Sharpened CG iteration bound for Schwarz-preconditioned high-contrast heterogeneous elliptic PDEs

Going beyond condition number

WI5005: Thesis Project

Philip Soliman

Delft University of Technology

Sharpened CG iteration bound for Schwarz-preconditioned high-contrast heterogeneous elliptic PDEs

Going beyond condition number

by

Philip Soliman

To obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on T.B.A.

Student number:	4945255	
Project duration:	December 2024 – September 2025	
Thesis committee:	Prof. H. Schuttelaars,	TU Delft, responsible supervisor
	Dr. A. Heinlein,	TU Delft, daily supervisor
	F. Camaru,	TU Delft, daily co-supervisor

This thesis is confidential and cannot be made public until December 31, 2025.

Preface

A preface...

Philip Soliman
Delft, April 2025

Summary

A summary...

Contents

Preface **i**

Summary **ii**

1 Introduction **1**

2 Theory **3**

2.1 Physical system 4

2.1.1 Derivation of system of equations 4

2.1.2 Variational formulation 4

2.2 Projection methods 5

2.2.1 Matrix representation 5

2.2.2 Interpretation in terms of projectors 5

2.2.3 Projection-based iterative methods (1D) 6

2.3 Schwarz domain decomposition 8

2.3.1 Schwarz methods as preconditioners 9

2.3.2 Convergence original Schwarz method 10

2.3.3 Need for a coarse space 11

2.3.4 Two-level additive Schwarz method 11

2.3.5 Convergence of two-level additive Schwarz 14

2.3.6 Spectral coarse spaces for scalar heterogeneous problems 15

2.4 Conjugate Gradient Method 17

2.4.1 Variants of the CG method 17

2.4.2 Krylov subspaces 17

2.4.3 Arnoldi's method 18

2.4.4 Arnoldi's method for linear systems 18

2.4.5 Lanczos' Algorithm 19

2.4.6 D-Lanczos 20

2.4.7 Derivation of CG 22

2.4.8 Convergence of CG 23

2.4.9 Preconditioning 29

2.5 Generalized Minimal Residual Method 32

2.5.1 GMRES implementation details 33

2.5.2 Relation between GMRES and CG 34

2.6 Nonlinear systems 35

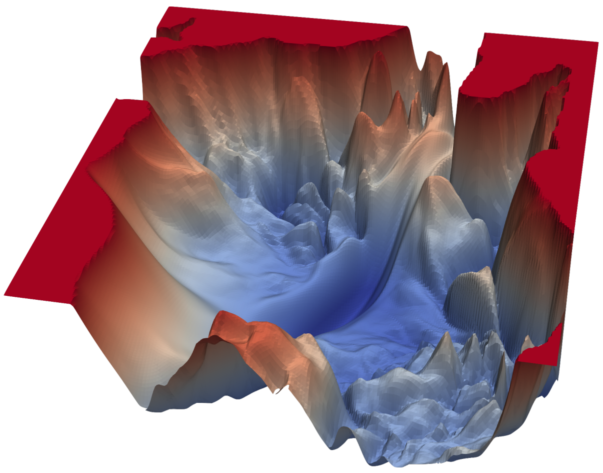
2.6.1 Inexact Newton-Raphson method 36

1 Introduction

Welcome to the introduction chapter.

2

Theory



2.1. Physical system

2.1.1. Derivation of system of equations

2.1.2. Variational formulation

2.2. Projection methods

This section is largely based on the extremely instructive book by Saad about iterative methods [8, Section 5: Projection Methods].

Most iterative methods that aim to solve (large) linear systems like

$$Ax = b$$

for $x, b \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times n}$ a linear operator, can be interpreted as projection methods.

A general description of projection methods is as follows let $x_0 \in \mathbb{R}^n$ be an initial guess, $\mathcal{K}, \mathcal{L}$ subspaces of $\mathbb{R}^n$, $r_0 = Ax_0 - b$ the residual.

$$\tilde{x} = x_0 + \delta, \quad \delta \in \mathcal{K} \tag{2.1a}$$

$$(r_0 - A\delta, w) = 0, \quad \forall w \in \mathcal{L}. \tag{2.1b}$$

where δ is the correction to the initial guess x_0 and $(\cdot,\cdot)$ is the inner product [8, Equation 5.5-6]. The second equation is the orthogonality condition.

Furthermore, orthogonal projection methods are a subclass of projection methods where $\mathcal{K} = \mathcal{L}$. The most well-known orthogonal projection method is the conjugate gradient method, section 2.4. In the case where $\mathcal{K} \perp \mathcal{L}$, the method is oblique projection method. Examples of oblique projection

2.2.1. Matrix representation

Let V and W be the bases for $\mathcal{K}$ and $\mathcal{L}$ respectively. Then equations (2.1a) and (2.1b) can be written as

$$\tilde{x} = x_0 + Vy, \tag{2.2}$$

$$W^T AVy = W^T r_0. \tag{2.3}$$

In case $W^T AVy$ is invertible, the solution is [8, Equation 5.7]

$$x = x_0 + (W^T AV)^{-1} W^T r_0. \tag{2.4}$$

2.2.2. Interpretation in terms of projectors

Residual projection methods

Let $\mathcal{L} = A\mathcal{K}$ and define the approximate residual

$$\tilde{r} = b - A\tilde{x} = r_0 + A\delta. \tag{2.5}$$

then $\tilde{r}$ is the residual is r_0 minus its orthogonal projection onto $\mathcal{L}$, $P_{\mathcal{L}}$. This can be expressed as [8, Proposition 5.4]

$$\tilde{r} = r_0 - P_{\mathcal{L}} r_0 = (I - P_{\mathcal{L}}) r_0. \tag{2.6}$$

Error projection methods

Similarly, suppose $\mathcal{K} = \mathcal{L}$ and A is SPD. Then the correction vector is obtained by constraining the residual vector $r_0 - A\delta$ to be orthogonal to $\mathcal{K}$.

$$(r_0 - A\delta, w) = 0, \quad \forall w \in \mathcal{K}.$$

Now, define the initial error

$$e_0 = x_* - x_0 \tag{2.7}$$

and the error after approximation

$$\tilde{e} = x_* - \tilde{x}. \tag{2.8}$$

Then,

$$\begin{aligned} A\tilde{e} &= A(x_* - x_0 - \delta) \\ &= A(e_0 - \delta) \\ &= r_0 - A\delta \end{aligned}$$

Hence, the orthogonality condition can be written as

$$(A\tilde{e}, w) = (e_0 - \delta, w)_A = 0, \quad \forall w \in \mathcal{K}.$$

It follows that the correction vector δ is the error e_0 projected onto $\mathcal{K}$ with respect to the inner product $(\cdot, \cdot)_A$ [8, Proposition 5.5].

$$\tilde{e} = (I - P^A_{\mathcal{K}})e_0. \tag{2.9}$$

2.2.3. Projection-based iterative methods (1D)

In general these kinds of methods are invoked when $\mathcal{K} = \text{span}\{v\}$ and $\mathcal{L} = \text{span}\{w\}$ for some vectors $v, w \in \mathbb{R}^n$ and may be described by the algorithm

$$\begin{aligned} r &\leftarrow b - Ax, \\ \alpha &\leftarrow \frac{(r, w)}{(Av, w)}, \\ x &\leftarrow x + \alpha v. \end{aligned}$$

Steepest descent

For the case where A is SPD

Algorithm 1 Steepest descent [8, Algorithm 5.2]

```
$r \leftarrow b - Ax$
$p = Ar$
while $r \neq 0$ do
    $\alpha \leftarrow \frac{(r, r)}{(p, r)}$
    $x \leftarrow x + \alpha r$
    $r \leftarrow r - \alpha p$
    $p \leftarrow Ar$
end while
```

