglomeration

When applied to finite element discretizations, some AMG methods can benefit from the construction of coarse elements. This is often referred to as element agglomeration.

For this purpose, the element-face and face-node connection information of the original mesh is needed. This connection information indicates which faces belong to an element and which nodes belong to a face respectively, and can be implemented efficiently using incidence matrices (see [66]). For instance, the face-node connection information can be represented by a matrix

$FN \in \{0,1\}^{|F|\times|N|}$, with $|F|$ and $|N|$ denoting the number of faces and nodes respectively, such that $FN(f,n) = 1$ if the node n belongs to the face f.

The following algorithm can be used to construct compact coarse elements on 2D meshes (see [66, 42]). It is presented here to the reader for completeness sake only. It will not be used further on in this thesis. Let all fine and coarse vertices be represented by the set of fine nodes belonging to them. Let $w(f)$ denote a weight function for all fine faces. Let E_i and i denote the actual coarse element being constructed and its index.

1. Initialization: $w(f) = 0$ and $i = 1$.
2. Stop if $w(f) = -1$ for all f. Else, select f with maximal $w(f)$.
   (a) Set $W = w(f)$ and $w(f) = -1$.
   (b) Add to E_i the fine elements to which f belongs.
   (c) Update the weights: $\forall$ face g, $w(g) \geq 0$, $g \cap f \neq \phi$:

$$\begin{array}{lll} \text{if} \quad \exists \text{ fine element } e \ : \ f,g \subset e & \text{then} & w(g) \leftarrow w(g)+2 \\ & \text{else} & w(g) \leftarrow w(g)+1 \end{array}$$

   (d) Test for completion of E_i:

$$\text{if } \forall \text{ fine face } g \in E_i, \quad w(g) < W \quad : \quad \left\{ \begin{array}{l} \forall g \in E_i \ : \ w(g) \leftarrow -1 \\ i \leftarrow i+1, \ E_i = \phi \\ \text{goto step 2.} \end{array} \right.$$

4.3 Prolongation

In this section, we will study the construction of the prolongator operator. First, in section 4.3.1, we formulate a series of constraints and properties to be satisfied by a prolongator. In section 4.3.2, we present a prolongator with ideal approximation properties. Next, we will discuss several algorithms to construct an AMG prolongator. We will consider in detail classical AMG (section 4.3.3), smoothed aggregation AMG (section 4.3.4), AMG based on energy minimization (section 4.3.5) and element-based AMG methods (section 4.3.6).

4.3.1 Constraints

In this section, we will discuss some common constraints, which a prolongator is usually required to satisfy.

Dirichlet boundary conditions

Dirichlet unknowns (see section 2.5.4) are typically resolved quickly by the application of the fine level smoother. For instance, they are resolved exactly by a single iteration of the Gauss-Seidel smoother. Consequently, there is no need any more to update these unknowns in the prolongation step of the multigrid

cycle. In other words, the rows of the prolongator P corresponding to Dirichlet unknowns, which will be denoted d, should be zero:

$$P(d,:) \quad = \quad 0. \tag{4.17}$$

Remark 4.3.1 Dirichlet unknowns in a system of the form (2.121) have no strong connections to any other node. Consequently, the coarsening algorithms based on a C/F-splitting or a node aggregation (see section 4.2) would identify each individual fine Dirichlet DoF as a coarse DoF. In other words, no coarsening would be achieved at all for the Dirichlet unknowns. The constraint (4.17) avoids this problem, since it apriori prevents Dirichlet unknowns to be part of a coarse DoF.

Field-circuit coupling

A field model, such as a finite element discretization of a PDE, can be coupled to a circuit model (see [45]). For instance, consider a model of an electric motor coupled to an external electric circuit. Typically, the resulting system can be written in the following block structure:

$$\begin{bmatrix} K_{field} & K_{coupling}^T \\ K_{coupling} & K_{circuit} \end{bmatrix} \begin{bmatrix} x_{field} \\ x_{circuit} \end{bmatrix} = \begin{bmatrix} b_{field} \\ b_{circuit} \end{bmatrix}. \tag{4.18}$$

Due to their different nature, the field and circuit variables need to be coarsened separately. In general, it is not clear what a coarse version of the circuit model should look like, and the circuit variables are typically not coarsened at all. Since the dimension of the circuit model $K_{circuit}$ is typically much smaller than the dimension of the field model K_{field}, this does not limit the coarsening speed significantly. This procedure imposes to following block structure for the prolongation operator P:

$$P \quad = \quad \begin{bmatrix} P_{field} & 0 \\ 0 & P_{circuit} \end{bmatrix}, \qquad \text{with } P_{circuit} = I. \tag{4.19}$$

Locality of coarse basis functions

The columns of the prolongator P represent linear combinations of the fine basis functions $N_i(\mathbf{x})$, which can be interpreted as coarse basis functions N_I. For the Ith column of P, we have:

$$N_I(\mathbf{x}) \quad := \quad \sum_i P(i,I) N_i(\mathbf{x}). \tag{4.20}$$

In order to obtain a sparse coarse operator, which will be constructed as a Galerkin product with P (see section 4.4), the coarse basis functions should have a limited support. In most AMG algorithms, this requirement is satisfied by construction.

Normalization of coarse basis functions

Similarly as the fine basis functions, the coarse basis functions can be required to be normalized, i.e.:

$$\forall I,\ \exists \mathbf{x}_I \quad : \quad N_J(\mathbf{x}_I) = \delta_{IJ}. \tag{4.21}$$

This normalization requirement relates each coarse basis function to a physical node, and is equivalent to requiring a C/F-splitting (see section 4.2) to exist. Consequently, the prolongator satisfies (4.14).

Partition of unity property

The partition of unity property requires all row sums of P to be 1. With $\mathbf{1}_f$ and $\mathbf{1}_c$ denoting the fine and coarse constant unit vectors, this property reads:

$$\forall i \ : \ \sum_j P(i,j) = 1, \quad \text{or} \quad \mathbf{1}_f = P\mathbf{1}_c. \tag{4.22}$$

In other words, a constant error on the fine level can be represented exactly on the coarse level, which allows this error component to be resolved on the coarse level. This is needed, because a constant error remains essentially untouched by the fine level smoothing.

The transpose of (4.22) allows the partition of unity property to be interpreted from the perspective of the restriction. More precisely, the restriction $b_c = P^T b_f$ of the righthand side in the fine system $K_f x_f = b_f$ is such that the sum of all elements of b_c and b_f are identical:

$$\sum_i b_c(i) \quad = \quad \mathbf{1}_c^T b_c \quad = \quad \mathbf{1}_c^T P^T b_f \quad = \quad \mathbf{1}_f^T b_f \quad = \quad \sum_i b_f(i). \tag{4.23}$$

In the context of electrostatics, for instance, this amounts to *charge conservation*, i.e. conservation of the total electric charge present in the system.

Remark 4.3.2 The partition of unity property (4.22) transfers the structure of the fine discrete de Rham sequence (2.99) partially to the coarse level:

$$\begin{array}{ccccccccc} \mathbb{R} & \xrightarrow{\mathbf{1}_f} & \mathcal{N}_f & \xrightarrow{G_f} & \mathcal{E}_f & \xrightarrow{C_f} & \mathcal{F}_f & \xrightarrow{D_f} & \mathcal{V}_f \\ & & \uparrow P_n & & & & & & \\ \mathbb{R} & \xrightarrow{\mathbf{1}_c} & \mathcal{N}_c & & & & & & \end{array} \tag{4.24}$$

Typically, the prolongators to be used by AMG methods in $\mathcal{E}_f$ and $\mathcal{F}_f$ will need to satisfy similar properties, which transfer the rest of the fine discrete de Rham sequence to the coarse level (see chapters 5 and 7).

4.3.2 Ideal prolongator

Recall that an algebraically smooth error e typically has a low energy norm (see section 4.1.3) and that ideally, e should belong (approximately) to the range of the prolongator P (cfr. (4.15)). This implies that the energy norms of the columns of P should be small. In this section, a prolongator will be derived with columns having an optimal energy norm.

Let a C/F-splitting be given and consider the block representation of the discretization matrix K, with the subscripts $\cdot_f$ and $\cdot_c$ referring to the fine and the coarse unknowns respectively:

$$\begin{bmatrix} K_{ff} & K_{fc} \\ K_{cf} & K_{cc} \end{bmatrix} \begin{bmatrix} x_f \\ x_c \end{bmatrix} = \begin{bmatrix} b_f \\ b_c \end{bmatrix}. \tag{4.25}$$

We assume K_{ff} to be invertible.

Theorem 4.3.1 *Let K be symmetric positive definite and let P be a prolongator satisfying (4.14). For any given x_c, the energy norm $\|Px_c\|_K$ is minimal for P defined as:*

$$P_{\min} = \begin{bmatrix} -K_{ff}^{-1}K_{fc} \\ I \end{bmatrix}. \tag{4.26}$$

Proof. The minimization problem to be solved reads, for a fixed x_c:

$$\min_{x_f} \left\| \begin{bmatrix} x_f \\ x_c \end{bmatrix} \right\|_K^2 = \min_{x_f} \left(x_f^T K_{ff} x_f + 2x_f^T K_{fc} x_c + x_c^T K_{cc} x_c \right).$$

Its solution is obtained by requiring the derivatives for all entries of x_f to be zero:

$$2K_{ff}x_f + 2K_{fc}x_c = 0.$$

This is equivalent to $[x_f \; x_c]^T = P_{\min} x_c$, with $P_{\min}$ defined as in (4.26). □

From a practical point of view, this prolongation scheme is not useful, because the calculation of $P_{\min}$, which requires the inversion of K_{ff}, is very expensive. Also, the coarse nodal elements have a global support, since $P_{\min}(f,:)$ is dense.

Remark 4.3.3 Note that the Galerkin coarse operator corresponding to the prolongator $P_{\min}$ is identical to the Schur complement of K for the coarse unknowns:

$$\begin{bmatrix} -K_{fc}^T K_{ff}^{-T} & I \end{bmatrix} \begin{bmatrix} K_{ff} & K_{fc} \\ K_{cf} & K_{cc} \end{bmatrix} \begin{bmatrix} -K_{ff}^{-1}K_{fc} \\ I \end{bmatrix} = K_{cc} - K_{cf}K_{ff}^{-1}K_{fc}.$$

4.3.3 Classical AMG

Classical AMG is based on the assumption that algebraically smooth errors vary slowly in the directions of strong connections. This typically holds for discretizations of diffusion-dominated problems (see section 4.1.2). This assumption motivates the use of a prolongation procedure based on interpolation along the directions of strong connection.

Let a C/F-splitting be available and let C denote the set of the coarse nodes. The so-called *direct interpolation* scheme, presented in [60], is based on the approximation (4.7) of the error equation for algebraically smooth errors. More precisely, a weighed averaging interpolation scheme is constructed for each fine node i by restricting (4.7) to the directions of strong connections:

$$\forall i \in F \quad : \quad e_i \leftarrow \frac{\sum_{j \in C \cap S_i} K_{ij} e_j}{\sum_{j \in C \cap S_i} K_{ij}}. \tag{4.27}$$

Note that this prolongation scheme satisfies the partition of unity property (4.22).

Several variants of (4.27) are described in [60]. The most popular is the so-called *standard interpolation* scheme, which takes the strong connections of i to its fine neighbours into account as well. For this purpose, all fine nodes are eliminated from (4.27). We refer the reader to [60] for more details.

4.3.4 Smoothed aggregation

Let an aggregation of nodes be available (see section 4.2.2). This aggregation can be used to construct a piecewise constant prolongation operator P_{pc}, which gives to each fine node the same value as the coarse aggregate to which it belongs. This prolongation operator P_{pc} satisfies:

$$\forall i \in C_j \quad : \quad P_{pc}(i,j) = 1. \tag{4.28}$$

Recall that C_j was defined as the set of fine nodes belonging to the j'th aggregate. The resulting prolongator contains one nonzero entry equal to 1 in each row. It satisfies the partition of unity property (4.22).

Due to its piecewise constant character, the approximation quality of P_{pc}, in the sense of (4.15), will be poor. This can be improved by applying the smoothing operation, which by definition yields algebraically smooth vectors (see section 4.1.3), to the columns of P_{pc} as well. In *smoothed aggregation AMG* [65], P_{pc} is smoothed by a damped Jacobi iteration with respect to the homogeneous problem $Kx = 0$. With D and d denoting the diagonal of K and the damping factor, the smoothed prolongator is obtained as:

$$P \quad = \quad (I - (dD)^{-1}K)P_{pc}. \tag{4.29}$$

If constant vectors belong to the kernel of K, which is the case for discretizations of the diffusion equation (3.11) with $\beta = 0$ in non-Dirichlet nodes, then the

prolongator resulting from (4.29) satisfies the partition of unity property (4.22) as well:

$$P\mathbf{1}_c \;=\; (I-(dD)^{-1}K)P_{pc}\mathbf{1}_c \;=\; \mathbf{1}_f-(dD)^{-1}K\mathbf{1}_f \;=\; \mathbf{1}_f. \tag{4.30}$$

This smoothing procedure causes the support of the coarse basis functions to grow. In order to prevent growth in the directions of weak connections, a filtered version of the matrix K, denoted as $\tilde{K}$, is often used in the smoothing procedure (4.29). More precisely, the off-diagonal entries of K corresponding to the weak connections are set to zero:

$$\forall j\neq i \quad : \quad \tilde{K}_{ij}=\begin{cases} K_{ij} & \text{if } j\in S_i, \\ 0 & \text{otherwise.} \end{cases} \tag{4.31}$$

In order to preserve the partition of unity property (4.22) for P, the diagonal entries are updated such that the row sums are equal to zero:

$$\forall i \quad : \quad \tilde{K}_{ii}=-\sum_{j\neq i}\tilde{K}_{ij}. \tag{4.32}$$

4.3.5 Energy minimization of a prolongator

In section 4.1, we showed that algebraically smooth errors typically have a low energy. In order to obtain a good approximation of low energy vectors by the prolongator, the columns of the prolongator should be low energy vectors themselves. A low energy prolongator with a prescribed nonzero pattern and satisfying the partition of unity property (4.22) can be found as the solution to a constrained minimization problem. With $J(P)$ defined as the sum of the energies, i.e. the K-norms of the columns of P:

$$J(P) \quad := \quad \frac{1}{2}\sum_j P(:,j)^T K P(:,j), \tag{4.33}$$

this minimization problem reads:

$$\min_{P\in\mathcal{Q}} J(P) \quad s.t. \quad \sum_j P(:,j)=\mathbf{1}_f. \tag{4.34}$$

The nonzero pattern is fixed by requiring P to belong to $\mathcal{Q}$, which is defined as the set of matrices with the same nonzero pattern as a reference binary matrix $Q\in\{0,1\}^{n\times m}$:

$$\mathcal{Q} \quad := \quad \{P\in\mathbb{R}^{n\times m} \mid Q(i,j)=0 \implies P(i,j)=0\}. \tag{4.35}$$

In [67], the columns of the optimized P are found as the (approximate) solution to the saddle point formulation of (4.34). In [48], (4.34) is solved using the following projected steepest descent algorithm:

1. Choose $P^{[0]}\in\mathcal{Q}$, satisfying (4.22).
2. $P^{[n+1]}=P^{[n]}+\omega Z(Q*(KP))$. (4.36)

Here, the $*$-operator denotes elementwise multiplication:

$$C = A * B \iff C(i,j) = A(i,j)B(i,j).$$

Z is the orthogonal projection operator mapping matrices of $\mathcal{Q}$ to the subset of matrices in $\mathcal{Q}$ that have zero rowsum. The action of this projection operator amounts to subtracting from each row the average of its nonzero elements:

$$Q = Z(S) \iff Q(i,:) = S(i:,) - \frac{S(i,:)Q(i,:)^T}{Q(i,:)Q(i,:)^T}Q(i,:).$$

This orthogonal projection guarantees that the resulting prolongator satisfies the partition of unity property (4.22). In order to achieve a convergent iteration, it can be proven (see [48]) that the damping factor ω needs to be chosen greater than or equal to $\frac{\rho(K)}{2}$). For proofs and further details, we refer to [48].

In chapter 7, we have used a slight variant (unpublished) of the energy minimization algorithm developed in [48]. This algorithm can also be used for the energy minimization of AMG prolongators for curl-curl problems. The full details of this algorithm can be found in Appendix A.

4.3.6 Element-based AMG methods

In element-based AMG methods, the characteristics of low energy modes are derived from *local* matrix representations of the differential operator. In the context of finite element discretizations, the contributions K_α of the finite elements α to the system matrix $K = \sum_\alpha K_\alpha$, i.e. the so-called *element matrices* (cfr. (2.120)), can be used for this purpose.

AMGe

Let a C/F-splitting be available. In AMGe, the coarse nodes prolongate to themselves exactly i.e. the prolongator satisfies (4.14). In order to calculate the row of the prolongator for a fine node i, a *local* representation of the differential operator in the neighbourhood of i is needed. Let $\mathcal{T}_i$ denote the neighbourhood of i, which is the set of all elements containing i, and let $\mathcal{N}_i$ denote the nodes belonging to $\mathcal{T}_i$:

$$\mathcal{T}_i := \{\alpha \mid \epsilon_i^T K_\alpha \epsilon_i \neq 0\}, \tag{4.37}$$

$$\mathcal{N}_i := \{j \mid \exists \alpha \in \mathcal{T}_i : \ j \in \alpha\}. \tag{4.38}$$

Here, ϵ_i denotes the ith unit vector. Unfortunately, the submatrix $K(\mathcal{N}_i, \mathcal{N}_i)$ of K for the nodes of $\mathcal{N}_i$ cannot be used as a local representation of the differential operator in $\mathcal{N}_i$. This is because the neighbour elements of $\mathcal{T}_i$ contribute to the entries of $K(\mathcal{N}_i, \mathcal{N}_i)$ as well, which disturbs the local representation of the differential operator. Instead, the proper discretization of the differential operator in $\mathcal{T}_i$, i.e. the assembling of the element matrices of all elements belonging to

$\mathcal{T}_i$, is to be used:

$$K_i \quad := \quad \sum_{\alpha \in \mathcal{T}_i} K_\alpha. \tag{4.39}$$

Clearly, the element matrices must be available for this purpose. As an example, consider the finite element discretization of the one-dimensional Poisson equation on an equidistant grid with grid size 1. In this case, $\mathcal{T}_i$ consists of the two intervals having i as end node, and $\mathcal{N}_i = \{i-1, i, i+1\}$. In this case, we have:

$$K(\mathcal{N}_i, \mathcal{N}_i) = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}, \qquad K_i(\mathcal{N}_i, \mathcal{N}_i) = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}.$$

Observe that K_i is the discretization matrix for the Poisson equation on the interval $[x_{i-1},\ x_{i+1}]$ with homogeneous Neumann boundary conditions in x_{i-1} and x_{i+1}.

Let f_i and c_i denote the vectors of the fine respectively coarse nodes of $\mathcal{N}_i$, and let $f_i(1) = i$. In AMGe, the nonzero entries of the ith row of the prolongation operator are obtained as a local approximation of the ideal prolongator (4.26). In particular, one has:

$$P(i, c_i) \quad = \quad \left(-K_i(f_i, f_i)^{-1} K_i(f_i, c_i)\right)(1, :). \tag{4.40}$$

Formula (4.40) imposes a constraint on the C/F-splitting. The C/F-splitting must be such that $K_i(f_i, f_i)$ is invertible for all fine nodes i. For a theoretical motivation of this procedure, we refer to [15].

In order to apply this procedure recursively, element matrices must be available on the coarse level as well. The coarse element matrices are constructed as the Galerkin product of the fine element matrices with P:

$$K_{\alpha,c} \quad = \quad P^T K_\alpha P. \tag{4.41}$$

In order to reduce the computational complexity and memory requirements, coarse elements operating on the same coarse unknowns are combined and their element matrices added.

Spectral AMGe (ρAMGe)

Let a partitioning of the elements of a mesh into compact agglomerates (see section 4.2.3) be available and let K_A denote the sum of the element matrices of the elements belonging to the agglomerate A:

$$K_A \quad := \quad \sum_{\alpha \in A} K_\alpha. \tag{4.42}$$

Let n_A denote the vector of the indices of the nodes belonging to the agglomerate A.

Spectral AMGe [21] is a variant of AMGe in which the coarse degrees of freedom are associated with agglomerates of elements. The coarse nodal elements are based on the eigenvalue decomposition of the agglomerate matrices K_A. Consider the eigenvalue decomposition of K_A from which the zero rows and columns are discarded:

$$K_A(n_A, n_A) \quad := \quad V_A \Lambda_A V_A^{-1}. \tag{4.43}$$

Here, $\Lambda_A = \mathrm{diag}(\lambda_1, \ \cdots, \ , \lambda_{|n_A|})$, where we assume that the eigenvalues are non-negative and sorted in ascending order:

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{|n_A|}.$$

Spectral AMGe uses the low energy modes of the *local* discretization matrix $K_A(n_A, n_A)$ to characterize the low energy modes of the *global* discretization matrix K. More precisely, for each agglomerate A, the eigenvectors of $K_A(n_A, n_A)$ corresponding to the m_A eigenvalues with the *smallest* magnitude will be used to build m_A columns of the prolongation operator. The columns of the prolongation operator are constructed as:

$$P(n_A, \cdot) \quad = \quad W_A V_A(:, 1{:}m_A). \tag{4.44}$$

The matrix $W_A \in \mathbb{R}^{|n_A| \times |n_A|}$ is a diagonal matrix defined as follows, with $n_i = n_A(i), \ 1 \leq i \leq |n_A|$:

$$W_A(i,i) = w_A(n_i) \quad \text{with} \quad w_A(n_i) = \frac{K_A(n_i, n_i)}{K(n_i, n_i)}. \tag{4.45}$$

Note that $w_A(n_i)$ constitutes a partition of unity weighing function:

$$\forall n_i \quad : \quad \sum_A w_A(n_i) = 1, \tag{4.46}$$

and satisfies:

$$w_A(n_i) \qquad \begin{cases} = 0 & \text{if} \quad n_i \notin A, \\ = 1 & \text{if} \quad n_i \in A \setminus \delta A, \\ \in [0 \ 1] & \text{if} \quad n_i \in \delta A. \end{cases} \tag{4.47}$$

Correspondingly, $w_A(n)$ can be interpreted as the strength of connection of a node n to an agglomerate A.

AMGe based on element agglomeration

In [42], a prolongator is constructed which matches the coarse topology which is implied by an element agglomeration. The coarse faces, edges and nodes of this topology are derived by the following principles:

1. Coarse faces are obtained as the intersections of element agglomerates. Let I_a denote the set of the intersections of any pair of element agglomerates. Each member of I_a which is not a subset of any other member of I_a, is a coarse face.

2. Coarse edges are obtained as the intersections of coarse faces. Let I_f denote the set of the intersections of any pair of coarse faces. Each member of I_f which is not a subset of any other member of I_f, is a coarse edge.

3. The coarse nodes are the intersections of any pair of coarse edges.

This coarse topology determines the nonzero pattern of the prolongator. More precisely, each coarse node is only prolongated from itself, and each fine node belonging to a coarse edge, face or element S is prolongated only from the coarse nodes belonging to S (see figure 4.3).

The entries of the prolongator are determined in four steps. First, the prolongation is set to the identity at the coarse nodes i.e., the prolongator satisfies (4.14). Next, the prolongation weights are determined for each fine node n belonging to a coarse edge E. For this purpose, we consider a local representation of the operator in the union of all element agglomerates to which n belongs, i.e.

$$\Omega(n) \quad := \quad \{A \mid n \in A\}.$$

This local representation is obtained as the sum of the element matrices of all elements belonging to $\Omega(n)$. We discard the zero rows and columns, and we consider the following block representation of the operator in $\Omega(n)$:

$$K_{\Omega(n)} \quad := \quad \sum_{\alpha \in \Omega(n)} K_\alpha \quad = \quad \begin{bmatrix} K_{ii} & K_{ib} \\ K_{bi} & K_{bb} \end{bmatrix}. \tag{4.48}$$

Here, b denotes the nodes belonging to the boundary ∂E of E. Note that all nonzero entries in $P(b,:)$ were determined in the previous step. Let the first row of $K_{\Omega(n)}$ correspond to n. The prolongation weights for n are determined by the following formula, which is a variant of the AMGe principle (4.40):

$$P(n,:) \quad = \quad \left(-K_{ii}^{-1} K_{ib} P(b,:)\right)(1,:). \tag{4.49}$$

Similarly, the prolongation weights for each fine node belonging to a coarse face F and volume V are derived from $P(\partial F,:)$ and $P(\partial V,:)$.

4.3.7 Adaptive AMG

Classical AMG is based on the assumption that algebraically smooth errors are geometrically smooth, i.e. varying slowly along directions of strong connections (see section 4.1.2). While this typically holds for discretizations of scalar elliptic PDEs, it is not true in general.

Adaptive AMG algorithms don't make an apriori assumption about algebraic smoothness. Instead, they determine the characteristics of algebraically smooth vectors themselves. For this purpose, the smoother is applied to the homogeneous problem $Kx = 0$, with a random initial guess. The resulting vector is used to build the prolongation operator. We refer the reader to [17] and [16] for further details.

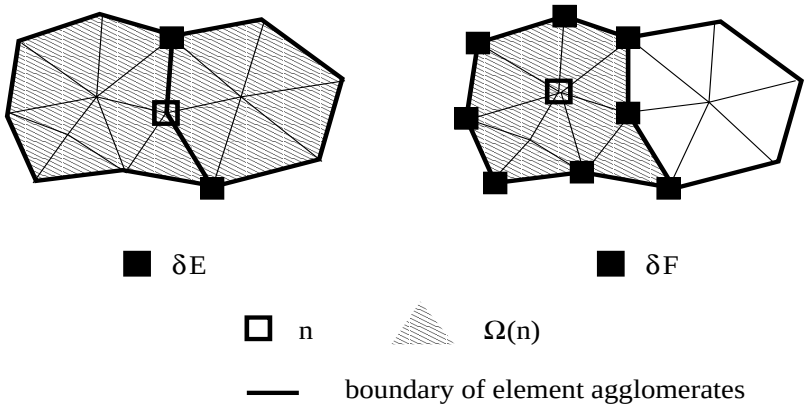


Figure 4.3: The prolongation pattern for a node n in element agglomeration AMGe depends on the local representation of the differential operator in $\Omega(n)$. It is derived from the prolongation patterns for the nodes belonging to ∂E if n belongs to a coarse edge E (left picture) or to ∂F if n belongs to a coarse face F (right picture).

4.4 Galerkin coarse operator

In an AMG context, the restriction operator R is usually chosen equal to the transpose of the prolongator P, and the coarse grid operator K_c is constructed as the Galerkin product of the fine grid operator K and the prolongator:

$$R = P^T \quad \text{and} \quad K_c = P^T K P. \tag{4.50}$$

These choices are optimal in the following sense. If the coarse system

$$K_c x_c \;=\; P^T(b - Kx)$$

is solved exactly, then the coarse grid correction

$$x \;\leftarrow\; x + Px_c$$

minimizes the K-norm of the error. Note, for this purpose, that from one iteration to the next, the error e is transformed as follows:

$$e \;\leftarrow\; e - Px_c.$$

The square of the K-norm of the error after the coarse grid correction is given by:

$$\begin{aligned} \|e - Px_c\|_K^2 &= (e - Px_c)^T K(e - Kx_c) \\ &= e^T Ke - 2x_c^T P^T Ke + x_c^T P^T KPx_c. \end{aligned}$$

This error is minimized when its gradient vanishes. This is, when:

$$P^T KPx_c = P^T Ke = P^T(b - Kx) \iff K_c x_c = P^T(b - Kx).$$

In other words, the optimal coarse grid correction x_c is obtained as the solution to the Galerkin coarse grid problem $K_c x_c = P^T(b - Kx)$. An immediate consequence of this is that an error belonging to the range of P is eliminated completely by the coarse grid correction:

$$e = Px_c \quad \Longrightarrow \quad \min_{x_c} \|e - Px_c\|_K = \min_{x_c} \|Px - Pe_c\|_K = 0.$$

4.5 Concluding remarks

Several highly efficient implementations of the AMG methods presented in this chapter are publicly available. For classical AMG, we refer to SAMG [61] and HYPRE [49]. For smoothed aggregation AMG, we refer to ML [28].

Chapter 5

Overview of multigrid for the curl-curl equation

Multigrid algorithms for node-based problems (see chapters 3 and 4) are not successful when applied to discretizations of the curl-curl equation:

$$\text{curl}(\alpha\,\text{curl}(\vec{A})) + \beta\vec{A} \quad = \quad \vec{b}. \tag{5.1}$$

More involved components will be needed both for the smoothing and prolongation operations.

In this chapter, we will present an overview of existing multigrid methods for the curl-curl equation. We will consider discretizations which are, up to the boundary conditions, of the following form (see section 2.5.4):

$$K_e x := \left(K_\alpha^{cc} + M_\beta^e\right) x = b, \quad \text{with} \quad K_\alpha^{cc} := C^T M_\alpha^f C. \tag{5.2}$$

In section 5.1, the basic smoothing and prolongation principles to be used for the curl-curl equation will be explained in the context of geometric multigrid. Next, we will present existing AMG methods for the curl-curl equation. They fall apart in two classes. In the first class of methods, each edge prolongator is derived from a nodal prolongator in such a way that the so-called node-edge commutation property is satisfied. This gives rise to a parallel multigrid hierarchy, with a node-based and an edge-based branch which are tightly coupled. In the second class of methods, a parallel multigrid hierarchy with decoupled branches is used. These methods will be presented in section 5.3.

5.1 Geometric multigrid

In this section, we will present the basic principles of geometric multigrid for problems in $H(\text{curl}, \Omega)$. Similarly as for problems in $H^1(\Omega)$, a classical multigrid cycle can be defined. However, other smoothers and prolongators will be needed.

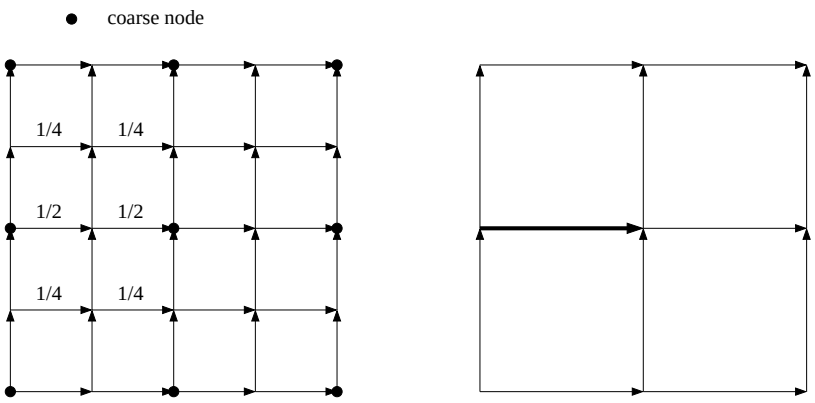


Figure 5.1: Regular 2D quadrilateral mesh (left) for the curl-curl problem, with the associated coarse mesh (right). The values indicated on the fine mesh are the prolongation values corresponding to the stencil (5.6) for the bold coarse edge in the right picture.

We consider the finite element discretization using edge elements of (5.1) with constant material parameters α and β on a regular 2D quadrilateral grid with grid size h, as depicted in figure 5.1. The corresponding stencils of the stiffness and mass part read for the horizontal edges:

$$\frac{\alpha}{h^2}\begin{bmatrix} \circ & -1 & \circ \\ -1 & & 1 \\ \bullet & 2 & \bullet \\ 1 & & -1 \\ \circ & -1 & \circ \end{bmatrix} + \beta \begin{bmatrix} \circ & \frac{1}{6} & \circ \\ 0 & & 0 \\ \bullet & \frac{2}{3} & \bullet \\ 0 & & 0 \\ \circ & \frac{1}{6} & \circ \end{bmatrix}, \tag{5.3}$$

and for the vertical edges:

$$\frac{\alpha}{h^2}\begin{bmatrix} \circ & -1 & \bullet & 1 & \circ \\ -1 & & 2 & & -1 \\ \circ & 1 & \bullet & -1 & \circ \end{bmatrix} + \beta \begin{bmatrix} \circ & 0 & \bullet & 0 & \circ \\ \frac{1}{6} & & \frac{2}{3} & & \frac{1}{6} \\ \circ & 0 & \bullet & 0 & \circ \end{bmatrix}. \tag{5.4}$$

5.1.1 Prolongation

Similarly as for multigrid in $H^1(\Omega)$, the prototype edge prolongation scheme is obtained by using coarse edge elements which are *isomorphic* to the fine edge elements. The entries of the corresponding edge prolongator P_e are calculated using (2.56) as the line integrals of the coarse edge element along the fine edges:

$$\vec{E}_{IJ} = \sum_{rs} P_e(rs, IJ)\vec{E}_{rs} \quad \Longrightarrow \quad \int_{e_{rs}} \vec{E}_{IJ} \cdot d\vec{l} = P_e(rs, IJ). \tag{5.5}$$

For the 2D regular quadrilateral grid depicted in figure 5.1, this yields the following prolongation stencils for horizontal and vertical coarse edges:

$$\begin{bmatrix} \cdot & 1/4 & \cdot & 1/4 & \cdot \\ 0 & & 0 & & 0 \\ \bullet & 1/2 & \cdot & 1/2 & \bullet \\ 0 & & 0 & & 0 \\ \cdot & 1/4 & \cdot & 1/4 & \cdot \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \cdot & 0 & \bullet & 0 & \cdot \\ 1/4 & & 1/2 & & 1/4 \\ \cdot & 0 & \cdot & 0 & \cdot \\ 1/4 & & 1/2 & & 1/4 \\ \cdot & 0 & \bullet & 0 & \cdot \end{bmatrix}. \tag{5.6}$$

Here, the fine and coarse nodes are indicated by $\cdot$ and $\bullet$ respectively. A geometric multigrid for non-nested grids has been developed in [23].

Edge prolongation has a different character than nodal prolongation. First, edges have a direction and an orientation, contrary to nodes. They are taken into account in the direction and orientation of the integration domains of the integrals (5.5). As a consequence, the sign of an entry of P_e flips if the direction of the fine or coarse edge involved is changed. In order to reduce the notational complexity in the theoretical calculations in the rest of this work, the orientation of an edge will be linked to the sign of its index, as follows:

$$P_e(:,\ -I) = -P_e(:,\ I), \qquad P_e(-i,\ :\ ) = -P_e(i,\ :\ ). \tag{5.7}$$

Note, however, that in practical calculations, each edge is represented by only one of its two possible orientations, and that this orientation does not change during the calculations. Another difference with nodal prolongation operators is that an edge prolongator P_e does not satisfy the partition of unity property (4.22).

In the same way as in (5.5), a face prolongator P_f and a volume prolongator P_v can be constructed corresponding to isomorphic coarse face and volume elements. The discrete gradient, curl and divergence operators for the coarse grid can be defined as in section 2.5.2. This allows the discrete de Rham sequence (2.99) to be transferred to the coarse level as follows:

$$\begin{array}{ccccccccc} \mathbb{R} & \xrightarrow{\text{cst}} & H^1(\Omega) & \xrightarrow{\text{grad}} & H(\text{curl},\Omega) & \xrightarrow{\text{curl}} & H(\text{div},\Omega) & \xrightarrow{\text{div}} & L^2(\Omega) \\ & & \uparrow \Pi_N & & \uparrow \Pi_E & & \uparrow \Pi_F & & \uparrow \Pi_V \\ \mathbb{R} & \xrightarrow{\mathbf{1}_f} & \mathcal{N}_f & \xrightarrow{G_f} & \mathcal{E}_f & \xrightarrow{C_f} & \mathcal{F}_f & \xrightarrow{D_f} & \mathcal{V}_f \\ & & \uparrow P_n & & \uparrow P_e & & \uparrow P_f & & \uparrow P_v \\ \mathbb{R} & \xrightarrow{\mathbf{1}_c} & \mathcal{N}_c & \xrightarrow{G_c} & \mathcal{E}_c & \xrightarrow{C_c} & \mathcal{F}_c & \xrightarrow{D_c} & \mathcal{V}_c. \end{array} \tag{5.8}$$

The coarse interpolation operators Π_{N_c}, Π_{E_c}, Π_{F_c} and Π_{V_c}, which map the vector spaces $\mathcal{N}_c$, $\mathcal{E}_c$, $\mathcal{F}_c$ and $\mathcal{V}_c$ onto $\mathcal{S}_N$, $\mathcal{S}_E$, $\mathcal{S}_F$ and $\mathcal{S}_V$, can be extracted from (5.8):

$$\begin{aligned} \Pi_{N_c} &:= \Pi_N \circ P_n, \\ \Pi_{E_c} &:= \Pi_E \circ P_e, \\ \Pi_{F_c} &:= \Pi_F \circ P_f, \\ \Pi_{V_c} &:= \Pi_V \circ P_v. \end{aligned}$$

Since the coarse Whitney elements are isomorphic to the fine Whitney elements, the coarse interpolation operators satisfy commutation properties similar to (2.95), (2.96), (2.97) and (2.98):

$$\begin{aligned} \text{cst} &= \Pi_{N_c} \circ \mathbf{1}_c, \\ \text{grad} \circ \Pi_{N_c} &= \Pi_{E_c} \circ G_c, \\ \text{curl} \circ \Pi_{E_c} &= \Pi_{F_c} \circ C_c, \\ \text{div} \circ \Pi_{F_c} &= \Pi_{V_c} \circ D_c. \end{aligned}$$

This allows to extract the following commutation properties for the prolongation operators and the discrete differential operators:

$$G_f P_n = P_e G_c, \tag{5.9}$$
$$C_f P_e = P_f C_c, \tag{5.10}$$
$$D_f P_f = P_v D_c. \tag{5.11}$$

For instance, (5.9) follows from:

$$\begin{aligned} \forall \alpha \in \mathcal{N}_c \quad : \quad \Pi_E \left(G_f P_n \alpha \right) &= \text{grad} \left(\Pi_N \left(P_n \alpha \right) \right) \\ &= \text{grad} \left(\Pi_{N_c} \alpha \right) \\ &= \Pi_{E_c} \left(G_c \alpha \right) \\ &= \Pi_E \left(P_e G_c \alpha \right). \end{aligned}$$

The node-edge commutation property (5.9) has been the key to many AMG methods for the curl-curl equation. It implies that the gradient subspaces of $\mathcal{E}_f$ and $\mathcal{E}_c$ are nested:

$$\text{ran} \left(P_e G_c \right) \subset \text{ran}(G_f).$$

Also, the node-edge commutation property is required to obtain an edge restriction that makes sense from a physical point of view. More precisely, consider a vector j_f representing a discretely divergence-free current, i.e. $G_f^T j_f = 0$. The edge-node commutation property (5.9) is needed to preserve the solenoidal character of j_f when restricted to the coarse level, i.e.:

$$G_c^T j_c = G_c^T P_e^T j_f = P_n^T G_f^T j_f = 0.$$

5.1.2 Smoothing

Classical Gauss-Seidel smoothing

The straightforward application of a multigrid cycle to discretizations of the curl-curl equation gives poor results. Typically, the convergence stagnates after a few iterations. This is illustrated in Table 5.1 with a numerical experiment. A significant reduction of the norm of the residual is achieved in the first few iterations. After that, the convergence stagnates. At this point, the convergence

iteration	$\|b - Kx\|/\|b\|$	ρ
1	1.05e-02	0.010
2	1.55e-03	0.148
3	2.67e-04	0.172
4	5.59e-05	0.230
5	1.52e-05	0.272
6	8.46e-06	0.556
7	1.62e-05	1.911
8	4.41e-05	2.727
9	5.32e-05	1.209
10	2.12e-05	0.395

Table 5.1: Typical convergence history of a multigrid cycle for the curl-curl equation using classical Gauss-Seidel smoothing. For this example, a V(2,2) AMG-cycle using the edge prolongator (7.6) and Gauss-Seidel smoothing is applied to the finite element discretization of the 2D curl-curl equation (5.1) with $\alpha = \beta = 1$ on an irregular triangular mesh containing 124741 edges.

factor is not always smaller than 1 any more. This problem can be traced back to the different behavior of classical smoothers for (5.2) in both parts of the discrete Helmholtz splitting (2.104).

In ran(C^T), classical smoothing satisfactorily reduces the high frequency components. This can be related to the fact that in ran(curl) and up to variations of the material parameter α, the curl-curl operator behaves like a vector Laplacian:

$$\begin{aligned} \operatorname{curl}\left(\alpha \operatorname{curl}(\vec{A})\right) &= -\alpha \begin{pmatrix} \operatorname{div}(\operatorname{grad}(A_x)) \\ \operatorname{div}(\operatorname{grad}(A_y)) \\ \operatorname{div}(\operatorname{grad}(A_z)) \end{pmatrix} \\ &+ \alpha \operatorname{grad}(\operatorname{div}(\vec{A})) \\ &+ \operatorname{grad}(\alpha) \times \operatorname{curl}(\vec{A}). \end{aligned} \tag{5.12}$$

Since the vector Laplacian is an elliptic operator, classical smoothing is expected to perform well. This will be confirmed by the local Fourier analysis of chapter 6.

In ran(G), however, the smoothing performance of classical smoothers is poor for practical values of α and β, and for sufficiently fine meshes. This will be illustrated for Gauss-Seidel smoothing. The error equation for the basic step (4.1) of Gauss-Seidel smoothing reads for an error $e^{[\nu]}$ belonging to ran(G):

$$e_r^{[\nu+1]} \leftarrow e_r^{[\nu]} - \frac{\sum_c K_e(r,c) e_c^{[\nu]}}{K_e(r,r)} = e_r^{[\nu]} - \frac{\sum_c M_\beta(r,c) e_c^{[\nu]}}{K_e(r,r)}. \tag{5.13}$$

Here, we have taken into account that G belongs to the kernel of the stiffness part K_α^{cc} of $K_e = K_\alpha^{cc} + M_\beta$. For the considered problem, the diagonal elements

of K_α^{cc} and M_β are related as follows:

$$K_\alpha^{cc}(r,r) \quad = \quad \frac{3\alpha}{\beta h^2} M_\beta(r,r),$$

and, hence

$$K_e(r,r) \quad = \quad \left(1 + \frac{3\alpha}{\beta h^2}\right) M_\beta(r,r). \tag{5.14}$$

This shows that (5.13) is equivalent to a step of a *damped* Gauss-Seidel iteration with a damping factor

$$d \quad = \quad \left(1 + \frac{3\alpha}{\beta h^2}\right)^{-1}$$

applied to $M_\beta x = b$. Consequently, the error remains essentially untouched for large values of $\frac{3\alpha}{\beta h^2}$.

To summarize, the poor smoothing performance of classical smoothers in $\mathrm{ran}(G)$ is essentially caused by the fact that gradient vectors belong to the kernel of K_α^{cc}, and by a size mismatch between the entries of K_α^{cc} and M_β. This size mismatch, which is expressed by (5.14) in this example, is generic and corresponds to the fact that K_α^{cc} and M_β are a discrete *second* order and a discrete *zero* order differential operator respectively. In general terms, this amounts to

$$\|K_\alpha^{cc}\| \quad \sim \quad \frac{1}{h^2} \|M_\beta^e\|. \tag{5.15}$$

In order to obtain satisfactory smoothing in $\mathrm{ran}(G)$, more involved smoothers are needed. We will discuss the hybrid smoother, presented by Hiptmair in [35] and an overlapping block smoother, presented by Arnold, Falk and Winther in [3]. A local Fourier analysis of these smoothers will be derived in chapter 6.

Hybrid smoother

Consider the discrete Helmholtz decomposition (2.106) of the error of (5.2):

$$e \quad = \quad e_g + e_s \quad = \quad G e_n + C^T e_f. \tag{5.16}$$

In order to smooth the gradient component e_g of the error, the so-called hybrid smoother of [35] performs an additional smoothing step on an auxiliary diffusion system:

$$K_n x_n \quad = \quad b_n. \tag{5.17}$$

Typically, the matrix of (5.17) is constructed as the Galerkin product of K_e with the discrete gradient G:

$$K_n \quad := \quad G^T K_e G \quad = \quad G^T M_\beta^e G. \tag{5.18}$$

Up to the boundary conditions, K_n is formally equivalent to a discretization of the diffusion operator $\mathrm{div}(\beta\,\mathrm{grad}(\cdot))$ (cfr. (2.122)). Note that the dominant part K_α^{cc} of K_e, which prevented the proper treatment of e_g by classical smoothing, does not contribute to K_n.

The righthand side of (5.17) is constructed as the restriction of the residual by G, with $\tilde{x}$ denoting the approximate solution of (5.2):

$$b_n \;:=\; G^T(b - K_e\tilde{x}) \;=\; G^T(b - M_\beta^e\tilde{x}). \tag{5.19}$$

The approximate solution $\tilde{x}_n$ of (5.17) is used to update $\tilde{x}$:

$$\tilde{x} \leftarrow \tilde{x} + G\tilde{x}_n, \quad \text{or,} \quad e_g \leftarrow e_g - G\tilde{x}_n. \tag{5.20}$$

For a rigorous and quantitative analysis of the hybrid smoother, we refer to [35] and to section 6.6.2. Here, we will explain this smoother in a more intuitive, qualitative way.

Obviously, the high frequency components of e_g are only reduced by (5.20) if they are approximately equal to the high frequency components of the correction $G\tilde{x}_n$. The conditions for this requirement to hold can be studied by considering the error equation for (5.2), multiplied from the left with G^T:

$$\begin{aligned} G^T K_e e = G^T(b - K_e\tilde{x}) \quad &\Longleftrightarrow \quad G^T M_\beta\,(Ge_n + e_s) = b_n \\ &\Longleftrightarrow \quad K_n e_n = b_n - G^T M_\beta e_s. \end{aligned} \tag{5.21}$$

This shows that the auxiliary system (5.17) corresponds to (5.21) up to an aliasing contribution from the solenoidal part of the error. Since we are only interested in the high frequency behavior of (5.21), the high frequency part of e_s should be sufficiently small. In other words, the solenoidal error should be sufficiently smooth. This is achieved by classical smoothing on (5.2).

Typically, the hybrid smoother is applied according to the following scheme:

1. ϵ_{pre} presmoothing steps in $\mathcal{E}$ on $K_e\tilde{x} = b$,
2. restriction to $\mathcal{N}$: $b_n = G^T(b - K\tilde{x})$,
3. ν smoothing steps in $\mathcal{N}$ on $K_n\tilde{x}_n = b_n$ with zero initial guess,
4. prolongation to $\mathcal{E}$: $\tilde{x} \leftarrow \tilde{x} + G\tilde{x}_n$,
5. ϵ_{post} postsmoothing steps in $\mathcal{E}$ on $K_e\tilde{x} = b$.