Each iteration is of the form $x \leftarrow x + \alpha d$, where d is the search direction or the negative of the gradient of

$$f(x) = ||x - x_*||^2_A.$$

The convergence rate is [8, Theorem 5.9]

Theorem 2.1. Let x_* be the solution to $Ax = b$. Then the steepest descent method converges to x_* with the rate

$$||x_* - x_{k+1}||^2_A \leq \frac{\lambda_{\text{max}} - \lambda_{\text{min}}}{\lambda_{\text{max}} + \lambda_{\text{min}}} ||x_* - x_k||^2_A \tag{2.10}$$

Minimal residual method

In this case we only assume A is positive definite. The algorithm is

Algorithm 2 Minimal residual method [8, Algorithm 5.3]

```
$r \leftarrow b - Ax$
$p = Ar$
while $r \neq 0$ do
    $\alpha \leftarrow \frac{(r, p)}{(p, p)}$
    $x \leftarrow x + \alpha r$
    $r \leftarrow r - \alpha p$
    $p \leftarrow Ar$
end while
```

Here, each iteration minimizes the residual $f(x) = ||b - Ax||^2_A$. The convergence rate is [8, Theorem 5.10]

Theorem 2.2. Let A be positive definite, and let

$$\mu = \lambda_{\text{min}}(A + A^T)/2, \quad \sigma = ||A||_2.$$

Then the minimal residual method converges with the rate

$$||r_{k+1}||^2 \leq \left(1 - \frac{\mu^2}{\sigma^2}\right)^{1/2} ||r_k||^2. \tag{2.11}$$

Residual norm steepest descent

Assume only that A is square non-singular matrix. The algorithm is

Algorithm 3 Residual norm the steepest descent [8, Alogrithm 5.4]

```
$r \leftarrow b - Ax$
while $r \neq 0$ do
    $v \leftarrow= A^T r$
    $p \leftarrow Av$
    $\alpha \leftarrow \frac{||v||_2}{||p||_2}$
    $x \leftarrow x + \alpha v$
    $r \leftarrow r - \alpha p$
end while
```

here is each step minimizes the residual norm $f(x) = ||b - Ax||_2$ in the direction of the negative gradient of f. This is equivalent to the steepest descent method applied to the normal equations $A^T Ax = A^T b$.

2.3. Schwarz domain decomposition

The content of this section is largely based on chapters 1,2, 4 and 5 of Dolean et al. about Schwarz methods.

The original Schwarz method was a way of proving that a Poisson problem on some complex domain Ω still has a solution.

$$\begin{aligned} -\Delta u &= f \text{ in } \Omega, \\ u &= 0 \text{ on } \partial\Omega. \end{aligned} \tag{2.12}$$

Existence is proved by splitting up the original complex domain in two (or more) simpler, possibly overlapping domains and solving the Poisson problem on each of these domains. The solution on the original domain is then the sum of the solutions on the subdomains. The method is named after Hermann Schwarz, who first introduced the method in 1869. The method has since been extended to more general problems and is now a popular method for solving partial differential equations.

Definition 2.1. The Schwarz algorithm is an iterative method based on solving subproblems alternatively in domains Ω_1 and Ω_2. It updates $(u_1^n, u_2^n) \to \left(u_1^{n+1}, u_2^{n+1}\right)$ by

$$\begin{aligned} -\Delta\left(u_1^{n+1}\right) &= f && \text{in } \Omega_1, & & & -\Delta\left(u_2^{n+1}\right) &= f && \text{in } \Omega_2, \\ u_1^{n+1} &= 0 && \text{on } \partial\Omega_1 \cap \partial\Omega, & &\text{then} & u_2^{n+1} &= 0 && \text{on } \partial\Omega_2 \cap \partial\Omega, \\ u_1^{n+1} &= u_2^n && \text{on } \partial\Omega_1 \cap \overline{\Omega_2}; & & & u_2^{n+1} &= u_1^{n+1} && \text{on } \partial\Omega_2 \cap \overline{\Omega_1}. \end{aligned}$$

The original Schwarz algorithm is sequential and, thereby, does not allow for parallelization. However, the algorithm can be parallelized. The Jacobi Schwarz method is a generalization of the original Schwarz method, where the subproblems are solved simultaneously and subsequently combined into a global solution. In order to combine local solutions into one global solution, an extension operator E_i, $i = 1, 2$ is used. It is defined as

$$E_i(v) = v \text{ in } \Omega_i, \quad E_i(v) = 0 \text{ in } \Omega \backslash \Omega_i.$$

Instead of solving for local solutions directly, one can also solve for local corrections stemming from a global residual. This is the additive Schwarz method (ASM). It is defined in algorithm 4.

Algorithm 4 Additive Schwarz method [3, Algorithm 1.2]

```
Compute residual $r^n = f - \Delta u^n$.
For $i = 1, 2$ solve for a local correction $v_i^n$:
    $-\Delta v_i^n = r^n$ in $\Omega_i$,    $v_i^n = 0$ on $\partial\Omega_i$
Update the solution: $u^{n+1} = u^n + \sum_{i=1}^{2} E_i(v_i)^n$.
```

The restrictive additive Schwarz method (RAS) is similar to ASM, but differs in the way local corrections are combined to form a global one. In the overlapping region of the domains it employs a weighted average of the local corrections. In particular, a partition of unity χ_i is used. It is defined as

$$\chi_i(x) = \begin{cases} 1, & x \in \Omega_i \setminus \Omega_{3-i}, \\ 0, & x \in \delta\Omega_i \setminus \delta\Omega \\ \alpha, & 0 \leq \alpha \leq 1, x \in \Omega_i \cap \Omega_{3-i} \end{cases}$$

such that for any function $w : \Omega \to \mathbb{R}$, it holds that

$$w = \sum_{i=1}^{2} E_i(\chi_i w_{\Omega_i}).$$

The RAS algorithm is defined in algorithm 5.

Algorithm 5 Restrictive additive Schwarz method [3, Algorithm 1.1]

```
Compute residual $r^n = f - \Delta u^n$.
For $i = 1, 2$ solve for a local correction $v_i^n$:
    $-\Delta v_i^n = r^n \text{ in } \Omega_i, \quad v_i^n = 0 \text{ on } \partial\Omega_i$
Update the solution: $u^{n+1} = u^n + \sum_{i=1}^{2} E_i(\chi_i v_i^n)$.
```

2.3.1. Schwarz methods as preconditioners

Let $\mathcal{N}$ be set containing all indices of degrees of freedom in the domain Ω and N_{sub} be the number of subdomains such that

$$\mathcal{N} = \sum_{i=1}^{N_{\text{sub}}} \mathcal{N}_i,$$

$\mathcal{N}_i$ is the set of indices of degrees of freedom in the subdomain Ω_i.