This is formally equivalent to a 2-level multigrid cycle using the gradient matrix G as the prolongation matrix. Different configurations of the hybrid smoother are possible. They depend on the number of pre- and postsmoothing steps for the edge smoother in steps 1 and 5 and on the configuration of the nodal smoother in step 3. For classical smoothers of Jacobi or Gauss-Seidel type, the configuration of the nodal smoother can be summarized as follows:

$$\text{nodal smoother} \;=\; \nu \times \begin{cases} 1. & \nu_{pre} \text{ iterations in presmoothing mode,} \\ 2. & \nu_{post} \text{ iterations in postsmoothing mode.} \end{cases}$$

The pre- and postsmoothing modes are identical for Jacobi and polynomial smoothers. For Gauss-Seidel smoothers, they differ only in the order in which the unknowns are updated: the updating order in postsmoothing mode is the reverse of the updating order in presmoothing mode. Popular configurations for the hybrid smoother are the so-called half and full hybrid smoother (see for instance [28]). The full hybrid smoother has $\epsilon_{pre} = \epsilon_{post} = \nu = \nu_{pre} = \nu_{post} = 1$. The half hybrid smoother has $\epsilon_{pre} = \nu = \nu_{pre} = 1$ and $\epsilon_{post} = \nu_{post} = 0$ in presmoothing mode, and $\epsilon_{pre} = \nu_{pre} = 0$ and $\epsilon_{post} = \nu = \nu_{post} = 1$ in postsmoothing mode.

Remark 5.1.1 The nodal part of the hybrid smoother has a physical meaning. Note, for this purpose, that the product of G^T and (5.2)

$$G^T K_e x \quad = \quad G^T M_\beta^e x \quad = \quad G^T b$$

corresponds to the charge conservation equation (2.15). Comparison with (5.17)-(5.18)-(5.19) shows that steps 2-3-4 of the hybrid smoother approximately restore the conservation of charge by means of gradient corrections to x.

Overlapping block smoother (AFW)

In [3], an appropriate smoothing in both ran(G) and ran(C^T) is obtained by using an overlapping blocksmoother. In the jth step of this smoother, the set E_j of the four edges containing the node j (see figure 5.2) are updated simultaneously:

$$x(E_j) \quad \leftarrow \quad x(E_j) + K_e(E_j, E_j)^{-1} \left(b(E_j) - K(E_j, :)x \right). \tag{5.22}$$

Note that each edge is updated twice, since each edge contains two nodes. Consequently, the blocksmoother is overlapping.

The support of each local solve corresponds to the nonzero pattern of one column of G:

$$G(i,j) \neq 0 \quad \Longleftrightarrow \quad i \in E_j.$$

Consequently, the local block solve (5.22) can generate corrections in ran($G(:,j)$). This is the key to the success of this smoother. For a rigorous and quantitative analysis, we refer to [3] and to section 6.6.3. Here, we will follow a more intuitive approach.

Recall that the poor smoothing performance of classical smoothers in ran(G) is essentially caused by the fact that ran(G) belongs to the kernel of the dominant part K_α^{cc} of the discrete operator K_e (cfr. (5.13)-(5.14)). As a consequence, there is a mismatch between the numerator and denominator in the update step (5.13) for gradient errors, which prevents gradient errors to be corrected in an efficient way. In the blocksmoother, this is automatically compensated for by the different orders of magnitude of the eigenvalues of $K_e(E_j, E_j)$. These eigenvalues play the same role as the denominator in (5.13). Intuitively, blocksmoothing amounts in this case to using different appropriate denominators in (5.13) for

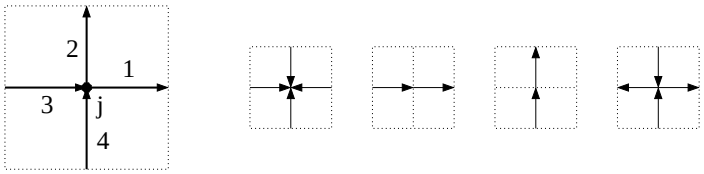


Figure 5.2: In the basic step (5.22) of the overlapping blocksmoother of [3] for the node j, the edges 1-2-3-4 are updated simultaneously. The patterns corresponding to the eigenvectors (5.23) of the diagonal block $K_e(E_j, E_j)$, with $E_j = [1\ 2\ 3\ 4]$, are depicted on the right.

gradient and solenoidal error components. For the choice of the edge indices depicted in figure 5.2, $K_e(E_j, E_j)$ reads, with $\epsilon = \frac{2\beta h^2}{3\alpha}$:

$$K_e(E_j, E_j) = \begin{bmatrix} 2+\epsilon & -1 & 0 & 1 \\ -1 & 2+\epsilon & 1 & 0 \\ 0 & 1 & 2+\epsilon & -1 \\ 1 & 0 & -1 & 2+\epsilon \end{bmatrix}.$$

The eigenvalue decomposition $K_e(E_j, E_j) = V\Lambda V^{-1}$, with

$$V = \begin{bmatrix} -1 & 1 & 0 & 1 \\ -1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \epsilon & 0 & 0 & 0 \\ 0 & 2+\epsilon & 0 & 0 \\ 0 & 0 & 2+\epsilon & 0 \\ 0 & 0 & 0 & 4+\epsilon \end{bmatrix}, \quad (5.23)$$

shows that the first eigenvalue, which corresponds to the gradient pattern of $G(:, j)$ (see figure 5.2), has a different order of magnitude than the other eigenvalues.

5.2 AMG methods satisfying the commutation property

A possible strategy to construct AMG prolongation operators, is to mimic the basic principles of their geometric counterparts. These principles, which were presented in section 5.1.1, can be summarized as follows. In geometric multigrid, a coarse grid and the corresponding edge elements are known apriori. These coarse edge elements are isomorphic to the fine edge elements and they satisfy similar properties. A crucial property is the fact that their range space is part of a coarse de Rham sequence. The node-edge commutation property (5.9) is a consequence of the coupling of this coarse de Rham sequence with the de Rham sequence on the fine level.

In algebraic multigrid methods, however, the coarse grid and the coarse edge elements are not known apriori, but they have to be constructed. Consequently,

one of the challenges is to construct coarse edge elements which satisfy similar properties as the fine edge elements, as in geometric multigrid. In the AMG methods which will be presented in this section, this is achieved by using the node-edge commutation property as a construction guideline. More precisely, each edge prolongator will be derived from a nodal prolongator, in such a way that the node-edge commutation property is satisfied.

First, we will discuss the AMG method developed by Reitzinger and Schöberl in [56]. This method was the first successful AMG method satisfying the node-edge commutation property. The edge prolongators used in this method can be considered piecewise constant. Several variants of this method have been developed, in which the approximation quality (4.15) of the edge prolongators is improved. An overview of these variants will be given. Finally, we will show how the derivation of an edge prolongator from a nodal prolongator gives rise to a parallel multigrid hierarchy, with a node-based and an edge-based branch which are tightly coupled.

5.2.1 Piecewise constant edge prolongation

In this section, we will discuss the first successful edge prolongator satisfying the commutation property (5.9). This edge prolongator was developed by Reitzinger and Schöberl in [56]. It is derived from a node aggregation and its associated piecewise constant nodal prolongator P_n (4.28).

Coarse gradient

First, a coarse topology is extracted from P_n. For this purpose, the coarse aggregates are defined as the nodes of the coarse topology. Two coarse nodes n_I and n_J are said to be connected by a coarse edge e_{IJ}, if there is a fine edge connecting a fine node of n_I to a fine node of n_J (see figure 5.3):

$$\forall I \neq J,\ \exists \pm e_{IJ} \quad \Longleftrightarrow \quad \exists \pm e_{kl} \ :\ P_n(k,I) \ = \ P_n(l,J) \ = \ 1. \tag{5.24}$$

The orientation of the coarse edges can be chosen arbitrarily.

This allows a gradient operator G_c to be defined which describes the coarse edges. Similarly as the fine level discrete gradient (2.76), the rows and columns of G_c correspond to the coarse edges and nodes respectively. The row for the coarse edge e_{IJ} is zero everywhere, except for its start and end nodes n_I and n_J, where it is -1 and 1 respectively:

$$\forall \text{ coarse edge } \ e_{IJ} \ \ : \ \ G_c(e_{IJ}, n_I) := -1 \ \text{ and } \ G_c(e_{IJ}, n_J) := 1. \tag{5.25}$$

Edge prolongator

The edge prolongator P_e proposed in [56] is constructed as follows. The row of P_e for a fine edge e_{ij} connecting the nodes n_i and n_j contains no nonzero entries if n_i and n_j belong to the same aggregate. If they belong to different aggregates n_I and n_J, then the coarse edge $\pm e_{IJ}$ exists, according to (5.24). In this case,

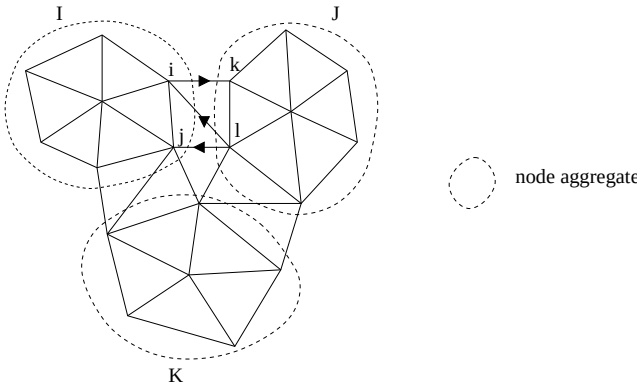


Figure 5.3: Identification of coarse edges implied by a node aggregation. Since there are connections between the fine nodes belonging to the aggregates I and J, they are connected by a coarse edge e_{IJ}. The prolongation values of the piecewise constant edge prolongator (5.26) for the coarse edge $e_{IJ} = n_I \rightarrow n_J$ are $P_e(e_{ik}, e_{IJ}) = -P_e(e_{li}, e_{IJ}) = -P_e(e_{lj}, e_{IJ}) = 1$.

e_{ij} is prolongated only from e_{IJ}. The corresponding entry $P_e(e_{ij}, e_{IJ})$ equals ± 1, depending on the relative orientation of e_{ij} and e_{IJ}:

$$P_e(kl, IJ) = \begin{cases} 1 & \text{if} \quad P_n(k, I) = 1 = P_n(l, J), \\ -1 & \text{if} \quad P_n(k, J) = 1 = P_n(l, I), \\ 0 & \text{else.} \end{cases} \tag{5.26}$$

In [56], it is proved that the coarse gradient G_c and the edge prolongator P_e satisfy the node-edge commutation property.

Theorem 5.2.1 *A piecewise constant nodal prolongator P_n (4.28), the edge prolongator P_e constructed as (5.26), the fine gradient G_f and the coarse gradient G_c constructed as (5.25) satisfy the node-edge commutation property (5.9).*

Proof. The coarse node to which a fine node n_i belongs, will be denoted using a capital subscript as n_I. An entry of the lefthand side of (5.9) reads:

$$\begin{aligned} \left(G_f P_n\right)(e_{ij}, n_K) &= P_n(n_j, n_K) - P_n(n_i, n_K) \\ &= \begin{cases} 0 & \text{if} \quad n_I = n_J \text{ or } \; n_I \neq n_K \neq n_J, \\ 1 & \text{if} \quad n_I \neq n_J = n_K, \\ -1 & \text{if} \quad n_J \neq n_I = n_K. \end{cases} \end{aligned}$$

The corresponding righthand side of (5.9) reads, using (5.7) and (5.26):

$$\left(P_e G_c\right)(e_{ij}, n_K) = \begin{cases} 0 & \text{if} \quad n_I = n_J, \\ G_c(e_{IJ}, n_K) & \text{if} \quad n_I \neq n_J. \end{cases}$$

The equality of the lefthand and righthand side is established by taking into account (5.25).
□

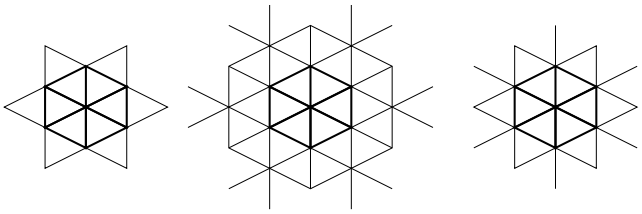


Figure 5.4: Effect on the support of coarse edge elements of the smoothing procedures. Left: smoothing as in (5.27); middle: smoothing as in (5.28) with unlumped M_β^e; right: smoothing as in (5.28) with lumped M_β^e. The original support of the coarse edge element is indicated by the bold edges.

5.2.2 Smoothed aggregation

Similarly as in section 4.3.4 for nodal prolongators, the approximation quality (4.15) of an edge prolongator can be improved by smoothing its columns. This approach has been developed by Bochev e.a. in [5] and by Hu e.a. in [39].

Let P_n, P_e, G_f and G_c denote a nodal and edge prolongator and a fine and coarse discrete gradient operator for which the node-edge commutation property (5.9) holds. For this purpose, the piecewise prolongators of section 5.2.1 are typically taken as a starting point.

Smoothing in edge space

Let P_e be smoothed by a damped Jacobi iteration with respect to the stiffness part K_α^{cc}, with D denoting the diagonal part of K_α^{cc} and d_e the damping factor:

$$\hat{P}_e \;=\; \left(I - (d_e D)^{-1} K_\alpha^{cc}\right) P_e. \tag{5.27}$$

Then, the commutation property (5.9) still holds (see [5]):

$$\begin{aligned} \hat{P}_e G_c &= \left(I - (d_e D)^{-1} K_\alpha^{cc}\right) P_e G_c \\ &= \left(I - (d_e D)^{-1} K_\alpha^{cc}\right) G_f P_n \;=\; G_f P_n. \end{aligned}$$

Here, we have taken into account that $K_\alpha^{cc} G_f \equiv 0$. The smoothing causes a growth of the support of the coarse edge elements, represented by the columns of P_e. That support is indicated in the left picture of figure 5.4.

In practice, P_e is often smoothed with respect to the total discretization matrix K_e instead of the stiffness part K_α^{cc}. In this case, the commutation property still holds approximately, thanks to the fact that K_α^{cc} is typically the dominant part of K_e (cfr. (5.15)).

Smoothing in nodal space

Next, we consider the case where P_n is smoothed by a damped Jacobi iteration with respect to the auxiliary matrix (5.18), with D_n denoting the diagonal part

of K_n:

$$\hat{P}_n \quad = \quad \left(I - (d_n D_n)^{-1} K_n\right) P_n.$$

In this case, the commutation can be restored by applying the following smoothing procedure to P_e (see [39]):

$$\hat{P}_e \quad = \quad \left(I - G_f (d_n D_n)^{-1} G_f^T K_e\right) P_e. \tag{5.28}$$

Since $K_n = G_f^T K_e G_f$, the commutation property reads:

$$\begin{aligned} \hat{P}_e G_c &= \left(I - G_f (d_n D_n)^{-1} G_f^T K_e\right) P_e G_c \\ &= \left(I - G_f (d_n D_n)^{-1} G_f^T K_e\right) G_f P_n \\ &= G_f \left(I - (d_n D_n)^{-1} K_n\right) P_n \quad = \quad G_f \hat{P}_n. \end{aligned}$$

In order to limit the growth of the support of the coarse edge elements, a lumped version of M_β^e can be used (see figure 5.4).

5.2.3 Constrained solution of the commutation equality

Let a partition of unity nodal prolongator P_n and a fine and coarse discrete gradient operator G_f and G_c be given. If additionally the nonzero pattern of the edge prolongator P_e is fixed, the commutation property (5.9) gives rise to an underdetermined linear system for each fine edge. Consider the fine edge e_{ij}, and let E denote the set of indices of the nonzero locations of $P_e(e_{ij}, :)$. Let N denote the set of indices of the nonzero columns of $G_c(E, :)$. The nonzero entries of $P_e(e_{ij}, E)$ satisfy:

$$Ax = b \quad \text{with} \quad \left\{ \begin{array}{lcl} x^T & = & P_e(e_{ij}, E), \\ A^T & = & G_c(E, N), \\ b^T & = & (G_f P_n)(e_{ij}, N). \end{array} \right. \tag{5.29}$$

In [5], the nonzero pattern of P_e is restricted such that all systems (5.29) have a unique solution which can be determined analytically. We refer the reader to [5] for more details.

5.2.4 Energy minimization

Similarly as in section 4.3.5, a low energy edge prolongator with a prescribed nonzero pattern and satisfying the commutation property (5.9) can be found as the solution to a constrained minimization problem. With $J(P_e)$ denoting the sum of the energies of the columns of P_e

$$J(P_e) \quad := \quad \frac{1}{2} \sum_j P_e(:, j)^T K_e P_e(:, j), \tag{5.30}$$

this minimization problem reads:

$$\min_{P_e \in \mathcal{N}_e} J(P_e) \quad s.t. \quad P_e G_c = G_f P_n. \tag{5.31}$$

Here, $\mathcal{N}_e$ denotes the space of the matrices with the prescribed nonzero pattern.

In [53], the columns of the optimized P_e are found as the (approximate) solution to the saddle point formulation of (5.31).

5.2.5 Parallel multigrid hierarchy

In the previous paragraphs, we showed how an edge prolongator P_e can be derived from a piecewise constant nodal prolongator P_n in such a way that the node-edge commutation property is satisfied. Now, we will consider the construction of the underlying nodal prolongator in some more detail.

For the construction of P_n, a diffusion-like matrix K_n is needed. For this purpose, the Galerkin product of the system matrix K_e (5.2) with the fine gradient G_f

$$K_n \quad = \quad G_f^T K_e G_f \quad = \quad G_f^T M_\beta^e G_f$$

is typically used. Recall that K_n is used by the nodal part of the hybrid smoother as well (cfr. (5.18)). Up to the boundary conditions, K_n is equal to the finite element discretization matrix K_β^{dg} (2.122) of the diffusion operator $\mathrm{div}(\beta(\mathrm{grad}(V)))$.

This procedure yields a parallel multigrid hierarchy of system matrices, with a node-based and an edge-based branch. Both branches are coupled by the discrete gradient operators:

$$\begin{array}{ccc} K_{nf} = G_f^T K_{ef} G_f & \xrightarrow{G_f} & K_{ef} \\ \uparrow P_n & & \uparrow P_e \\ K_{nc} = G_c^T K_{ec} G_c & \xrightarrow{G_c} & K_{ec} = P_e^T K_{ef} P_e \\ \uparrow \cdots & & \uparrow \cdots \\ \cdots & \xrightarrow{\cdots} & \cdots \end{array} \tag{5.32}$$

If the node-edge commutation property is satisfied, then the coarse system matrix K_{nc} of the nodal branch is equal to the Galerkin product of K_{nf} with P_n:

$$\begin{aligned} K_{nc} &= G_c^T K_{ec} G_c \\ &= G_c^T P_e^T K_{ef} P_e G_c \\ &= P_n^T G_f^T K_{ef} G_f P_n \\ &= P_n^T K_{nf} P_n. \end{aligned}$$

This allows the nodal branch to be considered as a genuine AMG hierarchy as well. Note, however, that due to the coupling of P_n and P_e in the node-edge commutation property, the coarsenings in both branches are not independent.

5.3 AMG methods with decoupled parallel hierarchy

The AMG methods described in the previous section are all based on a parallel multigrid hierarchy with branches coupled by the node-edge commutation property. In this section, we will consider AMG methods in which both branches are decoupled. In section 5.3.1, this is done by means of a block reformulation of the original problem. In section 5.3.2, multiple coarsening is used for this purpose.

5.3.1 Reformulation as a block system

In the coarsening step of AMG methods based on the node-edge commutation property, only the material parameter β is typically taken into account. This is because the coarsening, i.e. the selection of the coarse nodes, is part of the construction of the nodal prolongator P_n, which is derived from $K_n = G^T M_\beta G$. Due to the connection of P_n and P_e via the commutation property (5.9), this β-based coarsening will be reflected in the edge prolongator P_e as well. At most, the material parameter α can be taken into account via K_α^{cc} by smoothing the edge prolongator P_e (see section 5.2.2) or by minimizing its energy (see section 5.2.4).

This focus on β can be a problem if the spatial behavior of α and β is different. In this case, a β-based coarsening is only appropriate for the reduction of errors in $\mathrm{ran}(G)$, in which the α-related stiffness part K_α^{cc} has no impact. However, for the reduction of errors in $\mathrm{ran}(C^T)$, in which K_α^{cc} is dominating, a coarsening based on α might be more effective.

This can be achieved by giving up the coupling of P_n and P_e by the commutation property and by using an α- and β-based coarsening for P_e and P_n respectively. This approach has been developed by Jones and Lee in [41]. For this purpose, the fine level system (5.2) must be rewritten as a block system:

$$\begin{bmatrix} K_e & K_e G_f \\ G_f^T K_e & G_f^T K_e G_f \end{bmatrix} \begin{bmatrix} \bar{x} \\ \phi \end{bmatrix} = \begin{bmatrix} b \\ G_f^T b \end{bmatrix}. \tag{5.33}$$

The first block row of (5.33) is obtained immediately from (5.2) by using the following decomposition of the solution:

$$x = \bar{x} + G_f \phi, \quad \text{with} \quad \bar{x} \in \mathcal{E},\ \phi \in \mathcal{N}. \tag{5.34}$$

The second block row of (5.33) is obtained by multiplying its first block row by G_f^T. This implies that for a non-singular K_e, the dimension of the kernel of the system matrix of (5.33) equals the number of nodes. It is easy to see that the kernel equals $\{[-(G\varphi)^T\ \varphi^T]^T \mid \varphi \in \mathcal{N}\}$. This corresponds to the fact that the decomposition (5.34) is not unique:

$$x = (\bar{x} + G\varphi) + G(\phi - \varphi), \qquad \forall \varphi \in \mathcal{N}.$$

Consequently, all solutions of (5.33) describe the same unique solution of (5.2).

The decomposition (5.34) allows prolongation operators P_e and P_n to be used for $\bar{x}$ and ϕ, which are *not* coupled by the commutation property. They can be represented as a block prolongator:

$$P = \begin{bmatrix} P_e & 0 \\ 0 & P_n \end{bmatrix}. \tag{5.35}$$

In combination with classical smoothing, this block prolongator can be used in a classical multigrid cycle. Next, we will show that there is no need for a specialized smoothing procedure in this multigrid cycle.

Classical smoothing on the first block row of (5.33) is identical to classical smoothing on (5.2). Consequently, the update for $\bar{x}$ will typically not contain significant components in ran(G) (see section 5.1.2). While this is a major problem for multigrid for (5.2), it is an *advantage* here. More precisely, since $\bar{x}$ remains in ran(C^T) if the initial $\bar{x}$ belongs to ran(C^T), the (1,1)-block of (5.33) behaves like a vector Laplacian (cfr. (5.12)), which is an elliptic operator. Consequently, both diagonal blocks of (5.33) behave as elliptic operators. Since the off-diagonals blocks are typically relatively small, as they represent first order differential operators, classical smoothing is expected to yield good results for (5.33).

Remark 5.3.1 If the commutation property (5.9) holds, the structure of the fine level operator (5.33) is transferred to the coarse Galerkin operator. For instance, we have for the $(1,2)$-block, with $K_{ec} = P_e^T K_e P_e$:

$$P_e^T \left(K_e G_f\right) P_n \quad = \quad P_e^T K_e P_e G_c \quad = \quad K_{ec} G_c.$$

In this case, classical smoothing on (5.33) is equivalent to hybrid smoothing on (5.2), if after smoothing ϕ, the gradient $G\phi$ is added to $\bar{x}$ and ϕ is reset to 0:

$$\begin{aligned} \bar{x} &\leftarrow \bar{x} + G\phi \\ \phi &\leftarrow 0. \end{aligned}$$

5.3.2 Multiple coarsening AMG

The parallel multigrid hierarchy (5.32) can be decoupled by a multiple coarsening strategy as well. In a multiple coarsening approach, multiple multigrid hierarchies are constructed. For example, with two prolongators, P_1 and P_2:

$$\begin{array}{ccc} & K_e x = b & \\ & P_1 \swarrow \quad \searrow P_2 & \\ P_1^T K_e P_1 x_1 = P_1^T b & & P_2^T K_e P_2 x_2 = P_2^T b \\ \downarrow & & \downarrow \\ \cdots & & \cdots \end{array}$$

When applied to (5.2), this approach can be used to build a separate hierarchy tailored for each part of the discrete Helmholtz splitting (2.104).

For the methods considered in this section, the corresponding multigrid cycle amounts to smoothing on the fine level system, followed by a classical multigrid cycle on each hierarchy:

1. smoothing on $K_e x = b$
2. sequence of a multigrid cycle for each hierarchy i:
   2.1. restriction: $b_i = P_i^T (b - K_e x)$
   2.2. multigrid cycle for $P_i^T K_e P_i x_i = b_i$, with initial guess $x_i = 0$
   2.3. prolongation: $x \leftarrow x + P_i x_i$

In all AMG methods considered in this section, one of the top level prolongators equals the fine gradient operator $P_2 = G$. First, we will consider methods with $\text{ran}(P_1) = \text{ran}(C^T)$. Next, a component-splitting prolongator P_1 is discussed which yields a transformation from an edge-based representation of the solution to a node-based representation. Finally, this component-splitting prolongator is combined with a gauged block reformulation of the original problem.

Orthogonal subspaces

It is possible to isolate both parts of the discrete Helmholtz splitting by using C^T and G as the prolongation operators P_1 and P_2. Unfortunately, the solenoidal branch yields in this case a fourth-order problem in the face-based vector space $\mathcal{F}$. It is not clear how to build a suitable multigrid cycle for such problems.

In [46], a spanning tree approach is used to construct an alternative prolongator P_1 which generates $\text{ran}(C^T)$. Let the subscripts $\cdot_t$ and $\cdot_c$ refer to the edges belonging to the tree and the cotree respectively. A vector $x \in \text{ran}(C^T)$ satisfies:

$$G^T x = [G_t^T \; G_c^T] \begin{bmatrix} x_t \\ x_c \end{bmatrix} = 0, \quad \text{i.e.,} \quad G_t^T x_t + G_c^T x_c = 0. \tag{5.36}$$

Recall that the number of edges belonging to a spanning tree equals $n-1$, with n denoting the number of nodes. Consequently, G_t^T cannot be inverted in (5.36), since its size is $n \times n - 1$.

Note, however, that the equations of (5.36) are not linearly independent, since $G\mathbf{1} \equiv 0$. More precisely, each equation can be recovered as the negative sum of all other equations. This allows one equation to be dropped. Let the equation corresponding to the root node r of the tree be dropped, and let the matrices $\bar{G}_t$ and $\bar{G}_c$ denote G_t and G_c from which the column for r is removed. Now, $\bar{G}_t$ is square and can be inverted. This procedure yields a prolongator P_1 with the same range space as C^T:

$$x \;=\; \begin{bmatrix} -\bar{G}_t^{-T} \bar{G}_c^T \\ I \end{bmatrix} x_c \;:=\; P_1 x_c. \tag{5.37}$$

Component splitting

If the X-, Y- and Z-components of the unknown $\vec{A}$ of (5.1) can be made available in the nodes of the mesh, standard node-based multigrid is expected to perform well in $\text{ran}(C^T)$. This is because in $\text{ran}(C^T)$ and up to variations of α, each component of $\vec{A}$ satisfies a Laplace equation (cfr. (5.12)).

In [4], a prolongation operator P_{cs} is developed which relates the edge-based vector $a \in \mathcal{E}$ of DoFs to the node-based vectors $a_x, a_y, a_z \in \mathcal{N}$ of the corresponding X-, Y- and Z-components in the nodes, i.e.,

$$a \;=\; P_{cs} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \;:=\; \begin{bmatrix} P_x & P_y & P_z \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}. \tag{5.38}$$

The action of P_{cs} amounts to calculating the line integrals along all edges of the node-based representation of $\vec{A}$. Recall that the DoF $a(e_{ij})$ represents the line integral of $\vec{A}$ along e_{ij}. Since the node-based vectors a_x, a_y and a_z can be used to approximate $\vec{A}$:

$$\vec{A} \;\simeq\; \vec{A}_n \;:=\; \left(\sum_i a_x(n_i) N_i \;,\; \sum_i a_y(n_i) N_i \;,\; \sum_i a_z(n_i) N_i \right),$$

the line integral of $\vec{A}_n$ along e_{ij} yields an approximation of $a(e_{ij})$. Correspondingly, the action of P_{cs} is defined as follows:

$$\begin{aligned} a(e_{ij}) \;&=\; \int_{e_{ij}} \vec{A}_n \cdot d\vec{l} \\ &=\; \frac{(a_x(n_i) + a_x(n_j),\; a_y(n_i) + a_y(n_j),\; a_z(n_i) + a_z(n_j))}{2} \cdot (\mathbf{x}_j - \mathbf{x}_i). \end{aligned}$$

This allows the entries of the subprolongators P_x, P_y and P_z to be calculated as follows, with ϵ_x, ϵ_y and ϵ_z denoting the unit vectors in the X-, Y- and Z-direction:

$$\text{for } r = x, y, z \quad : \quad P_r(e_{ij}, n_k) = \begin{cases} \epsilon_r \cdot (\mathbf{x}_j - \mathbf{x}_i)/2 & \text{if k=i or k=j,} \\ 0 & \text{else.} \end{cases}$$

Note that this transformation requires the coordinates of the nodes of the mesh to be available.

This component splitting prolongator has been developed by Beck in [4] in the context of a multiple coarsening AMG with $P_1 = P_{cs}$ and $P_2 = G$. In this AMG method, both hierarchies are built using nodal AMG approaches from the second level on. This algorithm has been analyzed by Hiptmair and Xu in [38].

Remark 5.3.2 The Galerkin product of K_e with the component splitting prolongation operator P_{cs} has the following block structure:

$$P_{cs}^T K_e P_{cs} = \begin{bmatrix} K_{xx} & K_{xy} & K_{xz} \\ K_{yx} & K_{yy} & K_{yz} \\ K_{zx} & K_{zy} & K_{zz} \end{bmatrix} \quad \text{with} \quad K_{uv} = P_u^T K_e P_v. \tag{5.39}$$

In general, the nonzero pattern of all $P_u,\ u \in \{x, y, z\}$ is identical to the nonzero pattern of the discrete gradient operator G. Consequently, the nonzero patterns of all $K_{uv},\ u, v \in \{x, y, z\}$ are identical to the nonzero pattern of the discrete diffusion operator $G^T K_e G = G^T M_\beta^e G$. If no measures are taken to limit memory consumption, the memory needed to store this matrix is equal to the memory needed to store 9 discrete diffusion operators.

Compatible gauging

In [6], the original problem (5.2) is reformulated using gauging techniques, such that is becomes equivalent to a Laplace operator. Recall that (5.2) behaves like a vector Laplacian in $\text{ran}(C^T)$ (cfr. (5.12)). The gauging technique developed by Bochev e.a. in [6] results in a reformulation which is equivalent to a vector Laplacian in the *total* solution space.

Consider the so-called Hodge decomposition (see [6]) of the solution $x \in \mathcal{E}$ of (5.2):

$$x = G\varphi_n + \tilde{x}, \quad \text{with} \quad \begin{cases} \varphi_n \in \mathcal{N}, \\ \tilde{x} = (M_1^e)^{-1} C^T M_\alpha^f \varphi_f,\ \varphi_f \in \mathcal{F}. \end{cases} \tag{5.40}$$

Left multiplication with $G^T M_1^e$ shows that φ_n is the solution of the following system:

$$\begin{aligned} G^T M_1^e x &= G^T M_1^e G\varphi_n + G^T M_1^e (M_1^e)^{-1} C^T M_\alpha^f \varphi_f \\ &= G^T M_1^e G\varphi_n. \end{aligned}$$

Clearly, φ_n is unique up to components belonging to $\ker(G^T M_1^e G) = \ker(G)$. Consequently, the Hodge decomposition (5.40) is unique.

Similarly as in section 5.3.1, the Hodge decomposition (5.40) can be used to reformulate (5.2) as a block system. Additionally, the specific format of the Hodge decomposition allows a grad-div gauging term to be added to the $(1,1)$-block:

$$\begin{bmatrix} M_\beta^e + C^T M_\alpha^f C + \mathbf{M_1^e G M_\alpha^n G^T M_1^e} & M_\beta^e G \\ G^T M_\beta^e & G^T M_\beta^e G \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \varphi \end{bmatrix} = \begin{bmatrix} b \\ G^T b \end{bmatrix}. \tag{5.41}$$

This gauging term does not change the solution, since $\tilde{x}$ belongs to its kernel. However, it forces the solution to be unique. More precisely, the unique solution (5.34) of the ungauged system (5.33) is selected which satisfies the Hodge decomposition (5.40) as well.

The second order part of the $(1,1)$-block represents a compatible discretization of the following operator

$$\text{curl}(\alpha\, \text{curl}(\vec{A})) - \text{grad}(\alpha\, \text{div}(\vec{A})).$$

This differential operator equals a vector Laplacian for the X-, Y- and Z-components of $\vec{A}$, up to variations of α (cfr. (5.12)). This suggests to apply a component splitting approach to the $(1,1)$-block of (5.41), rather than to (5.2) itself, as in section 5.3.2. In [6], a multiple coarsening AMG solver for (5.2) is proposed which is based on classical AMG for both diagonal blocks of (5.41).

5.4 Concluding remarks

Today, AMG for curl-curl problems still is an active research domain. This is largely motivated by the observation that in general, AMG methods for curl-curl problems have not reached the efficiency of AMG methods for div-grad problems yet.

For publicly available implementations of the AMG methods presented in this chapter, we refer to ML (piecewise constant, smoothed aggregation and compatible gauging, [28]) and to HYPRE (component splitting, [49]).

Chapter 6

Fourier analysis of multigrid in $H(curl)$

6.1 Introduction

In this chapter, we develop a local Fourier analysis for geometric multigrid methods for the edge element discretization of the curl-curl equation

$$\text{curl}(\nu\,\text{curl}(\vec{A})) + \sigma\vec{A} \quad = \quad \vec{J}, \qquad \nu \neq 0. \tag{6.1}$$

A Fourier analysis of geometric multigrid for the curl-curl equation is not available yet. In fact, the nature of the degrees of freedom for discretizations of (5.1), which are related to edges instead of nodes, makes a straightforward application of the classical Fourier analysis techniques of section 3.3 impossible. The only existing related result is the derivation of the eigenvalues and eigenvectors of some curl-related operators in [22].

This chapter is organized as follows. Section 6.2 introduces some new definitions and notations. In section 6.3, the classical Fourier modes are generalized for edge-based discretizations. In section 6.4, the Helmholtz splitting is identified in the Fourier domain. In section 6.5, the LFA of the curl-curl and mass operators is developed. In section 6.6, several classical multigrid smoothers, the hybrid smoother of [35] and the overlapping block smoother of [3] are analyzed and their smoothing properties are quantified. Aliasing phenomena are analyzed in section 6.7.1. In sections 6.7.2 and 6.7.3, the restriction and prolongation are studied. Finally, the results of the previous sections are combined in section 6.8 into a LFA of the two-level multigrid cycle. Section 6.9 explains the numerical tests used to verify the analytical results presented in this chapter. In section 6.10, it is indicated how the approach used in this chapter can be generalized to other problem and mesh types.

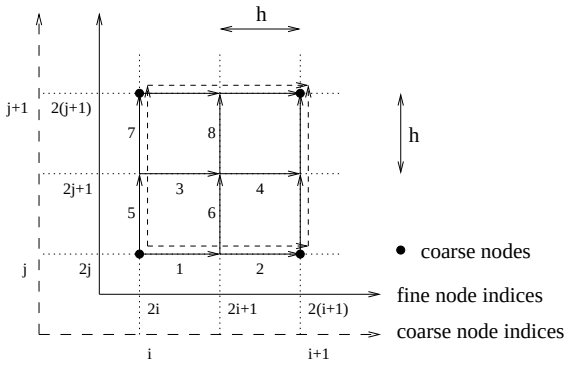


Figure 6.1: Fine and coarse rectangular grids. The coarse nodes are the fine nodes with even indices. The dotted arrows represent the coarse edges. The numbers indicate the colors of the fine edges in a 8-color scheme.

6.2 Definitions and notations

We consider the infinite, two-dimensional regular quadrilateral grid $\mathcal{G}_h$ with grid size h, as defined in (3.12). The orientation of the edges is chosen as indicated in figure 6.1. The horizontal and vertical orientation will be referred to by the subscripts $\cdot_1$ and $\cdot_2$ respectively.

The discretization of the curl-curl equation (6.1) on this grid is represented by the stencils (5.3) and (5.4). We will consider constant complex material parameters ν and σ. With an appropriate scaling, only the case $\nu = 1$ needs to be studied. The dependency on the grid size is eliminated by integrating h^2 into the material parameter σ as:

$$\sigma_h \quad := \quad h^2\sigma.$$

A node $\mathbf{x} = (k_1, k_2)h \in \mathcal{G}_h$ is referred to as (k_1, k_2), see section 3.3.1. Edges and faces will be referred to using their center points. For instance, $(\frac{r_1+s_1}{2}, \frac{r_2+s_2}{2})$ refers to the edge connecting the nodes (r_1, r_2) and (s_1, s_2). Further on, we will also need a *displaced index set* $I(\Delta_1, \Delta_2)$. With $\Delta_1, \Delta_2 \in \mathbb{R}$, this set is defined as

$$I(\Delta_1, \Delta_2) \quad := \quad \{(k_1 + \Delta_1, k_2 + \Delta_2) \mid k_1, k_2 \in \mathbb{Z}\}.$$

The sets of the indices of all nodes, horizontal and vertical edges and faces of $\mathcal{G}_h$ are defined as $N := I(0, 0)$, $E_1 := I(\frac{1}{2}, 0)$, $E_2 := I(0, \frac{1}{2})$ and $F := I(\frac{1}{2}, \frac{1}{2})$ respectively. The set of the indices of *all* edges is defined as $E := E_1 \cup E_2$.

The vector spaces of the degrees of freedom related to the edges and faces of $\mathcal{G}_h$ will be denoted $\mathcal{E}$ and $\mathcal{F}$. Similarly as the vector space $\mathcal{N}$ of the nodal degrees of freedom, these vectors spaces have infinite dimension, since the number of edges and faces in $\mathcal{G}_h$ is infinite.

The vector of the evaluations (3.13) of a function f in the nodes represented by the index set S will be denoted f_S. Similarly as for vectors belonging to $\mathcal{N}$,

this allows to represent a vector belonging to $\mathcal{E}$ by means of a function f as f_E. Recall that the evaluation operator $\cdot_{\mathbf{k}}$ can also be used as a selection operator, when applied to a vector. For example, $(f_S)_{\mathbf{k}} = f_{\mathbf{k}}$, if $\mathbf{k} \in S$.

We consider discrete edge-based operators L on $\mathcal{G}_h$ characterized by difference stencils with constant coefficients, similarly as (3.14). An important difference with stencils for node-based operators is that the orientation of the edges has to be taken into account. This implies that *two* difference stencils are needed to represent a 2D edge-based operator on a quadrilateral grid, one for the horizontal and one for the vertical edges. With $T = I(0,0) \cup I(\frac{1}{2},\frac{1}{2})$, the application of a discrete edge operator L to a vector w_E is defined as:

$$(Lw_E)_{\mathbf{k}} \quad := \quad \begin{cases} \sum_{\mathbf{t}\in T} s^{[1]}_{\mathbf{t}} w((\mathbf{k}+\mathbf{t})h) & \text{if } \mathbf{k} \in E_1, \\ \sum_{\mathbf{t}\in T} s^{[2]}_{\mathbf{t}} w((\mathbf{k}+\mathbf{t})h) & \text{if } \mathbf{k} \in E_2. \end{cases} \tag{6.2}$$

The coefficients of the horizontal and vertical stencil are denoted $s^{[1]}_{\mathbf{t}}$ and $s^{[2]}_{\mathbf{t}}$ respectively. The stencils corresponding to (6.2) will be represented as follows, for instance for the vertical stencil:

$$\mathcal{ST}_{L,2} \quad = \quad \begin{bmatrix} & \vdots & & \vdots & & \vdots & \\ \cdots & \circ & s^{[2]}_{-\frac{1}{2},\frac{1}{2}} & \bullet & s^{[2]}_{\frac{1}{2},\frac{1}{2}} & \circ & \cdots \\ & s^{[2]}_{-1,0} & & s^{[2]}_{0,0} & & s^{[2]}_{1,0} & \\ \cdots & \circ & s^{[2]}_{-\frac{1}{2},-\frac{1}{2}} & \bullet & s^{[2]}_{\frac{1}{2},-\frac{1}{2}} & \circ & \cdots \\ & \vdots & & \vdots & & \vdots & \end{bmatrix}. \tag{6.3}$$

Here, the subscripts $\cdot_{L,1}$ and $\cdot_{L,2}$ refer to the horizontal and vertical stencil of the operator L respectively. The nodes of the edge to which the result of the stencil operator is assigned, are indicated by filled circles, and all other nodes by open circles.

6.3 Fourier modes

Recall that a local Fourier analysis is based on modes of the following form (see section 3.3):

$$\varphi(\theta) := e^{i\theta\cdot\mathbf{x}/h} := e^{i\frac{\theta_1 x_1 + \theta_2 x_2}{h}} \qquad \text{with } \theta \in \mathbb{R}^2. \tag{6.4}$$

For node-based problems, these Fourier modes are continuous representations of the eigenvectors of grid operators that can be represented by a single stencil (see theorem 3.3.1). The apriori knowledge of the eigenvectors greatly reduces the complexity of the analysis.

In the case of edge-based problems, however, the relation between the Fourier modes and the eigenvectors of the problem is less obvious. More precisely, the vectors of the evaluations of (6.4) in the centers of the edges are *not* the eigenvectors of discrete edge operators L of the form (6.2). In fact, the application

of L to $\varphi(\theta)_E$ amounts to a *different* scaling for the horizontal and vertical components:

$$\begin{aligned}(L\varphi(\theta)_E)_{\mathbf{k}} &= \begin{cases} \sum_{\mathbf{t}\in T} s_{\mathbf{t}}^{[1]} e^{i\theta\cdot(\mathbf{k}+\mathbf{t})} & \text{if } \mathbf{k}\in E_1, \\ \sum_{\mathbf{t}\in T} s_{\mathbf{t}}^{[2]} e^{i\theta\cdot(\mathbf{k}+\mathbf{t})} & \text{if } \mathbf{k}\in E_2. \end{cases} \\ &= \varphi(\theta)_{\mathbf{k}} \cdot \begin{cases} \sum_{\mathbf{t}\in T} s_{\mathbf{t}}^{[1]} e^{i\theta\cdot\mathbf{t}} & \text{if } \mathbf{k}\in E_1, \\ \sum_{\mathbf{t}\in T} s_{\mathbf{t}}^{[2]} e^{i\theta\cdot\mathbf{t}} & \text{if } \mathbf{k}\in E_2. \end{cases}\end{aligned} \tag{6.5}$$

This decoupling is a direct consequence of the fact that the stencils for the horizontal and vertical edges are different. Consequently, $\varphi(\theta)_E$ is not directly useful in the LFA. Instead, we define $F_e(\theta)$ to be the two-dimensional space of all edge-based grid functions $\phi(\theta)_E$ of the form

$$\phi(\theta)_E = \alpha_1\phi_1(\theta)_E + \alpha_2\phi_2(\theta)_E, \qquad \alpha_1,\ \alpha_2 \in \mathbb{C}, \tag{6.6}$$

with

$$\phi_1(\theta)_E := \begin{bmatrix} \varphi(\theta)_{E_1} \\ 0_{E_2} \end{bmatrix}, \qquad \phi_2(\theta)_E := \begin{bmatrix} 0_{E_1} \\ \varphi(\theta)_{E_2} \end{bmatrix}. \tag{6.7}$$

This amounts to a decoupling of the horizontal and vertical components,

$$\phi(\theta)_{\mathbf{k}} = \begin{cases} \alpha_1\varphi(\theta)_{\mathbf{k}} & \text{if} \quad \mathbf{k}\in E_1 \\ \alpha_2\varphi(\theta)_{\mathbf{k}} & \text{if} \quad \mathbf{k}\in E_2, \end{cases} \qquad \text{with } \alpha_1,\alpha_2\in\mathbb{C}. \tag{6.8}$$

Further, we shall write (6.6) more compactly as

$$\phi(\theta)_E = \Phi(\theta)\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}, \qquad \text{with } \Phi(\theta) := \begin{bmatrix} \phi_1(\theta)_E & \phi_2(\theta)_E \end{bmatrix}. \tag{6.9}$$

$\Phi(\theta)$ will play the same role in the LFA of edge-based problems as the vector $\varphi(\theta)_N$ in the LFA of node-based problems. This role is based on the following theorem, which is the equivalent of theorem 3.3.1.

Theorem 6.3.1 *[Invariant subspace] For all $\theta\in\mathbb{R}^2$, $F_e(\theta)$ is invariant for a discrete edge operator L of the form (6.2).*

Proof. The application of L to $\phi(\theta)_E = \Phi(\theta)[\alpha_1\ \alpha_2]^T \in F_e(\theta)$ yields for a horizontal edge $\mathbf{k}\in E_1$:

$$\begin{aligned}(L\phi(\theta)_E)_{\mathbf{k}} &= \sum_{\mathbf{t}\in T} s_{\mathbf{t}}^{[1]}\phi(\theta)_{\mathbf{k}+\mathbf{t}} \\ &= \alpha_1 \sum_{\mathbf{t}\in I(0,0)} s_{\mathbf{t}}^{[1]} e^{i\theta\cdot(\mathbf{k}+\mathbf{t})} + \alpha_2 \sum_{\mathbf{t}\in I(\frac{1}{2},\frac{1}{2})} s_{\mathbf{t}}^{[1]} e^{i\theta\cdot(\mathbf{k}+\mathbf{t})} \\ &= e^{i\theta\cdot\mathbf{k}}\left(\alpha_1 \sum_{\mathbf{t}\in I(0,0)} s_{\mathbf{t}}^{[1]} e^{i\theta\cdot\mathbf{t}} + \alpha_2 \sum_{\mathbf{t}\in I(\frac{1}{2},\frac{1}{2})} s_{\mathbf{t}}^{[1]} e^{i\theta\cdot\mathbf{t}}\right).\end{aligned}$$

A similar result holds for the vertical edges. So, the combined result

$$L\phi(\theta)_E = L\Phi(\theta) \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \Phi(\theta) \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}, \tag{6.10}$$

with

$$\begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \sum_{\mathbf{t}\in I(0,0)} s_{\mathbf{t}}^{[1]} e^{i\theta\cdot\mathbf{t}} & \sum_{\mathbf{t}\in I(\frac{1}{2},\frac{1}{2})} s_{\mathbf{t}}^{[1]} e^{i\theta\cdot\mathbf{t}} \\ \sum_{\mathbf{t}\in I(\frac{1}{2},\frac{1}{2})} s_{\mathbf{t}}^{[2]} e^{i\theta\cdot\mathbf{t}} & \sum_{\mathbf{t}\in I(0,0)} s_{\mathbf{t}}^{[2]} e^{i\theta\cdot\mathbf{t}} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}, \tag{6.11}$$

belongs to $F_e(\theta)$ as well.
□

The 2×2 matrix in (6.11) is the representation of the edge operator L in $F_e(\theta)$ with respect to the basis $\Phi(\theta)$. It will be denoted $L_\Phi(\theta)$.

Theorem 6.3.2 *[Frequency aliasing] All spaces $F_e(\theta_1 + 2\pi a, \theta_2 + 2\pi b)$ with $a, b \in \mathbb{Z}$, are identical, for any $\theta_1, \theta_2 \in \mathbb{R}$.*

Proof. It is easy to check that the basis (6.9) of $F_e(\theta_1 + 2\pi a, \theta_2 + 2\pi b)$ and $F_e(\theta_1, \theta_2)$ are related as

$$\Phi(\theta_1 + 2\pi a, \theta_2 + 2\pi b) = \Phi(\theta_1, \theta_2) \begin{bmatrix} (-1)^a & 0 \\ 0 & (-1)^b \end{bmatrix}.$$

□

Theorem 6.3.2 shows that it is sufficient to consider any frequency interval of length 2π for θ_1 and θ_2. Similarly as in section 3.3, the two-dimensional frequency interval $\Theta := [-\pi/2, 3\pi/2)^2$ will be considered. For edge-based problems, this choice will lead to a considerable simplification of the LFA in comparison with the frequency interval $[-\pi, \pi)^2$ which is usually considered in a LFA (see for instance [63]).

Since multigrid smoothers aim at reducing the high frequency components of the error, their efficiency can be judged by studying their behavior for the high frequencies Θ^{high}.

Similarly as for node-based multigrid, aliasing phenomena will occur in the intergrid transfer operations restriction and prolongation. More precisely, aliasing will occur for the harmonic frequencies (3.22) of a frequency θ (see section 6.7). The aliasing of harmonic frequencies forces them to be studied together for the intergrid transfer operations restriction and prolongation. For the smoother of [3], which is not invariant for the spaces $F_e(\theta)$, harmonic frequencies will be coupled as well. In order to study these operators, we define the following 8-dimensional space:

$$F_8(\theta) := \bigcup_{a,b\in\{0,1\}} F_e(\theta^{(a,b)}), \qquad \theta \in \Theta^{\text{low}}. \tag{6.12}$$

The LFA in these spaces will be done by means of an 8-color parameterization. For any $\chi(\theta) \in F_8(\theta)$, this parameterization reads:

$$\chi(\theta)_{\mathbf{k}} \quad = \quad \delta_i \varphi(\theta)_{\mathbf{k}},\ \delta_i \in \mathbb{R} \qquad \text{if } \mathbf{k} \in E_i^8. \tag{6.13}$$

Here, E_i^8, with $1 \leq i \leq 8$, denotes the set of the indices of all edges with color i. The coloring scheme which will be used for (6.13), is indicated in figure 6.1. The transformation from the 8-color parameterization (6.13) to the frequency-based parameterization as a linear combination of vectors of the form (6.9), will be derived from the following lemma, which extends lemma 3.3.1.

Lemma 6.3.1 *Let $\boldsymbol{\Delta} = (\Delta_1, \Delta_2) \in \mathbb{R}^2$ be a fixed displacement and consider a grid function satisfying*

$$\chi(\theta)_{k_1+\Delta_1,k_2+\Delta_2} \quad = \quad \begin{cases} \zeta_{00}\varphi(\theta)_{k_1+\Delta_1,k_2+\Delta_2},\ \zeta_{00} \in \mathbb{C} & \text{if } k_1 \text{ and } k_2 \text{ even,} \\ \zeta_{10}\varphi(\theta)_{k_1+\Delta_1,k_2+\Delta_2},\ \zeta_{10} \in \mathbb{C} & \text{if } k_1 \text{ odd and } k_2 \text{ even,} \\ \zeta_{01}\varphi(\theta)_{k_1+\Delta_1,k_2+\Delta_2},\ \zeta_{01} \in \mathbb{C} & \text{if } k_1 \text{ even and } k_2 \text{ odd,} \\ \zeta_{11}\varphi(\theta)_{k_1+\Delta_1,k_2+\Delta_2},\ \zeta_{11} \in \mathbb{C} & \text{if } k_1 \text{ and } k_2 \text{ odd.} \end{cases}$$

Then, this grid function can be written as a linear combination of the grid functions $\varphi(\theta)_{N+\boldsymbol{\Delta}}$ for the harmonic frequencies of θ given in (3.22):

$$\chi(\theta)_{N+\boldsymbol{\Delta}} = \sum_{a,b \in \{0,1\}} \eta^{(a,b)} \varphi(\theta^{(a,b)})_{N+\boldsymbol{\Delta}}, \qquad \eta^{(a,b)} \in \mathbb{C}. \tag{6.14}$$

Proof. The functions $\varphi(\theta)_{N+\boldsymbol{\Delta}}$ for the harmonic frequencies of θ are related as:

$$\varphi(\theta^{(a,b)})_{k_1+\Delta_1,k_2+\Delta_2} \quad = \quad \varphi(\theta^{(0,0)})_{k_1+\Delta_1,k_2+\Delta_2} (-1)^{ak_1+bk_2} \gamma^{(a,b)}, \tag{6.15}$$

with $\gamma^{(a,b)} = e^{i\pi(a\Delta_1+b\Delta_2)}$. It suffices to note that (6.15) and (6.14) give rise to the following linear system:

$$\begin{bmatrix} \zeta_{00} \\ \zeta_{10} \\ \zeta_{01} \\ \zeta_{11} \end{bmatrix} \quad = \quad \begin{bmatrix} \gamma^{(0,0)} & \gamma^{(1,0)} & \gamma^{(0,1)} & \gamma^{(1,1)} \\ \gamma^{(0,0)} & -\gamma^{(1,0)} & \gamma^{(0,1)} & -\gamma^{(1,1)} \\ \gamma^{(0,0)} & \gamma^{(1,0)} & -\gamma^{(0,1)} & -\gamma^{(1,1)} \\ \gamma^{(0,0)} & -\gamma^{(1,0)} & -\gamma^{(0,1)} & \gamma^{(1,1)} \end{bmatrix} \begin{bmatrix} \eta^{(0,0)} \\ \eta^{(1,0)} \\ \eta^{(0,1)} \\ \eta^{(1,1)} \end{bmatrix}. \tag{6.16}$$

Let $\Gamma_{\Delta_1,\Delta_2}(\theta)$ denote the system matrix of (6.16). Its invertibility follows immediately from the fact that $|\gamma^{(a,b)}| = 1$ and from the factorization

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} \gamma^{(0,0)} & 0 & 0 & 0 \\ 0 & \gamma^{(1,0)} & 0 & 0 \\ 0 & 0 & \gamma^{(0,1)} & 0 \\ 0 & 0 & 0 & \gamma^{(1,1)} \end{bmatrix}.$$

□

Note that the classical definition (e.g. definition 4.4.1. in [63]) for the harmonics of a frequency, i.e.

$$\theta^{(a,b)} = (\theta_1^{(a)}, \theta_2^{(b)}),\ a,b \in \{0,1\} \tag{6.17}$$

with

$$\theta_i^{(c)} := \begin{cases} \theta_i + c\pi & \text{if } \theta_i < 0 \\ \theta_i - c\pi & \text{if } \theta_i \geq 0, \end{cases}$$

would require a separate treatment of the negative and positive frequencies.

Lemma 6.3.1 can be applied separately to the horizontal and vertical components of (6.13). This shows that a grid function of the form (6.13) can be transformed into a linear combination of vectors of the form (6.6)

$$\chi(\theta)_E = \begin{bmatrix} \Phi(\theta^{(0,0)}) & \Phi(\theta^{(1,0)}) & \Phi(\theta^{(0,1)}) & \Phi(\theta^{(1,1)}) \end{bmatrix} \begin{bmatrix} \epsilon^{(0,0)} \\ \epsilon^{(1,0)} \\ \epsilon^{(0,1)} \\ \epsilon^{(1,1)} \end{bmatrix}, \tag{6.18}$$

with $\epsilon^{(a,b)} = [\epsilon_1^{(a,b)}\ \epsilon_2^{(a,b)}]^T \in \mathbb{C}^2$ given by

$$\begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \end{bmatrix} = \Gamma_{\frac{1}{2},0}(\theta) \begin{bmatrix} \epsilon_1^{(0,0)} \\ \epsilon_1^{(1,0)} \\ \epsilon_1^{(0,1)} \\ \epsilon_1^{(1,1)} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \delta_5 \\ \delta_6 \\ \delta_7 \\ \delta_8 \end{bmatrix} = \Gamma_{0,\frac{1}{2}}(\theta) \begin{bmatrix} \epsilon_2^{(0,0)} \\ \epsilon_2^{(1,0)} \\ \epsilon_2^{(0,1)} \\ \epsilon_2^{(1,1)} \end{bmatrix}. \tag{6.19}$$

The combined transformation matrix will be denoted $T_8(\theta) \in \mathbb{C}^{8\times 8}$, with 0_4 denoting the zero matrix of size 4×4:

$$T_8(\theta) := \begin{bmatrix} \Gamma_{\frac{1}{2},0}(\theta) & 0_4 \\ 0_4 & \Gamma_{0,\frac{1}{2}}(\theta) \end{bmatrix}. \tag{6.20}$$

6.4 The Helmholtz splitting in Fourier space

In this section, it will be shown that the Helmholtz splitting (2.104), which plays a fundamental role in the analysis of discretization schemes and solvers for the curl-curl equation (see chapters 2 and 5 and [52, 13, 2, 35, 56]), is present in the Fourier domain as well. The Helmholtz splitting in the Fourier domain will be used to construct an alternative parameterization of the spaces $F_e(\theta)$. This parameterization will be useful in the LFA of the hybrid smoother of [35], and it will allow some well-known properties, such as $\text{ran}(K^{cc}) = \ker(G)$, to be identified in the Fourier domain.

The following theorem identifies the Helmholtz splitting in the spaces $F_e(\theta)$.

Theorem 6.4.1 *[Fourier-Helmholtz splitting] For all $\theta \in \Theta$, $F_e(\theta)$ can be split as*

$$F_e(\theta) = \psi_g(\theta)_E \oplus \psi_s(\theta)_E, \tag{6.21}$$

with $\psi_g(\theta)_E \in \text{ran}(G)$ and $\psi_s(\theta)_E \in \text{ran}(C^T)$.

Proof. If $\theta \neq (0,0)$, the vectors $\varphi(\theta)_N$ and $\varphi(\theta)_F$ of the evaluations of $\varphi(\theta)$ in the grid nodes N and the center nodes F of the faces can be used to construct vectors in $F_e(\theta)$ belonging to $\mathrm{ran}(G)$ and $\mathrm{ran}(C^T)$ respectively. The subscripts $\cdot_g$ and $\cdot_s$ will be used to refer to the gradient and solenoidal vector respectively,

$$\begin{aligned}
\psi_g(\theta)_{\mathbf{k}} := (G\varphi(\theta)_N)_{\mathbf{k}} &= \begin{cases} \varphi(\theta)_{k_1+\frac{1}{2},k_2} - \varphi(\theta)_{k_1-\frac{1}{2},k_2} & \text{if} \quad \mathbf{k} \in E_1, \\ \varphi(\theta)_{k_1,k_2+\frac{1}{2}} - \varphi(\theta)_{k_1,k_2-\frac{1}{2}} & \text{if} \quad \mathbf{k} \in E_2, \end{cases} \qquad (6.22) \\
&= \begin{cases} 2i\sin(\frac{\theta_1}{2})\varphi(\theta)_{\mathbf{k}} & \text{if} \quad \mathbf{k} \in E_1, \\ 2i\sin(\frac{\theta_2}{2})\varphi(\theta)_{\mathbf{k}} & \text{if} \quad \mathbf{k} \in E_2, \end{cases} \\
\psi_s(\theta)_{\mathbf{k}} := \left(C^T\varphi(\theta)_F\right)_{\mathbf{k}} &= \begin{cases} \varphi(\theta)_{k_1,k_2+\frac{1}{2}} - \varphi(\theta)_{k_1,k_2-\frac{1}{2}} & \text{if} \quad \mathbf{k} \in E_1, \\ \varphi(\theta)_{k_1-\frac{1}{2},k_2} - \varphi(\theta)_{k_1+\frac{1}{2},k_2} & \text{if} \quad \mathbf{k} \in E_2, \end{cases} \qquad (6.23) \\
&= \begin{cases} 2i\sin(\frac{\theta_2}{2})\varphi(\theta)_{\mathbf{k}} & \text{if} \quad \mathbf{k} \in E_1, \\ -2i\sin(\frac{\theta_1}{2})\varphi(\theta)_{\mathbf{k}} & \text{if} \quad \mathbf{k} \in E_2. \end{cases}
\end{aligned}$$

Since $\psi_g(\theta)_E$ and $\psi_s(\theta)_E$ are of the form (6.6) for all $\theta \in \Theta$, they belong to $F_e(\theta)$. By construction, they belong to $\mathrm{ran}(G)$ and $\mathrm{ran}(C^T)$ respectively. Since they are different from the 0-vector for all $\theta \neq (0,0)$ and $\mathrm{ran}(G) \perp \mathrm{ran}(C^T)$, they are linearly independent. Consequently, the columns of

$$\Psi(\theta) := \begin{bmatrix} \psi_g(\theta)_E & \psi_s(\theta)_E \end{bmatrix} \qquad (6.24)$$

constitute an orthogonal basis for $F_e(\theta)$ for all $\theta \neq (0,0)$.

The above procedure cannot be used for $\theta = (0,0)$, since it would yield $\psi_g = \psi_s \equiv 0_E$. In this case, the grid functions χ_N and χ_F, with $\chi(\mathbf{x}) := 2i(x_1+x_2)/h$, can be used instead, which yield:

$$\begin{aligned}
\psi_g(\mathbf{0})_{\mathbf{k}} := (G\chi_N)_{\mathbf{k}} &= \begin{cases} 2i = 2i\varphi(\mathbf{0})_{\mathbf{k}} & \text{if} \quad \mathbf{k} \in E_1, \\ 2i = 2i\varphi(\mathbf{0})_{\mathbf{k}} & \text{if} \quad \mathbf{k} \in E_2, \end{cases} \\
\psi_s(\mathbf{0})_{\mathbf{k}} := \left(C^T\chi_F\right)_{\mathbf{k}} &= \begin{cases} 2i = 2i\varphi(\mathbf{0})_{\mathbf{k}} & \text{if} \quad \mathbf{k} \in E_1, \\ -2i = -2i\varphi(\mathbf{0})_{\mathbf{k}} & \text{if} \quad \mathbf{k} \in E_2. \end{cases}
\end{aligned}$$

The conclusion that $\Psi(\mathbf{0})$ is an orthogonal basis for $F_e(\mathbf{0})$ follows.
□

The representation of an edge operator L of the form (6.2) in $F_e(\theta)$ with respect to the Fourier-Helmholtz basis (6.24) will be denoted $L_\Psi(\theta)$. This basis is related to the decoupled basis $\Phi(\theta)$ from (6.9) as follows:

$$\Psi(\theta) = 2i\Phi(\theta)H(\theta), \qquad (6.25)$$

with

$$H(\theta) := \begin{cases} \begin{bmatrix} \sin(\frac{\theta_1}{2}) & \sin(\frac{\theta_2}{2}) \\ \sin(\frac{\theta_2}{2}) & -\sin(\frac{\theta_1}{2}) \end{bmatrix} & \text{if} \quad \theta \neq (0,0), \\ \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} & \text{if} \quad \theta = (0,0). \end{cases} \qquad (6.26)$$

A representation of a grid function in $F_e(\theta)$ with respect to the basis $\Psi(\theta)$ can be transformed into a representation with respect to the basis $\Phi(\theta)$ as follows:

$$\Psi(\theta)\begin{bmatrix}\alpha_g\\ \alpha_s\end{bmatrix} = \Phi(\theta)\begin{bmatrix}\alpha_1\\ \alpha_2\end{bmatrix}, \quad \text{or,} \qquad 2iH(\theta)\begin{bmatrix}\alpha_g\\ \alpha_s\end{bmatrix} = \begin{bmatrix}\alpha_1\\ \alpha_2\end{bmatrix}. \tag{6.27}$$

The LFA presented in this chapter will be based on the basis (6.9). For the analysis of the hybrid smoother and for interpretation purposes, however, a transformation to the basis (6.24) will be advantageous. Further on, the argument θ will be left out whenever its value is clear from the context. The following abbreviations will be used:

$$\begin{aligned} &s_i = \sin(\tfrac{\theta_i}{2}), \quad \tilde{s}_i = \sin(\theta_i), \quad s_{i,a} = \sin(\tfrac{\theta_i+a\pi}{2}), \quad \tilde{s}_{i,a} = \sin(\theta_i+a\pi),\\ &c_i = \cos(\tfrac{\theta_i}{2}), \quad \tilde{c}_i = \cos(\theta_i), \quad c_{i,a} = \cos(\tfrac{\theta_i+a\pi}{2}), \quad \tilde{c}_{i,a} = \cos(\theta_i+a\pi). \end{aligned}$$

6.5 Fourier domain representation of the operator

6.5.1 Stiffness part

Application of (6.11) to the stencils defining K_{cc} yields:

$$\begin{bmatrix}\beta_1\\ \beta_2\end{bmatrix} = \begin{bmatrix}4s_2^2 & -4s_1s_2\\ -4s_1s_2 & 4s_1^2\end{bmatrix}\begin{bmatrix}\alpha_1\\ \alpha_2\end{bmatrix} =: K_{cc,\Phi}(\theta)\begin{bmatrix}\alpha_1\\ \alpha_2\end{bmatrix}. \tag{6.28}$$

Using (6.27), relation (6.28) can be transformed into:

$$\begin{aligned}\begin{bmatrix}\beta_g\\ \beta_s\end{bmatrix} &= H(\theta)^{-1}K_{cc,\Phi}(\theta)H(\theta)\begin{bmatrix}\alpha_g\\ \alpha_s\end{bmatrix} \\ &= \begin{bmatrix}0 & 0\\ 0 & 4(s_1^2+s_2^2)\end{bmatrix}\begin{bmatrix}\alpha_g\\ \alpha_s\end{bmatrix} =: K_{cc,\Psi}(\theta)\begin{bmatrix}\alpha_g\\ \alpha_s\end{bmatrix}.\end{aligned} \tag{6.29}$$

The nonzero structure of $K_{cc,\Psi}(\theta)$ shows that $\psi_g(\theta)_E$ and $\psi_s(\theta)_E$ are the eigenvectors of the discrete curl-curl operator. The zero eigenvalue for $\psi_g(\theta)_E$ corresponds to the fact that $\mathrm{ran}(G)$ is the kernel of the discrete curl-curl operator. The eigenvalue of $\psi_s(\theta)_E$ is identical to the Fourier domain representation (3.21) of the finite difference discretization of the div-grad operator on a two-dimensional rectangular grid. This can be interpreted as the Fourier domain counterpart of the equivalence of the vector Laplace operator and the curl-curl operator in the divergence-free part of the Helmholtz splitting, which follows from (5.12).

Remark. The equivalence of the vector Laplace operator and the curl-curl operator for divergence-free vectors can be recovered in the discrete setting as

well. For a divergence-free vector v, we have $G^T v = 0$, which reads using stencil notation:

$$\begin{bmatrix} & -1 & \\ 1 & \bullet & -1 \\ & 1 & \end{bmatrix} = 0 \iff \begin{bmatrix} & -1 & \\ 0 & \bullet & 0 \\ & 1 & \end{bmatrix} = \begin{bmatrix} & 0 & \\ -1 & \bullet & 1 \\ & 0 & \end{bmatrix}. \quad (6.30)$$

Consequently, the curl-curl stencil can be transformed into an equivalent div-grad stencil for divergence-free grid functions:

$$\begin{bmatrix} \circ & -1 & \circ \\ -1 & & 1 \\ \bullet & 2 & \bullet \\ 1 & & -1 \\ \circ & -1 & \circ \end{bmatrix} = \begin{bmatrix} & \circ & -1 & \circ & \\ & 0 & & 0 & \\ -1 & \bullet & 4 & \bullet & -1 \\ & 0 & & 0 & \\ & \circ & -1 & \circ & \end{bmatrix}. \quad (6.31)$$

6.5.2 Mass part

Application of theorem 6.3.1 for the mass part $\sigma_h M$ yields:

$$\sigma_h M_\Phi(\theta) := \frac{\sigma_h}{3} \begin{bmatrix} 2+\tilde{c}_2 & 0 \\ 0 & 2+\tilde{c}_1 \end{bmatrix}. \quad (6.32)$$

Formula (6.32) shows that the columns of $\Phi(\theta)$ for all θ are the eigenvectors of M.

6.6 Fourier domain representation of the smoother

6.6.1 Classical smoothers

Consider a smoother based on a matrix splitting iteration (3.1). Let the iteration matrix be denoted $S = -(K^+)^{-1} K^-$. Of particular interest is the case where the splitting of K corresponds to a splitting of the vertical and horizontal stencils defining K, i.e., for $i = 1, 2$:

$$\mathcal{ST}_{K,i} = \mathcal{ST}^+_{K,i} + \mathcal{ST}^-_{K,i} \quad \text{with} \quad \begin{cases} \mathcal{ST}^+_{K,i} = \mathcal{ST}_{K^+,i}\,, \\ \mathcal{ST}^-_{K,i} = \mathcal{ST}_{K^-,i}\,. \end{cases} \quad (6.33)$$

Then, theorem 6.3.1 holds for K^+ and K^- as well. More precisely, K^+ and K^- can be represented in the Fourier domain by 2×2 complex matrices, similarly as K (see section 6.5). Consequently, the error reduction in $F_e(\theta)$ reads:

$$\begin{aligned} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}^{[\nu+1]} &= -\left(K^+_\Phi(\theta)\right)^{-1} K^-_\Phi(\theta) \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}^{[\nu]} \\ &= -\left(K^+_\Phi(\theta)\right)^{-1} \left(K_\Phi(\theta) - K^+_\Phi(\theta)\right) \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}^{[\nu]} \\ &:= S_\Phi(\theta) \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}^{[\nu]}. \end{aligned} \quad (6.34)$$

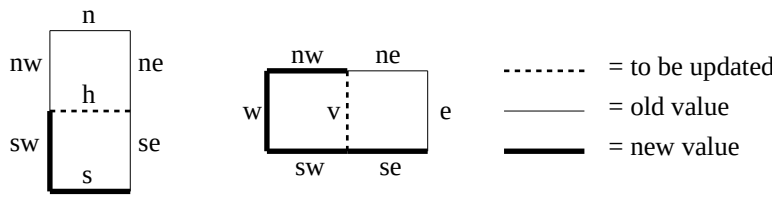


Figure 6.2: Node-based updating order for the lexicographic Gauss-Seidel method for a horizontal edge h and a vertical edge v.

Formula (6.34) implies that the spaces $F_e(\theta)$ are eigenspaces of S. This allows the error equation to be studied for each θ separately by means of the spectral radius of $S_\Phi(\theta)$:

$$\rho(S_\Phi(\theta)) \quad := \quad \max\{|\lambda| \ : \ \lambda \in \sigma(S_\Phi(\theta))\}.$$

First, we consider the Jacobi iteration. In this case K^+ equals the diagonal of K. This corresponds to the following stencil splittings:

$$\mathcal{ST}^+_{K,1} = \begin{bmatrix} \circ & 0 & \circ \\ 0 & & 0 \\ \bullet & 2+\frac{2\sigma_h}{3} & \bullet \\ 0 & & 0 \\ \circ & 0 & \circ \end{bmatrix}, \qquad \mathcal{ST}^+_{K,2} = \begin{bmatrix} \circ & 0 & \bullet & 0 & \circ \\ 0 & & 2+\frac{2\sigma_h}{3} & & 0 \\ \circ & 0 & \bullet & 0 & \circ \end{bmatrix}.$$

The Fourier domain representation is obtained by application of (6.11):

$$K^+_\Phi(\theta) \quad = \quad \begin{bmatrix} d_{cc}(2+\frac{2\sigma_h}{3}) & 0 \\ 0 & d_{cc}(2+\frac{2\sigma_h}{3}) \end{bmatrix}.$$

Next, we consider the lexicographic Gauss-Seidel method for a node-based updating order, in which edges with the same start node are updated consecutively. Figure 6.2 illustrates the update order. The nodes are traversed vertically from bottom left to top right. At every node, first the edge emanating to the right is updated, followed by the edge emanating to the top. This corresponds to the stencil splittings

$$\mathcal{ST}^+_{K,1} = \begin{bmatrix} \circ & 0 & \circ \\ 0 & & 0 \\ \bullet & 2+\frac{2\sigma_h}{3} & \bullet \\ 1 & & 0 \\ \circ & -1+\frac{\sigma_h}{6} & \circ \end{bmatrix}$$

and

$$\mathcal{ST}^+_{K,2} = \begin{bmatrix} \circ & -1 & \bullet & 0 & \circ \\ -1+\frac{\sigma_h}{6} & & 2+\frac{2\sigma_h}{3} & & 0 \\ \circ & 1 & \bullet & -1 & \circ \end{bmatrix}.$$

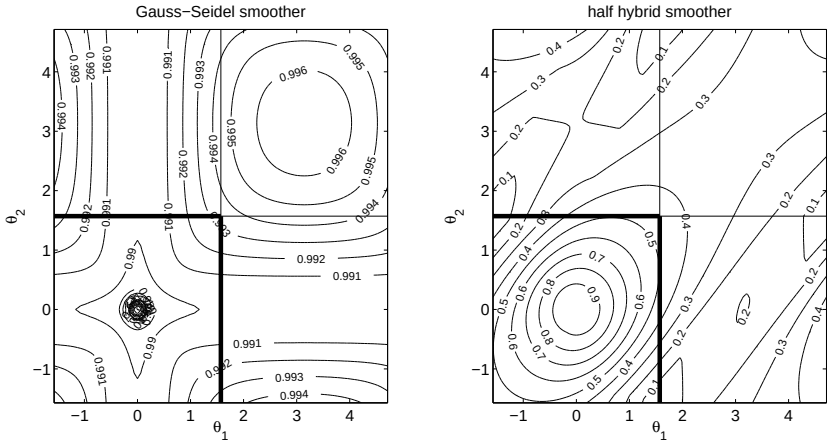


Figure 6.3: Contourplot of $\rho(S(\theta))$ as a function of θ with $\sigma_h = 0.01$ for the node-based lexicographic Gauss-Seidel smoother (left) and for the half hybrid smoother consisting of 1 node-based lexicographic Gauss-Seidel iteration in $\mathcal{E}$ followed by 1 lexicographic Gauss-Seidel iteration in $\mathcal{N}$ (right). The bold line separates the low frequency region Θ^{low} in the lower left corner from the high frequency region Θ^{high}. Contrary to the hybrid smoother, Gauss-Seidel iterations don't achieve satisfactory smoothing of the high frequency errors.

Again, the Fourier domain representation is obtained by application of (6.11):

$$K_\Phi^+(\theta) \quad = \quad \begin{bmatrix} (2+\frac{2\sigma_h}{3}) - (1-\frac{\sigma_h}{6})\varphi_{0,-1} & \varphi_{-\frac{1}{2},-\frac{1}{2}} \\ \varphi_{-\frac{1}{2},-\frac{1}{2}} - \varphi_{-\frac{1}{2},\frac{1}{2}} - \varphi_{\frac{1}{2},-\frac{1}{2}} & (2+\frac{2\sigma_h}{3}) - (1-\frac{\sigma_h}{6})\varphi_{-1,0} \end{bmatrix}.$$

Here, φ_{k_1,k_2} is used as a shorthand for $\varphi(\theta)_{\mathbf{k}}$, with $\mathbf{k} = (k_1, k_2)$.

The left picture of figure 6.3 shows $\rho(S(\theta))$ as a function of θ for the lexicographic Gauss-Seidel method applied to (5.2). Since in a multigrid context, it is the task of a smoother to reduce the high frequency error components, the smoothing properties of this smoother are very poor. This is due to the non-ellipticity of the curl-curl equation in the gradient part $\text{ran}(G)$ of the Helmholtz splitting (2.104). We refer the reader to [35] for a further analysis of this phenomenon.

Remark 6.6.1 With respect to the Fourier-Helmholtz basis (6.24), the Fourier domain representation of the Jacobi method with $\sigma = 0$ and damping factor d_{cc} reads

$$\begin{bmatrix} \alpha_g \\ \alpha_s \end{bmatrix}^{[\nu+1]} \quad = \quad \begin{bmatrix} 1 & 0 \\ 0 & 1 - \frac{2}{d_{cc}}(s_1^2 + s_2^2) \end{bmatrix} \begin{bmatrix} \alpha_g \\ \alpha_s \end{bmatrix}^{[\nu]}. \tag{6.35}$$

This illustrates the fact that undamped Jacobi, i.e., with $d_{cc} = 1$, diverges for the curl-curl equation. The damped iteration will converge in the solenoidal part

of the Helmholtz splitting for any $d_{cc} \geq 2$. This also reveals that in $\text{ran}(C^T)$, this smoother behaves in the same way as damped Jacobi iterations for the finite difference discretization of the Laplace equation $\text{div}(\text{grad}(V)) = \rho$ with damping factor $d_{dg} = d_{cc}/2$. Indeed, in the Fourier domain, the error equation for the latter satisfies (see section 3.3)

$$\alpha^{[\nu+1]} = \left(1 - \frac{1}{d_{dg}}(s_1^2 + s_2^2)\right) \alpha^{[\nu]}.$$

This result can be recovered using stencil notation as well. More precisely, in $\text{ran}(C^T)$, the horizontal stencil for damped Jacobi iterations for the curl-curl equation is equivalent to, using (6.30):

$$\frac{-1}{2d_{cc}} \begin{bmatrix} \circ & -1 & \circ \\ -1 & & 1 \\ \bullet & 2(1-d_{cc}) & \bullet \\ 1 & & -1 \\ \circ & -1 & \circ \end{bmatrix} = \frac{-1}{2d_{cc}} \begin{bmatrix} & \circ & -1 & \circ & \\ & 0 & & 0 & \\ -1 & \bullet & 4-2d_{cc} & \bullet & -1 \\ & 0 & & 0 & \\ & \circ & -1 & \circ & \end{bmatrix}.$$

The coefficient for the central edge in the equivalent stencil consists of two parts: one part arises from the original coefficient, the other part is contributed by the vertical edges via (6.30). This second part is *not* included in the damping. For damped Jacobi iterations for the diffusion equation, on the contrary, the *total* coefficient for the central edge is damped:

$$-\begin{bmatrix} & -1 & \\ -1 & 4(1-d_{dg}) & -1 \\ & -1 & \end{bmatrix} /4d_{dg}.$$

6.6.2 Hybrid smoother

In this section, a LFA of the hybrid smoother of [35], which is described in section 5.1.2, will be presented.

First, the nodal part of the hybrid smoother, consisting of steps 2, 3 and 4, is studied separately. We start with the left multiplication by G^T in step 2. The application of G^T to a grid function of the form (6.6) yields, with $\mathbf{k} = (k_1, k_2) \in N$:

$$\begin{aligned} \left(G^T \Phi(\theta) \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}\right)_{\mathbf{k}} &= -\alpha_1 \left(\varphi(\theta)_{k_1+\frac{1}{2},k_2} - \varphi(\theta)_{k_1-\frac{1}{2},k_2}\right) \\ &\quad -\alpha_2 \left(\varphi(\theta)_{k_1,k_2+\frac{1}{2}} - \varphi(\theta)_{k_1,k_2-\frac{1}{2}}\right) \\ &= \beta \varphi(\theta)_{\mathbf{k}}, \end{aligned}$$

with

$$\beta = -2i \begin{bmatrix} s_1 & s_2 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} =: G_\Phi^T(\theta) \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}. \tag{6.36}$$

Using (6.27), the relation (6.36) can be transformed into:

$$\beta \;=\; \begin{bmatrix} 4(s_1^2+s_2^2) & 0 \end{bmatrix} \begin{bmatrix} \alpha_g \\ \alpha_s \end{bmatrix} =: G_\Psi^T(\theta) \begin{bmatrix} \alpha_g \\ \alpha_s \end{bmatrix}. \tag{6.37}$$

Formula (6.37) is the Fourier domain counterpart of the transpose of the topological identity $CG \equiv 0$, as it shows that the restriction of vectors belonging to $\mathrm{ran}(C^T)$ by G^T is 0. Now, the restriction of the edge residual in step 2 can be represented in the Fourier domain. With $K_\Psi(\theta)$, $K_{cc,\Psi}(\theta)$ and $M_\Psi(\theta)$ the Fourier symbols of K, K_{cc} and M with respect to the Fourier-Helmholtz basis (6.24), we obtain:

$$\begin{aligned}
& G_\Psi^T(\theta) K_\Psi(\theta) \begin{bmatrix} \alpha_g \\ \alpha_s \end{bmatrix} \\
= \;& G_\Psi^T(\theta) \left(K_{cc,\Psi}(\theta) + \sigma_h M_\Psi(\theta) \right) \begin{bmatrix} \alpha_g \\ \alpha_s \end{bmatrix} \\
= \;& \sigma_h G_\Psi^T(\theta) M_\Psi(\theta) \begin{bmatrix} \alpha_g \\ \alpha_s \end{bmatrix} \\
= \;& \frac{\sigma_h}{3} \begin{bmatrix} 4(s_1^2+s_2^2) & 0 \end{bmatrix} H(\theta)^{-1} \begin{bmatrix} 2+\tilde{c}_2 & 0 \\ 0 & 2+\tilde{c}_1 \end{bmatrix} H(\theta) \begin{bmatrix} \alpha_g \\ \alpha_s \end{bmatrix} \\
= \;& \frac{4\sigma_h}{3} \begin{bmatrix} (2+\tilde{c}_2)s_1^2 + (2+\tilde{c}_1)s_2^2 & s_1 s_2 (\tilde{c}_2 - \tilde{c}_1) \end{bmatrix} \begin{bmatrix} \alpha_g \\ \alpha_s \end{bmatrix}.
\end{aligned}$$

For $\theta \neq (0,0)$, the prolongation by G of $\varphi(\theta)_N \alpha$, with $\alpha \in \mathbb{C}$, in step 4 yields a scaled gradient basis function by (6.22). For $\theta = (0,0)$, we refer to the remark at the end of this section. In the Fourier domain and using (6.24), step 4 reads:

$$\left(G\varphi(\theta)_N \alpha \right)_E \;=\; \Psi(\theta) \begin{bmatrix} \beta_g \\ \beta_s \end{bmatrix},$$

with

$$\begin{bmatrix} \beta_g \\ \beta_s \end{bmatrix} \;=\; \begin{bmatrix} 1 \\ 0 \end{bmatrix} \alpha \;=:\; G_\Psi(\theta)\alpha.$$

The auxiliary nodal matrix K_n for step 3 is usually built as the Galerkin product $G^T K G$. This corresponds to the stencil for the finite element discretization using bilinear finite elements of the Poisson equation on the same grid, scaled by σ_h:

$$\sigma_h \begin{bmatrix} -1/3 & -1/3 & -1/3 \\ -1/3 & 8/3 & -1/3 \\ -1/3 & -1/3 & -1/3 \end{bmatrix}, \tag{6.38}$$

and to the following Fourier domain representation:

$$\begin{aligned}
K_n(\theta) \;&=\; G_\Psi^T(\theta) K_\Psi(\theta) G_\Psi(\theta) = \sigma_h G_\Psi^T(\theta) M_\Psi(\theta) G_\Psi(\theta) \\
&=\; \frac{\sigma_h}{3} \begin{bmatrix} 4(s_1^2+s_2^2) & 0 \end{bmatrix} H(\theta)^{-1} \begin{bmatrix} 2+\tilde{c}_2 & 0 \\ 0 & 2+\tilde{c}_1 \end{bmatrix} H(\theta) \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
&=\; \frac{4\sigma_h}{3} \left((2+\tilde{c}_2)s_1^2 + (2+\tilde{c}_1)s_2^2 \right).
\end{aligned}$$

Now, the combined effect of steps 2, 3 and 4 can be studied. Suppose the smoother applied to the auxiliary system $K_n x_n = b_n$ is a linear operator S_n with eigenvectors $\varphi(\theta)_N$ and corresponding eigenvalues $S_n(\theta)$. This is the case if classical smoothers, such as Jacobi or lexicographic Gauss-Seidel, are used (see [63]). Let x_n^* denote the exact solution of the auxiliary system, i.e., $x_n^* = K_n^{-1} b_n = K_n^{-1} G^T (b - Kx)$, and let $x_n^{[1]}$ and $e_n^{[1]} = x_n^* - x_n^{[1]}$ be an approximate solution and the corresponding error after the smoothing step 3. Since this smoothing starts from a zero initial guess $x_n^{[0]} = 0$, the initial error $e_n^{[0]}$ is identical to x_n^*. Now, $x_n^{[1]}$ can be reconstructed from $e_n^{[1]} = x_n^* - x_n^{[1]} = S_n e_n^{[0]} = S_n x_n^*$, which gives

$$x_n^{[1]} \quad = \quad (I - S_n) x_n^* = (I - S_n) K_n^{-1} G^T (b - Kx).$$

Let $e = x^* - x$ denote the error of x as an approximation to the solution of $Kx^* = b$. Prolongation of $x_n^{[1]}$ yields $x \leftarrow x + G x_n^{[1]}$, or, equivalently,

$$e \leftarrow \left(I - G(I - S_n) K_n^{-1} G^T K \right) e.$$

In frequency space, this becomes:

$$\begin{aligned}
\begin{bmatrix} \alpha_g \\ \alpha_s \end{bmatrix} \quad &\leftarrow \quad \left(I - G_\Psi(\theta) \left(1 - S_n(\theta)\right) K_n^{-1}(\theta) G_\Psi^T(\theta) K_\Psi(\theta) \right) \begin{bmatrix} \alpha_g \\ \alpha_s \end{bmatrix} \\
&\leftarrow \quad \left(I - \begin{bmatrix} 1 \\ 0 \end{bmatrix} \left(1 - S_n(\theta)\right) \begin{bmatrix} 1 & \frac{s_1 s_2 (\tilde{c}_2 - \tilde{c}_1)}{(2+\tilde{c}_2) s_1^2 + (2+\tilde{c}_1) s_2^2} \end{bmatrix} \right) \begin{bmatrix} \alpha_g \\ \alpha_s \end{bmatrix} \\
&\leftarrow \quad \begin{bmatrix} S_n(\theta) & \frac{(S_n(\theta)-1) s_1 s_2 (\tilde{c}_2 - \tilde{c}_1)}{(2+\tilde{c}_2) s_1^2 + (2+\tilde{c}_1) s_2^2} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_g \\ \alpha_s \end{bmatrix} \\
&=: \quad S_{h(234)}(\theta) \begin{bmatrix} \alpha_g \\ \alpha_s \end{bmatrix}. \qquad (6.39)
\end{aligned}$$

The Fourier representation of steps 2, 3 and 4 of the hybrid smoother is represented by $S_{h(234)}(\theta)$. Formula (6.39) shows that the nodal part of this smoother leaves solenoidal errors untouched (lower right entry). If no such errors are present, the gradient errors are reduced in the same way as the error of the auxiliary system (upper left entry). The upper right entry represents the aliasing contribution of solenoidal errors.

With $S_{h(1)}$ and $S_{h(5)}$ denoting the Fourier domain representation with respect to the Fourier-Helmholtz basis (6.24) of the smoothers in steps 1 and 5, the frequency response of the hybrid smoother reads:

$$S_{h,\Psi}(\theta) \begin{bmatrix} \alpha_g \\ \alpha_s \end{bmatrix} \quad := \quad S_{h(5),\Psi}^{\epsilon_{post}}(\theta) \cdot S_{h(234),\Psi}(\theta) \cdot S_{h(1),\Psi}^{\epsilon_{pre}}(\theta) \cdot \begin{bmatrix} \alpha_g \\ \alpha_s \end{bmatrix}. \qquad (6.40)$$

Formula (6.40) shows that, similarly as for classical smoothers, the properties of the hybrid smoother can be studied for each frequency separately. In figure 6.3, the smoothing properties of a classical smoother and of its hybrid extension are

compared. The figures show that the hybrid smoother achieves a satisfactory smoothing of the high frequency error components, contrary to its classical counterpart.

Remark 6.6.2 The construction of K_n as the Galerkin product $G^T K G$ is optimal in the following sense. Let x^* satisfy $Kx^* = b$. If the error $e = x^* - x = K^{-1}b - x$ belongs to $\mathrm{ran}(G)$, i.e., $e = Gp$, then the exact solution of the auxiliary system $K_n x_n = G^T(b - Kx)$ is equal to p. This follows from $K_n p = G^T K G p = G^T(b - Kx)$. Hence, if the auxiliary system is solved exactly, the error e will be totally eliminated.

Remark 6.6.3 The above derivation does not hold for $\theta = (0,0)$, since $K_n((0,0)) = 0$ and $G((0,0)) = [0\ 0]^T$, which would give rise to a zero-by-zero division. This singularity of K_n is a consequence of the absence of boundary conditions, which is generic for a *local* Fourier analysis (cfr. [63], p.109). The appropriate introduction of boundary conditions in the auxiliary nodal system can be studied in a *global* Fourier analysis. For the results presented in this chapter, the nodal part of the hybrid smoother is not executed for $\theta = (0,0)$. This is motivated by the observation that the nodal part of the hybrid smoother only affects errors in $\mathrm{ran}(G)$, which belong to the kernel of the system matrix K for $\theta = (0,0)$.

6.6.3 Overlapping block smoother

The overlapping blocksmoother of [3] (see section 5.1.2) can be analyzed in $F_8(\theta)$ using the 8-color parameterization (6.13). The Fourier domain representation of the system matrix can be built directly by using the 8-color variant of formula (6.11) or by applying the similarity transformation (6.20) to the block diagonal matrix representing the operator for the harmonic frequencies of θ:

$$K_8(\theta) = T_8(\theta) \begin{bmatrix} K(\theta^{(0,0)}) & 0 & 0 & 0 \\ 0 & K(\theta^{(1,0)}) & 0 & 0 \\ 0 & 0 & K(\theta^{(0,1)}) & 0 \\ 0 & 0 & 0 & K(\theta^{(1,1)}) \end{bmatrix} T_8^{-1}(\theta).$$

First, we will consider the local block solves with center nodes having odd grid indices for the X- and Y-coordinate, i.e.:

$$\mathbf{x} \quad = \quad (2k_1 + 1, 2k_2 + 1)h, \qquad k_1, k_2 \in \mathbb{Z}. \tag{6.41}$$

The four edges containing a node of the form (6.41) have the colors 3, 4, 6 and 8 respectively (see figure 6.1). We shall collect those colors into the vector $c := [3\,4\,6\,8]$, and the remaining colors into the vector $\bar{c} := [1\,2\,5\,7]$. The values of the latter edges are *not* updated in the present step.

Consider $e(\theta)$, an error component that belongs to the subspace $F_8(\theta)$. By definition, it can be written in the form (6.13). The corresponding component of the residual, denoted as $r(\theta)$, can be found through the error equation

$K_8(\theta)e(\theta) = r(\theta)$. That residual component is also of the form (6.13). After the block update of the edges emanating from nodes of the form (6.41), the new residual is zero at all edges whose colors are in vector c. Hence, the new error, denoted as $e^{new}(\theta)$, satisfies a residual equation with zero right-hand side for those equations corresponding to edges with colors in c. Omitting the θ-dependence for notational simplicity, and using MATLAB notation, this can be written compactly as

$$K_8(c,:)\, e^{new} = 0. \tag{6.42}$$

Since the values at the edges with colors in $\bar{c}$ haven't changed, the corresponding errors remain unaltered. Hence, $e^{new}(\bar{c}) = e(\bar{c})$, and (6.42) becomes,

$$K_8(c,c)\, e^{new}(c) + K_8(c,\bar{c})\, e(\bar{c}) = 0.$$

This enables us to identify the iteration matrix, mapping $e([c\ \bar{c}])$ into $e^{new}([c\ \bar{c}])$:

$$S_{3468}([c\ \bar{c}],[c\ \bar{c}]) \quad = \quad \begin{bmatrix} 0 & -\left(K_8(c,c)\right)^{-1} K_8(c,\bar{c}) \\ 0 & I_4 \end{bmatrix}.$$

In a similar way, the iteration matrices for the sequence of the local block solves for all other possible center nodes

$$h(2k_1,2k_2), \quad h(2k_1+1,2k_2), \quad (2k_1,2k_2+1), \qquad \text{with} \qquad k_1,k_2 \in \mathbb{Z},$$

can be constructed. These matrices will be denoted S_{1257}, S_{1268} and S_{3457} respectively. The Fourier representation of the AFW-smoother itself, which is a sequence of all local block solves, reads, with respect to the 8-color parameterization (6.13):

$$\mathcal{S}_{AFW,8}(\theta) \quad = \quad S_{3457}(\theta) \cdot S_{1268}(\theta) \cdot S_{1257}(\theta) \cdot S_{3468}(\theta),$$

and with respect to the parameterization (6.18):

$$\mathcal{S}_{AFW}(\theta) \quad = \quad T_8(\theta)^{-1} \cdot \mathcal{S}_{AFW,8}(\theta) \cdot T_8(\theta).$$

Note that the order of the subiterations represented by S_{3468}, S_{1257}, S_{1268} and S_{3457} is arbitrary. In order to study the smoothing properties, we need to consider the effect of the smoother on the high frequencies only. Unfortunately, $S_{AFW}(\theta)$ is in general a dense matrix. The low harmonic frequency θ can be suppressed in an artificial way by means of the ideal coarse grid operator C (cfr. [63]):

$$C := \begin{bmatrix} 0_2 & 0_2 & 0_2 & 0_2 \\ 0_2 & I_2 & 0_2 & 0_2 \\ 0_2 & 0_2 & I_2 & 0_2 \\ 0_2 & 0_2 & 0_2 & I_2 \end{bmatrix}, \quad \text{with} \quad 0_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},\ I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Since the low harmonic frequency θ is removed, $C\mathcal{S}_{AFW}(\theta)$ can be used to represent the high frequency behavior of the AFW-smoother in $F_8(\theta)$. Consequently, the spectral radius of $C\mathcal{S}_{AFW}(\theta)$ is a measure for the smoothing properties of the AFW-smoother in $F_8(\theta)$. This function is plotted in figure 6.4.