Furthermore, let $R_i \in \mathcal{R}^{\#\mathcal{N}_i \times \#\mathcal{N}}$, R_i^T and D_i be the discrete versions of the restriction, extension and partition of unity operators such that

$$\mathcal{R}^{\#\mathcal{N}} \ni U = \sum_{i=1}^{\mathcal{N}_{\text{sub}}} R_i^T D_i R_i U.$$

Note that D_i is a diagonal matrix where the entries are the values of the partition of unity function χ_i evaluated for each degree of freedom. Consider for instance, a multidimensional FEM problem, in which $\mathcal{T}$ is the triangulation of the domain Ω and $\mathcal{T}_i$ is the triangulation of the subdomain Ω_i such that [3, Equation 1.27]

$$\Omega_i = \cup_{\tau \in \mathcal{T}_i} \tau.$$

In this case [3, Equation 1.28]

$$\mathcal{N}_i = \{k \in \mathcal{N} | \text{meas}(\text{supp}(\phi_k) \cap \Omega_i) > 0\},$$

and we can define

$$\mu_k = \#\{j | 1 \leq j \leq N_{\text{sub}} \text{ and } k \in \mathcal{N}_j\}.$$

Finally, this leads to

$$(D_i)_{kk} = \frac{1}{\mu_k},\ k \in \mathcal{N}_i. \tag{2.13}$$

Although the original Schwarz method is not a preconditioner, the ASM and RAS methods can be used as such. Originally the Schwarz method is a fixed point one [3, Definitions 1.12 and 1.13]

$$u^{n+1} = u^n + M^{-1} r^n,\ r^n = f - A u^n,$$

where M equals, but is not limited to, one of the following matrices;

$$M_{\text{ASM}} = \sum_{i=1}^{N_{\text{sub}}} R_i^T (R_i A R_i^T)^{-1} R_i, \tag{2.14a}$$

$$M_{\text{RAS}} = \sum_{i=1}^{N_{\text{sub}}} R_i^T D_i (R_i A R_i^T)^{-1} R_i. \tag{2.14b}$$

Both M_{ASM} and M_{RAS} are symmetric and positive definite and can be used as preconditioners.

Optimized Schwarz methods and corresponding preconditioners can also be constructed by including more interface conditions (Robin or Neumann) in the subproblems. One such example is the Optimized Restrictive Additive Schwarz method (ORAS) discussed in [3, Chapter 2].

2.3.2. Convergence original Schwarz method

In this section the Schwarz problem stated in definition 2.1 is solved in the one- and two-dimensional case. The convergence of the original Schwarz method is then discussed.

1D case

Let $L > 0$ and the domain $\Omega = (0, L)$. The domain is split into two subdomains $\Omega_1 = (0, L_1)$ and $\Omega_2 = (l_2, L)$ such that $l_2 \leq L_1$. Instead of solving for $u_{1,2}$ directly, we solve for the error $e^n_{1,2} = u^n_{1,2} - u_{|\Omega_i}$, which by linearity of the Poisson problem as well as the original Schwarz algorithm satisfies

$$\begin{aligned} -\frac{e_1^{n+1}}{dx^2} &= f \text{ in } (0, L_1), \\ e_1^{n+1}(0) &= 0, \\ e_1^{n+1}(L_1) &= e_2^n(L_1); \end{aligned} \qquad \text{then} \qquad \begin{aligned} -\frac{e_2^{n+1}}{dx^2} &= f \text{ in } (l_2, L), \\ e_2^{n+1}(l_2) &= e_1^{n+1}(l_2), \\ e_2^{n+1}(L) &= 0. \end{aligned}$$

The solution to the error problem is

$$e_1^{n+1}(x) = \frac{x}{L_1} e_2^n(L_1), \quad e_2^{n+1}(x) = \frac{L-x}{L-l_2} e_1^{n+1}(l_2).$$

Thes functions increase linearly from the boundary of the domain to the boundary of the overlapping region. The error at for instance $x = L_1$ is updated as

$$e_2^{n+1}(L_1) = \frac{1 - \delta/(L - l_2)}{1 + \delta/l_2} e_2^n(L_1),$$

where $\delta = L_1 - l_2 > 0$ is the overlap. The error is reduced by a factor of

$$\rho_{\text{1D}} = \frac{1 - \delta/(L - l_2)}{1 + \delta/l_2}, \tag{2.15}$$

which indicates the convergence becomes quicker as the overlap increases [3, Section 1.5.1].

2D case

In the 2D case two half planes are considered $\Omega_1 = (-\infty, \delta) \times \mathbb{R}$ and $\Omega_2 = (\delta, \infty) \times \mathbb{R}$. Following the example of Dolean et al. the problem is

$$\begin{aligned} -(\eta - \Delta)u &= f \text{ in } \mathbb{R}^2, \\ u & \text{ bounded at infinity.} \end{aligned}$$

Proceeding in similar fashion as the one-dimensional case, the error $e^{n+1}_{1,2}$ can be solved for in the two subdomains. This is done via a partial Fourier transform of the problem in the y-direction yielding an ODE for the transformed error $\hat{e}^{n+1}_{1,2}$, which can be solved explicitly with the ansatz

$$\hat{e}^{n+1}_{1,2}(x, k) = \gamma_1(k) e^{\lambda_+(k)x} + \gamma_2(k) e^{\lambda_-(k)x},$$

where $\lambda_\pm(k) = \pm\sqrt{k^2 + \eta}$. By using the interface conditions we find

$$\gamma_i^{n+1}(k) = \rho(k; \eta, \delta)^2 \gamma_i^{n-1}(k),$$

such that the convergence factor is [3, Equation 1.36]

$$\rho_{\text{2D}}(k; \eta, \delta) = e^{-\delta\sqrt{\eta + k^2}} \tag{2.16}$$

which indicates that the convergence is quicker as the overlap increases as before. Next to this, it also shows that the convergence is quicker for higher frequencies k.

2.3.3. Need for a coarse space

Following upon the results in the previous section 2.3.2 it is clear that the convergence of the Schwarz method not only depends on the extent of the overlap between various subdomains, but on the frequency components of the solution as well. In a general sense this means that low frequency modes need for instance at least N_{sub} steps to travel from one end of a square domain to the other. This in turns causes plateaus in the convergence of the Schwarz method. To overcome this, we can perform a Galerkin projection of the error onto a coarse space. That is we solve

$$\min_{\beta} ||A(x + R_0^T \beta) - f||^2,$$

where Z is a matrix representing the coarse space. The solution to this problem is

$$\beta = (R_0 A R_0^T)^{-1} R_0 r,$$

where $r = f - Ax$ is the residual.

The coarse space R_0 can be constructed in various ways. The classical way is called the Nicolaides space [3, Section 4.2], which uses the discrete partition of unity operators D_i as examplified in equation (2.13) to get

$$R_0 = \sum_{i=1}^{N_{\text{sub}}} R_i^T D_i R_i. \tag{2.17}$$

Note that the course space has a block-diagonal form.

Finally the coarse space correction term can be added to the Schwarz preconditioners equations (2.14a) and (2.14b) to get the following preconditioners

$$M_{\text{ASM,2}} = R_0^T (R_0 A R_0^T)^{-1} R_0 + \sum_{i=1}^{N_{\text{sub}}} R_i^T (R_i A R_i^T)^{-1} R_i, \tag{2.18a}$$

$$M_{\text{RAS,2}} = R_0^T (R_0 A R_0^T)^{-1} R_0 + \sum_{i=1}^{N_{\text{sub}}} R_i^T D_i (R_i A R_i^T)^{-1} R_i. \tag{2.18b}$$

2.3.4. Two-level additive Schwarz method

In this section the we will construct a coarse space for a Poisson problem with a constant scalar coefficient on arbitrary domain like in problem 2.12. However, the method is applicable to more general (highly) heterogeneous scalar problems, like the Darcy problem (see section 2.3.6). The coarse space is constructed using the eigenfunctions corresponding to the smallest m_j eigenvalues resulting from a local eigenproblem in each subdomain Ω_j. The coarse space is then constructed by taking the union of the m_j eigenvectors corresponding to the smallest eigenvalues in each subdomain glued together by the partition of unity functions χ_j. All of this can be found in [3, Sections 5.1-5.5].

This coarse space is subsequently used to construct the two level additive Schwarz preconditioner, and bounds for its condition number are provided as well.

Slowly convergent modes of the Dirichlet-to-Neumann map

As seen in section 2.3.2 the local error in any subdomain in the Schwarz method satisfies the original problem without forcing, i.e. right hand side $f = 0$. At the interface the local error has a Dirichlet boundary condition that equals the error of the neighbouring subdomain. Additionally, the convergence factor, e.g. ρ_{2D}, depends on the frequency of the modes present in the local error. In particular, small frequencies appear to have slow convergence. The question thus becomes how to get rid of these small frequency modes in the local errors of all subdomains.

One possible answer is the so-called Dirichlet-to-Neumann map [3, Definition 5.1]

Definition 2.2. (Dirichlet-to-Neumann map for a Poisson problem) For any function defined on the interface $u_{\Gamma_j} : \Gamma_j \mapsto \mathbb{R}$, we consider the Dirichlet-to-Neumann map

$$\mathrm{DtN}_{\Omega_j}\left(u_{\Gamma_j}\right) = \left.\frac{\partial v}{\partial \mathbf{n}_j}\right|_{\Gamma_j},$$

where $\Gamma_j := \partial\Omega_j \backslash \partial\Omega$ and v satisfies

$$\begin{aligned} -\Delta v &= 0 && \text{in } \Omega_j, \\ v &= u_{\Gamma_j} && \text{on } \Gamma_j, \\ v &= 0 && \text{on } \partial\Omega_j \cap \partial\Omega. \end{aligned} \tag{2.19}$$

The Dirichlet-to-Neumann map essentially solves for an error-like variable v that satisfies the Dirichlet local interface (or global boundary) conditions. DtN then maps the interface condition to the normal derivative of v on the interface, i.e. the Neumann condition. Now, as stated above and illustrated in [3, Figure 5.2] the low frequency modes of the error correspond to those modes that are nearly constant accross an interface, for which the Neumann condition is close to zero. So the problem of slowly convergent modes in the error of the Schwarz method is equivalent to a problem of finding eigenpairs of the DtN operator.

Hence we aim to solve the eigenvalue problem

$$\mathrm{DtN}_{\Omega_j}(v) = \lambda v,$$

which can be reformulated in the variational form. To that end let w be a test function that is zero on $\delta\Omega$. Multiply both sides of equation (2.19) by w, integrate over Ω_j and apply Green's theorem to get

$$\int_{\Omega_j} \nabla v \cdot \nabla w - \lambda \int_{\Omega_j} \frac{\partial v}{\partial \mathbf{n}_j} w, \quad \forall w.$$

Then, use the eigen property of v and fact that w is zero on $\delta\Omega$ to get the eigen problem in the variational form

$$\text{Find } (v,\lambda) \text{ s.t. } \int_{\Omega_j} \nabla v \cdot \nabla w - \lambda \int_{\Gamma_j} vw = 0, \quad \forall w. \tag{2.20}$$

FEM discretization

The discretisation is done in the context of the finite element method. To that end we consider a triangulation $\mathcal{T}$ of the domain Ω and a partition of unity χ_j. Denote by $V_{h,0} = V_h \cap H_0^1(\Omega)$ the space of piecewise continuous functions $v_h \in H_0^1(\Omega)$ with respect to $\mathcal{T}$. Let the basis of $V_{h,0}$ be given by $\{\phi_k\}_{\in\mathcal{N}}$. The FE formulation of the Poisson problem equation (2.12) follows from the variational form

$$a(u_h, v_h) = (f, v_h), \quad \forall u_h, v_h \in V_{h,0},$$

and is given by

$$A\mathbf{u} = b, \quad A_{ij} = a(\phi_j, \phi_i), \; b_i = (f, \phi_i) \quad \forall i, j \in \mathcal{N}. \tag{2.21}$$

Next to this we need a way of interpolating functions in $C(\Omega)$ to V_h. This is done by the interpolation operator $I_h : C(\Omega) \mapsto V_{h,0}$, which is defined by

$$\mathcal{I}_h v = \sum_{i\in\mathcal{N}} v(x_i)\phi_i,$$

where x_i are the nodes of the triangulation $\mathcal{T}$. $\mathcal{I}_h$ is stable with respect to the a-norm, that is

$$\|I_h(v)\|_a \leq C_{I_h} \|v\|_a.$$

As before we partition Ω into N_{sub} subdomains Ω_j, which overlap each other by one or several layers of elements in the triangulation $\mathcal{T}$. We make the the following observations

I For every degree of freedom $k \in \mathcal{N}$, there is a subdomain Ω_j such that ϕ_k has support in Ω_j [3, Lemma 5.3].

II The maximum number of subdomains a mesh element can belong to is given by

$$k_0 = \max_{\tau\in\mathcal{T}} \left(\#\{j | 1 \leq j \leq N_{\text{sub}} \text{ and } \tau \subset \Omega_j\}\right).$$

III The minimum number of colors needed to color all subdomains so that no two adjacent subdomains have the same color is given by

$$N_c \geq k_0$$

IV The minimum overlap for any subdomain Ω_j with any of its neihbouring subdomains is given by

$$\delta_j = \inf_{x \in \Omega_j \setminus \cup_{i \neq j} \bar{\Omega}_i} \text{dist}(x, \partial\Omega_j \setminus \partial\Omega).$$

V The partition of unity functions $\{\chi_j\}_{j=1}^{N_{\text{sub}}} \subset V_h$ are such that

V.a $\chi_j(x) \in [0,1], \quad \forall x \in \bar{\Omega}, j = 1, \dots, N_{\text{sub}},$

V.b $\text{supp}(\chi_j) \subset \bar{\Omega}_j,$

V.c $\sum_{j=1}^{N_{\text{sub}}} \chi_j(x) = 1, \quad \forall x \in \bar{\Omega},$

V.d $\|\nabla \chi_j(x)\| \leq \frac{C_\chi}{\delta_j},$

and are given by

$$\chi_j(x) = I_h\left(\frac{d_j(x)}{\sum_{j=1}^{N_{\text{sub}}} d_j(x)}\right),$$

where

$$d_j(x) = \begin{cases} \text{dist}(x, \partial\Omega_j), & x \in \Omega_j, \\ 0, & x \in \Omega \setminus \Omega_j. \end{cases}$$

VI The overlap region for any subdomain is given by

$$\Omega_j^\delta = \{x \in \Omega_j | \chi_j < 1\}.$$

The extension operator $E_j : V_{h,0}(\Omega_j) \to V_h$ is defined by

$$V_h = \sum_{j=1}^{N_{\text{sub}}} E_j V_{h,0}(\Omega_j),$$

which is gauranteed by item ASM observation I.

Note that using the extension operator we can show that all the local bilinear forms are positive definite as

$$a_{\Omega_j}(v, w) = a(E_j v, E_j w) \geq \alpha \|E_j v\|_a^2, \quad \forall v, w \in V_{h,0}(\Omega_j),$$

and a is positive definite.