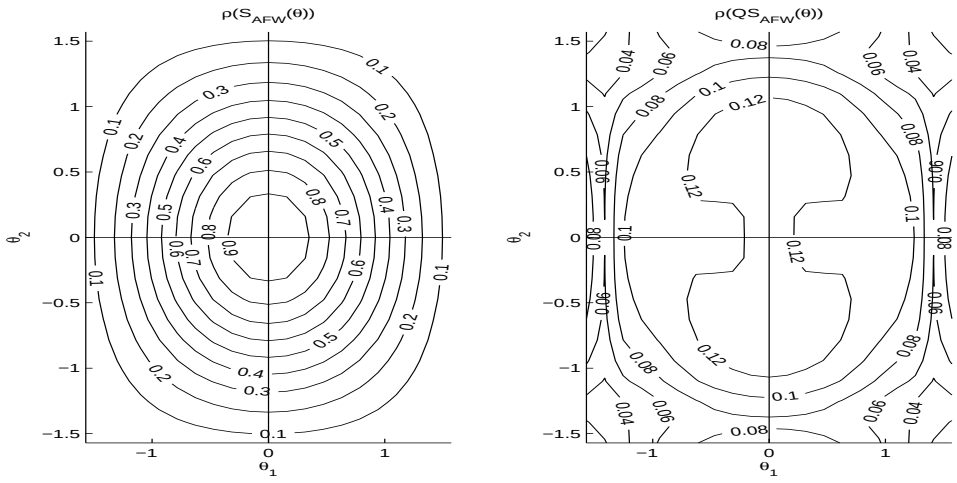


Figure 6.4: Contour plot of $\rho(\mathcal{S}_{AFW}(\theta))$ and $\rho(C\mathcal{S}_{AFW}(\theta))$ for $\sigma_h = 0.01$. Each value corresponds to the worst case reduction factor of the AFW-smoother for any combination of the 4 harmonic frequencies belonging to $F_8(\theta)$. These 4 harmonic frequencies are represented by the low frequency harmonic indicated on the axes. In the right picture, the low frequency harmonic itself is suppressed by the ideal coarse grid correction C. This gives an indication of the high frequency behavior of the AFW-smoother.

6.7 Fourier domain representation of the intergrid transfer operators

6.7.1 Coarse grid

We consider the coarse grid with mesh size $\tilde{h} = 2h$

$$\mathcal{G}_{\tilde{h}} \quad := \quad \{\mathbf{x} = \mathbf{k}\tilde{h} \mid \mathbf{k} \in \mathbb{Z}^2\}. \tag{6.43}$$

In the rest of this chapter, a superscript $\tilde{.}$ will be used to refer to the coarse grid. Similarly as (6.4), we define

$$\tilde{\varphi}(\theta) := e^{i\theta\cdot\mathbf{x}/\tilde{h}} \qquad \text{with } \theta \in \mathbb{R}^2. \tag{6.44}$$

To distinguish the evaluation operators (3.13) with respect to the fine and coarse grid, the grid size will be indicated in superscript as in (3.33). Recall that the fine and coarse evaluation operators and the fine and coarse Fourier modes are related as (3.36). Due to the displacement $\boldsymbol{\Delta} = (\Delta_1, \Delta_2)$ of the evaluation points, the aliasing of the fine Fourier modes on the coarse grid is more complicated than in an LFA for node-based problems. With $\theta \in \Theta^{\text{low}}$, $\mathbf{k} = (k_1, k_2) \in N$ and $a, b \in \{0, 1\}$, we have

$$\begin{aligned}
\varphi(\theta^{(a,b)})^{\tilde{h}}_{\mathbf{k}+\boldsymbol{\Delta}} &= e^{i\left((\theta_1+a\pi,\theta_2+b\pi)\cdot(k_1+\Delta_1,k_2+\Delta_2)\tilde{h}/h\right)} \\
&= \varphi(\theta^{(0,0)})^{\tilde{h}}_{\mathbf{k}+\boldsymbol{\Delta}} e^{2\pi i(a\Delta_1+b\Delta_2)} e^{2\pi i(ak_1+bk_2)} \\
&= \tilde{\varphi}(2\theta^{(0,0)})^{\tilde{h}}_{\mathbf{k}+\boldsymbol{\Delta}} e^{2\pi i(a\Delta_1+b\Delta_2)}.
\end{aligned} \tag{6.45}$$

6.7.2 Restriction

We consider the restriction corresponding to the transpose of the prolongation stencils (5.6). Since the horizontal and vertical components are decoupled with respect to this restriction, the decoupled parameterization (6.6) of $F_e(\theta)$ is a natural choice for the analysis of the restriction. Using (3.36) and (6.45), the restriction of the horizontal components reads:

$$\begin{aligned}
\left(R\phi_1(\theta^{(a,b)})\right)^{\tilde{h}}_{k_1+\frac{1}{2},k_2} &= \frac{1}{2}\left(\varphi(\theta^{(a,b)})^{h}_{2k_1+\frac{1}{2},2k_2} + \varphi(\theta^{(a,b)})^{h}_{2k_1+\frac{3}{2},2k_2}\right) + \\
&\quad \frac{1}{4}\begin{pmatrix} \varphi(\theta^{(a,b)})^{h}_{2k_1+\frac{1}{2},2k_2+1} + \varphi(\theta^{(a,b)})^{h}_{2k_1+\frac{3}{2},2k_2+1} \\ +\varphi(\theta^{(a,b)})^{h}_{2k_1+\frac{1}{2},2k_2-1} + \varphi(\theta^{(a,b)})^{h}_{2k_1+\frac{3}{2},2k_2-1} \end{pmatrix} \\
&= c_{1,a}(1+\tilde{c}_{2,b})\varphi(\theta^{(a,b)})^{h}_{2k_1+1,2k_2} \\
&= 2c_{1,a}c^2_{2,b}\tilde{\varphi}(2\theta^{(a,b)})^{\tilde{h}}_{k_1+\frac{1}{2},k_2} \\
&= 2c_{1,a}c^2_{2,b}(-1)^a\tilde{\varphi}(2\theta^{(0,0)})^{\tilde{h}}_{k_1+\frac{1}{2},k_2}.
\end{aligned} \tag{6.46}$$

Similarly, we get for the restriction of the vertical components:

$$\left(R\phi_2(\theta^{(a,b)})\right)^{\tilde{h}}_{k_1,k_2+\frac{1}{2}} = 2c_{2,b}c_{1,a}^2(-1)^b\tilde{\varphi}(2\theta^{(0,0)})^{\tilde{h}}_{k_1,k_2+\frac{1}{2}}. \quad (6.47)$$

Formulae (6.46) and (6.47) can be combined into the form (6.6):

$$R\Phi(\theta^{(a,b)})\begin{bmatrix}\alpha_1\\ \alpha_2\end{bmatrix} = \tilde{\Phi}(2\theta^{(0,0)})\begin{bmatrix}\tilde{\alpha}_1\\ \tilde{\alpha}_2\end{bmatrix}, \quad (6.48)$$

with

$$\begin{bmatrix}\tilde{\alpha}_1\\ \tilde{\alpha}_2\end{bmatrix} = \begin{bmatrix}2c_{1,a}c_{2,b}^2(-1)^a & 0\\ 0 & 2c_{2,b}c_{1,a}^2(-1)^b\end{bmatrix}\begin{bmatrix}\alpha_1\\ \alpha_2\end{bmatrix} =: R_\Phi(\theta^{(a,b)})\begin{bmatrix}\alpha_1\\ \alpha_2\end{bmatrix}, \quad (6.49)$$

and

$$\tilde{\Phi}(\theta) := \begin{bmatrix}\tilde{\phi}_1(\theta)^{\tilde{h}}_E & \tilde{\phi}_2(\theta)^{\tilde{h}}_E\end{bmatrix} := \begin{bmatrix}\tilde{\varphi}(\theta)^{\tilde{h}}_{E_1} & 0^{\tilde{h}}_{E_1}\\ 0^{\tilde{h}}_{E_2} & \tilde{\varphi}(\theta)^{\tilde{h}}_{E_2}\end{bmatrix}. \quad (6.50)$$

The space spanned by (6.50) is the coarse equivalent of $F_e(\theta)$, and will be denoted $\tilde{F}_e(\theta)$. Formula (6.48) shows that the spaces $F_e(\theta^{(a,b)})$ for the harmonic frequencies of $\theta^{(0,0)} \in \Theta^{\text{low}}$ are mapped to the *same* space $\tilde{F}_e(2\theta^{(0,0)})$ by the restriction operator.
Using the coarse equivalent of (6.25), i.e.,

$$\tilde{\Psi}(\theta) = 2i\tilde{\Phi}(\theta)H(\theta), \quad (6.51)$$

(6.48) and (6.49) can be transformed into:

$$R\Psi(\theta^{(a,b)})\begin{bmatrix}\alpha_g\\ \alpha_s\end{bmatrix} = \tilde{\Psi}(2\theta^{(0,0)})\begin{bmatrix}\tilde{\alpha}_g\\ \tilde{\alpha}_s\end{bmatrix}, \quad (6.52)$$

with

$$\begin{aligned}\begin{bmatrix}\tilde{\alpha}_g\\ \tilde{\alpha}_s\end{bmatrix} &= \left(H(2\theta^{(0,0)})\right)^{-1} R_\Phi(\theta^{(a,b)})H(\theta^{(a,b)})\begin{bmatrix}\alpha_g\\ \alpha_s\end{bmatrix}\\ &= \frac{1}{\tilde{s}_{1,a}^2+\tilde{s}_{2,b}^2}\begin{bmatrix}(-1)^a\tilde{s}_{1,a} & (-1)^b\tilde{s}_{2,b}\\ (-1)^b\tilde{s}_{2,b} & -(-1)^a\tilde{s}_{1,a}\end{bmatrix}R_\Phi(\theta^{(a,b)})\begin{bmatrix}s_{1,a} & s_{2,b}\\ s_{2,b} & -s_{1,a}\end{bmatrix}\\ &= \begin{bmatrix}\frac{\tilde{s}_{1,a}^2c_{2,b}^2+\tilde{s}_{2,b}^2c_{1,a}^2}{\tilde{s}_{1,a}^2+\tilde{s}_{2,b}^2} & 0\\ (-1)^{a+b}\frac{\tilde{s}_{1,a}\tilde{s}_{2,b}(c_{2,b}^2-c_{1,a}^2)}{\tilde{s}_{1,a}^2+\tilde{s}_{2,b}^2} & (-1)^{a+b}c_{1,a}c_{2,b}\end{bmatrix}\begin{bmatrix}\alpha_g\\ \alpha_s\end{bmatrix}\\ &:= R_\Psi(\theta^{(a,b)})\begin{bmatrix}\alpha_g\\ \alpha_s\end{bmatrix}.\end{aligned} \quad (6.53)$$

Remark 6.7.1 The nonzero structure of $R_\Psi(\theta)$ shows that the restriction of a fine solenoidal vector is a coarse solenoidal vector. This property corresponds to the transpose of the edge-face commutation property (5.10), which reads, with P_f, P_e, C_f and C_c denoting respectively the face and edge prolongators and the fine and coarse discrete curl operators:

$$P_f C_c = C_f P_e, \quad \text{or, equivalently,} \quad C_c^T P_f^T = P_e^T C_f^T. \tag{6.54}$$

6.7.3 Prolongation

We consider the prolongation corresponding to the stencils (5.6). The horizontal and vertical prolongation can be studied separately using the decoupled basis (6.50). This yields for the horizontal case:

$$\begin{aligned}\left(P\tilde{\phi}_1(2\theta)\right)^h_{k_1+\frac{1}{2},k_2} &= \begin{cases} \frac{1}{2}\varphi(\theta)^h_{k_1+1,k_2} & k_1,k_2 \text{ even},\\ \frac{1}{2}\varphi(\theta)^h_{k_1,k_2} & k_1 \text{ odd}, k_2 \text{ even},\\ \frac{1}{4}\left(\varphi(\theta)^h_{k_1+1,k_2-1}+\varphi(\theta)^h_{k_1+1,k_2+1}\right) & k_1 \text{ even}, k_2 \text{ odd},\\ \frac{1}{4}\left(\varphi(\theta)^h_{k_1,k_2-1}+\varphi(\theta)^h_{k_1,k_2+1}\right) & k_1,k_2 \text{ odd}, \end{cases}\\ &= \varphi(\theta)^h_{k_1+\frac{1}{2},k_2}\cdot\begin{cases} \frac{1}{2}\varphi(\theta)^h_{\frac{1}{2},0} & k_1,k_2 \text{ even},\\ \frac{1}{2}\varphi(\theta)^h_{-\frac{1}{2},0} & k_1 \text{ odd}, k_2 \text{ even},\\ \frac{1}{2}\tilde{c}_2\varphi(\theta)^h_{\frac{1}{2},0} & k_1 \text{ even}, k_2 \text{ odd},\\ \frac{1}{2}\tilde{c}_2\varphi(\theta)^h_{-\frac{1}{2},0} & k_1,k_2 \text{ odd}. \end{cases}\end{aligned} \tag{6.55}$$

Similarly, we have for the vertical prolongation $\left(P\tilde{\phi}_2(2\theta)\right)^h_{k_1,k_2+\frac{1}{2}}$:

$$\varphi(\theta)^h_{k_1,k_2+\frac{1}{2}}\cdot\begin{cases} \frac{1}{2}\varphi(\theta)^h_{0,\frac{1}{2}} & \text{if } k_1,k_2 \text{ even},\\ \frac{1}{2}\tilde{c}_1\varphi(\theta)^h_{0,\frac{1}{2}} & \text{if } k_1 \text{ odd}, k_2 \text{ even},\\ \frac{1}{2}\varphi(\theta)^h_{0,-\frac{1}{2}} & \text{if } k_1 \text{ even}, k_2 \text{ odd},\\ \frac{1}{2}\tilde{c}_1\varphi(\theta)^h_{0,-\frac{1}{2}} & \text{if } k_1,k_2 \text{ odd}. \end{cases} \tag{6.56}$$

Using (6.19) and lemma 6.3.1, (6.55) can be written in the form (6.18), with:

$$\begin{bmatrix} \beta_1^{(0,0)}\\ \beta_1^{(1,0)}\\ \beta_1^{(0,1)}\\ \beta_1^{(1,1)} \end{bmatrix} = \left(\Gamma_{\frac{1}{2},0}\right)^{-1}\begin{bmatrix} \frac{1}{2}\varphi(\theta)^h_{\frac{1}{2},0}\\ \frac{1}{2}\varphi(\theta)^h_{-\frac{1}{2},0}\\ \frac{1}{2}\tilde{c}_2\varphi(\theta)^h_{\frac{1}{2},0}\\ \frac{1}{2}\tilde{c}_2\varphi(\theta)^h_{-\frac{1}{2},0} \end{bmatrix} = \frac{1}{4}\begin{bmatrix} c_1(1+\tilde{c}_2)\\ s_1(1+\tilde{c}_2)\\ c_1(1-\tilde{c}_2)\\ s_1(1-\tilde{c}_2) \end{bmatrix}. \tag{6.57}$$

Note that the trigonometric functions figuring in (6.57) are evaluated for $\theta = \theta^{(0,0)}$. Using the trigonometric identities $1+\cos(2\alpha) = 2\cos^2(\alpha)$, $\cos(\alpha+\frac{\pi}{2}) = -\sin(\alpha)$ and $\cos(\alpha+\pi) = -\cos(\alpha)$, (6.57) can be recast into a general formula:

$$\beta_1^{(a,b)} = \frac{1}{2}(-1)^a c_{1,a}c_{2,b}^2, \qquad a,b\in\{0,1\}. \tag{6.58}$$

Similarly, we have for the vertical prolongation:

$$\beta_2^{(a,b)} \;=\; \frac{1}{2}(-1)^b c_{2,b} c_{1,a}^2, \qquad a,b \in \{0,1\}. \tag{6.59}$$

The formulae (6.58) and (6.59) can be combined into the format of (6.6):

$$P\tilde{\Phi}(2\theta^{(0,0)}) \begin{bmatrix} \tilde{\alpha}_1 \\ \tilde{\alpha}_2 \end{bmatrix} \;=\; \sum_{a,b\in\{0,1\}} \Phi(\theta^{(a,b)}) \begin{bmatrix} \alpha_1^{(a,b)} \\ \alpha_2^{(a,b)} \end{bmatrix}, \tag{6.60}$$

with

$$\begin{aligned} \begin{bmatrix} \alpha_1^{(a,b)} \\ \alpha_2^{(a,b)} \end{bmatrix} \;&=\; \frac{1}{2} \begin{bmatrix} (-1)^a c_{1,a} c_{2,b}^2 & 0 \\ 0 & (-1)^b c_{2,b} c_{1,a}^2 \end{bmatrix} \begin{bmatrix} \tilde{\alpha}_1 \\ \tilde{\alpha}_2 \end{bmatrix} \\ &=:\; P_\Phi(\theta^{(a,b)}) \begin{bmatrix} \tilde{\alpha}_1 \\ \tilde{\alpha}_2 \end{bmatrix}. \end{aligned} \tag{6.61}$$

Using (6.51), formula (6.60) can be transformed into:

$$P\tilde{\Psi}(2\theta^{(0,0)}) \begin{bmatrix} \tilde{\alpha}_g \\ \tilde{\alpha}_s \end{bmatrix} \;=\; \sum_{a,b\in\{0,1\}} \Psi(\theta^{(a,b)}) \begin{bmatrix} \alpha_g^{(a,b)} \\ \alpha_s^{(a,b)} \end{bmatrix}, \tag{6.62}$$

with

$$\begin{aligned} \begin{bmatrix} \alpha_g^{(a,b)} \\ \alpha_s^{(a,b)} \end{bmatrix} \;&=\; \Big(H(\theta^{(a,b)})\Big)^{-1} P_\Phi(\theta^{(a,b)}) H(2\theta^{(0,0)}) \begin{bmatrix} \tilde{\alpha}_g \\ \tilde{\alpha}_s \end{bmatrix} \\ &=\; \begin{bmatrix} c_{1,a}^2 c_{2,b}^2 & (-1)^{a+b}\, \tilde{s}_{1,a}\tilde{s}_{2,b} \frac{c_{2,b}^2 - c_{1,a}^2}{4\left(s_{1,a}^2 + s_{2,b}^2\right)} \\ 0 & (-1)^{a+b}\, c_{1,a} c_{2,b} \frac{\tilde{s}_{1,a}^2 + \tilde{s}_{2,b}^2}{4\left(s_{1,a}^2 + s_{2,b}^2\right)} \end{bmatrix} \begin{bmatrix} \tilde{\alpha}_g \\ \tilde{\alpha}_s \end{bmatrix} \\ &=:\; P_\Psi(\theta^{(a,b)}) \begin{bmatrix} \tilde{\alpha}_g \\ \tilde{\alpha}_s \end{bmatrix}. \end{aligned} \tag{6.63}$$

Remark 6.7.2 The nonzero structure of $P_\Psi(\theta^{(a,b)})$ shows that the prolongation of a coarse gradient vector is a fine gradient vector. This corresponds to the node-edge commutation property (5.9).

Remark 6.7.3 Similarly as for standard nodal prolongation and restriction, the 4-frequency representation of the restriction operator is equal to the scaled transpose of the 4-frequency representation of the prolongation operator :

$$\begin{bmatrix} R_\Phi(\theta^{(0,0)}) & R_\Phi(\theta^{(1,0)}) & R_\Phi(\theta^{(0,1)}) & R_\Phi(\theta^{(1,1)}) \end{bmatrix} \;=\; 4 \begin{bmatrix} P_\Phi(\theta^{(0,0)}) \\ P_\Phi(\theta^{(1,0)}) \\ P_\Phi(\theta^{(0,1)}) \\ P_\Phi(\theta^{(1,1)}) \end{bmatrix}^T.$$

Remark 6.7.4 Taking into account that $\tilde{s}_{i,a}^2 = \tilde{s}_i^2$ for $a \in \{0,1\}$, it can be seen that the formulae (6.63) and (6.53) are related as:

$$R_\Psi(\theta^{(a,b)}) \quad = \quad 4\frac{s_{1,a}^2 + s_{2,b}^2}{\tilde{s}_1^2 + \tilde{s}_2^2}\left(P_\Psi(\theta^{(a,b)})\right)^T = 4\frac{\det(S(\theta^{(a,b)}))}{\det(\tilde{S}(2\theta^{(0,0)}))}\left(P_\Psi(\theta^{(a,b)})\right)^T.$$

The additional scaling factor arises from the fact that the transformation matrices $H(\theta^{(a,b)})$ and $H(2\theta^{(0,0)})$ are not normalized.

Remark 6.7.5 The Fourier symbols $R_\Phi(\theta^{(a,b)})$ and $P_\Phi(\theta^{(a,b)})$ coincide with the Fourier symbols of the related grid transfer operators for velocities for discretizations on staggered grids (see for instance [63]).

6.8 Multigrid cycle

We consider a standard two-level multigrid cycle (cfr. section 3.2), We suppose the coarse system matrix K_c to be built as the Galerkin product RKP.

Due to the aliasing of the harmonic frequencies $\theta^{(a,b)}$ of $\theta \in \Theta^{\text{low}}$ on the coarse grid (see section 6.7.1), the multigrid cycle needs to be studied in $F_8(\theta)$. For this purpose, the Fourier domain representation of the multigrid cycle can be constructed from its constituent parts, similarly as in section 3.3.5:

$$C \quad = \quad S_{post}^{\mu_{post}}\left(I - PK_c^{-1}RK\right)S_{pre}^{\mu_{pre}}. \tag{6.64}$$

Let $\mathcal{K}(\theta), \mathcal{S}_{pre}(\theta), \mathcal{S}_{post}(\theta) \in \mathbb{C}^{8\times 8}$ represent the operator K and the pre- and postsmoothing operators in $F_8(\theta)$ with respect to the parameterization (6.18). The corresponding Fourier domain representations of the restriction and prolongation operators and the coarse system matrix read:

$$\begin{aligned}
\mathcal{R}(\theta) \quad &:= \quad \left[\begin{array}{cccc} R_\Phi(\theta^{(0,0)}) & R_\Phi(\theta^{(1,0)}) & R_\Phi(\theta^{(0,1)}) & R_\Phi(\theta^{(1,1)}) \end{array}\right] \in \mathbb{C}^{2\times 8},\\
\mathcal{P}(\theta) \quad &:= \quad \left[\begin{array}{cccc} P_\Phi(\theta^{(0,0)})^T & P_\Phi(\theta^{(1,0)})^T & P_\Phi(\theta^{(0,1)})^T & P_\Phi(\theta^{(1,1)})^T \end{array}\right]^T \in \mathbb{C}^{8\times 2},\\
\mathcal{K}_c(\theta) \quad &:= \quad \mathcal{R}(\theta)\mathcal{K}(\theta)\mathcal{P}(\theta) \in \mathbb{C}^{2\times 2}.
\end{aligned}$$

The Fourier domain representation of the multigrid cycle in $F_8(\theta)$ is obtained by replacing every component of (6.64) by its Fourier symbol:

$$\mathcal{C}(\theta) \quad = \quad \mathcal{S}_{post}(\theta)^{\mu_{post}}\left(I - \mathcal{P}(\theta)\mathcal{K}_c(\theta)^{-1}\mathcal{R}(\theta)\mathcal{K}(\theta)\right)\mathcal{S}_{pre}(\theta)^{\mu_{pre}}. \tag{6.65}$$

Now, the asymptotic convergence factor can be calculated as

$$\rho_\infty \quad = \quad \sup\{\rho(\mathcal{C}(\theta)) \ : \ \theta \in \Theta^{\text{low}}\}. \tag{6.66}$$

Figures 6.5 and 6.6 show the asymptotic convergence factors of a 2-level multigrid cycle using the hybrid smoother and the AFW-smoother as a function of σ_h. Of particular interest are the purely real and purely imaginary cases, depicted in figure 6.7. The latter corresponds to the time-harmonic curl-curl equation. The former arises, e.g., in the backward Euler discretization of the time-dependent curl-curl equation.

Remark 6.8.1 For $\theta = (0,0)$ and $\sigma_h = 0$, the above derivation does not hold, because $K_c(\theta)$ is singular. This is a consequence of the absence of boundary conditions, which is generic for a *local* Fourier analysis (cfr. [63], p.109). To avoid this complication, $\sigma_h = 0$ has been replaced by $\sigma_h = 10^{-6}$ to generate figures 6.5 and 6.6.

6.9 Remarks about the numerical tests

All results in this chapter have been obtained by LFA for the discretization of the two-dimensional curl-curl equation (5.2) on a square with periodic boundary conditions. A local Fourier analysis yields exact results for this problem setting. All analytical results were checked numerically. This can be done easily for all multigrid components presented in this chapter, except for the lexicographic Gauss-Seidel smoother. The classical implementation of Gauss-Seidel performs a sequence of local solves for each edge e, i.e.:

$$x(e) \quad = \quad \frac{b(e) - \sum_{c \neq e} K(e,c)x(c)}{K(e,e)}. \tag{6.67}$$

This is not compatible with the use of the *same* stencil for *all* edges, as required for the LFA. This can be seen by considering the first and the last edge to be updated in the iteration. The update of the first edge would obviously use the *old* values of all its neighbour edges, while the update of the last edge would use the *updated* values of all its neighbour edges. In order to check the correctness of the Gauss-Seidel analysis, an idealized implementation of the smoother has been used, which corresponds exactly to the scheme that is analyzed. In particular, the Gauss-Seidel smoother has been implemented as a matrix splitting iteration $x \leftarrow x + \tilde{K}^{-1}(b - Kx)$, with $\tilde{K}$ corresponding to the update patterns of figure 6.2. $\tilde{K}$ reads for each horizontal edge h and vertical edge v, using the indices indicated in figure 6.2:

$$\begin{aligned} \tilde{K}(h,[h\ s\ sw]) &= K(h,[h\ s\ sw]), \\ \tilde{K}(v,[v\ w\ nw\ sw\ se]) &= K(v,[v\ w\ nw\ sw\ se]). \end{aligned}$$

The remaining entries of matrix $\tilde{K}$ are set to zero. Note that the resulting smoothing procedure is *implicit* (cfr. Remark 4.6.4. in [63]), since it is not possible any more to transform this $\tilde{K}$ into a triangular matrix by means of a permutation.

All asymptotic convergence factors calculated by means of the LFA in this chapter were checked numerically by applying 2-level multigrid iterations for a zero righthand side and a random initial guess. This procedure benefits from the so-called non-normalized floating point representation, which allows the norm of the residual to be reduced much more than with a random righthand side. For all results presented in this chapter, it was possible to reduce the relative

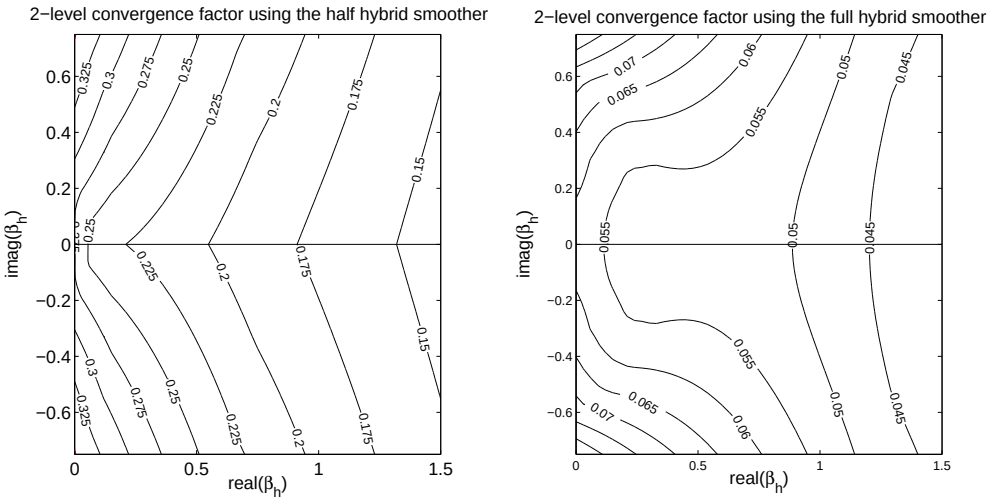


Figure 6.5: Asymptotic convergence factors as a function of σ_h for a 2-level multigrid cycle with $\mu_{pre} = \mu_{post} = 1$ using the half hybrid smoother (left) and the full hybrid smoother (right) consisting of node-based lexicographic Gauss-Seidel iterations in edge space and lexicographic Gauss-Seidel iterations in nodal space.

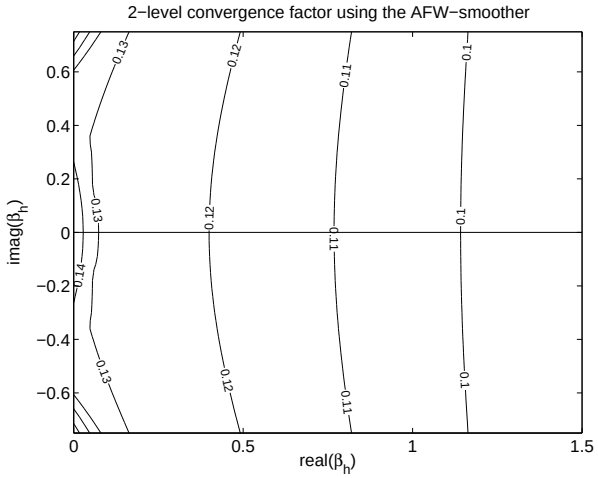


Figure 6.6: Asymptotic convergence factors as a function of σ_h for a 2-level multigrid cycle with $\mu_{pre} = \mu_{post} = 1$ using the AFW-smoother.

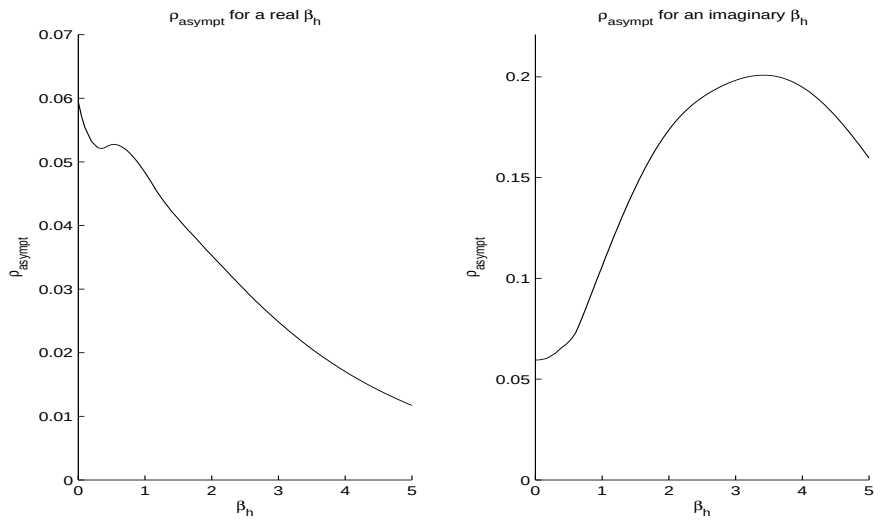


Figure 6.7: Asymptotic convergence factors as a function of a purely real (left) and a purely imaginary (right) σ_h for a 2-level multigrid cycle with $\mu_{pre} = \mu_{post} = 1$ using the full hybrid smoother consisting of node-based lexicographic Gauss-Seidel iterations in edge space and lexicographic Gauss-Seidel iterations in nodal space.

difference between the analytical and numerical values for the convergence factors to 0.1% by increasing the number of iterations. Table 6.1 shows an excerpt of the convergence history of a 2-level V(1,1) multigrid iteration using the full hybrid smoother.

Remark 6.9.1 We have also compared the the analytically obtained convergence factors using the idealized Gauss-Seidel scheme with the numerically computed factors obtained with the *classical* implementation of the lexicographic Gauss-Seidel method. They match fairly well, as can be seen from the values in table 6.2. Note, however, the growing deviation between the two convergence factors as σ_h approaches zero, i.e., as the discrete system becomes closer to being singular.

6.10 Generalization

The separate treatment of edges with different orientations, as introduced in section 6.3, can be applied to any translation-invariant set of stencils on a translation-invariant grid. More precisely, it can be applied when it is possible to represent the grid by means of a *single* node-edge pattern (see figure 6.8) and the operator by a stencil depending *only* on the orientation of the edge. In this case, theorems 6.3.1 and 6.3.2 can be generalized. For instance, the regular

Table 6.1: Convergence history of 2-level V(1,1) multigrid iterations using the full hybrid smoother for the finite element discretization of the two-dimensional curl-curl equation (5.2) with $\sigma_h = 0.1$ on a quadrilateral grid with 24×24 nodes. The boundary conditions are periodic. The righthand side is zero and the initial guess is random. The analytical convergence factor ρ_{anal} is calculated using LFA, the numerical convergence factor is calculated as $\rho_{num}^{[\nu]} = \|b - Kx^{[\nu]}\| / \|b - Kx^{[\nu-1]}\|$.

iteration	$\|\|b - Kx\|\|$	ρ_{num}	$\frac{\lvert\rho_{num} - \rho_{anal}\rvert}{\rho_{anal}}$
190	$9.21065e-240$	0.055395131977695	0.003494%
191	$5.10226e-241$	0.055395193735180	0.003382%
192	$2.82641e-242$	0.055395253466549	0.003274%
193	$1.56570e-243$	0.055395311241170	0.003170%
194	$8.67324e-245$	0.055395367125882	0.003069%
195	$4.80458e-246$	0.055395421185096	0.002971%
196	$2.66152e-247$	0.055395473480890	0.002877%
197	$1.47436e-248$	0.055395524073103	0.002786%
198	$8.16731e-250$	0.055395573019418	0.002697%
199	$4.52433e-251$	0.055395620375452	0.002612%
200	$2.50628e-252$	0.055395666194833	0.002529%

Table 6.2: Asymptotic convergence factor of a two-level cycle V(1,1) using the full hybrid smoother with lexicographic Gauss-Seidel subsmoothers as a function of σ_h. The analytical result is obtained using LFA with the idealized implementation (6.68) of the Gauss-Seidel subsmoothers of the hybrid smoother. The numerical result is the convergence factor after 100 iterations, and is obtained with the classical implementation (6.67) of the subsmoothers. Both the analytical and numerical results are generated for a grid with 100×100 nodes.

σ_h	0.1	0.5	1	1.5
numerical	0.0934	0.0693	0.0540	0.0438
analytical	0.0556	0.0527	0.0483	0.0411

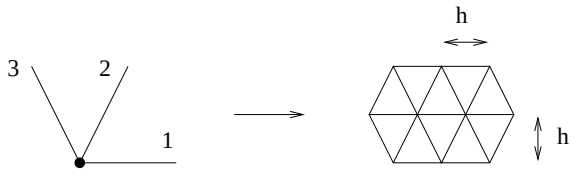


Figure 6.8: two-dimensional triangular mesh generated by a single node-edge pattern for each node. The node-edge pattern relates each node to edges of 3 different orientations.

two-dimensional triangular mesh of figure 6.8 contains 3 different types of edges. Correspondingly, 3 stencils will be needed to represent a discrete edge operator L and each frequency will be related to a three-dimensional eigenspace of L. Similarly as (6.9), the decoupled basis for this space corresponds to the columns of:

$$\Phi(\theta) \quad = \quad \begin{bmatrix} \varphi(\theta)_{E_1} & 0_{E_1} & 0_{E_1} \\ 0_{E_2} & \varphi(\theta)_{E_2} & 0_{E_2} \\ 0_{E_3} & 0_{E_3} & \varphi(\theta)_{E_3} \end{bmatrix}.$$

In a similar way, this approach can be generalized for face-based problems in $H(\mathrm{div})$ represented by translation-invariant stencils on a translation-invariant grid.

6.11 Conclusion

We presented a local Fourier analysis for multigrid for the finite element discretization of the two-dimensional curl-curl equation on a quadrilateral grid. Our approach is based on the decoupling in the Fourier domain of edges with different orientation. In this way, each frequency is related to a two-dimensional space. This procedure can be used to study the smoothing properties of multigrid smoothers and the asymptotic convergence behavior of the multigrid cycle. We identified in the Fourier domain several properties which are important in the context of the curl-curl equation, such as the Helmholtz splitting, topological identities and the node-edge and edge-face commutation properties. We showed how our approach can be extended to some other types of translation-invariant meshes and to translation-invariant problems in $H(\mathrm{curl})$ and $H(\mathrm{div})$.

Chapter 7

A new family of prolongation operators

7.1 Introduction

In this chapter, we will develop a new procedure to construct an edge prolongator P_e, which satisfies the node-edge commutation property (5.9). This edge prolongator will be derived from an arbitrary nodal prolongator P_n satisfying the partition of unity property (4.22). The resulting edge prolongation scheme will be used in AMG methods for the curl-curl equation (7.2). Numerical results will be presented to show that the resulting AMG methods are very efficient, both in terms of convergence speed and solve time.

A face and volume prolongator P_f and P_v satisfying the edge-face and face-volume commutation properties (5.10) and (5.11) respectively, can be derived from P_n by similar procedures. These procedures will reveal that a nodal prolongator implies a coarse gradient, curl and divergence, which together represent a coarse topology. We will show that the properties of this implied topology determine the memory complexity of AMG hierarchies for the div-grad, curl-curl and grad-div equations

$$
\begin{aligned}
-\operatorname{div}(\alpha \operatorname{grad}(X)) + \beta X &= b, && (7.1)\\
\operatorname{curl}(\alpha \operatorname{curl}(\vec{X})) + \beta \vec{X} &= \vec{b}, && (7.2)\\
-\operatorname{grad}(\alpha \operatorname{div}(\vec{X})) + \beta \vec{X} &= \vec{b}. && (7.3)
\end{aligned}
$$

The coarse topological operators and the edge, face and volume prolongators derived from the same nodal prolongator are connected by an exact coarse de Rham sequence (5.8), similarly as in geometric multigrid methods. This property essentially shows that the prolongators presented in this chapter transfer the typical characteristics of div-grad, curl-curl and grad-div problems to the coarse levels of a multigrid hierarchy. This is a requirement for the recursive application of the AMG coarsening and prolongation algorithms.

This chapter is organized as follows. In section 7.2, the formulas for the edge, face and volume prolongator will be derived. The implied coarse topology and the requirements for the normalization of the prolongators will be discussed in section 7.3 and 7.4. In section 7.5, the commutation properties and the existence of the coarse de Rham complex will be proved. In section 7.6, it will be shown that the resulting coarse system matrices have the same differential structure as the original discretization matrix. This is required for the application of the finest level smoother to make sense as a smoother on the coarse levels. Section 7.8 deals with implementation issues. In section 7.7, it will be shown that the presented prolongation principle, which is tailored for simplicial meshes, can be applied to quadrilateral and hexahedral meshes as well. Finally, in section 7.9, this prolongation principle will be applied in AMG for curl-curl problems, and some numerical results will be presented.

Throughout this chapter, we will assume that the fine mesh is simplicial, unless mentioned otherwise.

7.2 Prolongators derived from a nodal prolongator

In this section, an edge, face and volume prolongator will be derived from a nodal prolongator P_n satisfying the partition of unity property (5.9).

7.2.1 Edge prolongator

We will build coarse edge elements $\vec{E}_{IJ}$ by applying the analytical formula (2.64) for edge elements on a simplicial mesh to the coarse nodal elements (4.20) represented by the columns of P_n:

$$\begin{aligned}
\vec{E}_{IJ} &= N_I \ \mathrm{grad}(N_J) - N_J \ \mathrm{grad}(N_I) \qquad (7.4)\\
&= \left(\sum_k P_n(k,I)N_k\right) \mathrm{grad}\left(\sum_l P_n(l,J)N_l\right)\\
&\qquad - \left(\sum_l P_n(l,J)N_l\right) \mathrm{grad}\left(\sum_k P_n(k,I)N_k\right)\\
&= \sum_k \sum_l P_n(k,I)P_n(l,J)\left(N_k \,\mathrm{grad}(N_l) - N_l \,\mathrm{grad}(N_k)\right).
\end{aligned}$$

Since the fine mesh is simplicial, (2.64) can be used to write the coarse edge element $\vec{E}_{IJ}$ as a linear combination of fine edge elements:

$$\vec{E}_{IJ} = \sum_k \sum_l P_n(k,I)P_n(l,J)\vec{E}_{kl}. \qquad (7.5)$$

Considering that $\vec{E}_{kl} = -\vec{E}_{lk}$, this sum can be reorganized as:

$$\begin{aligned}\vec{E}_{IJ} &= \sum_{k<l} \left(P_n(k,I)P_n(l,J) - P_n(l,I)P_n(k,J)\right)\vec{E}_{kl} \\ &= \sum_{k<l} \begin{vmatrix} P_n(k,I) & P_n(k,J) \\ P_n(l,I) & P_n(l,J) \end{vmatrix} \vec{E}_{kl} \\ &:= \sum_{k<l} P_e(kl,IJ)\vec{E}_{kl}. \end{aligned} \tag{7.6}$$

The latter equality shows that the entries of the edge prolongator P_e are the determinants of 2×2 submatrices of P_n, and that $\vec{E}_{JI} = -\vec{E}_{IJ}$. For each combination of coarse nodes I and J for which $\vec{E}_{IJ} \neq 0$, either $\vec{E}_{IJ}$ or $\vec{E}_{JI}$ will be used in the calculations. This amounts to choosing an orientation for the coarse edge element, similarly as on the fine level.

Next, we will examine whether the coarse edge elements (7.6) can be considered genuine edge elements. For this purpose, we will examine whether they satisfy the requirements for Whitney elements, listed in section 2.5.1:

1. Since the coarse edge elements are linear combinations of fine edge elements, they are *conforming* in $H(\mathrm{curl}, \Omega)$.

2. Formula (7.4) shows that each coarse edge element $\vec{E}_{IJ}$ is associated with two coarse nodal elements N_I and N_J. Similarly as on the fine level, a coarse edge can be associated with each nonzero coarse edge element. The coarse edge e_{IJ} associated with the coarse edge element $\vec{E}_{IJ}$ can be defined by describing its coarse start and end nodes n_I and n_J:

   $$\mathrm{start}(e_{IJ}) := n_I, \quad \text{and} \quad \mathrm{end}(e_{IJ}) := n_J. \tag{7.7}$$

   Note that this is a formal definition, which only identifies the start and end nodes of a coarse edge. It does not allow a coarse edge to be traced on the fine mesh as a sequence of fine edges. In section 7.4, it will be shown that for this purpose, the coarse edge elements need to be normalized. This is not the case in general for the coarse edge elements obtained by (7.6).

3. From (7.4), it immediately follows that the coarse edge elements have a *local support*. More precisely, the support of $\vec{E}_{IJ}$ is a subset of the intersection of the supports of N_I and N_J:

   $$\mathrm{supp}(\vec{E}_{IJ}) \quad \subset \quad \mathrm{supp}(N_I) \cap \mathrm{supp}(N_J). \tag{7.8}$$

   In the general case, the lefthand and righthand side of (7.8) are identical.

4. The gradient of each coarse nodal element N_I equals a linear combination

of coarse edge elements:

$$\begin{aligned}
\operatorname{grad}(N_I) &= \operatorname{grad}\left(\sum_k P_n(k,I)N_k\right) \\
&= \sum_k P_n(k,I)\operatorname{grad}(N_k) \\
&= -\sum_k P_n(k,I)\sum_l \vec{E}_{kl} \\
&= -\sum_J\sum_l\sum_k P_n(k,I)P_n(l,J)\vec{E}_{kl} \quad = \quad \sum_J \vec{E}_{JI}. \qquad (7.9)
\end{aligned}$$

Here, we used subsequently (4.20), the linearity of the gradient operator, (2.79), (4.22) and (7.5). Consequently, the coarse equivalent of the *embedding property* (2.60) holds.

The only formal requirement in the derivation of (7.6) is that (2.64) holds on the fine level. This is the case for tetrahedral meshes. If P_e is built using (7.6), then, *by construction*, (2.64) holds on the coarse level as well. This allows (7.6) to be applied recursively.

Remark 7.2.1 The above also holds for triangular fine meshes. For quadrilateral and hexahedral meshes, we refer to section 7.7.

7.2.2 Face prolongator

We will build coarse face elements $\vec{F}_{IJK}$ by applying the analytical formula (2.65) for face elements on a tetrahedral mesh to the coarse nodal elements (4.20) represented by the columns of P_n and the coarse edge elements (7.6):

$$\begin{aligned}
\vec{F}_{IJK} &= N_I \operatorname{curl}(\vec{E}_{JK}) + N_J \operatorname{curl}(\vec{E}_{KI}) + N_K \operatorname{curl}(\vec{E}_{IJ}) \qquad (7.10) \\
&= \left(\sum_r P_n(r,I)N_r\right)\operatorname{curl}\left(\sum_s\sum_t P_n(s,J)P_n(t,K)\vec{E}_{st}\right) \\
&\quad + \left(\sum_s P_n(s,J)N_s\right)\operatorname{curl}\left(\sum_t\sum_r P_n(t,K)P_n(r,I)\vec{E}_{tr}\right) \\
&\quad + \left(\sum_t P_n(t,K)N_t\right)\operatorname{curl}\left(\sum_r\sum_s P_n(r,I)P_n(s,J)\vec{E}_{rs}\right)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_r \sum_s \sum_t P_n(r,I)P_n(s,J)P_n(t,K)N_r \operatorname{curl}(\vec{E}_{st}) \\
&\quad + \sum_s \sum_t \sum_r P_n(s,J)P_n(t,K)P_n(r,I)N_s \operatorname{curl}(\vec{E}_{tr}) \\
&\quad + \sum_t \sum_r \sum_s P_n(t,K)P_n(r,I)P_n(s,J)N_t \operatorname{curl}(\vec{E}_{rs}) \\
&= \sum_r \sum_s \sum_t P_n(r,I)P_n(s,J)P_n(t,K) \begin{pmatrix} N_r \operatorname{curl}(\vec{E}_{st}) \\ + N_s \operatorname{curl}(\vec{E}_{tr}) \\ + N_t \operatorname{curl}(\vec{E}_{rs}) \end{pmatrix}.
\end{aligned}
$$

Since the fine mesh is simplicial, (2.65) can be used to write the coarse face element $\vec{F}_{IJK}$ as a linear combination of fine face elements:

$$
\vec{F}_{IJK} = \sum_r \sum_s \sum_t P_n(r,I)P_n(s,J)P_n(t,K)\vec{F}_{rst}. \tag{7.11}
$$

Taking into account that

$$
\vec{F}_{rst} = \vec{F}_{str} = \vec{F}_{trs} = -\vec{F}_{srt} = -\vec{F}_{tsr} = -\vec{F}_{rts},
$$

this sum can be reorganized as:

$$
\begin{aligned}
&\sum_{r<s<t} \begin{pmatrix} P_n(r,I)P_n(s,J)P_n(t,K) - P_n(r,I)P_n(t,J)P_n(s,K) \\ - P_n(s,I)P_n(r,J)P_n(t,K) + P_n(s,I)P_n(t,J)P_n(r,K) \\ + P_n(t,I)P_n(r,J)P_n(s,K) - P_n(t,I)P_n(s,J)P_n(r,K) \end{pmatrix} \vec{F}_{rst} \\
= &\sum_{r<s<t} \begin{vmatrix} P_n(r,I) & P_n(r,J) & P_n(r,K) \\ P_n(s,I) & P_n(s,J) & P_n(s,K) \\ P_n(t,I) & P_n(t,J) & P_n(t,K) \end{vmatrix} \vec{F}_{rst} \\
:= &\sum_{r<s<t} P_f(rst,IJK)\vec{F}_{rst}.
\end{aligned} \tag{7.12}
$$

The latter equality shows that the entries of the face prolongator P_f are the determinants of 3×3 submatrices of P_n, and that each permutation of the coarse indices I, J and K yields a representative of either $\vec{F}_{IJK}$ or $\vec{F}_{JIK}$. For each combination of coarse nodes I, J and K for which $\vec{F}_{IJK} \neq 0$, either $\vec{F}_{IJK}$ or $\vec{F}_{JIK}$ will be used in the calculations. This amounts to choosing an orientation for the coarse face element, similarly as on the fine level.