Finally, we define the a-symmetric projection operators $\tilde{\mathcal{P}}_j : V_{h,0} \to V_h$ and $\mathcal{P}_j : V_h \to V_h$ defined by

$$a_{\Omega_j}(\tilde{\mathcal{P}}_j u, v_j) = a(u, E_j v_j) \quad \forall v_j \in V_{h,0},$$
$$\mathcal{P} = E_j \tilde{\mathcal{P}}_j.$$

then their matrix counterparts are given by

$$\tilde{P}_j = A_j^{-1} R_j^T A,$$
$$P_j = R_j^T A_j^{-1} R_j^T A,$$

where $A_j = R_j A R_j^T$. From this we can construct the two-level additive Schwarz method as

$$M_{\text{ASM},2}^{-1} A = \sum_{j=1}^{N_{\text{sub}}} P_j. \tag{2.22}$$

2.3.5. Convergence of two-level additive Schwarz

In the following we denote

$$\mathcal{P}_{\text{ad}} = \sum_{j=1}^{N_{\text{sub}}} \mathcal{P}_j,$$

and correspondingly,

$$P_{\text{ad}} = \sum_{j=1}^{N_{\text{sub}}} P_j.$$

In the context of this thesis the two-level additive Schwarz method is used in combination with a Krylov subspace method (section 2.4), in which case convergence rate depends on the entire spectrum of eigenvalues (section 2.4.8). However, an upperbound for the convergence rate (section 2.4.8) can be derived from the condition number of P_{ad} via equation (2.41).

Using the fact that P_{ad} is symmetric with respect to the a-norm, we can write

$$\kappa(P_{\text{ad}}) = \frac{\lambda_{\text{max}}}{\lambda_{\text{min}}},$$

where

$$\lambda_{\text{max}} = \sup_{v \in V_h} \frac{a(\mathcal{P}_{\text{ad}})}{a(v,v)}, \quad \lambda_{\text{min}} = \inf_{v \in V_h} \frac{a(\mathcal{P}_{\text{ad}})}{a(v,v)}.$$

Additionally, we can employ the a-orthogonality of the projection operators to get

$$\frac{a(\mathcal{P}_j u, u)}{\|u\|_a^2} = \frac{a(\mathcal{P}_j u, \mathcal{P}_j u)}{\|u\|_a^2} \leq 1.$$

Going further, we can pose that the projection operators defined by the sum of projection operators $\mathcal{P}_j$ of like-colored subdomains are a-orthogonal to each other. This is due to the fact that the partition of unity functions χ_j are such that they are zero on the interface of like-colored subdomains (see item ASM observation III). To that end, define

$$\mathcal{P}_{\Theta_i} = \sum_{j \in \Theta_i} \mathcal{P}_j,$$

where Θ_i is the set of indices of subdomains with color i and $i = 1, \ldots, N_c$. Then, we can write [3, Lemma 5.9]

$$\begin{aligned}\lambda_{\text{max}}(\mathcal{P}_{\text{ad}}) &= \sup_{v \in V_h} \sum_{i=1}^{N_c} \frac{a(\mathcal{P}_{\Theta_i} v, v)}{a(v,v)} \\ &\leq \sum_{i=1}^{N_c} \sup_{v \in V_h} \frac{a(\mathcal{P}_{\Theta_i} v, v)}{a(v,v)} \\ &\leq N_c + 1,\end{aligned}$$

where the extra one comes from the coarse projection operator $\mathcal{P}_0$. Note that this bound can be made sharper by using item ASM observation II to get $\lambda_{\text{max}}(\mathcal{P}_{\Theta_i}) \leq k_0 + 1$.

On the other hand, it can be shown that the minimum eigenvalue satisfies provided that $v \in V_h$ admits a C_0-stable decomposition [3, Theorem 5.11]

$$\lambda_{\text{min}}(\mathcal{P}_{\text{ad}}) \geq C_0^{-2}.$$

Finally, we can write the condition number of the two-level additive Schwarz preconditioner as

$$\kappa(P_{\text{ad}}) \leq (N_c + 1)\, C_0^2. \tag{2.23}$$

The value of C_0 depends on the projection operator Π_j to the chosen coarse space V_0 for each subdomain.

I. **Nicolaides coarse space** The projection operator is defined as

$$\Pi_j^{\text{Nico}} u = \begin{cases} (\frac{1}{|\Omega_j|} \int_{\Omega_j} u) \mathbf{1}_{\Omega_j}, & \delta\Omega_j \cap \delta\Omega = \emptyset, \\ 0, & \text{otherwise}, \end{cases} \tag{2.24}$$

which gives rise to the following basis functions in $V_{h,0}$

$$\Phi_j^{\text{Nico}} = I_h(\chi_j \mathbf{1}_{\Omega_j}).$$

Then,

$$V_0 = \text{span}\{\Phi_j^{\text{Nico}}\}_{j=1}^{N_{\text{sub}}},$$

and

$$\dim V_0 = \text{the number of floating subdomains},$$

that is the number of subdomains that are not connected to the boundary of the domain Ω. In this case [3, Theorem 5.16]

$$C_0^2 = \left(8 + 8C_\chi^2 \max_{j=1}^{N_{\text{sub}}} \left[C_P^2 + C_{\text{tr}}^{-1} \frac{H_j}{\delta_j}\right] k_0 C_{I_h}(k_0+1) + 1\right),$$

where H_j is the diameter of the subdomain Ω_j, C_p the Poincaré constant following from [3, Lemma 5.18] and C_{tr} is the trace constant.

II. **Local eigenfunctions coarse space** The projection operator is defined as

$$\Pi_j^{\text{spec}} u = \sum_{k=1}^{m_j} a_{\Omega_j}(u, v_k^{(j)}) v_k^{(j)},$$

where $v_k^{(j)}$ is the k^{th} eigenfunction resulting from the eigenproblem in equation (2.20). The basis functions in $V_{h,0}$ are then given by

$$\Phi_{j,k}^{\text{spec}} = I_h(\chi_j v_k^{(j)}),$$

resulting in the coarse space

$$V_0 = \text{span}\{\Phi_{j,k}^{\text{spec}}\}_{j=1,k=1}^{N_{\text{sub}},m_j},$$

with dimension

$$\dim V_0 = \sum_{j=1}^{N_{\text{sub}}} m_j.$$

In this case [3, Theorem 5.17]

$$C_0^2 = \left(8 + 8C_\chi^2 \max_{j=1}^{N_{\text{sub}}} \left[C_P^2 + C_{\text{tr}}^{-1} \frac{1}{\delta_j \lambda_{m_j+1}}\right] k_0 C_{I_h}(k_0+1) + 1\right).$$

2.3.6. Spectral coarse spaces for scalar heterogeneous problems

This section outlines various methods for constructing a coarse space that is both scalable as robust to high contrast in a problem coefficient. An example of such is the Darcy problem

$$-\nabla \cdot (\alpha \nabla u) = f \text{ in } \Omega,$$

where α is a scalar coefficient that can vary by several orders of magnitude in the domain Ω.

MsFEM

Main citation: [4]

To do ►*Write about the coarse space assumptions C1-C5*◄

To do ►*Local coarse grid basis functions as solutions to homogeneous version of system equation, as before*◄

To do ►*Write about the requirements for boundary conditions M1-M4*◄

To do ►*Distinguish between linear and oscillatory boundary conditions for local problems*◄

To do ►*Coarse grid basis functions are harmonic extensions of edge-(face-)restricted basis functions*◄

To do ►*Explain how R_0 is obtained from coarse grid basis functions [4, Equation 2.12]*◄

To do ►*Discuss robustness indicators $\pi(\alpha), \gamma(\alpha)$*◄

ACMS

Main citation: [7]

To do ▶*Separation of scales, fine and coarse triangulation $\mathcal{T}_h$ and $\mathcal{T}_H$*◀

To do ▶*Coarse problem is separated into two parts $u_c = u_I + u_\Gamma$, inner and interface*◀

To do ▶*Extension of MsFEM with vertex-specific, edge-specific and fixed-interface basis functions, i.e. MsFEM are simply only the vertex-specific basic functions*◀

To do ▶*Vertex-specific basis functions are harmonic extensions of trace values defined on the interface set Γ*◀

To do ▶*Edge-specific basis functions are harmonic extensions of eigenmodes resulting from an eigenvalue problem defined on an edge e*◀

To do ▶*Fixed-interface basis functions are the eigenmodes of the local eigenvalue problem defined on a coarse element T*◀

To do ▶*Two flavors for the coarse space; with DBCs or with NBCs*◀

To do ▶*With DBCs: MsFEM basis functions with edge-specific basis functions that agree on a shared edge(s) e_{ij} between subdomains $\Omega_{i,j}$ and are constructed from the harmonic extension (into the interior) of the eigenmodes resulting from an eigenvalue problem defined on the edge e_{ij} (albeit with a scaled bilinear form on the r.h.s.). The edge-specific coarse space is approximated by only taking eigenmodes corresponding to eigenfrequencies up to a certain tolerance*◀

To do ▶*With NBCs: both MsFEM basis functions and edge-specific basis functions are altered. The former are still defined solely on an edge e_{ij} and retain their Kronecker-delta vertex condition, but are now defined through a generalized eigenvalue problem on the slab of width kh; η_{ij}^{kh}. The latter are now solely defined through a generalized eigenvalue problem on the same slab, but no longer have a DBC*◀

To do ▶*Generalized eigenvalue problems may be solved through lumping the mass matrix*◀

(R)GDSW

Main citation: [2]

To do ▶*Splitting of problem into non-overlapping subdomains and subsequently into interior and interface nodes*◀

To do ▶*Only inputs is a coarse space G, whose columns span the rigid body modes of the subdomains*◀

To do ▶*Restriction operators to interface DOFs R_Γ, interior DOFs R_I and subdomain specific versions thereof, e.g. R_{Γ_j}*◀

To do ▶*Coarse solution on interface set $u_{0,\Gamma} = \sum_j^{N_{sub}} R_{\Gamma_j}^T G_{\Gamma_j} q_j = \Phi_\Gamma q$ + figure out what q resembles*◀

To do ▶*Coarse solution $u_0 = R_\Gamma^T u_{0,\Gamma} + R_I^T u_{0,I}$, where the interior DOF restriction operator R_I is obtained from energy minimizing extensions of the $u_{0,\Gamma}$ into subdomain interiors*◀

To do ▶*Energy minimization principle applied to the potential $u_0 A u_0$ leads to Φ_I*◀

AMS

Main citation: [1]

To do ▶*Find original reference outlining AMS method*◀

To do ▶*Splitting of problem into non-overlapping subdomains and subsequently into interior and interface nodes*◀

To do ▶*Further splitting of interface nodes into edge, vertex and face (for 3D) nodes*◀

To do ▶*Elimination of lower diagonal blocks*◀

To do ▶*Same energy minimization principle as in (R)GDSW leads to Φ_I*◀

2.4. Conjugate Gradient Method

This section is largely based on the extremely instructive book by Saad about iterative methods [8, Section 6: Krylov Subspace Methods Part I]

The conjugate gradient method is a special instance of a projection method discussed in section 2.2 where

$$\mathcal{K}_m(A_0, r_0) = \text{span}\{r_0, Ar_0, A^2r_0, \dots, A^{m-1}r_0\}, \tag{2.25}$$

or $\mathcal{K}_m$ as a shorthand. The approximate answer is then given by

$$x_m = x_0 + \sum_{i=0}^{m-1} c_i A^i r_0 = x_0 + q_{m-1}(A)r_0, \tag{2.26}$$

where $q_{m-1}(A)$ is a polynomial of degree $m-1$ in A. It is shown later in this section how the coefficients c_i are obtained (equation (2.47)).

2.4.1. Variants of the CG method

Variants of the CG method differ in the way A is preconditioned (see section section 2.4.9) d the choices for the constraint subspace $\mathcal{L}_m$. The former type of variations result in the preconditioned CG method PCG and these are described in section 2.4.9. The latter type of variations branch off into two major categories:

i $\mathcal{L}_m = \mathcal{K}_m$ and $\mathcal{L}_m = A\mathcal{K}_m$;

ii $\mathcal{L}_m = \mathcal{K}_m(A^T, r_0)$.

Note that item CG-type i correspond to the residual and error projection methods from section 2.2.2. The former results in Arnoldi's method, which is discussed in section 2.4.3, as well as variants thereof like Full Orthogonalization Method (FOM), Incomplete Orthogonalization Method (IOM) and Direct Incomplete Orthogonalization Method (DIOM). The latter on the other hand results in the Generalized Minimum Residual Method (GMRES).

2.4.2. Krylov subspaces

Definition 2.3. The grade of a vector v with respect to a matrix A is the lowest degree of the polynomial q such that $q(A)v = 0$.

Consequently,

Theorem 2.3. The Krylov subspace $\mathcal{K}_m$ is of dimension m if and only if the grade μ of v with respect to A is not less than m [8, proposition 6.2],

$$\dim(\mathcal{K}_m) = m \iff \mu \geq m,$$

such that

$$\dim(\mathcal{K}_m) = \min\{m, \text{grade}(v)\}. \tag{2.27}$$

Definition 2.4. The action of a matrix A can be thought of as a mapping

$$\mathbb{R}^n \to \mathbb{R}^n : v \mapsto Av$$

Thus the domain and codomain of A are $\mathbb{R}^n$. Let $X \subset \mathbb{R}^n$, we can consider the map

$$X \to \mathbb{R}^n : v \mapsto Av$$

instead. The only difference from A is that the domain is X. This map is defined as the restriction $A_{|_X}$ of A to X.

Definition 2.5. Let Q be a projector onto the subspace X. Then the section of the operator A onto X is defined as $QA_{|_X}$.

Theorem 2.4. Let Q_m be any projector onto $\mathcal{K}_m$ and let A_m be the section of A to $\mathcal{K}_m$, that is, $A_m = Q_m A_{|\mathcal{K}_m}$. Then for any polynomial q of degree not exceeding $m-1$ [8, proposition 6.3],

$$q(A)v = q\left(A_m\right)v$$

and for any polynomial of degree $\leq m$,

$$Q_m q(A)v = q\left(A_m\right)v$$

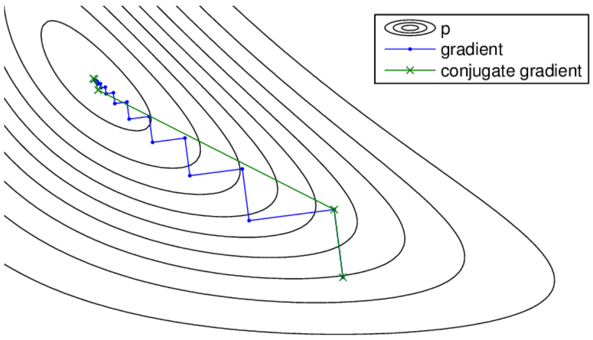


Figure 2.1: Conjugate Gradient Method

2.4.3. Arnoldi's method

The Arnoldi method is a generalization of the Gram-Schmidt orthogonalization process to Krylov subspaces. The Arnoldi method is used to generate an orthonormal basis for the Krylov subspace $\mathcal{K}_m(A, r_0)$.