Next, we will examine whether the coarse face elements (7.12) can be considered genuine face elements. For this purpose, we will examine whether they satisfy the requirements for Whitney elements, listed in section 2.5.1:

1. Since the coarse face elements are linear combinations of fine face elements, they are *conforming* in $H(\operatorname{div},\Omega)$.

2. Formula (7.11) shows that each coarse face element $\vec{F}_{IJK}$ is associated with three coarse nodal elements N_I, N_J and N_K and three coarse edge elements $\vec{E}_{IJ}$, $\vec{E}_{JK}$ and $\vec{E}_{KI}$. Similarly as on the fine level, a coarse face

can be associated with each nonzero coarse face element. These coarse faces are logically triangular, i.e. they contain 3 coarse nodes. The coarse face f_{IJK} associated with the coarse face element $\vec{F}_{IJK}$ can be defined by describing its oriented boundary, which consists of the oriented coarse edges e_{IJ}, e_{JK} and e_{KI}:

$$\partial f_{IJK} \quad := \quad \{e_{IJ}, e_{JK}, e_{KI}\}. \tag{7.13}$$

Note that this is a formal definition, which does not allow a coarse face or its boundary to be traced on the fine mesh as a sequence of fine faces or edges. In section 7.4, it will be shown that for this purpose, the coarse face elements need to be normalized. This is not the case in general for the coarse face elements obtained by (7.12).

3. From (7.11) and (7.8), it immediately follows that the coarse face elements have a *local support*. More precisely, the support of $\vec{F}_{IJK}$ is a subset of the intersection of the supports of N_I, N_J and N_K:

$$\mathrm{supp}(\vec{F}_{IJK}) \quad \subset \quad \mathrm{supp}(N_I) \cap \mathrm{supp}(N_J) \cap \mathrm{supp}(N_K). \tag{7.14}$$

In the general case, the lefthand and righthand side of (7.14) are identical.

4. The curl of each coarse edge element $\vec{E}_{IJ}$ equals a linear combination of coarse face elements:

$$\begin{aligned}
\mathrm{curl}(\vec{E}_{IJ}) &= \sum_r \sum_s P_n(r,I) P_n(s,J)\, \mathrm{curl}(\vec{E}_{rs}) \\
&= \sum_r \sum_s P_n(r,I) P_n(s,J) \sum_t \vec{F}_{rst} \\
&= \sum_K \sum_r \sum_s \sum_t P_n(r,I) P_n(s,J) P_n(t,K) \vec{F}_{rst} \\
&= \sum_K \vec{F}_{IJK}.
\end{aligned} \tag{7.15}$$

Here, we used subsequently (7.5), the linearity of the curl operator, (2.80), (4.22) and (7.11). Consequently, the coarse equivalent of the *embedding property* (2.61) holds.

The only formal requirement in the derivation of (7.12) is that (2.65) holds on the fine level. This is the case for tetrahedral meshes. Note that if P_f is built using (7.12), then *by construction*, (2.65) holds on the coarse level as well. This allows (7.12) to be applied recursively.

Remark 7.2.2 In the 2D parallel mode for a computational domain Ω in the XY-plane (see section 2.5.1), the face elements are associated with the edges of the mesh, which can be considered degenerate representations of faces perpendicular to Ω. The analytical formula for these face elements reads:

$$\vec{F}_{ij} \quad = \quad \vec{E}_{ij} \times \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

The application of this formula to the coarse level yields

$$\vec{F}_{IJ} \quad = \quad \sum_{k<l} \begin{vmatrix} P_n(k,I) & P_n(k,J) \\ P_n(l,I) & P_n(l,J) \end{vmatrix} \vec{F}_{kl},$$

which is the same formula as for edge elements (7.6). This corresponds to the fact that the grad-div operator is equivalent to a curl-curl operator in this case (cfr. (2.48)). For quadrilateral and hexahedral meshes, we refer to section 7.7.

7.2.3 Volume prolongator

In an analogous way, coarse volume elements can be derived from the formula (2.66) for volume elements V_{ijkl} on tetrahedral meshes. The resulting coarse volumes are logically tetrahedral, i.e. they contains 4 coarse nodes. The coarse volume elements are linear combinations of fine volume elements:

$$\begin{aligned} V_{IJKL} \quad &= \quad N_I \operatorname{div}(\vec{F}_{JKL}) + N_J \operatorname{div}(\vec{F}_{KLI}) \qquad (7.16) \\ &\quad +N_K \operatorname{div}(\vec{F}_{LIJ}) + N_L \operatorname{div}(\vec{F}_{IJK}) \\ &= \quad \sum_{r<s<t<u} \begin{vmatrix} P_n(r,I) & P_n(r,J) & P_n(r,K) & P_n(r,L) \\ P_n(s,I) & P_n(s,J) & P_n(s,K) & P_n(s,L) \\ P_n(t,I) & P_n(t,J) & P_n(t,K) & P_n(t,L) \\ P_n(u,I) & P_n(u,J) & P_n(u,K) & P_n(u,L) \end{vmatrix} V_{rstu} \\ &:= \quad \sum_{r<s<t<u} P_v(rstu, IJKL) V_{rstu}. \qquad (7.17) \end{aligned}$$

The latter equality shows that the entries of the volume prolongator P_v are the determinants of 4×4 submatrices of P_n, and that each permutation of the coarse indices I, J, K and L yields either V_{IJKL} or $-V_{IJKL}$.

We define the coarse volume v_{IJKL} associated with the coarse volume element V_{IJKL} by means of its oriented boundary, which consists of the oriented coarse faces f_{IJK}, f_{IKL}, f_{ILJ} and f_{JLK}.

$$\partial v_{IJKL} \quad := \quad \{f_{IJK}, f_{IKL}, f_{ILJ}, f_{JLK}\}. \qquad (7.18)$$

From (7.16) and (7.14), it immediately follows that the coarse volume have support which is limited by:

$$\operatorname{supp}(V_{IJKL}) \quad \subset \quad \operatorname{supp}(N_I) \cap \operatorname{supp}(N_J) \cap \operatorname{supp}(N_K) \cap \operatorname{supp}(N_L). \qquad (7.19)$$

In the general case, the lefthand and righthand side of (7.19) are identical.

By a similar derivation as for (7.9) and (7.15), the coarse volume elements are found to be related to the coarse face elements as follows:

$$\operatorname{div}(\vec{F}_{IJK}) \quad = \quad \sum_{L} V_{IJKL}. \qquad (7.20)$$

7.2.4 Piecewise constant prolongators

When derived from a piecewise constant nodal prolongator P_n, the edge, face and volume prolongators (7.6), (7.12) and (7.17) will be called piecewise constant as well. This is motivated by the following theorem.

Theorem 7.2.1 *The nonzero entries of the edge, face and volume prolongators (7.6), (7.12) and (7.17) derived from piecewise constant nodal prolongator are either* ± 1 *or* 0*, and each row contains at most one nonzero entry.*

Proof. We will only give the proof for the derived edge prolongator P_e. The proofs for the derived face and volume prolongators are analogous.

The entries of P_e are calculated as the determinants of submatrices of P_n of size 2×2. Since each row of a piecewise constant nodal prolongator contains exactly one nonzero entry, which is identical to 1, these submatrices can contain at most two nonzero entries, each of which must be identical to 1. The determinants of such matrices are either ± 1 or 0.

In order to prove that each row of P_e contains at most one nonzero entry, assume that the row of P_e corresponding to e_{ij} contains more than one nonzero entry. This is only possible if together, the rows of P_n corresponding to n_i and n_j contain more than two nonzero entries. This is not the case for a piecewise constant nodal prolongator. □

The following theorem shows that the piecewise constant edge prolongator corresponds to the prolongator proposed by Reitzinger and Schöberl in [56] (see section 5.2.1).

Theorem 7.2.2 *When applying (7.6) on a fine simplicial mesh to a piecewise constant nodal prolongator* P_n*, the edge prolongator (5.26) derived from* P_n *is obtained.*

Proof. It is sufficient to compare the prolongation contributions of the two schemes for all possible combinations of fine nodes n_k, n_l and coarse nodes n_I, n_J:

$P_n(k,I)$	$P_n(k,J)$	$P_n(l,I)$	$P_n(l,J)$	(7.6)	[56]
1	0	1	0	$\det\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$	0
1	0	0	1	$\det\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	1
0	1	1	0	$\det\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	-1
0	1	0	1	$\det\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$	0
else				0	0

□

Theorem 7.2.2 shows that on simplicial meshes, the new edge prolongator (7.6) contains the prolongator (5.26) as the special case for a piecewise constant

nodal prolongator. Hence, on simplicial meshes, (7.6) can be considered a generalization of (5.26) for *arbitrary* partition of unity nodal prolongators, and the foundation of (7.6) in formula (2.64) carries over to (5.26).

7.2.5 Remark on determinants in finite element discretizations

For finite element discretizations on simplicial meshes, relations using determinants of submatrices, similar to (7.6), (7.12) and (7.17), also exist at the level of the discretization matrix. In particular, the element matrix K_T of a tetrahedron T for the curl-curl or grad-div operator can be derived from the element matrix of T for the div-grad operator, if lowest order Whitney elements of the first kind are used and if the material parameters are constant in each tetrahedron.

First, we consider the curl-curl operator. With V_T the volume of T, and using the formula for edge elements (2.64) and the fact that the nodal elements are linear inside T, we have:

$$\begin{aligned} K_T^{cc}(ij,kl) &= V_T \ \mathrm{curl}(\vec{E}_{ij}) \cdot \mathrm{curl}(\vec{E}_{kl}) \\ &= 4V_T \ (\mathrm{grad}(N_i) \times \mathrm{grad}(N_j)) \cdot (\mathrm{grad}(N_k) \times \mathrm{grad}(N_l)). \end{aligned}$$

Using the vector law $(\vec{a} \times \vec{b}) \cdot (\vec{c} \times \vec{d}) \equiv (\vec{a} \cdot \vec{c})(\vec{b} \cdot \vec{d}) - (\vec{a} \cdot \vec{d})(\vec{b} \cdot \vec{c})$, this becomes:

$$\begin{aligned} K_T^{cc}(ij,kl) &= 4V_T \begin{pmatrix} (\mathrm{grad}(N_i) \cdot \mathrm{grad}(N_k))(\mathrm{grad}(N_j) \cdot \mathrm{grad}(N_l)) \\ -(\mathrm{grad}(N_i) \cdot \mathrm{grad}(N_l))(\mathrm{grad}(N_j) \cdot \mathrm{grad}(N_k)) \end{pmatrix} \\ &= \frac{4}{V_T} \begin{vmatrix} K_T^{dg}(i,k) & K_T^{dg}(i,l) \\ K_T^{dg}(j,k) & K_T^{dg}(j,l) \end{vmatrix}. \end{aligned}$$

A similar line of reasoning for the grad-div operator leads to:

$$K_T^{gd}(ijk,rst) = \frac{36}{V_T^2} \begin{vmatrix} K_T^{dg}(i,r) & K_T^{dg}(i,s) & K_T^{dg}(i,t) \\ K_T^{dg}(j,r) & K_T^{dg}(j,s) & K_T^{dg}(j,t) \\ K_T^{dg}(k,r) & K_T^{dg}(k,s) & K_T^{dg}(k,t) \end{vmatrix}.$$

7.3 Implied coarse topology

In this section, we will focus on the coarse edges, faces and volumes, which were identified in the previous section. In the rest of this thesis, they will be referred to as coarse simplices. This is motivated by the fact that they contain 2, 3 and 4 coarse nodes respectively, which is characteristic for simplicial edges, faces and volume (see section 2.5).

First, we will present a criterium that allows to determine whether a group of coarse nodes is connected by a coarse simplex. Next, we will show that the coarse edges, faces and volumes can be described by coarse versions of the discrete differential operators gradient, curl and divergence respectively. Together, these differential operators describe a mesh-like topology. This coarse topology

can be considered simplicial. However, an important difference with simplicial meshes is that it contains in general crossing edges and faces, and overlapping faces and volumes. We will show that the coarse topology determines the memory requirements for AMG methods for the div-grad, curl-curl and grad-div equations.

7.3.1 Existence criteria for coarse simplices

In the previous section, all possible combinations of 2, 3 and 4 coarse nodal elements were considered in the definition of the coarse edge, face and volume elements respectively. However, most combinations give rise to coarse shape functions which are identical to zero. This can be seen most easily by considering the support of the coarse nodal elements involved. More precisely, it follows immediately from (7.4), (7.11) and (7.16) that the resulting shape function is identical to zero if the intersection of the supports of the coarse nodal elements involved is empty:

$$\begin{aligned} \mathrm{supp}(N_I) \cap \mathrm{supp}(N_J) = 0 \quad &\Longrightarrow \quad \vec{E}_{IJ} \equiv 0, \\ \mathrm{supp}(N_I) \cap \mathrm{supp}(N_J) \cap \mathrm{supp}(N_K) = 0 \quad &\Longrightarrow \quad \vec{F}_{IJK} \equiv 0, \\ \mathrm{supp}(N_I) \cap \mathrm{supp}(N_J) \cap \mathrm{supp}(N_K) \cap \mathrm{supp}(N_L) = 0 \quad &\Longrightarrow \quad V_{IJKL} \equiv 0. \end{aligned}$$

Consequently, overlapping support of the coarse nodes involved is a necessary condition for the existence of a coarse simplex. In the following theorems, we will formulate sufficient and necessary requirements for the existence of a coarse simplex.

Theorem 7.3.1 *Suppose all coarse nodal elements are linearly independent and have a compact support, and let the underlying fine mesh be simplicial. The coarse edge element $\vec{E}_{IJ}$ obtained from (7.4) is different from zero if and only if there is there is a fine edge connecting a fine node prolongated from N_I and a fine node prolongated from N_J:*

$$\vec{E}_{IJ} \not\equiv 0 \quad \Longleftrightarrow \quad \exists \pm e_{rs} \ : \ P_n(r, I) \neq 0 \neq P_n(s, J). \tag{7.21}$$

Proof. If $\vec{E}_{IJ} \not\equiv 0$, at least one entry of the IJth column of the edge prolongator P_e must be different from zero:

$$\exists \pm e_{rs} \quad : \quad P_e(rs, IJ) = \begin{vmatrix} P_n(r, I) & P_n(r, J) \\ P_n(s, I) & P_n(s, J) \end{vmatrix} \neq 0.$$

The righthand side of the equivalence (7.21) follows from the fact that it is possible to apply a row- and column permutation to a regular 2×2 matrix such that the resulting diagonal elements are all different from zero.

In order to prove the implication from the right to the left, suppose that the coarse edge element $\vec{E}_{IJ}$ is identical to zero. Consequently, $P_e(rs, IJ)$ must be

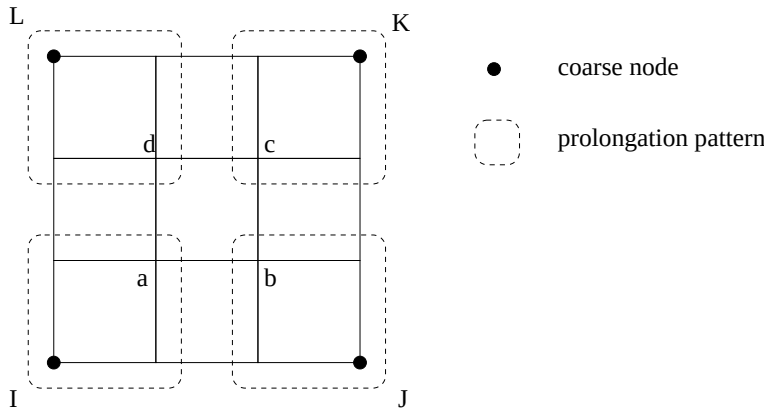


Figure 7.1: Counterexample for theorem 7.3.1 on a quadrilateral mesh. The intersection of the supports of the coarse nodal elements N_I and N_K equals the quadrilateral abcd. However, there is no fine edge connecting fine nodes prolongated from N_I and N_K.

zero:

$$\left| \begin{array}{cc} P_n(r,I) & P_n(r,J) \\ P_n(s,I) & P_n(s,J) \end{array} \right| = 0 \quad \Longrightarrow \quad \exists \alpha \in \mathbb{R}^2 \ : \ \left[\begin{array}{cc} P_n(r,I) & P_n(r,J) \\ P_n(s,I) & P_n(s,J) \end{array} \right] \alpha = \left[\begin{array}{c} 0 \\ 0 \end{array} \right].$$

Since $P_n(r,I) \neq 0 \neq P_n(s,J)$, α is different from zero. Since the supports of N_I and N_J are compact, this relation is transfered throughout the entire supports of N_I and N_J, i.e.

$$\forall t \quad : \quad \left[\begin{array}{cc} P_n(t,I) & P_n(t,J) \end{array} \right] \alpha = 0.$$

Consequently, N_I and N_J are not linearly independent. Since this is not the case, the assumption that $\vec{E}_{IJ}$ is identical to zero is wrong.
□

Note that for non-simplicial meshes, the implication from the left to the right in (7.21) is not guaranteed to hold. See figure 7.1 for a counterexample.

Theorem 7.3.2 *Suppose all coarse nodal elements are linearly independent and have a compact support, and let the underlying fine mesh be simplicial. The coarse face element $\vec{F}_{IJK}$ obtained from (7.11) is different from zero if and only if there is there is a fine face connecting fine nodes prolongated from N_I, N_J and N_K:*

$$\vec{F}_{IJK} \not\equiv 0 \quad \Longleftrightarrow \quad \exists \pm f_{rst} \ : \ P_n(r,I), P_n(s,J), P_n(t,K) \neq 0. \qquad (7.22)$$

Proof. If $\vec{F}_{IJK} \not\equiv 0$, at least one entry of the IJKth column of the face prolongator P_f must be different from zero:

$$\exists \pm f_{rst} \quad : \quad P_f(rst, IJK) = \begin{vmatrix} P_n(r,I) & P_n(r,J) & P_n(r,K) \\ P_n(s,I) & P_n(s,J) & P_n(s,K) \\ P_n(t,I) & P_n(t,J) & P_n(t,K) \end{vmatrix} \neq 0.$$

The righthand side of the equivalence (7.22) follows from the fact that it is possible to apply a row- and column permutation to a regular 3×3 matrix such that the resulting diagonal elements are all different from zero.

In order to prove the implication from the right to the left, suppose that the coarse face element $\vec{F}_{IJK}$ is identical to zero. Consequently, $P_f(rst, IJK)$ must be zero:

$$\begin{vmatrix} P_n(r,I) & P_n(r,J) & P_n(r,K) \\ P_n(s,I) & P_n(s,J) & P_n(s,K) \\ P_n(t,I) & P_n(t,J) & P_n(t,K) \end{vmatrix} = 0$$

$$\Longrightarrow \exists \alpha \in \mathbb{R}^3 \ : \ \begin{bmatrix} P_n(r,I) & P_n(r,J) & P_n(r,K) \\ P_n(s,I) & P_n(s,J) & P_n(s,K) \\ P_n(t,I) & P_n(t,J) & P_n(t,K) \end{bmatrix} \alpha = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Since $P_n(r,I), P_n(s,J), P_n(t,K) \neq 0$, α is different from zero. Since the supports of the coarse nodal elements are compact, this relation is transfered throughout the entire supports of N_I, N_J and N_K, i.e.

$$\forall t \quad : \quad \begin{bmatrix} P_n(t,I) & P_n(t,J) & P_n(t,K) \end{bmatrix} \alpha = 0.$$

Consequently, N_I, N_J and N_K are not linearly independent. Since this is not the case, the assumption that $\vec{F}_{IJK}$ is identical to zero is wrong.
□

Theorem 7.3.3 *Suppose all coarse nodal elements are linearly independent and have a compact support, and let the underlying fine mesh be simplicial. The coarse volume element V_{IJKL} obtained from (7.16) is different from zero if and only if there is there is a fine volume connecting fine nodes prolongated from N_I, N_J, N_K and N_L:*

$$V_{IJKL} \not\equiv 0 \iff \exists \pm v_{rstu} : P_n(r,I), P_n(s,J), P_n(t,K), P_n(u,L) \neq 0. \quad (7.23)$$

The formulae (7.21), (7.22) and (7.23) show that the existence of coarse edges, faces and volumes depends on the sparsity pattern of P_n. As a consequence, coarse simplices can be eliminated from the coarse topology implied by P_n by reducing the number of nonzeros in P_n:

Corollary 7.3.1 *If the nonzero locations of $\tilde{P}_n$ form a subset of the nonzero locations of P_n, then the sets of coarse edges, faces and volumes implied by $\tilde{P}_n$ are subsets of the sets of coarse edges, faces and volumes implied by P_n.*

Proof. Straightforward application of (7.21), (7.22) and (7.23). □

Note that a reduction of the sparsity pattern of P_n cannot result in a reduction of the number of coarse nodes, which is identical to the number of columns of P_n.

7.3.2 Coarse differential operators

The properties (7.9), (7.15) and (7.20) and the definitions of the boundary of the coarse simplices (7.7), (7.13) and (7.18) can be used to represent the actions of the gradient, curl and divergence operators on the coarse level by matrices G_c, C_c and D_c, similarly as on the fine level (see section 2.5.2). These matrices will be called the coarse discrete gradient, curl and divergence respectively.

$$G_c(e_{IJ}, n_K) := \begin{cases} 1 & \text{if} \quad n_K = \text{end}(e_{IJ}), \\ -1 & \text{if} \quad n_K = \text{start}(e_{IJ}), \\ 0 & \text{else}, \end{cases} \tag{7.24}$$

$$C_c(f_{IJK}, e_{RS}) := \begin{cases} 1 & \text{if} \quad e_{RS} \in \partial f_{IJK}, \\ -1 & \text{if} \quad -e_{RS} \in \partial f_{IJK}, \\ 0 & \text{else}, \end{cases} \tag{7.25}$$

$$D_c(v_{IJKL}, f_{RST}) := \begin{cases} 1 & \text{if} \quad f_{RST} \in \partial v_{IJKL}, \\ -1 & \text{if} \quad -f_{RST} \in \partial v_{IJKL}, \\ 0 & \text{else}. \end{cases} \tag{7.26}$$

Since the boundary of a coarse edge, face and volume contains exactly two, three and four coarse nodes, edges and faces, each row of G_c, C_c and D_c contains exactly two, three and four nonzeros respectively.

Remark 7.3.1 The direct calculation of the entries of G_c, C_c and D_c as:

$$\begin{aligned} G_c(e,n) &= \int_e \text{grad}(N_n) \cdot d\vec{l} &&= N_n(end(e)) - N_n(start(e)), \\ C_c(f,e) &= \int_f \text{curl}(\vec{E}_e) \cdot d\vec{\Omega} &&= \int_{\partial f} \vec{E}_e \cdot d\vec{l}, \\ D_c(v,f) &= \int_v \text{div}(\vec{F}_f)\, dV &&= \int_{\partial v} \vec{F}_f \cdot d\vec{\Omega}, \end{aligned}$$

similarly as in section 2.5.2, is not possible, since this would require the boundary of all coarse edges, faces and volumes to be localized. The requirements for this localization will be explained in section 7.4.

7.3.3 Properties of the implied coarse topology

In many respects, the coarse topology implied by a nodal prolongator has a simplicial character. First, its edge, face and volume elements are obtained by applying the analytical formulae for Whitney elements on simplicial meshes

(see sections 7.2.1, 7.2.2 and 7.2.3). Also, its faces and volumes contain 3 and 4 coarse nodes respectively, which is a characteristic property of simplicial faces and volumes (see section 2.5). However, an important difference with simplicial meshes is that in general, the coarse topology implied by a nodal prolongator contains *crossing and overlapping simplices*. First, we will define this terminology more precisely.

Definition 7.3.1 *Volumes are called overlapping if the supports of their associated volume elements are overlapping.*

Recall that the support of a coarse volume element equals in general the intersection of the supports of the coarse nodal elements of its nodes (see (7.19)). Definition 7.3.1 is illustrated for the 2D setting depicted in figure 7.2. Note that in 2D, the role of volumes is played by faces. There are 6 coarse edges ($e_{IJ}, e_{IK}, e_{IL}, e_{JK}, e_{JL}$ and e_{KL}) and 4 coarse volumes ($v_{IJK}, v_{IJL}, v_{IKL}$ and v_{JKL}). It is immediately clear that e_{IL} and e_{KJ} are crossing and that v_{IJK} and v_{IJL} are overlapping. But also v_{IJK} and v_{JKL} are overlapping, see tables 7.1 and 7.2.

Definition 7.3.2 *The type of a simplex indicates whether it is a node, edge, face or volume.*

Definition 7.3.3 *Two simplices of the same type are called neighbours if they belong to a common volume. The set of neighbours of a simplex x is defined as:*

$$neighbours(x) \quad = \quad \{y \mid type(y) = type(x) \quad and \quad \exists \ volume \ v : x, y \in v\}.$$

Definition 7.3.4 *Two coarse simplices of the same type are called crossing if they belong to overlapping coarse volumes. A discrete topology is called crossing, if it contains overlapping coarse volumes. The set of simplices crossing a simplex x is defined as:*

$$cross(x) \quad = \quad \{y \mid type(y) = type(x) \quad and \quad x \ and \ y \ are \ crossing\}.$$

Note that the topology of a fine mesh (2.49) is not crossing.

The following lemma shows that for a fixed set of coarse nodes, it is possible to restrict the sparsity structure of the nodal prolongator in such a way that the resulting implied coarse topology is not crossing.

Lemma 7.3.1 *For any partition of unity nodal prolongator P_n with respect to a non-crossing fine topology, there exists a partition of unity nodal prolongator $\tilde{P}_n$ with a sparsity pattern that is a subset of the sparsity pattern of P_n, such that the coarse topology implied by $\tilde{P}_n$ is not crossing as well.*

Proof. The sparsity pattern of a partition of unity nodal prolongator P_n can always be restricted to the sparsity pattern of a piecewise constant $\tilde{P}_n$ by keeping exactly 1 nonzero entry per row. By setting all these remaining entries to one, $\tilde{P}_n$ will satisfy the partition of unity property as well. So, it suffices to prove that the topology implied by $\tilde{P}_n$ is not crossing if the fine topology is not

	supp(N_I)	supp(N_J)	supp(N_K)	supp(N_L)
left	acfhg	acig	acig	bcigd
middle	abeg	acfhg	dfig	cihe
right	acg	acig	acig	cig

Table 7.1: Supports of the coarse nodal elements represented by the prolongators depicted in figure 7.2. The supports are indicated by their fine boundary nodes.

	supp(V_{IJK})	supp(V_{JKL})	supp(V_{IJL})	supp(V_{IKL})
left	acfhg	bcigd	bcfhgd	bcfhgd
middle	deg	efh	\	\
right	acg	cig	\	\

Table 7.2: Supports of the coarse volume elements derived from the nodal prolongators depicted in figure 7.2. The supports are indicated by their fine boundary nodes.

crossing. According to (7.23), v_{ijkl} can belong to the support of more than one coarse volume element only if $\tilde{P}_n([i,j,k,l],:)$ contains more than 4 nonzeros. This is not possible, since each row of a piecewise constant $\tilde{P}_n$ contains at most 1 nonzero. So, if the fine volumes are not overlapping, the coarse volumes are not overlapping as well. By definition 7.3.4, this implies a non-crossing coarse topology.
□

Lemma 7.3.1 is illustrated in figure 7.2. Recall that in 2D, the role of volumes is played by faces. We call the prolongation pattern of a coarse node n_I the set of fine nodes n_r for which $P_n(r,I) \neq 0$. The left picture represents the prolongation patterns for a nodal prolongator implying a crossing coarse topology. The implied coarse edges and volumes are $\{e_{IJ}, e_{IK}, e_{IL}, e_{JK}, e_{JL}, e_{KL}\}$ and $\{v_{IJK}, v_{IJL}, v_{IKL}, v_{JKL}\}$. The supports of any two coarse volume elements overlap, see table 7.2. In the middle picture, the sparsity pattern of P_n is restricted to the pattern for the piecewise constant nodal prolongator defined by $P_n([a,d],I) = P_n([b,c,e],J) = P_n([f,i],L) = P_n([g,h],K) = 1$. The implied coarse edges and volumes are $\{e_{IJ}, e_{IK}, e_{JK}, e_{JL}, e_{KL}\}$ and $\{v_{IJK}, v_{JKL}\}$. The supports of the volume elements $supp(V_{IJK}) = \Delta deg$ and $supp(V_{JKL}) = \Delta efh$ have no overlap. So, the coarse topology is not crossing.

Note that P_n does not need to be piecewise constant to imply a non-crossing coarse topology. For instance, all crossing coarse simplices can be eliminated by requiring $P_n(e,I) = P_n(e,L) = 0$, as depicted in the right picture of figure 7.2. The corresponding coarse topology contains the same non-overlapping coarse volumes as in the middle picture. However, the union of the supports of the coarse volume elements in the right picture equals the *entire* computational domain.

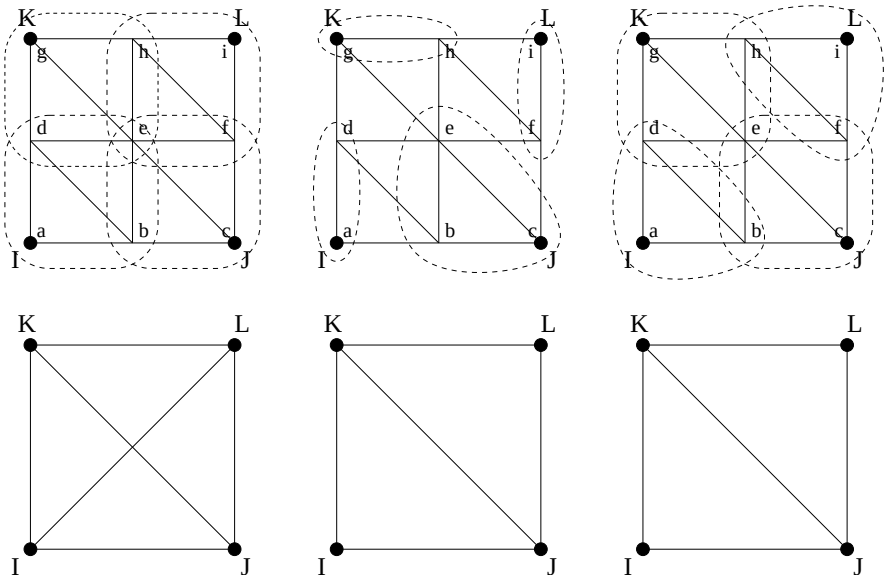


Figure 7.2: The dotted lines in the top row pictures indicate different possible prolongation patterns of P_n for the same number of coarse nodes. The nonzero patterns of the nodal prolongators P_n represented in the middle and right pictures are subsets of the nonzero pattern of the nodal prolongator P_n represented in the left picture. The bottom row pictures show the corresponding coarse topologies. The left coarse topology is crossing. The middle and right coarse topologies are identical and not crossing. The middle pictures correspond to a piecewise constant P_n.

7.3.4 Impact of the coarse topology on the memory requirements of AMG methods

In this section, we will study the memory requirements for the coarse Galerkin matrices in AMG methods which use the edge prolongators (7.6) or face prolongators (7.12). We will show that the coarse topology implied by the underlying nodal prolongator determines the nonzero pattern of these coarse AMG matrices.

Consider a nodal prolongator P_n which implies a crossing topology τ. According to lemma 7.3.1, it is possible to restrict the sparsity pattern of P_n such that the implied coarse topology becomes non-crossing. Let $\tilde{P}_n$ denote *any* nodal prolongator with a sparsity pattern satisfying this requirement, and let $\tilde{\tau}$ be the coarse topology implied by $\tilde{P}_n$. In figure 7.2, P_n would correspond to the prolongator of the left picture, while $\tilde{P}_n$ is any of the remaining cases.

Let s_n, s_e, s_f and s_v, and $\tilde{s}_n$, $\tilde{s}_e$, $\tilde{s}_f$ and $\tilde{s}_v$ denote the sets of the coarse nodes, edges, faces and volumes belonging to τ and $\tilde{\tau}$ respectively. By corrolary 7.3.1, we have:

$$\tilde{s}_n = s_n, \qquad \tilde{s}_e \subset s_e, \qquad \tilde{s}_f \subset s_f, \qquad \tilde{s}_v \subset s_v.$$

Let r_n, r_e, r_f and r_v denote the sets of the coarse nodes, edges, faces and volumes of τ which are eliminated from $\tilde{\tau}$:

$$\begin{aligned} r_n &:= s_n \setminus \tilde{s}_n = \phi, \\ r_e &:= s_e \setminus \tilde{s}_e, \\ r_f &:= s_f \setminus \tilde{s}_f, \\ r_v &:= s_v \setminus \tilde{s}_v. \end{aligned}$$

Depending on the problem at hand, let P_i and $\tilde{P}_i$ denote the nodal, edge or face prolongator derived from P_n or $\tilde{P}_n$ respectively, and $K_{ci} = P_i^T K_{fi} P_i$ and $\tilde{K}_{ci} = \tilde{P}_i K_{fi} \tilde{P}_i$ the corresponding coarse Galerkin matrices. With an appropriate permutation, K_{ci} can be written in the following block format:

$$K_{ci} \sim \begin{bmatrix} K_{ci}(\tilde{s}_i, \tilde{s}_i) & K_{ci}(\tilde{s}_i, r_i) \\ K_{ci}(r_i, \tilde{s}_i) & K_{ci}(r_i, r_i) \end{bmatrix}. \tag{7.27}$$

Clearly, the elimination of the coarse simplices r_i yields a size reduction of the coarse system matrix. With $size(K)$ denoting the number of rows and columns of a square matrix K, we have:

$$\begin{aligned} \#\tilde{s}_i = size(\tilde{K}_{ci}) &= size(K_{ci}(\tilde{s}_i, \tilde{s}_i)) \\ &= size(K_{ci}) - \#r_i = \#s_i - \#r_i \end{aligned}$$

and a reduction of the number of nonzeros, corresponding to the off-diagonal blocks of (7.27). Recall that this is not the case for node-based problems, since $r_n = \phi$.

Additionally, the number of nonzeros of $\tilde{K}_{ci}$ is smaller than the number of nonzeros of the $(1,1)$-block $K_{ci}(\tilde{s}_i, \tilde{s}_i)$ of (7.27). This can be seen by observing that each off-diagonal nonzero entry of a node-, edge and face-based problem corresponds to some edges, faces and volumes respectively. In the example of figure 7.2, for instance, the elimination of the coarse edge e_{IL} and the coarse faces f_{IJL} and f_{IKL} corresponds to:

$$\begin{aligned} \text{elimination of } e_{IL} &\longrightarrow K_{cn}(I,L) = K_{cn}(L,I) = 0, \\ \text{elimination of } f_{IJL} &\longrightarrow K_{ce}(IJ,JL) = K_{ce}(JL,IJ) = 0, \\ \text{elimination of } f_{IKL} &\longrightarrow K_{ce}(IK,KL) = K_{ce}(KL,IK) = 0. \end{aligned}$$

7.4 Normalization

Edge elements, face elements and volume elements on a fine simplicial mesh are normalized (see section 2.5). In this section, we will study the requirements for the normalization of the coarse nodal, edge, face and volume elements presented in section 7.2.

The classical definitions of normalization (2.54) cannot be applied immediately here. This is because it is not clear which node, path, surface or volume to choose as the integration domain. A generalized definition of normalization will be needed for the coarse shape functions. For this purpose, some new concepts and notations will be introduced first.

Definition 7.4.1 *[Representative] A representative of a node is any point in the support of its nodal function. A representative of an edge is any path belonging to the support of its edge element, connecting representatives of its end nodes. A representative of a face is any surface belonging to the support of its face element, having a boundary defined by representatives of its edges. A representative of a volume is any volume belonging to the support of its volume element, having a boundary defined by representatives of its faces.*

Definition 7.4.2 *[Discrete representative] A discrete representative of a simplex is a representative of that simplex entirely consisting of nodes, edges, faces or volumes of the topology of the mesh. A discrete representative of a shape function X is a discrete representative of its associated simplex x. The set of discrete representatives of a shape function X will be denoted by drep(X).*

Definition 7.4.2 is illustrated on figure 7.2 and tables 7.1 and 7.2. Consider the right picture. The discrete representatives of n_I are n_a, n_b and n_d. A discrete representative of e_{IJ} is for example $n_d \rightarrow n_e \rightarrow n_c$: it connects a discrete representative n_d of n_I to a discrete representative n_c of n_J, and it belongs to $supp(\vec{E}_{IJ}) = supp(N_I) \cap supp(N_J) = \Delta acg$. The square $\square abed$ is a discrete representative of the coarse triangle v_{IJK}: its boundary is defined by discrete representatives $a \rightarrow b \rightarrow e$, $e \rightarrow d$ and $d \rightarrow a$ of e_{IJ}, e_{JK} and e_{KI} respectively, and it belongs to $supp(V_{IJK}) = supp(N_I) \cap supp(N_J) \cap supp(N_K) = \Delta acg$.

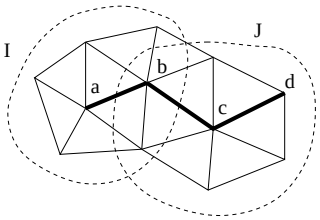


Figure 7.3: Illustration of the definitions of focus and normalization for a coarse edge element. The prolongation patterns of the coarse nodes n_I and n_J are indicated by the curved dotted lines. If the line integral along the discrete representative *abcd* of e_{IJ} equals ± 1 for $\vec{E}_{IJ}$ and 0 for all other edge elements, then *abcd* is a focus of e_{IJ} and $\vec{E}_{IJ}$ is normalized.

Definition 7.4.3 *[Focus] A focus of a nodal element N_i, edge element $\vec{E}_i$, face element $\vec{F}_i$ or volume element V_i is a discrete representative n, p, s or v for which holds:*

$$N_j(n) = \int_p \vec{E}_j \cdot d\vec{l} = \iint_s \vec{F}_j \cdot d\vec{\Omega} = \iiint_v V_j \, dV = \delta_{ij}.$$

The set of foci of a shape function X will be denoted by $focus(X)$.

Now, the requirements (2.55), (2.56), (2.57), (2.58) for the normalization of nodal, edge, face and volume elements can be generalized:

Definition 7.4.4 *[Normalization] A shape function is normalized if it has a focus.*

For example, a coarse edge element $\vec{E}_{IJ}$ is called normalized if there exists a path on the fine mesh connecting discrete representatives of n_I and n_J, along which the path integral of any $\vec{E}_{KL}$ evaluates to ± 1 if $\vec{E}_{KL} = \pm \vec{E}_{IJ}$, and to 0 otherwise. This is illustrated on figure 7.3.

Definition 7.4.4 contains (2.55)-(2.56)-(2.57)-(2.58) as a special case. Note that the existence of multiple foci for a shape function is not precluded by this definition.

Next, the requirements for the normalization of the coarse edge, face and volume elements will be studied in detail.

Theorem 7.4.1 *Suppose the coarse nodal elements defined by (4.20) with P_n a partition of unity nodal prolongator, are normalized and nonnegative. Then, the coarse edge elements defined by (7.6) are normalized if each coarse edge element $\vec{E}_{IJ}$ has a discrete representative p_{IJ} satisfying*

$$\forall x \in p_{IJ} : \; N_I(x) + N_J(x) = 1, \tag{7.28}$$

$$begin(p_{IJ}) \in focus(N_I), \; end(p_{IJ}) \in focus(N_J). \tag{7.29}$$

Proof. Using (7.4), (7.28), (7.29) and the normalization of the coarse nodal elements, the line integral of $\vec{E}_{IJ}$ along p_{IJ} reads:

$$\begin{aligned}\int_{p_{IJ}} \vec{E}_{IJ}\, d\vec{l} &= \int_{p_{IJ}} (N_I \text{ grad}(N_J) - N_J \text{ grad}(N_I))\, d\vec{l} \\ &= \int_{p_{IJ}} (N_I \text{ grad}(1-N_I) - (1-N_I) \text{ grad}(N_I))\, d\vec{l} \\ &= \int_{p_{IJ}} -\text{grad}(N_I)\, d\vec{l} \\ &= N_I(begin(p_{IJ})) - N_I(end(p_{IJ})) \;=\; 1. \end{aligned} \tag{7.30}$$

Also, (4.22), (7.28) and the non-negativity of the coarse nodal elements imply that any other coarse edge element $\vec{E}_{KL}$ and its line integral are 0 along p_{IJ}:

$$\begin{aligned} e_{KL} \neq \pm e_{IJ} \;&\Longrightarrow\; N_K(p_{IJ}) \equiv 0 \text{ or } N_L(p_{IJ}) \equiv 0 \\ &\Longrightarrow\; \vec{E}_{KL}(p_{IJ}) \equiv \vec{0} \text{ and } \int_{p_{IJ}} \vec{E}_{KL}\, d\vec{l} = 0 \end{aligned} \tag{7.31}$$

(7.30) and (7.31) imply that p_{IJ} is a focus of $\vec{E}_{IJ}$. Since this holds for any $\vec{E}_{IJ}$, the coarse edge elements are normalized.
□

Theorem 7.4.2 *Suppose the coarse nodal elements defined by (4.20) with P_n a partition of unity nodal prolongator, are normalized and nonnegative. Then, the coarse face elements defined by (7.12) are normalized if each coarse face element $\vec{F}_{IJK}$ has a discrete representative s_{IJK} satisfying*

$$\forall x \in s_{IJK}: \; N_I(x) + N_J(x) + N_K(x) = 1, \tag{7.32}$$

$$\left.\begin{array}{l} \exists p_{IJ} \in focus(\vec{E}_{IJ}) \\ \exists p_{JK} \in focus(\vec{E}_{JK}) \\ \exists p_{KI} \in focus(\vec{E}_{KI}) \end{array}\right\} \;:\quad \partial s_{IJK} = \{p_{IJ},\; p_{JK},\; p_{KI}\}. \tag{7.33}$$

Proof. Using (7.11), (7.32), (7.33), the normalization of the coarse nodal elements and

$$\begin{aligned}\text{curl}\left(\vec{E}_{IJ}\right) &= \text{curl}\left(N_I \,\text{grad}(N_J) - N_J \,\text{grad}(N_I)\right) \\ &= 2\,\text{grad}(N_I) \times \text{grad}(N_J),\end{aligned}$$

the surface integral of $\vec{F}_{IJK}$ over s_{IJK} reads:

$$\int_{s_{IJK}} \vec{F}_{IJK} \cdot d\vec{\Omega} \;=\; 2\int_{s_{IJK}} \left(\begin{array}{rl} & N_I \,\text{grad}(N_J) \times \text{grad}(N_K) \\ + & N_J \,\text{grad}(N_K) \times \text{grad}(N_I) \\ + & N_K \,\text{grad}(N_I) \times \text{grad}(N_J) \end{array}\right) \cdot d\vec{\Omega}$$

$$
\begin{aligned}
&= 2\int_{s_{IJK}} \left(\begin{array}{ll} - & N_I \operatorname{grad}(N_J) \times \operatorname{grad}(N_I + N_J) \\ - & N_J \operatorname{grad}(N_I + N_J) \times \operatorname{grad}(N_I) \\ + & (1 - N_I - N_J) \operatorname{grad}(N_I) \times \operatorname{grad}(N_J) \end{array} \right) \cdot d\vec{\Omega} \\
&= 2\int_{s_{IJK}} \left(\begin{array}{ll} & N_I \operatorname{grad}(N_I) \times \operatorname{grad}(N_J) \\ + & N_J \operatorname{grad}(N_I) \times \operatorname{grad}(N_J) \\ + & (1 - N_I - N_J) \operatorname{grad}(N_I) \times \operatorname{grad}(N_J) \end{array} \right) \cdot d\vec{\Omega} \\
&= \int_{s_{IJK}} \operatorname{curl}(\vec{E}_{IJ}) \cdot d\vec{\Omega} \;=\; \int_{\partial s_{IJK}} \vec{E}_{IJ} \cdot d\vec{l} \;=\; 1. \qquad (7.34)
\end{aligned}
$$

Also, (4.22), (7.32) and the non-negativity of the coarse nodal elements imply that any other coarse face element $\vec{F}_{RST} \neq \pm\vec{F}_{IJK}$ and its surface integral are identical to 0 over s_{IJK}:

$$
\begin{aligned}
N_R(s_{IJK})N_S(s_{IJK})N_T(s_{IJK}) \equiv 0 \quad &\Longrightarrow \quad \vec{F}_{RST}(s_{IJK}) \equiv \vec{0} \\
&\Longrightarrow \quad \iint_{s_{IJK}} \vec{F}_{IJK} \cdot d\vec{\Omega} = 0.
\end{aligned}
$$

Consequently, s_{IJK} is a focus of $\vec{F}_{IJK}$. Since this holds for any $\vec{F}_{IJK}$, the coarse face elements are normalized.
□

Theorem 7.4.3 *Suppose the coarse nodal elements defined by (4.20) with P_n a partition of unity nodal prolongator, are normalized and nonnegative. Then, the coarse volume elements defined by (7.17) are normalized if each coarse volume element V_{IJKL} has a discrete representative v_{IJKL} satisfying*

$$\forall x \in v_{IJKL} : \; N_I(x) + N_J(x) + N_K(x) + N_L(x) \equiv 1, \qquad (7.35)$$

$$
\left.
\begin{array}{l}
\exists s_{IJK} \in focus(\vec{F}_{IJK}) \\
\exists s_{IJL} \in focus(\vec{F}_{IJL}) \\
\exists s_{IKL} \in focus(\vec{F}_{IKL}) \\
\exists s_{JKL} \in focus(\vec{F}_{JKL})
\end{array}
\right\} \; : \; \partial v_{IJKL} = \{s_{IJK},\; s_{IJL},\; s_{IKL},\; s_{JKL}\}. \qquad (7.36)
$$

Proof. Analogous to the proof of theorems 7.4.1and 7.4.2. □

Note that theorems 7.4.1, 7.4.2 and 7.4.3 only prove that (7.28-7.29), (7.32-7.33) and (7.35-7.36) are *sufficient* requirements for normalized coarse edge, face and volume elements. If (7.28)-(7.29) is not satisfied for example, (7.30) becomes:

$$
\begin{aligned}
\int_{p_{IJ}} \vec{E}_{IJ} \cdot d\vec{l} \;=\; &\int_{p_{IJ}} -\operatorname{grad}(N_I) \cdot d\vec{l} \\
&- \int_{p_{IJ}} (N_I \operatorname{grad}(X_{IJ}) - X_{IJ} \operatorname{grad}(N_I)) \cdot d\vec{l}, \qquad (7.37)
\end{aligned}
$$

with $N_I + N_J + X_{IJ} \equiv 1$. It will be difficult *in practice* to find for every $\vec{E}_{IJ}$ a discrete representative p_{IJ} such that (7.37) equals 1, if (7.28) and (7.29) can*not*

always be satisfied. For the same reason, the foci of the coarse face and volume elements satisfy in the generic case (7.32) and (7.35).