Algorithm 6 Arnoldi's method [8, Algorithm 6.1]

```
Choose a vector $v_1$ with $||v_1|| = 1$
for $j = 1, 2, \dots, m$ do
    $w_j \leftarrow Av_j$
    for $i = 1, 2, \dots, j$ do
        $h_{ij} \leftarrow (w, v_i)$
        $w_j \leftarrow w_j - h_{ij}v_i$
    end for
    $h_{j+1,j} \leftarrow ||w_j||$
    $v_{j+1} \leftarrow w_j / h_{j+1,j}$
end for
```

The Arnoldi method generates an orthonormal basis $V_m = [v_1, v_2, \dots, v_m]$ for the Krylov subspace $\mathcal{K}_m(A, r_0)$ and an upper Hessenberg matrix $\bar{H}_m$ such that

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T, \tag{2.28a}$$

$$= V_{m+1} \bar{H}_m, , \tag{2.28b}$$

$$V_m^T A V_m = H_m, \tag{2.28c}$$

where e_m is the mth unit vector and H_m is obtained from $\bar{H}_m$ by removing the last row [8, Proposition 6.5].

Note that the Arnoldi method stops at j, if and only if the minimal polynomial of A with respect to v_1 is of degree j [8, proposition 6.6].

2.4.4. Arnoldi's method for linear systems

The Arnoldi method can be used to solve linear systems of the form $Ax = b$. If $v_1 = r_0/||r_0||_2$ and $\beta = ||r_0||_2$, then by equation (2.28c) we have

$$V_m^T A V_m = H_m \text{ and } V_m^T r_0 = V_m^T \beta v_1 = \beta e_1 \implies \begin{array}{c} H_m y = \beta e_1, \\ x_m = x_0 + V_m y. \end{array} .$$

This results in the following algorithm

Algorithm 7 Arnoldi's method for linear systems (FOM) [8, Algorithm 6.4]

```
Compute $r_0 = b - Ax_0$, $\beta = ||r_0||_2$ and $v_1 = r_0/\beta$
Define $H_m = \{0\}$
Define $V_1 = \{v_1\}$
for $j = 1, 2, \dots, m$ do
    $w_j = Av_j$
    for $i = 1, 2, \dots, j$ do
        $h_{ij} = (w_j, v_i)$ and store $h_{ij}$ in $H_m$
        $w_j = w_j - h_{ij}v_i$
    end for
    $h_{j+1,j} = ||w_j||_2$
    if $h_{j+1,j} = 0$ then
        $m = j$
        break
    end if
    $v_{j+1} = w_j/h_{j+1,j}$ and store $v_{j+1}$ into $V_{j+1}$
end for
Solve $H_m y_m = \beta e_1$ for $y_m$
$x_m = x_0 + V_m y_m$
```

Note that a stopping criterion can be derived from the residual vector $r_m = b - Ax_m$. Theorem 2.5 gives a way of calculating the size of the residual vector.

Theorem 2.5. The residual vector $r_m = b - Ax_m$ satisfies

$$\begin{aligned} r_m &= b - Ax_m && (2.29a)\\ &= r_0 - AV_m y_m && (2.29b)\\ \mathit{equation}\ \text{(2.28a)} \rightarrow &= \beta v_1 - V_m H_m y_m - h_{m+1,m} e_m^T y_m v_{m+1} && (2.29c)\\ &= -h_{m+1,m} e_m^T y_m v_{m+1}. && (2.29d) \end{aligned}$$

Therefore, the residual vector is orthogonal to the Krylov subspace $\mathcal{K}_m(A, r_0)$ and its size is given by

$$||r_m||_2 = h_{m+1,m}|e_m^T y_m|, \tag{2.30}$$

where $h_{m+1,m} \geq 0$ [8, Proposition 6.7].

2.4.5. Lanczos' Algorithm

In the special case where A is symmetric, the Arnoldi method can be simplified to the Lanczos algorithm. The Lanczos algorithm is a special case of the Arnoldi method where the matrix $T_m = V_m^T A V_m^T$ is tridiagonal given by

$$T_m = \begin{pmatrix} \delta_1 & \eta_2 & 0 & \dots & 0 \\ \eta_2 & \delta_3 & \eta_3 & \dots & 0 \\ 0 & \eta_3 & \delta_4 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \eta_m \\ 0 & 0 & 0 & \eta_m & \delta_m \end{pmatrix} \tag{2.31}$$

This leads to the following algorithm

Algorithm 8 Lanczos [8, algorithm 6.15]

```
Choose a vector $v_1$ with $||v_1|| = 1$
$\eta_1 = 0$
$v_0 = 0$
for $j = 1, 2, \dots, m$ do
    $w_j = Av_j - \eta_j v_{j-1}$
    $\delta_j = (w_j, v_j)$
    $w_j = w_j - \delta_j v_j$
    $\eta_{j+1} = ||w_j||$
    if $\eta_{j+1} = 0$ then
        Break
    end if
    $v_{j+1} = w_j / \eta_{j+1}$
end for
```

Similarly, we can apply the Lanczos algorithm to linear systems resulting in algorithm 9.

Algorithm 9 Lanczos algorithm for linear systems [8, Algorithm 6.16]

```
Compute $r_0 = b - Ax_0$, $\beta = ||r_0||_2$, $v_0 = 0$ and $v_1 = r_0/\beta$
$V_1 = \{v_1\}$
for $j = 1, 2, \dots, m$ do
    $w_j = Av_j - \eta_j v_{j-1}$
    $\delta_j = (w_j, v_j)$
    $w_j = w_j - \delta_j v_j$
    $\eta_{j+1} = ||w_j||_2$
    if $\eta_{j+1} = 0$ then
        $m = j$
        Break
    end if
    $v_{j+1} = w_j / \eta_{j+1}$ and store $v_{j+1}$ into $V_{j+1}$
end for
Solve the tridiagonal system $T_m y_m = \beta e_1$ for $y_m$
$x_m = x_0 + V_m y_m$
```

2.4.6. D-Lanczos

The direct version of the Lanczos algorithm or 'D-Lanczos' is obtained by performing a LU-factorisation T_m

$$T_m = L_m U_m = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ \tilde{\eta}_2 & 1 & 0 & \dots & 0 \\ 0 & \tilde{\eta}_3 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \tilde{\eta}_m & 1 \end{pmatrix} \times \begin{pmatrix} \tilde{\delta}_1 & \eta_2 & 0 & \dots & 0 \\ 0 & \tilde{\delta}_2 & \eta_3 & \dots & 0 \\ 0 & 0 & \tilde{\delta}_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \eta_m \\ 0 & 0 & 0 & \dots & \tilde{\delta}_m \end{pmatrix} \tag{2.32}$$

Then, the approximate solution is given by

$$\begin{aligned} x_m &= x_0 + V_m y_m \\ &= x_0 + V_m U_m^{-1} L_m^{-1} \beta e_1 \\ &= x_0 + V_m U_m^{-1} (L_m^{-1} \beta e_1) \\ &= x_0 + P_m z_m, \end{aligned}$$

where $P_m = V_m U_m^{-1}$ and $z_m = L_m^{-1} \beta e_1$. Considering the definition of U_m in equation (2.32), we have that the m^{th} column of P_m is given by

$$p_m = \frac{1}{\tilde{\delta}_m} \left[v_m - \eta_m p_{m-1} \right]. \tag{2.33}$$

Furthermore, from the LU factorization of T_m we have that

$$\begin{aligned}\tilde{\eta}_m &= \frac{\eta_m}{\tilde{\delta}_{m-1}},\\ \tilde{\delta}_m &= \delta_m - \tilde{\eta}_m \eta_m,\ m > 1.\end{aligned}$$

Now the solution can be incrementally updated by realizing that

$$z_m = \begin{pmatrix} z_{m-1} \\ \zeta_m \end{pmatrix} = \begin{pmatrix} z_{m-2} \\ \zeta_{m-1} \\ \zeta_m \end{pmatrix},$$

and

$$L_m = \begin{pmatrix} L_{m-1} & \mathbf{0}_{m-1} \\ \mathbf{0}^T_{m-2} \quad \tilde{\eta}_m & 1 \end{pmatrix}.$$

Then,

$$L_m z_m = \begin{pmatrix} L_{m-1} z_{m-1} \\ \tilde{\eta}_m \zeta_{m-1} + \zeta_m \end{pmatrix} = \begin{pmatrix} \beta e_1 \\ 0 \end{pmatrix},$$

where the last equality follows from definition of z_m. Consequently, we have that

$$\zeta_m = -\tilde{\eta}_m \zeta_{m-1}.$$

Finally, we obtain

$$\begin{aligned} x_m &= x_0 + P_m z_m \\ &= x_0 + [P_{m-1}\ p_m] \begin{pmatrix} z_{m-1} \\ \zeta_m \end{pmatrix} \\ &= x_0 + P_{m-1} z_{m-1} + p_m \zeta_m \\ &= x_{m-1} + p_m \zeta_m. \end{aligned}$$

Putting it all together, we obtain algorithm 10.