In general, the coarse edge, face and volume elements derived from an arbitrary partition of unity nodal prolongator are not normalized. However, the following theorem will show that normalization is automatically obtained if the underlying nodal prolongator is piecewise constant. This theorem is illustrated in figure 7.4.

Theorem 7.4.4 *The coarse edge, face and volume elements (7.6), (7.12) and (7.17) derived from a piecewise constant nodal prolongator are normalized, if the fine level edge, face and volume elements are normalized respectively.*

Proof. Since the fine shape functions are normalized, each entry of an edge, face or volume prolongator equals the line, face or volume integral of the coarse edge, face or volume element represented by its column, over the fine edge, face or volume corresponding to its row:

$$\begin{aligned}\int_{e_{rs}} \vec{E}_{IJ} \cdot d\vec{l} &= P_e(rs, IJ),\\ \int_{f_{rst}} \vec{F}_{IJK} \cdot d\vec{\Omega} &= P_f(rst, IJK),\\ \int_{v_{rstu}} V_{IJKL}\, dV &= P_v(rstu, IJKL).\end{aligned}$$

Since all nonzero entries of these prolongators equal ± 1 (see theorem 7.2.1), we have:

$$\begin{aligned}P_e(rs, IJ) \neq 0 &\implies \int_{e_{rs}} \vec{E}_{IJ} \cdot d\vec{l} = \pm 1,\\ P_f(rst, IJK) \neq 0 &\implies \int_{f_{rst}} \vec{F}_{IJK} \cdot d\vec{\Omega} = \pm 1,\\ P_v(rstu, IJKL) \neq 0 &\implies \int_{v_{rstu}} V_{IJKL}\, dV = \pm 1.\end{aligned}$$

Since each column of an edge, face or volume prolongator contains at least one nonzero entry, all coarse edge, face and volume elements are normalized.
□

7.5 Coarse de Rham complex

In this section, we will prove that the range spaces of a nodal prolongator P_n, the edge, face and volume prolongators P_e, P_f and P_v derived from P_n by the formulas (7.6), (7.12) and (7.17) and the coarse topological operators (7.24), (7.25) and (7.26) implied by P_n, form a de Rham complex if all prolongators have full rank. Similar results have been obtained in [23] in the context of

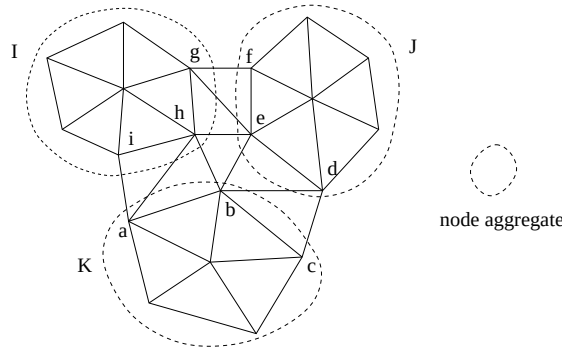


Figure 7.4: Illustration of the normalization of the edge and face prolongators P_e and P_f derived from a piecewise constant nodal prolongator P_n. The prolongation patterns of the coarse nodes n_I, n_J and n_K are indicated by the dotted curves. All fine edges prolongated by P_e from e_{IJ}, i.e. e_{gf}, e_{ge} and e_{he} in this example, are foci of the coarse edge e_{IJ}. Consequently, $\vec{E}_{IJ}$ is normalized. All fine faces prolongated by P_f from f_{IJK}, i.e. f_{beh} in this example, are foci of the coarse edge e_{IJ}. Consequently, $\vec{F}_{IJK}$ is normalized.

geometric multigrid for discretizations using the finite integration technique (FIT, [24]).

First, we will prove several commutation properties relating the prolongators P_n, P_e, P_f, P_v and the fine and coarse topological operators. Next, these commutation properties will be used to prove the existence of the coarse de Rham complex if all prolongators have full rank. Finally, it will be shown that normalization of the coarse shape functions is a sufficient condition for the prolongators to have full rank. Throughout this section, it will be assumed that the formulas (7.6), (7.12), (7.17) and (7.24), (7.25), (7.26) are applicable.

Lemma 7.5.1 (node-edge commutation) *If an edge prolongator P_e can be derived from a partition of unity nodal prolongator P_n by (7.6), then P_n and P_e satisfy the node-edge commutation property*

$$P_e\, G_c \quad = \quad G_f\, P_n, \tag{7.38}$$

with the discrete gradient operators G_f and G_c defined in (2.76) and (7.24) respectively.

Proof. Consider the righthand side of (7.38). Using (2.76), we have

$$(G_f\, P_n)(ij, K) = G_f(ij, :)\, P_n(:, K) = P_n(j, K) - P_n(i, K).$$

The same entry of the lefthand side reads, using (7.24) and (4.22),

$$\begin{aligned}(P_e\,G_c)(ij,K) &= P_e(ij,:)\,G_c(:,K)\\ &= \sum_{L\neq K} -\begin{vmatrix} P_n(i,K) & P_n(i,L)\\ P_n(j,K) & P_n(j,L)\end{vmatrix}\\ &= \begin{vmatrix} P_n(j,K) & \sum_L P_n(j,L)\\ P_n(i,K) & \sum_L P_n(i,L)\end{vmatrix}\\ &= P_n(j,K)-P_n(i,K).\end{aligned}$$

Since the equality holds for all entries, the commutation property is proved. □

Lemma 7.5.2 (edge-face commutation) *If an edge and face prolongator P_e and P_f can be derived from a partition of unity nodal prolongator P_n by (7.6) and (7.12), then P_e and P_f satisfy the edge-face commutation property*

$$P_f\,C_c \;=\; C_f\,P_e, \tag{7.39}$$

with the discrete curl operators C_f and C_c defined in (2.77) and (7.25) respectively.

Proof. Consider the righthand side of (7.39). Using (2.77), we have

$$\begin{aligned}(C_f\,P_e)(rst,IJ) &= C_f(rst,:)\,P_e(:,IJ)\\ &= P_e(rs,IJ)+P_e(st,IJ)+P_e(tr,IJ).\end{aligned}$$

The same entry of the lefthand side reads, using (7.25), (4.22) and (7.6):

$$\begin{aligned}(P_f\,C_c)(rst,IJ) &= \sum_K P_f(rst,IJK)\\ &= \sum_K \begin{vmatrix} P_n(r,I) & P_n(r,J) & P_n(r,K)\\ P_n(s,I) & P_n(s,J) & P_n(s,K)\\ P_n(t,I) & P_n(t,J) & P_n(t,K)\end{vmatrix}\\ &= \begin{vmatrix} P_n(r,I) & P_n(r,J) & \sum_K P_n(r,K)\\ P_n(s,I) & P_n(s,J) & \sum_K P_n(s,K)\\ P_n(t,I) & P_n(t,J) & \sum_K P_n(t,K)\end{vmatrix}\\ &= \begin{vmatrix} P_n(s,I) & P_n(s,J)\\ P_n(t,I) & P_n(t,J)\end{vmatrix} - \begin{vmatrix} P_n(r,I) & P_n(r,J)\\ P_n(t,I) & P_n(t,J)\end{vmatrix}\\ &\quad + \begin{vmatrix} P_n(r,I) & P_n(r,J)\\ P_n(s,I) & P_n(s,J)\end{vmatrix}\\ &= P_e(st,IJ)+P_e(tr,IJ)+P_e(rs,IJ).\end{aligned}$$

Since the equality holds for all entries, the commutation property is proved. □

Lemma 7.5.3 (face-volume commutation) *If a face and volume prolongator P_f and P_v can be derived from a partition of unity nodal prolongator P_n by (7.12) and (7.17), then P_f and P_v satisfy the face-volume commutation property*

$$P_v\, D_c \quad = \quad D_f\, P_f. \tag{7.40}$$

with the discrete divergence operators D_f and D_c defined in (2.78) and (7.26) respectively.

Proof. analogous to the proofs of lemmas 7.5.1 and 7.5.2 □

Theorem 7.5.1 (Coarse De Rham sequence) *Suppose the fine discrete nodal, edge, face and volume spaces $\mathcal{N}_f$, $\mathcal{E}_f$, $\mathcal{F}_f$ and $\mathcal{V}_f$ and the fine discrete gradient, curl and divergence G_f, C_f and D_f form a de Rham complex (2.99). If an edge, face and volume prolongator P_e, P_f and P_v can be derived from a partition of unity nodal prolongator P_n by (7.6), (7.12) and (7.17), then their range spaces $\mathcal{N}_c$, $\mathcal{E}_c$, $\mathcal{F}_c$ and $\mathcal{V}_c$ and the coarse differential operators (7.24), (7.25) and (7.26) implied by P_n form a de Rham complex, if P_e, P_f and P_v have full rank.*

Proof. The total sequence of commutation properties (7.38), (7.39) and (7.40) can be represented as in (5.8):

$$\begin{array}{ccccccccc}
\mathbb{R} & \xrightarrow{\mathbf{1}_f} & \mathcal{N}_f & \xrightarrow{G_f} & \mathcal{E}_f & \xrightarrow{C_f} & \mathcal{F}_f & \xrightarrow{D_f} & \mathcal{V}_f \\
 & & \uparrow P_n & & \uparrow P_e & & \uparrow P_f & & \uparrow P_v \\
\mathbb{R} & \xrightarrow{\mathbf{1}_c} & \mathcal{N}_c & \xrightarrow{\mathbf{G}_c} & \mathcal{E}_c & \xrightarrow{\mathbf{C}_c} & \mathcal{F}_c & \xrightarrow{\mathbf{D}_c} & \mathcal{V}_c.
\end{array} \tag{7.41}$$

Hence, the coarse spaces have the correct dimensions to form a de Rham complex. It remains to prove the topological identities (cfr. section 2.4.2)

$$\mathrm{ran}(\mathbf{1}_c) \quad = \quad \ker(G_c), \tag{7.42}$$
$$\mathrm{ran}(G_c) \quad = \quad \ker(C_c), \tag{7.43}$$
$$\mathrm{ran}(C_c) \quad = \quad \ker(D_c). \tag{7.44}$$

First, we prove that the lefthand sides are subsets of the righthand sides. Using the partition of unity property (4.22) for P_n, the commutation properties (7.38), (7.39) and (7.40), the properties (2.100), (2.101) and (2.102) of the fine de Rham complex and the assumptions that P_e, P_f and P_v have full rank, we have:

$$\begin{array}{rclcrcl}
P_e\, G_c\, 1_c & = & G_f\, P_n\, 1_c & = & G_f\, 1_f & \equiv & 0, \quad \Longrightarrow \quad G_c\, 1_c \equiv 0, \\
P_f\, C_c\, G_c & = & C_f\, P_e\, G_c & = & C_f\, G_f\, P_n & \equiv & 0, \quad \Longrightarrow \quad C_c\, G_c \equiv 0, \\
P_v\, D_c\, C_c & = & D_f\, P_f\, C_c & = & D_f\, C_f\, P_e & \equiv & 0 \quad \Longrightarrow \quad D_c\, C_c \equiv 0.
\end{array}$$

Next, we prove that the righthand sides of (7.42), (7.43) and (7.44) are subsets of the lefthand sides. For (7.42), this follows from the definition (7.24) of G_c and from the fact that coarse topology is connected:

$$\forall e_{IJ},\ (G_c\alpha_c)(e_{IJ}) = \alpha_c(n_J) - \alpha_c(n_I) = 0 \quad \Longrightarrow \quad \alpha_c \in \mathrm{ran}(\mathbf{1}_c).$$

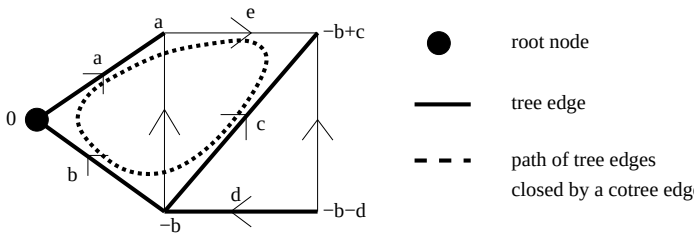


Figure 7.5: Spanning tree of edges for constructing α satisfying $C\alpha = 0$ when the entries of α for the tree edges are known. First, an auxiliary nodal vector φ is determined such that $G\varphi = \alpha$ is satisfied for all tree edges. Next, the entry for each cotree edge is determined by requiring the circulation of α along the path of tree edges closed by this cotree edge to be zero. The loop indicated in the picture allows e to be determined as $e - c + b + a = 0$, which corresponds to the gradient formula $(G\varphi)(e) = \alpha(e)$.

The proof for (7.43) is based on a spanning tree for the coarse edges (see figure 7.5). Note that the presence of crossing edges (see section 7.3) does not prevent the construction of such spanning tree. For each $\alpha_c \in \mathcal{E}_c$, there is a coarse nodal potential vector φ_c, such that $\alpha_c(t) = (G_c\varphi_c)(t)$ for each coarse edge t belonging to the tree. The entries of φ_c can be calculated by a top-down approach, as follows. The entry for the root node of the tree can be chosen arbitrarily. The entry for any other node n_I can be calculated from the entry for its parent node n_J as:

$$(G_c\varphi_c)(e_{IJ}) \quad = \quad \varphi_c(n_J) - \varphi_c(n_I) \quad = \quad \alpha_c(e_{IJ}). \tag{7.45}$$

The entries of α_c for the coarse edges not belonging to the tree are determined in a unique way by the requirement $C_c\alpha_c = 0$. Note, for this purpose, that for each such edge e, there is exactly one path p of tree edges connecting its start and end nodes. Since $C_c\alpha_c = 0$ implies that the circulation of α_c along the loop of e and p is zero, the entry of α_c for e satisfies (7.45) as well, which concludes the proof.

The proof for (7.44) is based on a spanning tree for the coarse faces, which is equivalent to a spanning tree for the dual edges, and is analogous.
□

Lemma 7.5.4 *If all fine edge, face and volume elements and all coarse edge, face and volume elements derived from a full rank partition of unity nodal prolongator P_n using (7.6), (7.12) and (7.17) are normalized, then the associated prolongators P_e, P_f, P_v have full rank.*

Proof. We only prove the result for P_e. The proofs for P_f and P_v are analogous.

Because of the normalization of P_e, $\vec{E}_{KL}$ has a focus p_{KL}. Using (7.6) and the normalization of the fine and coarse edge elements, the line integral of $\vec{E}_{IJ}$ along p_{KL} reads:

$$\begin{aligned}\int_{p_{KL}} \vec{E}_{IJ} \cdot d\vec{l} &= \int_{p_{KL}} \sum_{e_{rs}} P_e(rs, IJ)\, \vec{E}_{rs} \cdot d\vec{l} \\ &= \sum_{e_{rs} \in p_{KL}} P_e(rs, IJ) = \delta_{IJ,KL}.\end{aligned}$$

Consider the oriented sum corresponding over the fine edges belonging to p_{IJ} of any linear combination of the columns of P_e. Using the previous result, this becomes:

$$\begin{aligned}\sum_{e_{rs} \in p_{IJ}} \sum_{e_{UV}} \alpha_{UV}\, P_e(rs, UV) &= \sum_{e_{UV}} \alpha_{UV} \sum_{e_{rs} \in p_{IJ}} P_e(rs, UV) \\ &= \sum_{e_{UV}} \alpha_{UV}\, \delta_{UV,IJ} = \alpha_{IJ}.\end{aligned}$$

Hence, the IJ^{th} column of P_e is linearly independent of the other columns of P_e. Since this holds for any column IJ, P_e has full rank.
□

7.6 Structure of the coarse system matrices

Recall that compatible discretizations of div-grad, curl-curl and grad-div problems can be factorized as the Galerkin product of a discrete differential operator and a mass matrix, cfr. (2.122), (2.123) and (2.124). This factorization mimics the structure of the differential operator. In this section, it will be shown that the node-edge, edge-face and face-volume commutation properties (7.38), (7.39) and (7.40) transfer this structure to the coarse levels of a multigrid hierarchy. Similar factorization results are obtained in [23] in the context of geometric multigrid for discretizations using the finite integration technique (FIT, [24]).

The preservation of this structure is very important in a multigrid context, since it is the basic requirement for the recursive application of a multigrid cycle to make sense. Intuitively speaking, the multigrid cycle has to match the problem to which it is applied on *all* levels of the hierarchy.

Theorem 7.6.1 *If the fine system matrices K_f^{dg}, K_f^{cc} and K_f^{gd} can be factorized as (2.122),(2.123) and (2.124):*

$$\begin{aligned}K_f^{dg} &= G_f^T M_{\alpha,f}^e G_f + M_{\beta,f}^n, \\ K_f^{cc} &= C_f^T M_{\alpha,f}^f C_f + M_{\beta,f}^e, \\ K_f^{gd} &= D_f^T M_{\alpha,f}^v D_f + M_{\beta,f}^f,\end{aligned}$$

and if the prolongators P_n, P_e, P_f and P_v satisfy the node-edge, edge-face or face-volume commutation properties (7.38), (7.39) and (7.40):

$$P_e G_c = G_f P_n, \quad P_f C_c = C_f P_e, \quad P_v D_c = D_f P_f,$$

then the Galerkin coarse system matrices can be factorized as (2.122), (2.123) and (2.124) as well:

$$\begin{aligned} K_c^{dg} &= G_c^T M_{\alpha,c}^e G_c + M_{\beta,c}^n, \\ K_c^{cc} &= C_c^T M_{\alpha,c}^f C_c + M_{\beta,c}^e, \\ K_c^{gd} &= D_c^T M_{\alpha,c}^v D_c + M_{\beta,c}^f, \end{aligned}$$

with the coarse mass matrices defined as:

$$M_{x,c}^n := P_n^T M_{x,f}^n P_n, \tag{7.46}$$
$$M_{x,c}^e := P_e^T M_{x,f}^e P_e, \tag{7.47}$$
$$M_{x,c}^f := P_f^T M_{x,f}^f P_f, \tag{7.48}$$
$$M_{x,c}^v := P_v^T M_{x,f}^v P_v. \tag{7.49}$$

Proof. The mass part holds by definition. The coarse stiffness part is obtained by applying the appropriate commutation property. For node-based problems, for instance, we have, using the node-edge commutation property and (7.47):

$$\begin{aligned} P_n^T G_f^T M_{\alpha,f}^e G_f P_n &= G_c^T P_e^T M_{\alpha,f}^e P_e G_c \\ &= G_c^T M_{\alpha,c}^e G_c. \end{aligned}$$

□

Remark 7.6.1 For finite element discretizations, the coarse mass matrices satisfy the same formula as the fine mass matrices. For instance, the mass matrix for the coarse edge elements reads:

$$\begin{aligned} M_{\alpha,c}^e(I,J) &= (P_e^T\, M_{\alpha,f}^e\, P_e)(I,J) \\ &= \iiint_V \alpha \left(\sum_r P_e(r,I) \vec{E}_r \right) \cdot \left(\sum_s P_e(s,J) \vec{E}_s \right) dV \\ &= \iiint_V \alpha\, \vec{E}_I \cdot \vec{E}_J \, dV. \end{aligned}$$

This shows that the coarse system matrices can be considered genuine finite element discretizations of the original problem using the coarse shape functions on the coarse topology implied by P_n.

7.7 Extension to quadrilateral and hexahedral meshes

The edge, face and volume prolongators derived in section 7.2 formally require the fine mesh to be simplicial. In this section, fine quadrilateral and hexahedral meshes will be considered. In section 7.7.1, we will show that the formulae for Whitney elements on non-simplicial meshes cannot always be applied to the discrete setting. In section 7.7.2, however, we will show how the prolongation formulae derived in section 7.2 can be changed, such that they can always be applied, regardless of the type of the underlying fine mesh.

7.7.1 Application of the formulae for quadrilateral and hexahedral meshes

The application of the analytical formulas for edge and face elements on quadrilateral or hexahedral fine meshes do not, in general, lead to a satisfactory prolongator. Consider for instance the case of edge elements. With $F_{i\bar{j}}$ denoting the set of all faces containing node i and *not* containing node j, the analytical formula for edge elements on quadrilateral and hexahedral meshes reads:

$$\vec{E}_{ij} \quad = \quad N_j \sum_{r \in F_{\bar{i}j}} \operatorname{grad}(N_r) - N_i \sum_{r \in F_{i\bar{j}}} \operatorname{grad}(N_r), \tag{7.50}$$

see [27]. A counterexample is illustrated in figure 7.6. Using (4.20) and the following partition of unity nodal prolongator

$$P_n = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ a & 0 & 1-a & 0 \\ 0 & b & 0 & 1-b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

formula (7.50) applied to the coarse edge e_{IJ} of the quadrilateral that is the union of A and B, becomes:

$$\begin{aligned} \vec{E}_{IJ} \quad &= \quad N_J \operatorname{grad}(N_J + N_L) - N_I \operatorname{grad}(N_I + N_K) \\ &= \quad (N_j + b\,N_s) \operatorname{grad}(N_j + N_s + N_l) - (N_i + a\,N_r) \operatorname{grad}(N_i + N_r + N_k) \\ &= \quad \vec{E}_{ij} + b\,N_s \operatorname{grad}(N_j + N_s + N_l) - a\,N_r \operatorname{grad}(N_i + N_r + N_k). \end{aligned}$$

In the last step, we took into account that $N_j \operatorname{grad}(N_l) = 0 = N_i \operatorname{grad}(N_k)$. If $a = b$, the coarse edge element $\vec{E}_{IJ}$ can be written as a linear combination of fine edge elements $\vec{E}_{IJ} = \vec{E}_{ij} + a\,\vec{E}_{rs}$. If $a \neq b$, this is not possible anymore. Suppose fine element A is the unit square, and consider the restriction of $\vec{E}_{IJ}$

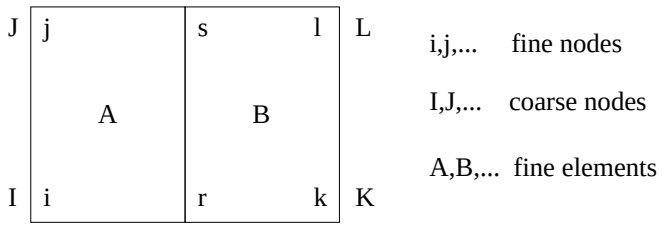


Figure 7.6: Coarsening on a 2D grid of quadrilaterals.

to A:

$$\begin{aligned}\left(\vec{E}_{IJ}\right)_{|A} &= \begin{bmatrix} 0 \\ 1-x \end{bmatrix} + b\,xy \begin{bmatrix} 0 \\ 1 \end{bmatrix} - a\,(1-y)x \begin{bmatrix} 0 \\ -1 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 1-(1-a)x+(b-a)xy \end{bmatrix}.\end{aligned}$$

Clearly, the coarse edge elements based on (7.50) will in general be piecewise polynomials of second order. So, they cannot be rewritten as linear combinations of fine edge elements, which are piecewise polynomials of first order.

7.7.2 Application of the formulae for simplicial meshes

In theory, the formulae (7.6), (7.12) and (7.17) to construct an edge, face or volume prolongator require the underlying fine mesh to be simplicial. This is because they rely on the assumption that all fine simplices exist which correspond to nonzero prolongation entries:

$$\begin{vmatrix} P_n(r,I) & P_n(r,J) \\ P_n(s,I) & P_n(s,J) \end{vmatrix} \neq 0 \implies \exists\ \pm e_{rs}, \tag{7.51}$$

$$\begin{vmatrix} P_n(r,I) & P_n(r,J) & P_n(r,K) \\ P_n(s,I) & P_n(s,J) & P_n(s,K) \\ P_n(t,I) & P_n(t,J) & P_n(t,K) \end{vmatrix} \neq 0 \implies \exists\ \pm f_{rst}, \tag{7.52}$$

$$\begin{vmatrix} P_n(r,I) & P_n(r,J) & P_n(r,K) & P_n(r,L) \\ P_n(s,I) & P_n(s,J) & P_n(s,K) & P_n(s,L) \\ P_n(t,I) & P_n(t,J) & P_n(t,K) & P_n(t,L) \\ P_n(u,I) & P_n(u,J) & P_n(u,K) & P_n(u,L) \end{vmatrix} \neq 0 \implies \exists\ \pm v_{rstu}. \tag{7.53}$$

Let $\hat{G}_f$, $\hat{C}_f$ and $\hat{D}_f$ denote the extended discrete differential operators, which describe *all* fine simplices which are assumed to exist. With E, F and V denoting the sets of the existing fine edges, faces and volumes, these extended differential operators satisfy:

$$\hat{G}_f(E,:) = G_f, \quad \hat{C}_f(F,:) = C_f, \quad \hat{D}_v(V,:) = D_v.$$

This allows the assumptions (7.51),(7.52) and (7.53) to be reformulated as:

$$\hat{G}_f = G_f, \quad \hat{C}_f = C_f, \quad \hat{D}_f = D_f. \tag{7.54}$$

These assumptions are only satisfied for simplicial meshes. In other words, if the underlying fine mesh is not simplicial, then (7.6), (7.12) and (7.17) calculate prolongation entries for *non-existent* fine simplices. For instance, in figure 7.1, the edge prolongation procedure (7.6) would calculate prolongation entries for the non-existent fine edges e_{ac} and e_{bd}. These fine edge belong to $\hat{G}_f$, but they don't belong to G_f.

Note, however, that this does not prevent the construction formulas (7.6), (7.12) and (7.17) to be applied. It suffices to calculate the prolongation entries *only for existent fine simplices*. In the discrete setting, this amounts to eliminating the rows corresponding to all non-existent fine simplices from the prolongation matrices:

$$P_e \leftarrow P_e(E,:), \quad P_f \leftarrow P_f(F,:), \quad P_v \leftarrow P_v(V,:).$$

The commutation properties proved in section 7.5 are satisfied for these prolongators. This can be seen as follows. The proofs of the commutation properties are based on the assumption that the underlying fine mesh is simplicial, i.e. on (7.54). In other words, what has been proved reads, in a general setting:

$$P_e G_c = \hat{G}_f P_n, \quad P_f C_c = \hat{C}_f P_e, \quad P_v D_c = \hat{D}_f P_n.$$

By eliminating all rows corresponding to non-existing fine simplices, the classical commutation properties are obtained:

$$\begin{aligned} P_e(E,:)G_c &= \hat{G}_f(E,:)P_n = G_f P_n, \\ P_f(F,:)C_c &= \hat{C}_f(F,:)P_e = C_f P_e, \\ P_v(V,:)D_c &= \hat{D}_f(V,:)P_n = D_f P_f. \end{aligned}$$

In section 7.9, we will show for curl-curl problems that this prolongation procedure yields efficient AMG solvers.

Remark 7.7.1 Note that this prolongation procedure does not transfer the quadrilateral or hexahedral character of the fine topology to the implied coarse topology, which is simplicial *by construction*.

7.8 Implementation

The edge prolongation scheme (7.6) has been implemented in an AMG method for curl-curl problems. In this section, we will discuss some implementation details. In section 7.8.1, the specific AMG algorithm used to construct the underlying nodal prolongator will be described. In section 7.8.2, we will discuss the strategy used to control the implied coarse topology.

7.8.1 Construction of the underlying nodal prolongator

The edge prolongator obtained from (7.6) can be derived from any partition of unity nodal prolongator. Such nodal prolongator can be built by any of the AMG algorithms presented in chapter 4.

Tests have been done with edge prolongators derived from the nodal prolongators obtained by the direct interpolation scheme of classical AMG (section 4.3.3), smoothed aggregation AMG (section 4.3.4) and AMG based on energy minimization (section 4.3.5). The results obtained with the smoothed aggregation and energy minimization variants proved to be significantly better than the results obtained with the classical AMG variant, both with respect to convergence rates and memory requirements. In particular, the classical AMG nodal prolongators proved to yield coarse topologies of poor quality, i.e. a coarse topology containing many crossing edges and overlapping volumes.

Smoothed aggregation methods cannot be combined easily with the coarse topology control strategy of section 7.8.2. Hence, we have chosen to use nodal prolongators obtained by energy minimization nodal AMG in the numerical experiments of section 7.9. More precisely, energy minimization is applied to a simple prolongation operator which is based on the node connectivity. In a first step, the nonzero pattern of P_n is determined, as follows:

1. Aggregation of the nodes (see section 4.2.2), *without* taking into account the strength of the connection, i.e. $S_i = \text{neighbours}(i)$. Each aggregate will correspond to a coarse node.

2. Extending each aggregate N with one hop distance:

$$N \quad \leftarrow \quad N \ \cup \ \{f \mid \exists n \in N \cap \text{neighbours}(f)\}.$$

3. Each fine node f is prolongated from all aggregates to which it belongs:

$$P_n(n, N) \neq 0 \quad \Longleftrightarrow \quad n \in N.$$

The entries of the initial prolongator are determined as follows, with $nz(n)$ denoting the nonzero locations of $P_n(n, :)$:

$$\forall N \in nz(n) \quad : \quad P_n(n, N) = 1/\#nz(n).$$

Clearly, the resulting P_n satisfies the partition of unity property (4.22).

Finally, the energy of P_n is minimized with respect to a diffusion matrix. For this purpose, we have used a slight variant (unpublished) of the energy minimization algorithm developed in [48] by Mandel, Vanek and Brezina. The full details of the algorithm that we have used can be found in Appendix A.1. In all experiments reported in this chapter and unless mentioned explicitly otherwise, the energy of P_n will be reduced by two minimization iterations.

7.8.2 Controlling the implied coarse topology

The second step of the algorithm of section 7.8.1 can be adapted in such a way that the coarse topology implied by the piecewise constant nodal prolongator does not change. This can be achieved by allowing an aggregate N to be extended with a fine node f *only* if this causes no additional coarse simplices to be introduced:

$$N \leftarrow N \cup \left\{ \begin{array}{l} f \in \text{neighbours}(N) \mid \\ P_n(f,N) \neq 0 \text{ does not change the coarse topology} \end{array} \right\}. \quad (7.55)$$

This constraint requires a list of all coarse edges, faces and volumes to be available. For this purpose, the coarse topological operators can be used. Since the coarse topology, implied by the original node aggregation of the first step is not crossing (see the proof of lemma 7.3.1), this procedure yields a non-crossing coarse topology.

Controlling the coarse topology reduces the memory requirements for the coarse system matrices in the multigrid hierarchy (see section 7.3.4). More precisely, the nonzero pattern of a coarse system matrix obtained by the Galerkin product with P_n, P_e or P_f will be identical in this case to the nonzero pattern obtained with the piecewise constant nodal, edge or face prolongator, i.e. *without* extending the aggregates.

A drawback of this approach is the reduction of the number of nonzeros of P_n, P_e and P_f. Consequently, the prolongation quality may deteriorate. However, this can be compensated by extending the aggregates by more than one hop distance, i.e. by repeating the constrained extension step (7.55). Note that this will not affect the memory complexity of coarse system matrices obtained by the Galerkin product with P_n, P_e or P_f for the same reason as in the previous paragraph.

7.9 Numerical results

In this section, the edge prolongator (7.6) will be used to solve finite element discretizations of the curl-curl equation (7.2). The piecewise constant edge prolongator (5.26) presented in [56], the smoothed edge prolongator (5.27) presented in [5], and the edge prolongator (7.6) will be compared with respect to convergence speed, setup and solve timings and memory requirements.

For the smoothed edge prolongator (5.27), the implementation of ML is used. ML, which is part of the Trilinos project [62] developed at SANDIA National Laboratory (US), contains a state-of-the art parallel implementation of AMG for the curl-curl equation. The algorithm (5.27) is its default configuration.

Both the half and full hybrid smoother will be used in the experiments. The number of iterations of the edge and nodal subsmoother (see section 5.1.2) are always chosen identical, and will be indicated between brackets. For instance, the full hybrid smoother with 1 iteration of the edge and nodal subsmoother is

indicated as follows:

$$\text{full(1)} \quad : \quad \epsilon = \epsilon_{pre} = \epsilon_{post} = 1 \quad \text{and} \quad \nu = \nu_{pre} = \nu_{post} = 1.$$

The experiments were done on an Opteron 875, 2.2 GHz machine of the cluster of the KULeuven. All codes were compiled with the g++-3.3 compiler, using the optimizing flags "-O2 -NDEBUG -funroll-all-loops".

Throughout this section, different AMG solvers will be referred to as follows. PC refers to AMG using piecewise constant edge prolongators (5.26). ML refers to the implementation in ML of AMG using smoothed edge prolongators (5.27). The edge AMG algorithm based on the nodal AMG method described in section 7.8.1, will be referred to as WP, which stands for Whitney Prolongator. This naming is motivated by the fact that this edge prolongation scheme is derived from an analytical formula for Whitney edge elements. The AMG algorithm in which the implied coarse topology is controlled, as described in section 7.8.2, will be referred to as WP(tc), which stands for Whitney Prolongator with topology control.

In this section, only Neumann boundary conditions will be considered, unless mentioned explicitly otherwise. In this way, the influence of boundary effects on the multigrid convergence is reduced, which allows to focus on the quality of the prolongation scheme. More precisely, Dirichlet boundary conditions will prove to have an adverse effect on the convergence speed for the WP algorithm.

Unless mentioned explicitly otherwise, the righthand side of all linear systems solved in this section is a random vector, and the iteration starts from a zero initial guess for the solution. The iteration is stopped when the norm of the residual is a factor 10^{-14} smaller than the norm of the righthand side:

$$\frac{\|b - Kx\|}{\|b\|} \quad < \quad 10^{-14}. \tag{7.56}$$

The use of the same stopping criterium for all problem sizes is common practice in multigrid literature, and it used to compare the behavior of the solver as a function of the mesh size. The high required accuracy reduces the impact of the convergence factors in the first few iterations, which are typically lower than the average.

The memory complexity of a hierarchy of matrices is defined as follows, with K_l denoting the matrix on level l, $l = 1$ and $l = L$ being the finest and coarsest level, and with $nnz(K_l)$ denoting the number of nonzero entries of K_l:

$$\text{memory complexity} \quad := \quad \frac{\sum_{l=1}^{L} nnz(K_l)}{nnz(K_1)}. \tag{7.57}$$

7.9.1 Two-dimensional problems

First, we consider finite element discretizations of the curl-curl equation (7.2) on the unit square, which is split into two equal parts Ω_1 and Ω_2. In each of both subdomains, the material parameters are isotropic and constant, and will be denoted (α_1, β_1) and (α_2, β_2) respectively.

Problem 1: triangular mesh and homogeneous materials

First, we consider homogeneous material parameters $\alpha = \alpha_1 = \alpha_2$ and $\beta = \beta_1 = \beta_2$.

Configuration of the hybrid smoother. Table 7.3 and 7.4 show convergence and timing results for WP for $\alpha = \beta = 1$ as a function of the configuration of the hybrid smoother. The use of the full hybrid smoother results in a faster convergence than the use of the half hybrid smoother. However, the time per iteration is larger for the full hybrid smoother than for the half hybrid smoother. Additional iterations of the subsmoothers of the hybrid smoother result in a faster convergence. However, this causes the solve time to be reduced only for the half hybrid smoother. In the rest of the experiments, the half(2) hybrid smoother, i.e. the half hybrid smoother with 2 iterations of the subsmoothers, will be used. This corresponds to the default configuration of ML.

Configuration of the cycle configuration. Table 7.5 and 7.6 show convergence and timing results for WP for $\alpha = \beta = 1$ as a function of the configuration of the cycle. Additional pre- and postsmoothing steps yield lower convergence factors, but increased solve times. In the rest of the experiments, only the V(1,1)-cycle will be used.

Influence of β. Table 7.7 shows that for relatively small values of β , the convergence factor of WP is essentially independent of β. For increasing values of β, the convergence benefits from the growing weight of the mass part, which causes the condition of the system matrix to improve.

Comparison of edge prolongation schemes. In tables 7.8, 7.9 and 7.11, the convergence factors, timings and memory complexities of PC, ML, WP and WP(tc) are compared. This comparison shows that the convergence factors of the WP-algorithms are significantly better than the convergence factors of the other algorithms. Also, the WP-algorithms scale much better with the system size, i.e. the (relative) increase of the number of iterations as a function of the system size is much smaller. The overall solve time of the WP-algorithms is up to a factor 4 smaller than ML. On the other hand, the memory complexity is higher for WP than for ML. However, this can be compensated by using WP(tc) instead. The smaller memory complexity of WP(tc) in comparison with WP explains the smaller time per iteration of WP(tc) (see table 7.10). For the problem at hand, this causes the overall solve time to be smaller for WP(tc) than for WP, despite the higher number of iterations needed by WP(tc).

Influence of Dirichlet boundary conditions. Table 7.12 shows that the convergence of the WP algorithm degrades in the presence of Dirichlet boundary conditions. ML, on the other hand, seems to benefit a little from Dirichlet

nbNodes	nbEdges	half(1)	half(2)	full(1)	full(2)
10547	31244	0.33(30)	0.20(20)	0.23(23)	0.14(17)
41846	124741	0.35(31)	0.22(21)	0.25(24)	0.16(18)
166061	496586	0.37(33)	0.24(23)	0.28(26)	0.19(20)
663190	1986373	0.41(36)	0.28(26)	0.32(28)	0.22(22)

Table 7.3: Geometric mean convergence factors and number of iterations for a WP V(1,1)-cycle accelerated by CG for problem 1 with $\alpha = \beta = 1$, as a function of the configuration of the hybrid smoother.

nbNodes	nbEdges	half(1)	half(2)	full(1)	full(2)
10547	31244	0.80	0.55	0.61	0.66
41846	124741	3.21	2.80	3.17	3.40
166061	496586	18.70	17.17	18.07	19.02
663190	1986373	109.52	100.37	96.08	110.97

Table 7.4: Solve time in seconds (setup time not included) for a WP V(1,1)-cycle accelerated by CG for problem 1 with $\alpha = \beta = 1$, as a function of the configuration of the hybrid smoother.

boundaries conditions. The appropriate treatment of Dirichlet boundary conditions by the WP algorithm may require further research.

Problem 2: triangular mesh and heterogeneous materials

Table 7.13 shows the influence of material jumps on the convergence of the WP AMG solver. If the value of one parameter increases and the value of the other decreases, the number of iterations approximately doubles. In all other cases, the increase of the number of iterations is smaller. On the other hand, table 7.14 shows that the convergence rate of ML typically improves slightly for the same material jumps, although it remains way below that of WP AMG.

As mentioned in section 7.8.1, the energy of the nodal prolongators, from

		half(2)		full(2)	
nbNodes	nbEdges	V(1,1)	V(2,2)	V(1,1)	V(2,2)
10547	31244	0.20(20)	0.12(16)	0.14(17)	0.10(14)
41846	124741	0.22(21)	0.15(17)	0.16(18)	0.12(16)
166061	496586	0.24(23)	0.18(19)	0.19(20)	0.15(17)
663190	1986373	0.28(26)	0.21(21)	0.22(22)	0.18(19)

Table 7.5: Geometric mean convergence factors and number of iterations for a WP AMG-cycle accelerated by CG for problem 1 with $\alpha = \beta = 1$, as a function of the configuration of the cycle and the hybrid smoother.

		half(2)		full(2)	
nbNodes	nbEdges	V(1,1)	V(2,2)	V(1,1)	V(2,2)
10547	31244	0.55	0.69	0.66	0.90
41846	124741	2.80	3.50	3.40	4.95
166061	496586	17.17	21.47	19.02	29.50
663190	1986373	100.37	126.89	110.97	175.59

Table 7.6: Solve time in seconds (setup time not included) for a WP AMG-cycle accelerated by CG for problem 1 with $\alpha = \beta = 1$, as a function of the configuration of the cycle and the hybrid smoother.

nbNodes	nbEdges	10^6	10^3	1	10^{-3}	10^{-6}
10547	31244	0.002(6)	0.12(16)	0.20(20)	0.20(20)	0.20(20)
41846	124741	0.001(5)	0.15(18)	0.22(21)	0.22(22)	0.22(22)
166061	496586	0.01(7)	0.18(19)	0.24(23)	0.24(23)	0.24(23)
663190	1986373	0.06(12)	0.20(20)	0.28(26)	0.28(26)	0.28(26)

Table 7.7: Geometric mean convergence factors and number of iterations for a WP V(1,1)-cycle accelerated by CG using the half(2) hybrid smoother for problem 1 with $\alpha = 1$, as a function of β.

		div-grad	curl-curl			
nbNodes	nbEdges		PC	ML	WP(tc)	WP
10547	31244	0.15(17)	0.74(107)	0.59(61)	0.20(21)	0.20(20)
41846	124741	0.19(20)	0.84(184)	0.69(86)	0.21(21)	0.22(21)
166061	496586	0.20(20)	0.90(307)	0.76(116)	0.27(25)	0.24(23)
663190	1986373	0.22(21)	0.94(532)	0.83(168)	0.33(29)	0.28(26)

Table 7.8: Geometric mean convergence factors and number of iterations for a V(1,1)-cycle accelerated by CG using the half(2) hybrid smoother for problem 1 with $\alpha = \beta = 1$, for different solvers. The convergence of a node-based V(2,2)-cycle accelerated by CG using the nodal prolongator described on section 7.8.1 for the div-grad equation (7.1) with $\alpha = \beta = 1$ discretized on the same mesh, is shown as a reference.

	div-grad	curl-curl			
nbEdges		PC	ML	WP(tc)	WP
31244	0.04/0.09	0.05/2.35	0.23/1.82	0.21/0.68	0.17/0.77
124741	0.23/0.63	0.36/20.07	1.24/14.70	1.28/3.40	1.11/4.05
496586	1.59/3.30	2.28/221.25	5.42/79.84	7.36/15.48	6.85/25.36
1986373	9.36/23.84	12.89/1756.76	25.93/589.29	30.74/100.84	28.60/144.47

Table 7.9: Setup/solve time in seconds for a V(1,1)-cycle accelerated by CG using the half hybrid smoother with $\alpha = \beta = 1$, for the same problems and solvers as in table 7.8.

	div-grad	curl-curl			
nbEdges		PC	ML	WP(tc)	WP
31244	0.0053	0.022	0.030	0.032	0.040
124741	0.0315	0.109	0.171	0.162	0.193
496586	0.1650	0.721	0.688	0.619	1.103
1986373	1.1352	3.302	3.508	3.477	5.557

Table 7.10: Time in seconds needed for one V(1,1)-cycle accelerated by CG using the half(2) hybrid smoother with $\alpha = \beta = 1$, for the same problems and solvers as in table 7.8.

nbEdges	PC	ML	WP(tc)	WP
31244	1.095(4)	1.163(4)	1.106(4)	1.390(4)
124741	1.096(5)	1.161(5)	1.105(4)	1.398(4)
496586	1.097(5)	1.160(5)	1.102(5)	1.409(5)
1986373	1.097(6)	1.160(6)	1.100(6)	1.408(5)

Table 7.11: Memory complexity and number of levels of the edge hierarchy for the same problem and solvers as in table 7.8.

		WP		ML	
nbNodes	nbEdges	Neumann	Dirichlet	Neumann	Dirichlet
10547	31244	0.20(20)	0.35(31)	0.59(61)	0.55(55)
41846	124741	0.22(21)	0.38(34)	0.69(86)	0.66(77)
166061	496586	0.24(23)	0.42(38)	0.76(116)	0.73(102)
663190	1986373	0.28(26)	0.48(45)	0.83(168)	0.81(149)

Table 7.12: Geometric mean convergence factors and number of iterations for a V(1,1)-cycle accelerated by CG using the half(2) hybrid smoother for problem 1 with $\alpha = \beta = 1$ and Dirichlet and Neumann boundary conditions applied to all boundaries.

nbEdges	ref	α_1		β_1		$\alpha_1 = \beta_1$		$\alpha_1 = \beta_2$	
		10^{-2}	10^{-4}	10^{-2}	10^{-4}	10^{-2}	10^{-4}	10^{-2}	10^{-4}
31244	20	20/23	21/21	23/22	25/24	22/22	27/27	26/29	43/42
124741	21	23/28	24/23	27/23	29/25	26/25	33/31	28/32	47/46
496586	23	26/28	26/25	28/25	29/28	28/27	32/31	32/34	51/51
1986373	26	26/32	27/28	27/27	28/31	28/29	33/36	33/38	55/53

Table 7.13: Number of iterations needed for a WP V(1,1)-cycle accelerated by CG using the half(2) hybrid smoother for problem 2 with different types of material jumps. All material parameters except the ones indicated in the first row of the table, are identical to 1. The result for $\alpha_1 = \alpha_2 = \beta_1 = \beta_2 = 1$ is indicated as a reference. The first and second result are obtained by using edge prolongators which are based on nodal prolongators optimized with respect to $G^T M_\alpha^e G$ and $G^T M_\beta^e G$ respectively (see section 7.8.1).

nbEdges	ref	α_1		β_1		$\alpha_1 = \beta_1$		$\alpha_1 = \beta_2$	
		10^{-2}	10^{-4}	10^{-2}	10^{-4}	10^{-2}	10^{-4}	10^{-2}	10^{-4}
31244	61	54	55	61	61	59	60	58	65
124741	86	78	75	86	86	85	82	82	92
496586	116	102	106	113	113	111	111	109	119
1986373	168	152	148	166	166	164	162	157	170

Table 7.14: Number of iterations needed for a ML V(1,1)-cycle accelerated by CG using the half(2) hybrid smoother for problem 2 with the same types of material jumps as in table 7.13.

which the edge prolongators are derived, can be minimized with respect to either $G^T M_\alpha^e G$ or $G^T M_\beta^e G$. The numerical results of table 7.13 don't allow to conclude which approach is the best.

The performance of the WP solver might be improved by incorporating semi-coarsening techniques in the nodal prolongators from which the edge prolongators are derived (see section 7.8.1) and/or by decoupling the nodal and edge hierarchy as presented in [41] (see section 5.3.1). This has not been tested here.

Problem 3: quadrilateral mesh

As described in section 7.7.2, the WP AMG algorithm can be applied to curl-curl problems discretized on quadrilateral meshes as well. Table 7.15 shows convergence and timing results for the curl-curl equation with $\alpha = \beta = 1$, discretized on a regular quadrilateral grid for the unit square. Although the convergence speed of the WP algorithm clearly degrades in comparison with the convergence for problems discretized on a triangular mesh, the performance of the WP-solver is still better than the performance of ML.

		WP		ML	
nbNodes	nbEdges	$\bar{\rho}(nbIter)$	Tsolve	$\bar{\rho}(nbIter)$	Tsolve
2601	5100	0.26(24)	0.10	0.37(33)	0.11
10201	20200	0.30(27)	0.71	0.44(39)	0.85
40401	80400	0.38(33)	4.22	0.54(52)	5.38
160801	320800	0.42(37)	21.38	0.65(76)	36.12
641601	1281600	0.46(42)	55.25	0.72(100)	210.30

Table 7.15: Geometric mean convergence factors, number of iterations and solve time in seconds (setup time not included) for a V(1,1)-cycle accelerated by CG using the half(2) hybrid smoother, for problem 1 discretized on a regular quadrilateral mesh with $\alpha = \beta = 1$.

7.9.2 Three-dimensional problems

Finally, we consider finite element discretizations of the curl-curl equation (7.2) with constant material parameters α and β on a tetrahedral discretization of the unit cube. Tables 7.16 and 7.17 compare convergence and timing results and memory complexities for WP and ML. The convergence and timing results are still better for WP than for ML, but the difference is smaller than for 2D problems. The reduction of the timing differences can be explained partially by the significant growth of the memory complexity of WP. More precisely, a higher memory complexity implies a higher amount of floating point operations to be executed per cycle, which causes the time per cycle to increase.

The memory complexities of WP, observed in the experiments, are significantly higher for 3D problems than for 2D problems. Corresponding to section 7.3.4, this indicates that the coarse topologies implied by the nodal prolongators, from which the edge prolongators are derived, are more problematic in 3D than in 2D. This indicates that controlling the coarse topologies is more important for 3D problems than for 2D problems. Unfortunately, the topology control algorithm of section 7.8.2, which was applied successfully for 2D problems, does not yield good results for 3D problems. More precisely, the constraints imposed by (7.55) prove to be much more restrictive in 3D. This results in a significant degradation of the convergence.

7.10 Concluding remarks

In this chapter, we developed a procedure to derive an edge, face and volume prolongator from an arbitrary nodal prolongator, which satisfies the partition of unity property. We showed that this procedure implies the existence of coarse edges, faces and volumes. This coarse topology can be represented by a coarse gradient, curl and divergence operator.

This procedure can be considered very natural, as it mimics the relations which exist between Whitney elements. For instance, the coarse shape functions

		WP			ML		
nbNodes	nbEdges	$\bar{\rho}$(nbIter)	Tsolve	mem	$\bar{\rho}$(nbIter)	Tsolve	mem
12231	84461	0.26(25)	5.28	1.581	0.52(49)	7.72	1.153
92127	657907	0.43(39)	88.27	1.725	0.66(77)	123.25	1.177
178358	1281831	0.40(35)	168.71	1.746	0.67(81)	267.11	1.177
417239	3019710	0.46(42)	511.64	1.750	0.68(85)	721.05	1.181

Table 7.16: Geometric mean convergence factors, number of iterations and solve time in seconds (setup time not included) and memory complexities for a V(1,1)-cycle accelerated by CG using the half(2) hybrid smoother, for problem 3 with homogeneous Neumann boundary conditions and $\alpha = \beta = 1$.

		WP			ML		
nbNodes	nbEdges	$\bar{\rho}$(nbIter)	Tsolve	mem	$\bar{\rho}$(nbIter)	Tsolve	mem
12231	84461	0.30(27)	5.70	1.581	0.52(50)	7.90	1.148
92127	657907	0.43(39)	88.28	1.725	0.65(74)	117.59	1.174
178358	1281831	0.42(37)	178.43	1.746	0.68(83)	282.39	1.174
417239	3019710	0.46(42)	515.82	1.750	0.67(82)	694.97	1.179

Table 7.17: Geometric mean convergence factors, number of iterations and solve time in seconds (setup time not included) and memory complexities for a V(1,1)-cycle accelerated by CG using the half(2) hybrid smoother, for problem 3 with Dirichlet boundary conditions and $\alpha = \beta = 1$.

represented by the columns of the nodal, edge, face and volume prolongators, and the coarse topological operators, form a coarse de Rham sequence. This property is very important in the context of multigrid algorithms. However, the implied coarse topology contains crossing edges and faces and overlapping volumes. We showed that this causes the memory complexity of AMG algorithms to grow. We developed a procedure to control the coarse topology. This procedure imposes constraints on the nonzero structure of the nodal prolongator from which the other prolongators are derived.

The edge prolongator developed in this chapter, has been used in an AMG code for the curl-curl equation. For 2D problems, this code is significantly faster than ML, both with respect to convergence speed and timings. The topology control algorithm is able to reduce the memory complexity significantly in this case. For 3D problems, the performance difference between both codes reduces. This is caused, at least partially, by the character of the topologies on the coarse levels of the multigrid hierarchy.

Numerical tests show that the presented AMG algorithm is more sensitive to jumps of the material parameters than ML. Also, and contrary to ML, its convergence speed decreases if Dirichlet boundary conditions are applied. Together with the control of the coarse topology in 3D, these subjects need further research.

Chapter 8

AMG for higher order time discretization

8.1 Introduction

Implicit Runge-Kutta (IRK) and Boundary Value Method (BVM) time discretizations have many appealing properties, such as high order of accuracy and good stability [20, 33, 34, 19]. However, when applied to time-dependent partial differential equations, they give rise to linear systems which are difficult to solve. First of all, the size of these systems is a multiple of the size of the systems to be solved for the backward Euler (BE) discretization of the same problem. Also, existing solvers are typically tailored for BE discretizations, and their efficiency usually significantly deteriorates when applied to higher order time discretizations.

In the case of IRK methods, the computational cost can be reduced significantly by using *diagonally implicit* (DIRK) Runge-Kutta methods. The resulting system is then block-triangular, and can be solved as a sequence of smaller problems. These smaller problems have a similar structure as the one to be solved in the BE method, hence efficient solvers for that method can be reused. Unfortunately, DIRK methods have a lower order of accuracy and poorer stability properties than fully coupled IRK methods of the same size [20, 34]. This order and stability reduction can be avoided by using a DIRK method as a *preconditioner* for the fully coupled IRK system [50]. The algorithm developed in [50] allows the reuse of existing BE software components. However, the number of iterations required for solving the preconditioned IRK-system increases rapidly with the number of stages in the IRK method.

In [64], it is shown how geometric multigrid techniques can be used to solve efficiently *fully* coupled IRK and BVM time discretizations of finite difference spatial discretizations of the time-dependent div-grad equation

$$-\operatorname{div}(\alpha \operatorname{grad}(U)) + \beta \frac{\partial U}{\partial t} \quad = \quad f. \tag{8.1}$$

The key is the use of a blocksmoother updating all unknowns related to a grid point simultaneously. In this chapter, the approach of [64] will be extended to *algebraic* multigrid (AMG) for (8.1) and for the time-dependent curl-curl equation

$$\text{curl}(\alpha \text{ curl}(\vec{U})) + \beta \frac{\partial \vec{U}}{\partial t} \quad = \quad \vec{f}. \tag{8.2}$$

We will consider both IRK and BVM discretizations.

A theoretical estimate of the complexity of the AMG algorithm developed in this chapter would lead one to conclude that the number of floating point operations and the memory requirements scale cubically and quadratically respectively as a function of the number of stages in the IRK or BVM method. It will be shown, however, that the algorithm offers several opportunities for implementation optimization, yielding a significant increase of the floating point execution rate and a significant reduction of the memory requirements. The timing results of the optimized implementation will be shown to scale rather linearly as a function of the number of stages. Thanks to the favorable timings and memory requirements and to the high order of accuracy of the considered time discretization methods, the presented AMG algorithm is very useful in practice.

This chapter is organized as follows. In section 8.2, we will present a brief overview of the time discretization methods used in the rest of this chapter. In section 8.3, the block AMG algorithm is discussed, and in section 8.4 a convergence analysis is presented. Section 8.5 deals with important implementation issues. In section 8.6, some numerical results are presented that illustrate the efficiency of the method.

8.2 Discretization in time

In this section, we will give a brief overview of the time discretization methods that will be used in the rest of this chapter.

8.2.1 Time discretization methods

Consider the system of n ordinary differential equations:

$$\frac{du}{dt} = f(t,u), \qquad \text{with} \quad u(t_0) = u_0 \in \mathbb{R}^n. \tag{8.3}$$

The considered time discretization methods use a time stepping procedure. Let t_0 and Δt denote the initial time and the time step. A time stepping procedure yields an approximation u_1 of the solution at the time $t_1 := t_0 + \Delta t$:

$$u_1 \quad \simeq \quad u(t_1). \tag{8.4}$$

The forward Euler, backward Euler and Crank-Nicholson discretization schemes are based on the first order approximation of the time derivative:

$$\frac{du}{dt} \quad \simeq \quad \frac{u_1 - u_0}{\Delta t}. \tag{8.5}$$

In a **forward Euler** time discretization, the excitation is evaluated in t_0, which yields an explicit time stepping scheme:

$$u_1 \quad = \quad u_0 + \Delta t\, f(t_0, u_0). \tag{8.6}$$

The cost per iteration of forward Euler time discretization is very low, but stability requirements typically impose very small values for Δt.

In **backward Euler (BE)** time discretization, the excitation is evaluated in t_1. This discretization method is always stable, but it yields a more expensive, implicit scheme:

$$u_1 \quad = \quad u_0 + \Delta t\, f(t_1, u_1). \tag{8.7}$$

The order of accuracy of both forward and backward Euler time discretizations is $\mathcal{O}(\Delta t)$, i.e. they have first order accuracy. A second order scheme, which is called **Crank-Nicholson** time discretization, is obtained by averaging (8.6) and (8.7):

$$u_1 \quad = \quad u_0 + \Delta t\, \frac{f(t_0, u_0) + f(t_1, u_1)}{2}. \tag{8.8}$$

A **Runge-Kutta (RK)** method (see [20, 33, 34, 43]) calculates the approximation u_1 of $u(t_1)$ as:

$$u_1 \quad = \quad u_0 + \Delta t \sum_{j=1}^{s} b_j f(t_0 + c_j \Delta t, x_j). \tag{8.9}$$

For this purpose, s auxiliary vectors $x_i \in \mathbb{R}^n$ are used. These vectors are called *stage vectors*, and they are required to satisfy the following system of equations:

$$x_i \quad = \quad u_0 + \Delta t \sum_{j=1}^{s} a_{ij} f(t_0 + c_j \Delta t, x_j), \quad i = 1 \ldots s. \tag{8.10}$$

Let the coefficients a_{ij} and b_i of the RK method be assembled in a matrix $A_{irk} \in \mathbb{R}^{s \times s}$ and a vector $b_{irk} \in \mathbb{R}^s$ as:

$$A_{irk} = \begin{bmatrix} a_{11} & \ldots & a_{1s} \\ \vdots & & \vdots \\ a_{s1} & \ldots & a_{ss} \end{bmatrix}, \qquad b_{irk} = \begin{bmatrix} b_1 \\ \vdots \\ b_s \end{bmatrix},$$

and let the vector $f_j \in \mathbb{R}^n$ and the multivectors $F, X, U_0 \in \mathbb{R}^{n \times s}$ be defined as follows:

$$\begin{aligned} f_j &:= f(t_0 + c_j \Delta t, x_j), \\ F &:= \left[\begin{array}{ccc} f_1 & \ldots & f_s \end{array} \right], \\ X &:= \left[\begin{array}{ccc} x_1 & \ldots & x_s \end{array} \right], \\ U_0 &:= \left[\begin{array}{ccc} u_0 & \ldots & u_0 \end{array} \right]. \end{aligned}$$

This allows (8.9) and (8.10) to be written compactly as:

$$u_1 = u_0 + \Delta t \, F \, b_{irk} \tag{8.11}$$
$$X = U_0 + \Delta t \, F \, A_{irk}^T. \tag{8.12}$$

A strictly lower triangular A_{irk} yields an explicit scheme. Explicit Runge-Kutta methods can have a high order of accuracy, but similarly as for the forward Euler method, the time step is limited due to stability requirements.

If A_{irk} is lower triangular and all diagonal elements are nonzero, the method is called a *diagonally implicit Runge-Kutta (DIRK)* method. A *singly diagonally implicit Runge-Kutta (SDIRK)* method is a DIRK method, with all diagonal entries of A_{irk} being identical. The most general case of a dense A_{irk} is referred to as a *fully implicit Runge-Kutta method (FIRK)*. Typically, FIRK methods have a higher order of accuracy than (S)DIRK methods for the same number of stages.

Boundary value methods (BVM) (see [19]) can be considered a generalization of linear multistep methods (LMM, see [33]). In each step of a BVM, the solution values at a series of n points in time $t_1, t_2, \ldots, t_n$ are approximated. Here, for simplicity of notation, we shall only consider the equidistant case, where $t_j = t_0 + j\Delta t$. A k-step BVM for (8.3) has for each $i = k_1, \ ..., \ n - k_2$, with $k_1 + k_2 = k$, an equation of the form

$$\sum_{j=-k_1}^{k_2} \alpha_{k_1+j} u_{i+j} = \Delta t \sum_{j=-k_1}^{k_2} \beta_{k_1+j} f(t_{i+j}, u_{i+j}). \tag{8.13}$$

The main difference with traditional LMM time discretization is the use of *future* points in time. As such, a BVM discretization cannot be solved in a time-marching way. An additional $k_1 - 1$ initial and k_2 final equations are needed of the form

$$\begin{array}{rcll} \sum_{j=0}^{k} \alpha_j^{(i)} u_j & = & \Delta t \sum_{j=0}^{k} \beta_j^{(i)} f(t_j, u_j) & i = 1, \ ..., \ k_1 - 1 \\ \sum_{j=0}^{k} \alpha_j^{(i)} u_{n-k+j} & = & \Delta t \sum_{j=0}^{k} \beta_j^{(i)} f(t_{n-k+j}, u_{n-k+j}) & i = n - k_2 + 1, \ ..., \ n. \end{array} \tag{8.14}$$

The coefficients $\alpha_j^{(i)}$ and $\beta_j^{(i)}$ are chosen such that the truncation errors are of the same order as for the basic method (8.13). When applied to (8.3), the BVM problem reads in compact format:

$$[u_0 \ U] \ [a_0 \ A_{bvm}]^T = \Delta t [f_0 \ F] \ [b_0 \ B_{bvm}]^T, \tag{8.15}$$

with U and F defined as follows:

$$\begin{aligned} U &:= [\, u_1 \ \cdots \ u_n \,], \\ F &:= [\, f_1 \ \cdots \ f_n \,]. \end{aligned}$$

The vectors u_0 and f_0 represent the initial condition and the corresponding right-hand side value at time t_0. The matrices $A_{bvm},\ B_{bvm} \in \mathbb{R}^{n\times n}$ have a quasi-Toeplitz structure. For instance, $[a_0\ A_{bvm}]$ is given by:

$$\left[\begin{array}{c|ccc|ccc} \alpha_0^1 & \alpha_1^1 & \dots & \alpha_k^1 & & & \\ \vdots & & & \vdots & & & \\ \alpha_0^{k_1-1} & \alpha_1^{k_1-1} & \dots & \alpha_k^{k_1-1} & & & \\ \alpha_0 & \alpha_1 & \dots & \alpha_k & & & \\ & & & \ddots & & & \\ & & & & \alpha_0 & \dots & \alpha_k \\ & & & & \alpha_0^{n-k_2+1} & \dots & \alpha_k^{n-k_2+1} \\ & & & & \vdots & & \vdots \\ & & & & \alpha_0^n & \dots & \alpha_k^n \end{array}\right]. \tag{8.16}$$

The structure of B_{bvm} is similar, and composed of the β-coefficients from (8.13) and (8.14). Assuming that A_{bvm} is invertible, (8.15) can be transformed into a system of the form (8.12):

$$\begin{aligned} U &= \left(-u_0 a_0^T + \Delta t f_0 b_0^T\right) A_{bvm}^{-T} + \Delta t F B_{bvm}^T A_{bvm}^{-T} \\ &:= \tilde{U}_0 + \Delta t F \tilde{A}^T. \end{aligned} \tag{8.17}$$

The formal equivalence of these systems allows a BVM problem to be studied as an IRK problem with A_{irk} replaced by $A_{bvm}^{-1} B_{bvm}/n$. The scaling factor $1/n$ takes into account the specific meaning of the time step Δt. For IRK methods, it corresponds to the *global* time step; for BVM methods, it is the time step *between the intermediate steps.*

8.2.2 Application to a semi-discretized PDE

Next, these time discretization methods will be applied to the spatial discretizations of (8.1) and (8.2). For both problem types, the spatial discretization methods presented in chapter 2 give rise to the following set of equations:

$$Ku + M\frac{du}{dt} = b. \tag{8.18}$$

When applied to a system of the form (8.18), forward Euler, backward Euler and Crank-Nicholson time discretization yield respectively:

$$Mu_1 = Mu_0 + \Delta t\,(b_0 - Ku_0), \tag{8.19}$$

$$(M + \Delta t K)\, u_1 = Mu_0 + \Delta t\, b_1, \tag{8.20}$$

$$\left(M + \frac{\Delta t}{2} K\right) u_1 = Mu_0 + \Delta t\, \frac{b_1}{2} + \Delta t\, \frac{(b_0 - Ku_0)}{2}. \tag{8.21}$$

For IRK methods, the main computational cost is the solution of (8.10). The application of (8.10) to (8.18) yields a system of the following form:

$$\begin{bmatrix} M + \Delta t K a_{11} & \dots & \Delta t K a_{1s} \\ \vdots & & \vdots \\ \Delta t K a_{s1} & \dots & M + \Delta t K a_{ss} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \vdots \\ \tilde{x}_s \end{bmatrix} = \tilde{b}. \tag{8.22}$$

Here, the unknowns are ordered block-wise; the vectors $\tilde{x}_i \in \mathbb{R}^n$ correspond to the unknowns of the ith IRK stage, and the righthand side $\tilde{b}$ can readily be computed from known quantities. With I_s the unit matrix of size s and $\otimes$ the Kronecker matrix product symbol

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1m}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \cdots & a_{nm}B \end{bmatrix}, \quad \text{with } A \in \mathbb{R}^{n \times m},$$

(8.22) can be written in tensor format as:

$$(I_s \otimes M + \Delta t A_{irk} \otimes K)\, \tilde{x} = \tilde{b}. \tag{8.23}$$

If A_{irk} is triangular, as for (S)DIRK methods, (8.22) can be solved efficiently with block back-substitution using solvers optimized for the corresponding BE problem (8.20). This is because the diagonal blocks of (8.22) are formally equivalent to BE discretizations with a scaled time step $\Delta\tilde{t} = a_{ii}\Delta t$. In FIRK discretizations, however, it is not possible any more to solve for the stage vectors sequentially.

Reordering the unknowns in (8.22) yields a system of the following form:

$$\begin{bmatrix} m_{11}I_s + \Delta t k_{11} A_{irk} & \cdots & m_{1n}I_s + \Delta t k_{1n} A_{irk} \\ \vdots & \ddots & \vdots \\ m_{n1}I_s + \Delta t k_{n1} A_{irk} & \cdots & m_{nn}I_s + \Delta t k_{nn} A_{irk} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = b, \tag{8.24}$$

which reads in tensor format:

$$(M \otimes I_s + \Delta t K \otimes A_{irk})\, x = b. \tag{8.25}$$

Here, $x_i \in \mathbb{R}^s$ denotes the vector of the stage values related to the same physical unknown u_i. A third formulation stems from the application of (8.12) to (8.18). This yields:

$$MX + \Delta t K X A_{irk}^T = B. \tag{8.26}$$

Note that (8.23), (8.25) and (8.26) are mathematically equivalent. Once these systems have been solved, (8.9) or (8.11) can be used to propagate the solution values from t_0 to t_1.

8.3 A block algebraic multigrid algorithm

In this section, an algebraic multigrid algorithm is developed for problems of the form (8.23)-(8.25)-(8.26), denoted in short as $Lx = b$. A standard multigrid cycle is considered, which is the recursive application of the following two-level scheme:

1. smoothing on $Lx = b$,
2. restriction: $b_c = R_{irk}(b - Lx)$,
3. (approximate) solve of $L_c x_c = b_c$,
4. prolongation: $x \leftarrow x + P_{irk} x_c$,
5. smoothing on $Lx = b$.

Here, the subscript c refers to the coarse level; the matrices P_{irk} and R_{irk} are the prolongation and restriction matrices respectively. In order to characterize the algorithm, a procedure to build P_{irk} and R_{irk} and the coarse system matrices L_c is needed. Also, the smoothing operation is to be specified. For the solution of the system at the coarsest level of the multigrid hierarchy, a direct solver or a preconditioned Krylov solver can be used.

8.3.1 Prolongation and restriction

We suggest to derive the prolongation matrix P_{irk} from the prolongation matrix P built for the stationary problem $Kx = b$, where K is the stiffness matrix from (8.18). The matrix P can be built using all AMG methods of chapter 4 for div-grad problems, and by the AMG methods based on the node-edge commutation property, i.e. the methods presented in section 5.2 and chapter 7 for curl-curl problems. More precisely, for formulation (8.25), the following construction is suggested for P_{irk}:

$$P_{irk} \quad = \quad P \otimes I_s. \tag{8.27}$$

Note that this amounts to stage decoupling for the prolongation operator. Following the standard AMG practice, we set the restriction operator equal to the prolongator's transpose, that is

$$R_{irk} \quad := \quad P_{irk}^T = P^T \otimes I_s. \tag{8.28}$$

Using the compact formulation of (8.26), the restriction of the residual and the prolongation of the coarse grid correction read:

$$B_c \quad = \quad P^T \left(B - MX - \Delta t K X A_{irk}^T \right), \tag{8.29}$$

$$X \quad \leftarrow \quad X + P X_c. \tag{8.30}$$

8.3.2 Construction of the coarse system

As for any AMG method, the coarse system matrices are built as the Galerkin product of the fine system matrices and the prolongation matrices. For the formulation used in (8.25), we have

$$\begin{aligned} L_c &= P_{irk}^T \left(M \otimes I_s + \Delta t K \otimes A_{irk}\right) P_{irk} \\ &= \left(P^T M P\right) \otimes I_s + \Delta t \left(P^T K P\right) \otimes A_{irk}. \end{aligned} \tag{8.31}$$

Using the compact format of (8.26), the coarse system reads:

$$\left(P^T M P\right) X_c + \Delta t \left(P^T K P\right) X_c A_{irk}^T = B_c. \tag{8.32}$$

Note that (8.32) corresponds to the classical IRK time-discretization of the coarse grid equivalent problem to (8.18), i.e.

$$K_c u + M_c \frac{du}{dt} = h_c, \qquad \text{with} \quad K_c = P^T K P \quad \text{and} \quad M_c = P^T M P.$$

8.3.3 Definition of the smoother

For the div-grad equation (8.1), the blocksmoother suggested in [64] for use with a geometric multigrid method, will be used. This smoother consists of a sequence of local solves, one corresponding to each node i, in which all stage values for that node are updated simultaneously:

$$\left(m_{ii} I_s + \Delta t \; k_{ii} A_{irk}\right) x_i = b_i - \sum_{j \neq i} \left(m_{ij} I_s + \Delta t \; k_{ij} A_{irk}\right) x_j. \tag{8.33}$$

This blocksmoother is a block Gauss-Seidel type matrix splitting iteration applied to (8.24). Using the matrix splittings $M = M^+ + M^-$ and $K = K^+ + K^-$, with M^+ and K^+ denoting the lower triangular parts of M and K, (8.33) can be formulated in tensor notation as:

$$\left(M^+ \otimes I_s + \Delta t K^+ \otimes A_{irk}\right) x^{[\nu+1]} = b - \left(M^- \otimes I_s + \Delta t K^- \otimes A_{irk}\right) x^{[\nu]}. \tag{8.34}$$

Here, superscript $[\nu]$ is used to denote the iteration index. The nonzero structure of the lefthand side matrix is indicated in Figure 8.1. Note that the same matrix splittings, applied to a backward Euler discretization of (8.18), correspond to a classical Gauss-Seidel iteration for the problem $(M + \Delta t K)x = b$,

$$\left(M^+ + \Delta t K^+\right) x^{[\nu+1]} = b - \left(M^- + \Delta t K^-\right) x^{[\nu]}. \tag{8.35}$$

For the curl-curl equation, a block version of the hybrid smoother (see section 5.1.2) will be used. Similarly as for the scalar case (5.18), an auxiliary matrix is needed. This matrix is constructed as the Galerkin product of the original system matrix with the decoupled tensor version $G_{irk} = G \otimes I_s$ of the gradient:

$$\begin{aligned} L_{aux} &= G_{irk}^T \left(M \otimes I_s + \Delta t K \otimes A_{irk}\right) G_{irk} \\ &= \left(G^T M G\right) \otimes I_s + \Delta t \left(G^T K G\right) \otimes A_{irk}. \end{aligned} \tag{8.36}$$

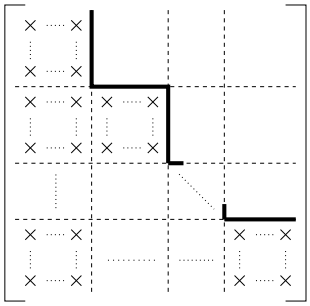


Figure 8.1: Nonzero structure of $M^+ \otimes I_s + \Delta t K^+ \otimes A_{irk}$.

Using the compact format of (8.26), the auxiliary system reads:

$$\left(G^T M G\right) X_{aux} + \Delta t \left(G^T K G\right) X_{aux} A_{irk}^T = B_{aux}. \tag{8.37}$$

Note that $G^T K G$ is zero away from the Dirichlet boundaries. One step of the block hybrid smoother is defined as follows:

1. smoothing on (8.26),
2. block restriction: $B_{aux} = G^T(B - MX - \Delta t K X A_{irk}^T)$,
3. smoothing on (8.37),
4. block prolongation: $X \leftarrow X + G X_{aux}$,
5. smoothing on (8.26).

Formally, this is equivalent to a 2-level multigrid cycle with prolongator G_{irk} and coarse system matrix (8.36). The smoother in steps (1), (3) and (5) is the block smoother of [64].

8.4 Convergence analysis

8.4.1 A formula for the asymptotic convergence factor

The convergence analysis for the AMG method presented in this paper is similar to the convergence analysis for the geometric multigrid method in [64]. First, we analyze the blocksmoother. To this end, we consider the matrix function

$$S(z) \quad = \quad -(zK^+ + M^+)^{-1}(zK^- + M^-). \tag{8.38}$$

Note that $S(\Delta t)$ is the iteration matrix for the Gauss-Seidel iteration defined by (8.35) for the BE discretization of (8.18) with a time step Δt. The iteration matrix for the blocksmoother (8.34) for the IRK discretization of the same

problem can be written in terms of the matrix function (8.38) as well. It is given by $S(\Delta t A_{irk})$, which equals

$$-\left(\Delta t A_{irk} \otimes K^+ + \ I_s \otimes M^+\right)^{-1} \left(\Delta t A_{irk} \otimes K^- + \ I_s \otimes M^-\right). \tag{8.39}$$

Using the eigenvalue decomposition $A_{irk} = V\Lambda V^{-1}$ and after applying the Butcher transformation, which is the similarity transformation corresponding to $I_N \otimes V$ (see [20]), (8.39) can be shown to be equivalent to

$$-\left(\Delta t\Lambda \otimes K^+ + I_s \otimes M^+\right)^{-1} \left(\Delta t\Lambda \otimes K^- + I_s \otimes M^-\right).$$

This shows that (8.39) is spectrally equivalent to the following block diagonal matrix:

$$S(\Delta t A_{irk}) \sim \begin{bmatrix} S(\Delta t\lambda_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & S(\Delta t\lambda_s) \end{bmatrix}, \quad \lambda_i \in \sigma(A_{irk}).$$

Here, $\sigma(A_{irk})$ denotes the spectrum of A_{irk}, which is complex for most fully coupled IRK and BVM methods. Note that the same result holds for an A_{irk} matrix that is not diagonalizable. In that case, the Jordan decomposition is to be used, and (8.39) is transformed into a block bidiagonal matrix by the similarity transformation $I_N \otimes V$.

So, the asymptotic convergence behavior of the blocksmoother (8.34) for the IRK discretization can be analyzed using the convergence results for the corresponding Gauss-Seidel smoother (8.35) for the BE discretization of the same problem. With $\rho(A)$ denoting the spectral radius of A, the asymptotic convergence factor for the blocksmoother is given by:

$$\rho\left(S(\Delta t A_{irk})\right) = \max_{\lambda \in \sigma(A_{irk})} \rho\left(S(\Delta t\lambda)\right). \tag{8.40}$$

A similar analysis can be done for a two level multigrid cycle using the matrix function

$$T(z) = S(z)^{\nu_2} \left(I_N - P\left(P^T\left(zK + M\right)P\right)^{-1} P^T(zK + M)\right) S(z)^{\nu_1},$$

where ν_1 and ν_2 are the number of pre- and postsmoothing iterations. Here, the invariance of the prolongation matrix P_{irk} for the similarity transformation $I_N \otimes V$ is used:

$$\left(I_N \otimes V^{-1}\right) \cdot \left(P \otimes I_s\right)\left(I_N \otimes V\right) = \left(I_N \cdot P \cdot I_N\right) \otimes \left(V^{-1} \cdot I_s \cdot V\right) = P \otimes I_s.$$

The results are analogous to (8.40), i.e.

$$\rho\left(T(\Delta t A_{irk})\right) \quad = \quad \max_{\lambda \in \sigma(A_{irk})} \rho\left(T(\Delta t\lambda)\right). \tag{8.41}$$

By recursion, the results for the two level cycle immediately carry over to a multilevel cycle. The analysis for the block hybrid smoother is formally identical to the analysis for the two level cycle, with P replaced by G.

To summarize, the asymptotic convergence of the presented IRK AMG algorithm is related to the asymptotic convergence behavior of the corresponding AMG algorithm applied to the BE discretization of the same problem, but with a scaled time step. That is, the asymptotic multigrid convergence factor for the fully coupled IRK system (8.24) corresponds to the worst case of the asymptotic multigrid convergence for the problems

$$(\lambda_i \Delta t K + M)\, x = b, \quad \text{with } \lambda_i \in \sigma(A_{irk}). \tag{8.42}$$

For model problems, the latter can be analyzed quantitatively by a so-called local Fourier mode analysis [63, 69, 40, 11] and chapter 6. Note that (8.40) and (8.41) continue to hold for BVM time discretizations if A_{irk} is replaced by $A_{bvm}^{-1} B_{bvm}/n$, as mentioned in section 8.2.

8.4.2 Discussion

The convergence speed of multigrid for the BE problem $(\Delta t K + M)x = b$ depends on the value of Δt. Hence, in order to be able to interpret (8.42), knowledge of the location of the eigenvalues λ_i is of great importance. The location in the complex plane of the eigenvalues of A_{irk} and $A_{bvm}^{-1} B_{bvm}/n$ for some popular IRK and BVM methods is indicated in Figure 8.2. The picture is very similar for the classical IRK families of type RadauIA, RadauIIA, Gauss and LobattoIIIC and for the BVM family GAM. For the IRK families LobattoIIIA and LobattoIIIB, A_{irk} is singular and one eigenvalue is 0. For the BVM family GBDF with more than 5 intermediate steps, some eigenvalues have a (small) negative real part. Typically, the modulus of the eigenvalues decreases with increasing number of stages for IRK or intermediate steps for BVM, and the decrease becomes smaller with increasing number of stages or steps.

For a model problem on a regular grid, the asymptotic convergence factor of a scalar multigrid algorithm applied to $(zK + M)x = b$ for $z \in \mathbb{C}$ can be calculated by a so-called local Fourier mode analysis. We refer to [40, 63, 69] for the div-grad problem and to [11] and chapter 6 for the curl-curl problem. Qualitatively, the results of the analysis for the model problem carry over to a more general problem setting. By using the Fourier analysis of [11] and [40], we have computed $\rho(T(z))$ for a specific div-grad and curl-curl problem. The results are depicted in Figure 8.3.

Taking into account the typical location of the eigenvalues of $\Delta t A_{irk}$, we are especially interested in the convergence factors for values of z with a positive real part and a sufficiently large modulus. This is because, except for the GBDF methods, a negative real part of z is only relevant for indefinite problems, i.e. problems with $\beta < 0$. Multigrid methods are known to perform poorly for this type of problems. Also, since z is proportional to $\frac{\Delta t}{h^2}$, values of z with a small modulus are typically related to very small timesteps, which are typically not used in practice for the considered time discretization methods.

The local Fourier mode analysis shows that for z having a positive real part and a sufficiently large modulus, the asymptotic convergence factor typically improves with decreasing modulus of z. Hence, the convergence analysis presented

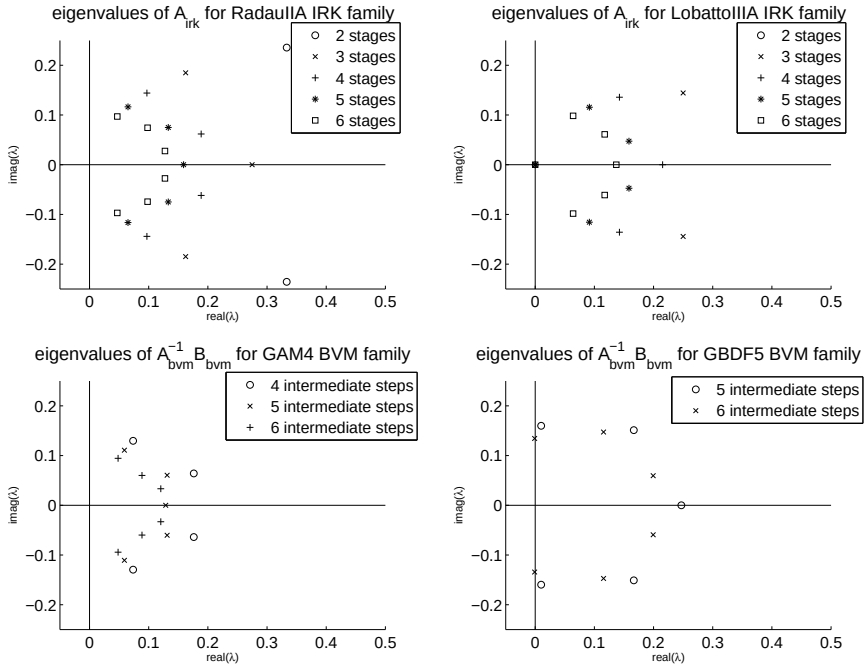


Figure 8.2: Eigenvalues of A_{irk} for the RadauIIA and LobattoIIIA IRK families and of $A_{bvm}^{-1}B_{bvm}/n$ for the GAM4 and GBDF5 BVM families. The 1-stage variant of the RadauIIA method corresponds to backward Euler; its eigenvalue is located at $(1, 0)$. For the BVM methods, the smallest possible number of intermediate steps n equals the number of steps k of the LMMs it is composed of, which is 4 and 5 for the GAM4 and GBDF5 methods respectively.

in this section allows to predict a faster multigrid convergence for IRK methods with more stages or BVM methods with more intermediate steps. Also, as expected, the use of a smaller time-step will typically lead to a faster multigrid convergence.

8.5 Implementation issues

The AMG algorithm presented in this chapter offers several opportunities for optimization, which will yield a significant increase of the floating point execution rate. This is important, because the dense linear system to be solved in the basic operation (8.33) of the blocksmoother causes the number of floating point operations to scale *cubically* as a function of the number of stages s. In this section, several aspects of the implementation, which are crucial for its memory and CPU efficiency, will be explained.

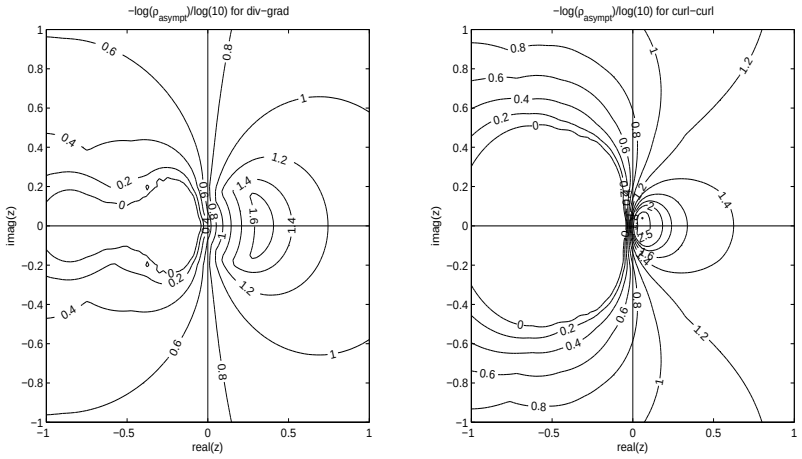


Figure 8.3: Asymptotic convergence factors of a two-level V(1,1) multigrid cycle for the finite element discretization $(zK + M)x = b$ of the 2D div-grad and curl-curl equations on an infinite quadrilateral grid with mesh size $h = 1$ as a function of z, calculated using local Fourier analysis. For the div-grad equation, Gauss-Seidel smoothing and classical bilinear prolongation (see [63]) are used. For the curl-curl equation, the full hybrid smoother (see [11]) and canonical edge prolongation (see [35]) are used. The restriction operator is chosen equal to the transpose of the prolongation operator, and the coarse system matrix is built as the Galerkin product of the fine level system matrix and the prolongation operator. The results presented in this picture can be reused for other grid sizes, by normalizing z as z/h^2 with respect to the grid size.

8.5.1 Storage of the multivectors

A row-by-row storage of the multivectors offers opportunities for cache optimization for the product $Y = AX$ of a sparse matrix $A \in \mathbb{R}^{N \times N}$ and a multivector $X \in \mathbb{R}^{N \times s}$. In this case, optimal reuse of the cache is achieved if the inner of the three loops for the calculation of $Y = AX$ runs over the columns of the multivectors instead of over their rows:

```
FOR all rows r of Y,
    FOR all nonzeros locations i of A(r,:),
        FOR all columns c of Y,
            Y(r,c) = Y(r,c) + A(r,i)*X(i,c)
```

In this way, the data access patterns for X and Y match their storage layout. Note that in a classical column-by-column calculation, this loop is the *outer* loop. This procedure can be applied directly for the matrix-vector product (8.26), the block restriction (8.29) and the block prolongation (8.30). A similar cache optimization can be done for the blocksmoother in the calculation of the righthand sides of (8.33).

Note that a similar cache optimization is not possible if the multivectors are stored column-by-column. This is because the loop over the rows r of the multivectors cannot be moved to become the inner loop, as the values of i depend on r for sparse matrices.

8.5.2 Storage of the tensor system matrices

The compact format of (8.26) shows that *only* Δt, A_{irk} and the appropriate stiffness and mass matrices K and M need to be stored to represent the tensor system matrices. Correspondingly, the memory requirements are essentially independent of the number of stages and amount to $nnz(K) + nnz(M)$ matrix entries, with $nnz(X)$ denoting the number of nonzero elements of X. Obviously, the system matrices are never expanded to standard matrix form, as this would result in memory requirements scaling quadratically as $nnz(K + M)s^2$ as a function of the number of stages. Note that the compact format allows to change the time discretization scheme, which is represented by A_{irk}, without the need to rebuild the system matrix.

In finite element applications, Dirichlet boundary conditions are often introduced by eliminating the columns of Dirichlet unknowns and by replacing their equations by $x_i = v_i$, with v_i the required solution value for the unknown x_i. With d and $\bar{d}$ denoting the set of the indices of the Dirichlet unknowns and the non-Dirichlet unknowns respectively, this is achieved as follows in the compact format:

$$\begin{aligned} &\begin{bmatrix} I & 0 \\ 0 & M_{\bar{d}\bar{d}} \end{bmatrix} \begin{bmatrix} X_d \\ X_{\bar{d}} \end{bmatrix} + \Delta t \begin{bmatrix} 0 & 0 \\ 0 & K_{\bar{d}\bar{d}} \end{bmatrix} \begin{bmatrix} X_d \\ X_{\bar{d}} \end{bmatrix} A_{irk}^T \\ = &\begin{bmatrix} V_d \\ B_{\bar{d}} - M_{\bar{d}d} V_d - \Delta t K_{\bar{d}d} V_d A_{irk}^T \end{bmatrix}. \end{aligned}$$

8.5.3 Blocksmoother

The basic operation (8.33) of the blocksmoother consists of three steps. First, the local righthand side $b = b_i - \sum_{j \neq i} (m_{ij} I_s + \Delta t k_{ij} A_{irk})$ and the local system matrix $A = m_{ii} I_s + \Delta t k_{ii} A_{irk}$ are constructed. Next, the local $s \times s$ system denoted as $Ax = b$ is solved. In this procedure, several optimizations are possible to reduce the computational and memory complexity and to increase the floating point execution rate.

All local matrices A can be inverted and stored in a setup phase. This allows $Ax = b$ to be implemented as a matrix-vector product $x = A^{-1}b$, which causes the number of floating point operations to scale only quadratically, instead of cubically, as a function of the number of stages. The price to be paid is an additional memory requirement, scaling quadratically as a function of the number of stages. This additional memory requirement can be reduced by a factor 2 by storing all A^{-1} in *single* precision.

In this case, the accuracy of the blocksmoother and, consequently, of the AMG cycle, is limited to single precision as well. This can be avoided by calculating the *increments* of the stage values instead of the stage values themselves:

$$A\Delta x_i = \left(b_i - \sum_j (m_{ij} I_s + \Delta t \; k_{ij} A_{irk}) \, x_j \right) \tag{8.43}$$

$$x_i \leftarrow x_i + \Delta x_i. \tag{8.44}$$

For this purpose, x and b are to be stored in double precision and the calculation of the righthand side of (8.43) and the update formula (8.44) are to be carried out using double precision. Now, the precision of x_i is limited by the lowest possible *exponent* of the single precision floating point format instead of by the length of the single precision *mantissa*, which allows to achieve double precision. Note that this precision reduction does not occur if Krylov acceleration is used, since in this case, the preconditioner is applied to the error equation with zero initial guess, which is equivalent to (8.43)-(8.44).

For the dense system solves and the dense matrix-vector products used to solve (8.33) or (8.43), it is crucial *not* to use standard software components, such as LAPACK or BLAS routines. This is because the local system sizes are typically rather small, too small for the cache optimization strategies of BLAS and LAPACK to pay off. Instead, it is more appropriate to use *ad-hoc* routines. This allows to avoid parameter checking, which is typical for general purpose routines, and it allows to use cheap, highly tuned algorithms. For instance, the linear solve needed in the blocksmoother can be based on LU-factorization *without* pivoting.

8.5.4 Loop unrolling

Many important parts of the AMG code contain small loops over the number of stages. Consider for instance the inner loop of the product of a sparse matrix and a multivector (see Section 8.5.1) and the routines for the linear solve and

matrix-vector product used in the blocksmoother (see Section 8.5.3). If the body of the loop is small, the loop overhead, consisting of incrementing and checking the loop parameter, is not negligible. Template programming allows the compiler to optimize such loops for each number of stages *separately* by loop unrolling, resulting in the elimination of this overhead.

8.6 Numerical experiments

The presented algorithms have been implemented in C++. The underlying scalar AMG algorithms are smoothed aggregation nodal multigrid [65] for the div-grad problem and the algorithm of [8] for the curl-curl equation. All numerical results presented in this section are generated on a Pentium IV 2.4GHz machine using the g++ 3.3.5 compiler. In all experiments, we consider the finite element discretization of the 2D div-grad and curl-curl equations with $\alpha = \beta = 1$ on a triangular mesh for the unit square $[0\ 1]^2$ containing 41624 nodes and 124075 edges with homogeneous Dirichlet boundary conditions. The righthand side is always a random vector, and the iteration starts from a zero initial guess.

We will show two types of results. First, we illustrate the convergence of the AMG solver for a variety of IRK and BVM discretizations. Second, we discuss the effect of various implementation issues. We show, among other things, that for up to 6 stages, the actual cost per iteration scales almost linearly with the number of stages and not quadratically or cubically, as could be expected from a floating point operation count.

Tables 8.1 and 8.2 show for some fully coupled IRK and BVM methods the number of iterations as a function of the number of stages, needed to solve a 2-dimensional div-grad and curl-curl problem to a high accuracy. The accuracy is chosen high to make the differences in the results more clear. For standalone AMG, the number of iterations typically decreases with increasing number of stages, and the decrease becomes smaller with increasing number of stages. Table 8.3 shows for the RadauIIA family of IRK methods that the AMG methods converge faster for a smaller time step. Both observations confirm the results of the convergence analysis of Section 8.4. Similar as for scalar AMG, Krylov acceleration results in a significant reduction of the number of iterations. Note that the convergence results are similar or slightly better than for scalar AMG for the BE problem, which is equivalent to the RadauIIA case with 1 stage.

Different possible implementations of the matrix-vector products or linear solves in (8.33) are compared in Tables 8.4 and 8.5. The use of ad-hoc templated routines instead of the corresponding BLAS or LAPACK routines will be responsible for the main part of the reduction of the time per AMG cycle. The corresponding time reduction is of the order of magnitude of 10. Additional time reductions, achieved by the optimizations explained in Section 8.5, are illustrated in Figure 8.4.

The time needed for a fully-optimized block AMG cycle scales only quasi-*linearly* as a function of the number of stages for up to 6 stages (see Figure 8.4).

	div-grad			curl-curl		
nbStages	RadauIIA	Gauss	LobattoIIIA	RadauIIA	Gauss	LobattoIIIA
1	50 (22)	41 (19)	/	65 (29)	57 (27)	/
2	41 (22)	38 (19)	42 (19)	55 (26)	52 (26)	56 (26)
3	37 (22)	36 (21)	38 (21)	52 (26)	50 (24)	52 (25)
4	35 (22)	34 (20)	35 (21)	49 (26)	51 (25)	51 (25)
5	35 (22)	34 (20)	36 (21)	51 (25)	49 (25)	52 (24)
6	35 (22)	34 (21)	35 (21)	49 (25)	50 (25)	51 (27)

Table 8.1: Number of iterations as a function of the number of stages for some fully coupled IRK methods, needed for a V(2,2) AMG cycle as a standalone solver and with BiCGstab acceleration (between brackets) to reach an accuracy $\|residual\|/\|rhs\| < 10^{-14}$ for a time step of 0.01.

	div-grad		curl-curl	
nbSteps	gam4	gbdf5	gam4	gbdf5
5	36 (21)	37 (29)	50 (26)	54 (27)
10	34 (23)	38 (27)	50 (26)	53 (28)
15	35 (26)	36 (26)	48 (27)	52 (31)
20	34 (24)	36 (26)	49 (28)	55 (31)
25	34 (24)	36 (27)	48 (27)	53 (30)
30	34 (26)	36 (26)	50 (29)	54 (30)

Table 8.2: Number of iterations as a function of the number of intermediate steps of some BVM methods, needed for a V(2,2) AMG cycle as a standalone solver and with BiCGstab acceleration (between brackets) to reach an accuracy $\|residual\|/\|rhs\| < 10^{-14}$ a time step of 0.01.

	div-grad			curl-curl		
nbStages \ Δt	10^{-1}	10^{-2}	10^{-3}	10^{-1}	10^{-2}	10^{-3}
1	37 (16)	22 (12)	11 (7)	34 (17)	24 (13)	18 (11)
2	33 (14)	18 (12)	10 (7)	29 (15)	23 (13)	17 (10)
3	33 (16)	17 (11)	10 (7)	29 (17)	22 (12)	16 (10)
4	32 (17)	17 (11)	10 (7)	28 (15)	22 (13)	16 (10)
5	32 (17)	17 (10)	9 (7)	29 (15)	23 (13)	16 (10)
6	32 (19)	17 (11)	9 (7)	29 (16)	21 (12)	16 (10)

Table 8.3: Number of iterations as a function of the number of stages and the time step for a V(2,2) AMG cycle as a standalone solver and with BiCGstab acceleration (between brackets) to reach an accuracy $\|residual\|/\|rhs\| < 10^{-8}$.

This is remarkable, as the corresponding number of floating point operations scales quadratically, due to the matrix-vector products $x = A^{-1}b$ used to solve (8.33). However, thanks to their dense character, these matrix-vector products cause very little cache misses, contrary to the construction of the righthand sides of (8.33), which is a sparse operation. Due to these cache effects, the cost of the blocksmoother is dominated by the construction of the local righthand sides (see the right picture of Figure 8.5). The cost of the local dense linear solve $Ax = b$ proves to be much higher than the cost of the corresponding matrix-vector product $x = A^{-1}b$, and has the same order of magnitude as the cost of the construction of the righthand sides. Correspondingly, if the inverses of the local system matrices are *not* constructed and stored in the setup phase, the cost of the blocksmoother scales more than linearly (see the left picture of Figure 8.5).

Figure 8.4 shows the cost of s scalar AMG cycles as a reference. This amounts essentially to the cost of a DIRK-based AMG solver. The figure shows that for the same number of stages, the solution by AMG of the linear system arising in a fully implicit Runge-Kutta discretization is not much more expensive (typically a factor of 2 or less) than solving the sequence of s linear systems appearing in a DIRK-based time-discretization. Obviously, for the same number of stages, the order of accuracy of DIRK time discretization is significantly lower. Hence, a DIRK method will require many more (smaller) time steps than a carefully chosen fully coupled IRK method to reach a similar accuracy. Note that a more precise and quantitative comparison of DIRK versus fully implicit RK-methods would require taking into account the detailed aspects of the time step control strategy, the cost of error estimation and many more implementation heuristics. This is outside the scope of the present study.

8.7 Conclusion

We have shown by a theoretical analysis and by numerical experiments that it is possible to develop efficient algebraic multigrid methods for fully coupled implicit Runge-Kutta and Boundary Value Method time discretizations of the div-grad and curl-curl equations. Both the analysis and the experiments showed that the convergence behavior is similar to the convergence behavior of the underlying scalar multigrid method for the backward Euler time discretization of the same problem. Also, we demonstrated that these multigrid methods offer opportunities to achieve very high cache efficiency and floating point execution rates. Considering the high order of accuracy of the time discretization methods used and the favorable convergence and timings results, the presented algorithms show great promise.

size	templated	non-templated		BLAS	
2	7.00e-09	2.90e-08	(+ 314 %)	3.66e-07	(+ 5129 %)
3	1.80e-08	6.20e-08	(+ 244 %)	4.42e-07	(+ 2356 %)
4	3.00e-08	5.60e-08	(+ 87 %)	5.40e-07	(+ 1700%)
5	4.70e-08	1.04e-07	(+ 121 %)	5.49e-07	(+ 1068 %)
6	6.80e-08	1.62e-07	(+ 138 %)	5.80e-07	(+ 753 %)

Table 8.4: Average time in seconds of dense matrix-vector products using an ad-hoc templated implementation, an ad-hoc non-templated implementation and BLAS as a function of the matrix size. The compiler optimization flags -O2 and -funroll-loops are used.

size	templated	non-templated		LAPACK	
2	8.00e-08	1.00e-07	(+ 25 %)	7.60e-07	(+ 850 %)
3	1.80e-07	2.40e-07	(+ 33 %)	1.78e-06	(+ 889 %)
4	3.70e-07	4.20e-07	(+ 14 %)	2.06e-06	(+ 457 %)
5	5.10e-07	5.90e-07	(+ 16 %)	3.18e-06	(+ 524 %)
6	7.10e-07	8.70e-07	(+ 23 %)	4.52e-06	(+ 537 %)

Table 8.5: Average time in seconds of dense system solves using an ad-hoc templated implementation, an ad-hoc non-templated implementation and LAPACK as a function of the matrix size. The compiler optimization flags -O2 and -funroll-loops are used.

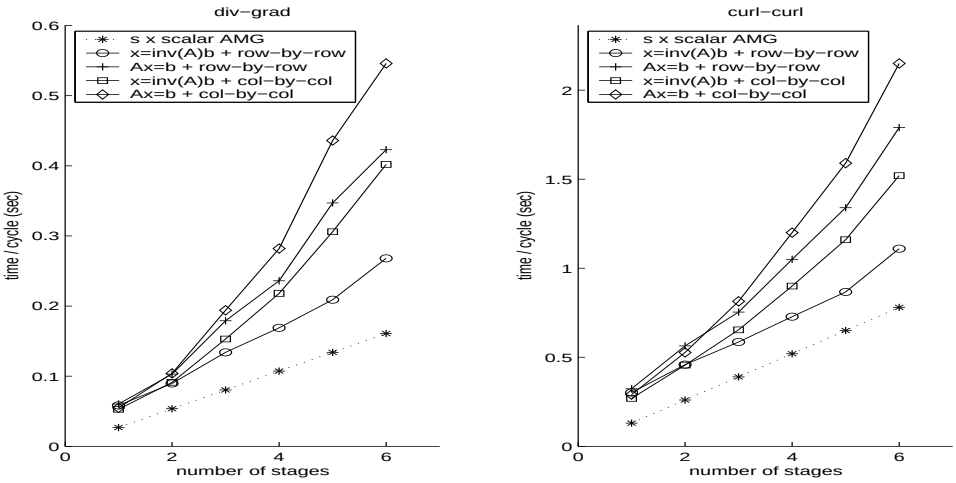


Figure 8.4: Average time of an AMG cycle for a 2D div-grad and curl-curl problem discretized on the same triangular mesh containing 41624 nodes and 124075 edges as a function of the number of stages, using different optimizations. "$Ax = b$" and "$x = inv(A)b$" indicate whether x in (8.33) is calculated by a linear solve or by a matrix-vector product (see Section 8.5.3). "row-by-row" and "col-by-col" refer to the storage layout of the multivectors (see Section 8.5.1). The time needed for s scalar AMG cycles is indicated for comparison.

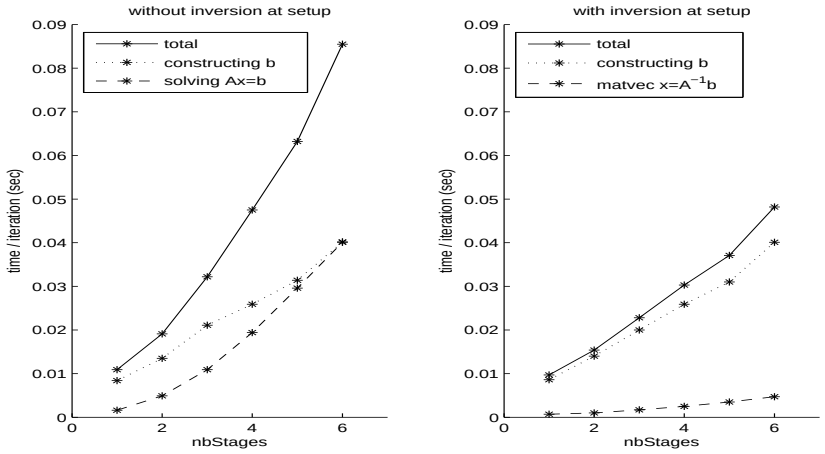


Figure 8.5: Average time in seconds needed for the basic operation (8.33) of the blocksmoother and for its dominating components with and without inversion and storage of the local system matrices A in the setup phase. The problem is the discretization of the 2D div-grad equation on a triangular mesh with 41624 nodes on the unit square $[0\ 1]^2$.

Chapter 9

Conclusions

9.1 Overview of the central results

9.1.1 LFA of multigrid for curl-curl problems

Fourier analysis is an important tool for the analysis of multigrid methods. However, this kind of analysis was not available for curl-curl problems. This is essentially due to the fact that the degrees of freedom of curl-curl problems are related to the mesh edges. Since standard Fourier techniques use a node-based approach, they cannot be applied directly here.

In this thesis, we extended Fourier techniques such that they can be applied to the two-dimensional curl-curl problem on a regular quadrilateral grid. Our approach is based on the decoupling in the Fourier domain of edges with different orientation. This allows two-dimensional invariant subspaces to be identified, which form the basis for the Fourier analysis. This technique was used to analyze the smoothing properties of multigrid smoothers and the asymptotic convergence behavior of the multigrid cycle. As a side-effect, several properties, which are known to be crucial for multigrid for curl-curl problems, were recovered in the Fourier domain.

This new Fourier technique is not restricted to the problem setting studied in this thesis. Its only formal requirement is that the discrete problem is translation-invariant and can be represented in stencil form. It can be applied to three-dimensional problems, to face-based problems and to other types of translation-invariant meshes as well.

9.1.2 AMG for curl-curl problems

In AMG methods for curl-curl problems, the nature of the degrees of freedom, which are related to the edges of the mesh, need to be taken into account appropriately by the coarsening and prolongation strategy. This is typically done by requiring the edge prolongator to satisfy the so-called node-edge commutation property.

In this thesis we developed a new algorithm to construct an edge prolongator which satisfies this commutation property. This edge prolongator is derived from an arbitrary partition of unity nodal prolongator by applying the analytical formula of edge elements on simplicial meshes to the coarse nodal elements represented by the nodal prolongator. Similar algorithms are used to construct face and volume prolongation operators.

These procedures allow a coarse topology to be identified. This topology can be described by a coarse gradient, curl and divergence operator. In general, this coarse topology contains crossing edges and faces and overlapping faces and volumes. We showed that this problematic character causes the memory requirements of AMG algorithms for node-, edge- and face-based problems to increase. We developed an algorithm to avoid this, by imposing additional constraints for the nonzero structure of the nodal prolongator.

We showed that the derived prolongation operators transfer the differential structure of the problem to be solved to the coarse levels of the multigrid hierarchy. This structure is embedded in an exact de Rham sequence, similarly as the structure of the original problem.

The edge prolongation scheme was used in an AMG method for curl-curl problems. Numerical experiments showed that this method is very efficient in terms of convergence speed and solve time.

9.1.3 AMG for high order time discretizations

Multigrid methods are typically developed for stationary problems or backward Euler (BE) time discretizations of time-dependent problems. They cannot be applied immediately to fully implicit Runge-Kutta time discretizations and Boundary Value Methods discretizations, which are often preferred because of their superior stability and accuracy properties.

In this thesis, we extended an existing approach for geometric multigrid for time-dependent div-grad problems, to algebraic multigrid for time-dependent div-grad and curl-curl problems. This approach uses a blocksmoother, in which all unknowns related to a mesh node or edge are updated simultaneously. The prolongators are based on the prolongators obtained by any scalar AMG algorithm for the backward Euler time discretization of the same problem. More precisely, all unknowns related to different points in time are decoupled for the prolongation operation, and are prolongated separately by the prolongators of the underlying scalar AMG method.

In a theoretical analysis and by numerical experiments, we showed that the convergence behavior of the resulting block AMG method is similar to the convergence behavior of the underlying scalar AMG method. Also, we demonstrated that these block multigrid methods offer opportunities to achieve very high cache efficiency and floating point execution rates.

9.2 Some suggestions for further research

9.2.1 Application to industrial problems

The AMG algorithms developed in this thesis were tested using geometrically simple problems with and without jumps in the material parameters. This gives a good idea about the potential of these methods. Anisotropic problems have not been considered yet, and testing using real-life industrial problems remains to be done as well.

9.2.2 Coarse topology control

We showed that controlling the coarse topology implied by a nodal prolongator is important for the computational and memory efficiency of AMG methods for curl-curl problems. In this thesis, we developed a strategy to control this implied coarse topology. This strategy proved to be very successful for two-dimensional curl-curl problems.

- It would be interesting to investigate whether this control strategy is beneficial in the context of AMG for div-grad problems as well.
- This control strategy is not successful for three-dimensional problems. This requires further research.

9.2.3 Commutation properties

The node-edge commutation property is a key property for multigrid methods for curl-curl problems. It would be interesting to investigate whether similar commutation properties could be useful for other problem types as well.

Commutation can be considered a generic way to transfer any matrix property, which can be represented as a Galerkin product with a matrix Q_f:

$$K_{f,q} = Q_f^T K_f Q_f \tag{9.1}$$

to the coarse level of a multigrid hierarchy. More precisely, if a coarse version Q_c of the matrix property Q_f can be determined which commutes with Q_f with respect to the prolongator P:

$$Q_f P = P Q_c,$$

then the structure (9.1) of the fine level system matrix is transferred to the coarse level:

$$\begin{aligned} P^T K_{f,q} P &= P^T \left(Q_f^T K_f Q_f \right) P \\ &= Q_c^T \left(P^T K_f P \right) Q_c. \end{aligned}$$

A Galerkin representation (9.1) can be used for instance to impose complicated boundary conditions, such as Robin, (anti-)periodic or floating potential boundary conditions [31], or coupling conditions [30].

9.2.4 AMG based on energy minimization

We developed a new energy minimization algorithm for nodal and edge prolongators. The theory for this algorithm has been fully developed, and is included as an appendix to this thesis. This algorithm has only been used in the numerical experiments presented in this thesis to generate the nodal prolongators from which the edge prolongators are derived. Further testing and comparison with other energy minimization algorithms is needed to assess its efficiency.

Appendix A

Energy minimization of multigrid prolongators

In this appendix, we will propose a new iterative procedure to solve the constrained minimization problems (4.34) and (5.31) to optimize a nodal and edge prolongator respectively. In both cases, this procedure has the same structure. In its basic step, the nonzero locations of a single row of the prolongator $P \in \mathbb{R}^{n\times m}$ are updated:

$$\begin{array}{l} \text{WHILE not converged,} \\ \quad \text{FOR r = 1 : nbRows,} \\ \quad\quad P(r,:) \longleftarrow P(r,:) + \Delta p, \quad \Delta p \in \mathbb{R}^{1\times m}. \end{array} \tag{A.1}$$

The update Δp for a row will be calculated by solving the constrained minimization problem for the nonzeros of this row only, while keeping all other nonzeros of P fixed.

A.1 Energy minimization of a nodal prolongator

Let the initial prolongator $P_n \in \mathbb{R}^{n\times m}$ satisfy the partition of unity property (4.22). We consider the minimization problem (4.34) restricted to the nonzero locations nz_r of the rth row of P_n

$$nz_r \quad := \quad \{c \mid P_n(r,c) \neq 0\}.$$

In other words, only the nonzero locations of the rth row of P_n are allowed to change. This restricted minimization problems reads, with ϵ_r denoting the rth unit vector:

$$\min_{\Delta p\in\mathbb{R}^{1\times m}} \sum_{c=1}^{m} \left(P_n(:,c) + \Delta p(c)\epsilon_r\right)^T K_n \left(P_n(:,c) + \Delta p(c)\epsilon_r\right) \tag{A.2}$$

subject to

$$\sum_{c=1}^{m} \Delta p(c) \quad = \quad 0 \tag{A.3}$$

and

$$\forall c \notin nz_r \quad : \quad \Delta p(c) = 0. \tag{A.4}$$

The constraint (A.3) guarantees that the updated prolongator $P_n + \Delta p \cdot \epsilon_r$ satisfies the partition of unity property as well. The constraint (A.4) guarantees that the nonzero structure of P_n is preserved. This localized constrained minimization problem can be reformulated using Lagrange multipliers as follows:

$$\min \sum_{c \in nz_r} \left(P_n(:,c) + \Delta p(c)\epsilon_r\right)^T K_n \left(P_n(:,c) + \Delta p(c)\epsilon_r\right) + \lambda \sum_{c \in nz_r} \Delta p(c). \tag{A.5}$$

For a symmetric K_n, (A.5) becomes:

$$\min \sum_{c \in nz_r} \left(\begin{array}{lll} & P_n(:,c)^T K_n P_n(:,c) \\ + & 2\Delta p(c) K_n(r,:) P_n(:,c) \\ + & \left(\Delta p(c)\right)^2 K_n(r,r) \end{array} \right) + \lambda \sum_{c \in nz_r} \Delta p(c). \tag{A.6}$$

The minimum of (A.6) is obtained by requiring the derivatives of (A.6) for $\Delta p(c),\ c \in nz_r$ and for λ to be 0:

$$\frac{\partial (A.6)}{\partial \Delta p(c)} = 0 \quad \longrightarrow \quad 2K_n(r,r)\Delta p(c) + \lambda = -2K_n(r,:)P_n(:,c) \tag{A.7}$$

$$\frac{\partial (A.6)}{\partial \lambda} = 0 \quad \longrightarrow \quad \sum_{c \in nz_r} \Delta p(c) = 0 \tag{A.8}$$

Let $I \in \mathbb{R}^{\#nz_r \times \#nz_r}$ and $\mathbf{1} \in \mathbb{R}^{\#nz_r}$ denote respectively the unit matrix and the vector with all entries equal to 1. Let $b \in \mathbb{R}^{\#nz_r}$ be defined as:

$$b^T \quad := \quad K_n(r,:)P_n(:,nz_r).$$

The system of equations (A.7)-(A.8) reads in matrix form:

$$\left[\begin{array}{cc} 2K_n(r,r)I & \mathbf{1} \\ \mathbf{1}^T & 0 \end{array} \right] \left[\begin{array}{c} \Delta p(nz_r)^T \\ \lambda \end{array} \right] = \left[\begin{array}{c} -2b \\ 0 \end{array} \right]. \tag{A.9}$$

The solution of (A.9) reads, with $\bar{b}$ denoting the average value of b:

$$\Delta p(nz_r)^T \quad = \quad \frac{-b + \mathbf{1}\left(\mathbf{1}^T\mathbf{1}\right)^{-1}\mathbf{1}^T b}{K_n(r,r)} = \frac{-b + \mathbf{1}\bar{b}}{K_n(r,r)}. \tag{A.10}$$

This is a scaled projection of b orthogonal to ran($\mathbf{1}$). It is very similar to the orthogonal projection Z (4.37) used in [48]. The major differences between both energy minimization methods are:

n	ρ	nbIter
0	0.43	39
1	0.22	21
2	0.20	20
3	0.20	20

Table A.1: Geometric mean convergence factors and number of iterations for a V(2,2)-cycle accelerated by CG using the nodal prolongator described in Section 7.8.1 for the finite element discretization using nodal elements of $\mathrm{div}(\alpha\,\mathrm{grad}(U)) + \beta U = f$ with $\alpha = \beta = 1$ and homogeneous Neumann boundary conditions on an irregular triangular mesh for the unit square with 166061 nodes, as a function of the number of energy minimization iterations n for the prolongators. The required accuracy is $\|b - Kx\|/\|b\| < 10^{-14}$.

- In [48], all nonzero entries of P are updated in one single update step, while (A.2) changes the nonzeros of P row by row.
- (A.2) is guaranteed to converge if K is positive definite, while for the convergence of the method of [48] a suitable damping factor is needed.

This allows both methods to be considered related in the same way as classical Gauss-Seidel and Jacobi iterations.

Table A.1 shows convergence results of node-based AMG for a div-grad problem as a function of the number of energy minimization iterations for the prolongators.

A.2 Energy minimization of an edge prolongator

Let the initial prolongator $P_e \in \mathbb{R}^{n\times m}$ satisfy the node-edge commutation property (5.9). We consider the minimization problem (5.31) restricted to the nonzero locations nz_r of the rth row of P_e

$$nz_r \quad := \quad \{c \mid P_e(r,c) \neq 0\}.$$

In other words, only the nonzero locations of the rth row of P_e are allowed to change. This restricted minimization problems reads, with ϵ_r denoting the rth unit vector:

$$\min_{\Delta p\in\mathbb{R}^{1\times m}} \sum_{c=1}^{m} \left(P_e(:,c) + \Delta p(c)\epsilon_r\right)^T K_e \left(P_e(:,c) + \Delta p(c)\epsilon_r\right)$$

subject to

$$\Delta p \cdot G_c \quad = \quad 0 \tag{A.11}$$

and

$$\forall c \notin nz_r \quad : \quad \Delta p(c) = 0. \tag{A.12}$$

The set of constraints represented by (A.11) guarantees that the updated prolongator $P_e + \Delta p \cdot \epsilon_r$ satisfies the node-edge commutation property as well. The constraint (A.12) guarantees that the nonzero structure of P_e is preserved. This localized constrained minimization problem can be reformulated using Lagrange multipliers as follows:

$$\begin{aligned} \min \quad & \sum_{c \in nz_r} \left(P_e(:,c) + \Delta p(c)\epsilon_r\right)^T K_e \left(P_e(:,c) + \Delta p(c)\epsilon_r\right) \\ & + \sum_{c \in nz_r} \Delta p(c) G_c(c,:)\Lambda. \end{aligned} \tag{A.13}$$

Here, Λ denotes a column vector of Lagrange multipliers. Each Lagrange multiplier corresponds to a coarse node. Let nz_c denote the nonzero columns of $G_c(nz_r,:)$. Only the Lagrange multiplier $\Lambda(nz_c)$ influence the solution of (A.13). This allows all other Lagrange multipliers to be eliminated from the problem. For a symmetric K_e, (A.13) becomes:

$$\begin{aligned} \min \quad & \sum_{c \in nz_r} \left(\begin{array}{lll} & P_e(:,c)^T K_e P_e(:,c) \\ + & 2\Delta p(c) K_e(r,:) P_e(:,c) \\ + & \left(\Delta p(c)\right)^2 K_e(r,r) \end{array} \right) \\ & + \sum_{c \in nz_r} \Delta p(c) G_c(c, nz_c)\Lambda(nz_c). \end{aligned} \tag{A.14}$$

The minimum of (A.14) is obtained by requiring the derivatives for $\Delta p(c),\ c \in nz_r$ and for each Lagrange multiplier to be 0:

$$\begin{aligned} \frac{\partial (A.14)}{\partial \Delta p(c)} = 0 \quad \longrightarrow \quad & 2K_e(r,r)\Delta p(c) + G_c(c,nz_c)\Lambda(nz_c) \\ & = -2K_e(r,:)P_e(:,c) \end{aligned} \tag{A.15}$$

$$\frac{\partial (A.14)}{\partial \Lambda(nz_c(i))} = 0 \quad \longrightarrow \quad G_c(nz_r, nz_c(i))^T \Delta p(nz_r)^T = 0 \tag{A.16}$$

Let $I \in \mathbb{R}^{\#nz_r \times \#nz_r}$ and $\mathbf{1} \in \mathbb{R}^{\#nz_r}$ denote respectively the unit matrix and the vector with all entries equal to 1. Let b, $\tilde{\Lambda}$ and $\tilde{G}_c$ be defined as:

$$\begin{aligned} b^T &:= K_n(r,:)P_n(:,nz_r) \\ \tilde{\Lambda} &:= \Lambda(nz_c) \\ \tilde{G}_c &:= G_c(nz_r, nz_c) \end{aligned}$$

The system of equations (A.7)-(A.8) reads in matrix form:

$$\left[\begin{array}{cc} 2K_e(r,r)I & \tilde{G}_c \\ \tilde{G}_c^T & 0 \end{array} \right] \left[\begin{array}{c} \Delta p(nz_r)^T \\ \tilde{\Lambda} \end{array} \right] = \left[\begin{array}{c} -2b \\ 0 \end{array} \right] \tag{A.17}$$

e \ n	0	1	2	3
0	0.65(76)	0.27(25)	0.24(23)	0.24(23)
1	0.45(40)	0.23(22)	0.20(21)	0.21(21)
2	0.43(39)	0.22(22)	0.21(21)	0.21(21)
3	0.43(39)	0.23(22)	0.21(21)	0.21(21)

Table A.2: Geometric mean convergence factor and number of iterations for a V(2,2)-cycle accelerated by CG using edge prolongators derived from the nodal prolongators described in Section 7.8.1 for the finite element discretization using edge elements of $\mathrm{curl}(\alpha\,\mathrm{curl}(\vec{U})) + \beta\vec{U} = \vec{f}$ with $\alpha = \beta = 1$ and homogeneous Neumann boundary conditions on an irregular triangular mesh for the unit square with 166061 nodes and 496586 edges, as a function of the number of energy minimization iterations for the nodal and edge prolongators, indicated by n and e respectively. The required accuracy is $\|b - Kx\|/\|b\| < 10^{-14}$.

Note that the system matrix of (A.17) is singular, since $\tilde{G}_c\mathbf{1} \equiv 0$. This implies that one of the non-trivial constraints of (A.11) is redundant:

$$\left.\begin{array}{l} \tilde{G}_c\mathbf{1} \equiv 0 \\ \forall j \neq i \;:\; \Delta p(nz_r)\tilde{G}_c(:,j) = 0 \end{array}\right\} \implies \Delta p(nz_r)\tilde{G}_c(:,i) = 0$$

Consequently, one additional Lagrange multiplier and the corresponding column of $\tilde{G}_c$ can be eliminated from the problem. For instance:

$$\begin{array}{rcl} \tilde{\Lambda} & \leftarrow & \tilde{\Lambda}(2{:}\#nz_c) \\ \tilde{G}_c & \leftarrow & \tilde{G}_c(:\,,2{:}\#nz_c). \end{array}$$

The solution of (A.17) reads:

$$\tilde{\Lambda} \;=\; \frac{-b + \tilde{G}_c\left(\tilde{G}_c^T\tilde{G}_c\right)^{-1}\tilde{G}_c^T b}{K_e(r,r)}.$$

This is a scaled projection of b orthogonal to $\mathrm{ran}(\tilde{G}_c)$.

Table A.2 shows that the energy minimization of the nodal prolongators from which the edge prolongators are derived, yields a larger convergence speedup than energy minimization of the edge prolongators themselves.

A.3 The importance of prolongator normalization

Consider the energy minimization problem (4.34) for a nodal prolongator P_n with respect to a positive (semi-)definite matrix K_n. As such, the partition of unity constraint does not prevent individual columns of P_n to become zero. In practice, however, this is prevented by the normalization requirement (4.14) of

the coarse nodal elements. More precisely, normalization implies that for each coarse node n_I, there is (at least) one fine node n_i that is prolongated from n_I only:

$$\forall n_I,\ \exists n_i \quad : \quad P_n(n_i, n_J) \neq 0 \quad \Longrightarrow \quad n_J = n_I.$$

In other words, $P_n(n_i, :)$ contains exactly one nonzero entry. In order for the partition of unity property to be satisfied, this entry needs to be 1. The partition of unity constraint of the energy minimization prevents this value to be changed. Consequently, $P_n(:, n_I)$ cannot become identical to the zero vector.

For the energy minimization (4.34) of an edge prolongator P_e with respect to a positive (semi-)definite matrix K_e, a similar problem exists. More precisely, the node-edge commutation as such does not prevent individual columns of P_e to become zero. For this purpose, the following property needs to hold:

$$\forall e_{IJ},\ \exists e_{ij} \quad : \quad P_e(e_{ij}, e_{KL}) \neq 0 \quad \Longrightarrow \quad e_{KL} = e_{IJ},$$

which follows immediately from the normalization of the coarse edge elements.

This is numerically confirmed by the application of the edge minimization procedure of Section A.2 to the unnormalized edge prolongator (7.6). The experiments show that the L_2-norm of some columns of the edge prolongator are of the order of magnitude of the machine precision.

Additionally, the node-edge commutation property as such does not prevent individual columns of P_e to belong to $\mathrm{ran}(G)$. This is observed numerically in the edge minimization procedure of Section A.2. More precisely, some entries of some update vectors Δp become very large, while the energy of P_e decreases. This can only be explained by the introduction of gradient components, which have a much smaller energy than solenoidal vectors.

Bibliography

[1] M. Adams, M. Brezina, J. Hu, and R. Tuminaro. Parallel multigrid smoothing: polynomial versus Gauss-Seidel. *Journal of Computational Physics*, 188(2):593–610, 2003.

[2] D. Arnold. Differential complexes and numerical stability. In L. Tatsien, editor, *Proceedings of the International Congress of Mathematicians, Vol.I: Plenary Lectures and Ceremonies*, pages 137–157. Higher Education Press, Beijing, 2002.

[3] D. Arnold, R. Falk, and R. Winther. Multigrid in H(div) and H(curl). *Numerische Mathematik*, 85:197–217, 2000.

[4] R. Beck. Algebraic multigrid by component splitting for edge elements on simplicial triangulations. *Tech. Report SC 99-40, Konrad-Zuse-Zentrum fur Informationstechnik, Berlin, Germany*, December 1999.

[5] P. Bochev, C. Garasi, J. Hu, A. Robinson, and R. Tuminaro. An improved algebraic multigrid method for solving Maxwell's equations. *SIAM Journal on Scientific Computing*, 25(2):623–642, 2003.

[6] P. Bochev, J. Hu, C. Siefert, and R. Tuminaro. An algebraic multigrid approach based on a compatible gauge reformulation of Maxwell's equations. *submitted to SIAM Journal on Scientific Computing*, 2007.

[7] P. Bochev and J. Hyman. Principles of mimetic discretizations. In D. Arnold, P. Bochev, R. Lehoucq, R. Nicolaides, and M. Shashkov, editors, *IMA*, volume 142. Springer Verlag, 2006.

[8] T. Boonen, G. Deliége, and S. Vandewalle. On algebraic multigrid methods derived from partition of unity nodal prolongators. *Numerical Linear Algebra with Applications*, 13(2-3):105–131, 2006.

[9] T. Boonen, J. Van lent, H. De Gersem, J. Driesen, and S. Vandewalle. Algebraic multigrid for implicit Runge-Kutta discretizations of the eddy current problem. *IEEE Transactions on Magnetics*, 43:1265–1268, 2007.

[10] T. Boonen, J. Van lent, and S. Vandewalle. An algebraic multigrid method for high order time-discretization of the div-grad and the curl-curl equations. *Applied Numerical Mathematics*, accepted.

[11] T. Boonen, J. Van lent, and S. Vandewalle. Local Fourier analysis of multigrid for the curl-curl equation. *SIAM Journal on Scientific Computing*, accepted.

[12] A. Bossavit. Mixed finite elements and the complex of Whitney forms. In J. R. Whiteman, editor, *The Mathematics of Finite Elements and Applications VI*, pages 137–144. Academic Press, London, 1988.

[13] A. Bossavit. *Computational Electromagnetism*. Academic Press, Boston, 1998.

[14] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Math. Comp.*, 31:333–390, 1977.

[15] M. Brezina, A. Cleary, R. Falgout, V. Henson, J. Jones, T. Manteuffel, S. McCormick, and J. Ruge. Algebraic multigrid based on element interpolation (AMGe). *SIAM Journal on Scientific Computing*, 22(5):1570–1592, 2001.

[16] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, , and J. Ruge. Adaptive smoothed aggregation (αSA) multigrid. *SIAM Review*, 47:317–346, 2005.

[17] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge. Adaptive algebraic multigrid. *SIAM Journal on Scientific Computing*, 27:1261–1286, 2006.

[18] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A multigrid tutorial*. SIAM, Philadelphia, 2000.

[19] L. Brugnano and D. Trigiante. *Solving Differential Problems by Multistep Initial and Boundary Value Methods*, volume 6 of *Stability and Control: Theory, Methods and Application*. Gordon & Breach, Amsterdam, 1998.

[20] J.C. Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons Ltd., Chichester, 2003.

[21] T. Chartier, R. D. Falgout, V. E. Henson, J. Jones, T. Manteuffel, S. McCormick, J. Ruge, and P. S. Vassilevski. Spectral AMGe. *SIAM Journal on Scientific Computing*, 25:1–26, 1.

[22] E. Chung and J. Zou. The eigenvalues and eigenspaces of some discrete div- and curl-related operators. *SIAM Journal on Matrix Analysis and Applications*, 24(4):1149–1160, 2002.

[23] M. Clemens, S. Feigh, and T. Weiland. Geometric multigrid algorithms using the conformal finite integration technique. *IEEE Transactions on Magnetics*, 40(2):1065–1068, 2004.

[24] M. Clemens and T. Weiland. Discrete electromagnetism with the finite integration technique. *Progress in Electromagnetics Research (PIER)*, 32:65–87, 2001.

[25] H. K. Dirks. Quasi-stationary fields for microelectronic applications. *Electrical Engineering*, 79(2):145–155, 1996.

[26] P. Dular, F. Henrotte, and W. Legros. A general and natural method to define circuit relations associated with magnetic vector potential formulations. *III Transactions on Magnetics*, 35(3):1630–1633, 1999.

[27] P. Dular, J.-Y. Hody, A. Nicolet, A. Genon, and W. Legros. Mixed finite elements associated with a collection of tetrahedra, hexahedra and prisms. *IEEE Transactions on Magnetics*, 30(5(2)):2980–2983, 1994.

[28] M.W. Gee, C.M. Siefert, J.J. Hu, R.S. Tuminaro, and M.G. Sala. ML 5.0 smoothed aggregation user's guide. Technical Report SAND2006-2649, Sandia National Laboratories, 2006.

[29] H. De Gersem, K. Hameyer, and T. Weiland. Field-circuit coupled models in electromagnetic simulation. *Journal of Computational and Applied Mathematics*, 168:125–133, 2004.

[30] H. De Gersem and T. Weiland. Harmonic weighting functions at the sliding interface of a finite-element machine model incorporating angular displacement. *IEEE Transactions on Magnetics*, 40(2):545–548, 2004.

[31] H. De Gersem, M. Wilke, M. Clemens, and T. Weiland. Efficient modelling techniques for complicated boundary conditions applied to structured grids. *International Journal for Computation and Mathematics in Electrical and Electronic Engineering (COMPEL)*, 23(4):904–912, 2004.

[32] V. Gradinary and R. Hiptmair. Whitney elements on pyramids. *ETNA*, 8:154–168, 1999.

[33] E. Hairer, S. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems.*, volume 8 of *Springer Series in Comput. Mathematics*. Springer-Verlag, 1993.

[34] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.*, volume 14 of *Springer Series in Comput. Mathematics*. Springer-Verlag, 1996.

[35] R. Hiptmair. Multigrid method for Maxwell's equations. *SIAM Journal on Numerical Analysis*, 36(1):204–255, 1998.

[36] R. Hiptmair. Canonical construction of finite elements. *Mathematics of Computation*, 68(228):1325–1346, 1999.

[37] R. Hiptmair. Finite elements in computational electromagnetism. *Acta Numerica*, 11:237–339, 2002.

[38] R. Hiptmair and J. Xu. Nodal auxiliary space preconditioning in H(curl) and H(div). *submitted to SIAM Numerical Analysis*, 2006.

[39] J. Hu, R. Tuminaro, P. Bochev, C. Garasi, and A. Robinson. Towards an h-independent algebraic multigrid method for Maxwell's equations. *SIAM Journal on Scientific Computing*, 27(5):1669–1688, 2006.

[40] J. Janssen and S. Vandewalle. Multigrid waveform relaxation on spatial finite element meshes: the discrete-time case. *SIAM Journal on Scientific Computing*, 17(1):133–155, 1996.

[41] J. Jones and B. Lee. A multigrid method for variable coefficient Maxwell's equations. *SIAM Journal on Scientific Computing*, 27(5):1689–1708, 2006.

[42] J. Jones and P. Vassilevski. Amge based on element agglomeration. *SIAM Journal on Scientific Computing*, 23:109–133, 2001.

[43] A. Kværnø, S. P. Nørsett, and B. Owren. Runge-Kutta research in Trondheim. *Applied Numerical Mathematics*, 22(1-3):263–277, 1996.

[44] D. Lahaye, H. De Gersem, S. Vandewalle, and K. Hameyer. Algebraic multigrid for complex symmetric systems. *IEEE Transactions on Magnetics*, 36(4):1535–1538, 2000.

[45] D. Lahaye, S. Vandewalle, and K. Hameyer. An algebraic multilevel preconditioner for field-circuit coupled problems. *IEEE Transactions on Magnetics*, 38(2):413–416, 2002.

[46] B. Lee and C. Tong. A novel algebraic multigrid-based approach for Maxwell's equations. *Preprint UCRL-JC-218750*, 2006.

[47] J. Van lent. *Multigrid methods for time-dependent partial differential equations*. PhD thesis, Department of Computer Science, K.U.Leuven, Leuven, Belgium, February 2006.

[48] J. Mandel, P. Vanek, and M. Brezina. Energy optimization of algebraic multigrid bases. *Computing*, 62(3):205–228, 1999.

[49] HYPRE User's manual. Casc, Lawrence Livermore National Laboratory. *http://www.llnl.gov/CASC/linear_solvers*, 2006.

[50] K.-A. Mardal, T.K. Nilssen, and G.A. Staff. Order optimal preconditioners for implicit Runge-Kutta schemes applied to parabolic PDEs. *SIAM Journal on Scientific Computing*, 29(1):361–375, 2007.

[51] MATLAB. Mathworks. *http://www.mathworks.com*.

[52] P. Monk. *Finite element methods for Maxwell's equations*. Clarendon Press, Oxford, 2003.

[53] F. Musy, L. Nicolas, and R. Perrussel. Compatible coarse nodal and edge elements through energy functionals. *SIAM Journal on Scientific Computing*, 29(3):1315–1337, 2007.

[54] J. C. Nédélec. Mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*, 35:315–341, 1980.

[55] J. C. Nédélec. A new family of mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*, 50:57–81, 1986.

[56] S. Reitzinger and J. Schoeberl. An algebraic multigrid method for finite element discretizations with edge elements. *Numerical Linear Algebra and Applications*, 9:223–238, 2002.

[57] E. Rosseel, T. Boonen, and S. Vandewalle. Algebraic multigrid for stationary and time-dependent partial differential equations with stochastic coefficients. *Numer. Linear Algebra Appl.*, accepted.

[58] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, Philadelphia, 2003.

[59] K. Stüben. Algebraic multigrid (amg): experiences and comparisons. *Applied mathematics and computation*, 13:419–452, 1983.

[60] K. Stüben. A review of algebraic multigrid. *Journal of Computational and Applied Mathematics*, 128:281–309, 2001.

[61] K. Stüben and T. Klees. *SAMG User's manual*, 2005.

[62] Trilinos. Sandia national laboratories. *http://trilinos.sandia.gov/*.

[63] U. Trottenberg, C. Oosterlee, and A. Schuller. *Multigrid*. Academic Press, London, 2001.

[64] J. Van lent and S. Vandewalle. Multigrid methods for implicit Runge-Kutta and boundary value method discretizations of parabolic PDEs. *SIAM Journal on Scientific Computing*, 27(1):67–92, 2005.

[65] P. Vanek, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56(3):179–196, 1996.

[66] P. Vassilevski. Sparse matrix topology with application to AMG(e) and preconditioning. *Numerical Linear Algebra with Applications*, 9:429–444, 2002.

[67] W. L. Wan, T. F. Chan, and B. Smith. An energy-minimizing interpolation for robust multigrid methods. *SIAM J. Sci. Computing*, 21(4):1632–1649, 2000.

[68] P. Wesseling. *Introduction to multigrid methods.* John Wiley & Sons Ltd., Chichester, 1992.

[69] R. Wienands and W. Joppich. *Practical Fourier analysis for multigrid methods.* Chapman and Hall/CRC Press, 2005.

Curriculum Vitae

Higher education

1. **1994-1998**: Burgerlijk elektrotechnisch-werktuigkundig ingenieur, optie energie, (Electromechanical Engineer, option energy), Katholieke Universiteit Leuven, Leuven, Belgium
   Thesis: Wavelet-based analysis of polluted electric currents.

2. **1998-2000**: Gediplomeerde in aanvullende studies filosofie (Complementary Studies Master of Philosophy), Katholieke Universiteit Leuven, Leuven, Belgium.

3. **2001-2003**: Burgerlijk ingenieur in de Computerwetenschappen (Engineer in Computer Science), Katholieke Universiteit Leuven, Leuven, Belgium
   Thesis: Fast solvers for the simulation of three-dimensional electromagnetic systems.

4. **2003-2007**: Ph.D. in Engineering, Katholieke Universiteit Leuven, Leuven, Belgium
   Thesis: Multigrid algorithms for electromagnetic field computations.

Teaching

- **2003-2007**: Teaching assistant for *Development of technical and scientific software*.

- **2005-2007**: Teaching assistant for *Computer-aided problem solving in physics*.

- **2004-2005**: Co-supervision of master thesis *Numerical modelling and simulation of grain growth*, Liesbeth Vanherpe.

Publications in reviewed journals

Publications of the results presented in this thesis:

- T. Boonen, J. Van lent, and S. Vandewalle, Local Fourier analysis of multigrid for the curl-curl equation, SIAM J. Sci. Comput., pp. 1-25, September, 2007, accepted.

- T. Boonen, J. Van lent, and S. Vandewalle, An algebraic multigrid method for high order time-discretization of the div-grad and the curl-curl equations, Applied Numerical Mathematics, pp. 1-17, June, 2007, accepted.

- T. Boonen, J. Van lent, H. De Gersem, J. Driesen, and S. Vandewalle, Algebraic multigrid for implicit Runge-Kutta discretizations of the eddy current problem, IEEE Trans. Magn. 43 (4), pp. 1265-1268, April, 2007.

- T. Boonen, G. Deliége, and S. Vandewalle, On algebraic multigrid methods derived from partition of unity nodal prolongators, Numer. Linear Algebra Appl. 13 (2-3), pp. 105-131, March, 2006.

Other publications:

- E. Rosseel, T. Boonen, and S. Vandewalle, Algebraic multigrid for stationary and time-dependent partial differential equations with stochastic coefficients, Numer. Linear Algebra Appl., pp. 1-21, 2007, accepted.

- B. Doliwa, T. Boonen, H. De Gersem, and T. Weiland, An optimized electromagnetic 3D field solver for frequencies below the first resonance, IET Science, Measurement and Technology 1 (1), pp. 53-56, January, 2007.

- H. Vande Sande, T. Boonen, I. Podoleanu, F. Henrotte, and K. Hameyer, Simulation of a three-phase transformer using an improved anisotropy model, IEEE Trans. Magn. 40 (2), pp. 850-855, March, 2004.

Publications in conference proceedings

- T. Boonen, G. Deliége, and S. Vandewalle, On the commutation property for arbitrary partition of unity nodal prolongators, Proceedings ACOMEN 2005 (CDROM) (E. Dicks, J. Vierendeels, L. Vandevelde, L. Dupré, M. Slodicka, R. Van Keer, eds.), pp. 1-11, 2005.

- H. Vande Sande, T. Boonen, H. De Gersem, F. Henrotte, and K. Hameyer, Accelerating non-linear time-harmonic problems by a hybrid Picard-Newton approach, Proceedings 10th International IGTE Symposium on Numerical Field Calculation in Electrical Engineering (IGTE'02) (Biro, O., ed.), pp. 1-6, 2003.

Conference presentations

Presentations of the results of this thesis. The presenting author is underlined.

- T. Boonen, J. Van lent, S. Vandewalle, Fourier analysis of multigrid for the curl-curl equation, 16th International Conference on the Computation of Electromagnetic Fields, COMPUMAG, Aachen, Germany, July 24-28, 2007.

- S. Vandewalle, T. Boonen, Algebraic multigrid for high-order time-discretizations of the div-grad and curl-curl equations, 13th Copper Mountain Conference on Multigrid Methods, Copper Mountain, Colorado, USA, March 17-23, 2007.

- S. Vandewalle, T. Boonen, An algebraic multigrid method for high order time-discretizations of the div-grad and curl-curl equations, 11th Seminar on Numerical Solution of Differential and Differential-Algebraic Equations, NUMDIFF11, Halle, Germany, September 4-8, 2006.

- T. Boonen, J. Van lent, H. De Gersem, J. Driesen, S. Vandewalle, Algebraic multigrid for implicit Runge-Kutta discretizations of the eddy current problem, Twelfth Biennial IEEE Conference on Electromagnetic Field Computation, CEFC, Miami, Florida, USA, April 30th - May 3rd, 2006.

- T. Boonen, J. Van lent, S. Vandewalle, Algebraic multigrid for implicit Runge-Kutta discretizations of parabolic PDE's, 8th European Multigrid Conference, EMG 2005, Scheveningen (The Hague), The Netherlands, September 27-30, 2005,

- T. Boonen, G. Deliége, S. Vandewalle, A new prolongator for algebraic multigrid for edge element discretizations, Third International Conference on Advanced Computational Methods in Engineering, ACOMEN 2005, Gent, Belgium, May 30 - Juni 2, 2005.

- T. Boonen, S. Vandewalle, A new prolongator for the curlcurl equations, Twelfth Copper Mountain Conference on Multigrid Methods, CM2005, Copper, Colorado, VS, April 3-8, 2005.

Seminars

- Fourier analysis of multigrid for the curl-curl equation, HYPRE-meeting, CASC, Lawrence Livermore National Laboratory, California, USA, July 25, 2006.

- Algebraic multigrid for implicit Runge-Kutta discretizations of the eddy current problem, Study day of the IAP "Dynamical systems and control: computation, identification and modelling", UCL, May 16, 2006.

- Algebraic multigrid for the curlcurl equation, Invited presentation, Fraunhofer Institute for Algorithms and Scientific Computing (FhI-SCAI), Sankt Augustin, February 10, 2005.
- Algebraic multigrid for fully implicit Runge-Kutta discretizations of the time-dependent divgrad and curlcurl equations, Elektromaschinenkolloquium, Institut für Elektrische Maschinen, RWTH Aachen, December 7, 2005.
- Polynomiale iteratie, KULeuven, Belgium, October 15, 2005.
- AMG for the curl-curl equation, KULeuven, Belgium, June 3, 2004.
- Interpolation strategies for AMG methods, KULeuven, Belgium, December 11, 2003.
- Interpolation strategies for AMG methods, TEMF, TUDarmstadt, Germany, November 19, 2003.