Algorithm 10 D-Lanczos [8, Algorithm 6.17]

```
$r_0 = b - Ax_0$, $\beta = ||r_0||_2$, $v_1 = r_0/\beta$
$\tilde{\eta}_1 = \beta_1 = 0$, $p_0 = 0$
for $m = 1, 2, \dots, m$ until convergence do
    $w = Av_m - \beta_m v_{m-1}$
    $\delta_m = (w, v_m)$
    if $m > 1$ then
        $\tilde{\eta}_m = \frac{\beta_m}{\tilde{\delta}_{m-1}}$
        $\zeta_m = -\tilde{\eta}_m \zeta_{m-1}$
    end if
    $\tilde{\delta}_m = \delta_m - \tilde{\eta}_m \beta_m$
    $p_m = \frac{1}{\tilde{\delta}_m} [v_m - \beta_m p_{m-1}]$
    $x_m = x_{m-1} + p_m \zeta_m$
    if convergence criterion is met (using equation (2.30) for example) then
        break
    end if
    $w = w - \delta_m v_m$
    $\beta_{m+1} = ||w||_2$
    $v_{m+1} = w/\beta_{m+1}$
end for
```

Note that the residual vectors produced by algorithm 10 are orthogonal to each other due to theorem 2.5. Additionally,

$$\begin{aligned} P_m^T A P_m &= U_m^{-T} V_m^T A V_m U_m^{-1} \\ &= U_m^{-T} T_m U_m^{-1} \\ &= U_m^{-T} L_m, \end{aligned}$$

where U_m^{-T} and L_m are both lower diagonal matrices. Their product must be symmetric, since $P_m^T A P_m$ is (symmetry of A). Therefore, $U_m^{-T} L_m$ must be a diagonal matrix. Consequently, the vectors p_m are A-orthogonal to each other.

2.4.7. Derivation of CG

From observations made in the in algorithm 10, we can derive the conjugate gradient method. We start by constraining subsequent residuals r_j to be orthogonal and search directions p_j to be A-orthogonal to each other. To that end, let

$$x_{j+1} = x_j + \alpha_j p_j, \tag{2.34}$$

and, thereby,

$$r_{j+1} = r_j - \alpha_j A p_j. \tag{2.35}$$

If the residuals are to be orthogonal, then

$$(r_{j+1}, r_j) = 0 \implies (r_j - \alpha_j A p_j, r_j) = 0 \implies \alpha_j = \frac{(r_j, r_j)}{(A p_j, r_j)}.$$

Now, using equations (2.29d) and (2.33), we can write the next search direction as a linear combination of the previous search direction and the next residual

$$p_{j+1} = r_{j+1} + \beta_j p_j. \tag{2.36}$$

Substituting equation (2.36), we obtain

$$(A p_{j+1}, r_j) = (A p_j, p_j - \beta_{j-1} p_{j_1}) = (A p_j, p_j),$$

since p_j is A-orthogonal to p_{j-1}. This allows us to write

$$\alpha_j = \frac{(r_j, r_j)}{(A p_j, p_j)}. \tag{2.37}$$

Additionally, taking the inner product with $A p_j$ on both sides of equation (2.36) gives

$$\beta_j = \frac{(r_{j+1}, A p_j)}{(p_j, A p_j)}.$$

Now, rewriting equation (2.35) gives

$$A p_j = \frac{1}{\alpha_j}(r_j - r_{j+1}),$$

which we substitute into the equation for β_j to obtain

$$\beta_j = \frac{1}{\alpha_j} \frac{(r_{j+1}, (r_{j+1} - r_j))}{(A p_j, r_j)} = \frac{(r_{j+1}, r_{j+1})}{((r_j - r_{j-1}), r_j)} = \frac{(r_{j+1}, r_{j+1})}{(r_j, r_j)}. \tag{2.38}$$

Finally, we can write the conjugate gradient method as algorithm 11.

Algorithm 11 Conjugate Gradient Method [8, Algorithm 6.18]

```
$r_0 = b - Ax_0$, $p_0 = r_0$, $\beta_0 = 0$
for $j = 0, 1, 2, \ldots, m$ do
    $\alpha_j = (r_j, r_j)/(Ap_j, p_j)$
    $x_{j+1} = x_j + \alpha_j p_j$
    $r_{j+1} = r_j - \alpha_j A p_j$
    $\beta_j = (r_{j+1}, r_{j+1})/(r_j, r_j)$
    $p_{j+1} = r_{j+1} + \beta_j p_j$
end for
```

Hessenberg matrix coefficients The coefficients of the Hessenberg matrix T_m can be calculated as follows

$$\delta_{j+1} = \begin{cases} \frac{1}{\alpha_j} + \frac{\beta_{j-1}}{\alpha_{j-1}} & j > 0, \\ \frac{1}{\alpha_0} & j = 0, \end{cases} \tag{2.39}$$

and

$$\eta_{j+1} = \frac{\sqrt{\beta_{j-1}}}{\alpha_{j-1}}. \tag{2.40}$$

here we have used the definition of $T_m = V_m^T A V_m$ and the fact that the residuals are just multiples of the Lanczos vectors $r_j = \text{scalar} \times v_j$ by equation (2.29d) [8, Equation 6.103].

2.4.8. Convergence of CG

Convergence rate

It can be shown [8, lemma 6.28 and theorem 6.29] that the error of the m^{th} iterate of the CG algorithm $\epsilon_m = x^* - x_m$ minimizes the A-norm of the error in the affine Krylov subspace $\mathcal{K}_m(A, r_0)$, that is

$$||(I - Aq_m(A))\epsilon_0||_A = \min_{q \in \mathcal{P}_{m-1}} ||(I - Aq(A))\epsilon_0||_A = \min_{r \in \mathcal{P}_{m-1}, r(0)=1} ||r(A)\epsilon_0||_A,$$

where the equality follows, since there exists an isomorphic mapping between the affine Krylov subspace and the polynomial space $\mathcal{P}_{m-1}$ of degree $m-1$ and the polynomial $tq(t)$ equals 0 at $t = 0$. The right-hand side can be further bounded by letting λ_i, ξ_i be the eigenvalues of A and the components of ϵ_0 in the eigenvector basis of A, respectively. Then

$$||r(A)\epsilon_0||_A = \sqrt{\sum_{i=1}^{n} |r(\lambda_i)|^2 |\xi_i|^2} \leq \max_{\lambda \in \sigma(A)} |r(\lambda)| ||\epsilon_0||_A,$$

where $\sigma(A)$ is the spectrum of A. This gives

$$\begin{aligned} ||e_m|| &\leq \min_{r \in \mathcal{P}_{m-1}, r(0)=1} \max_{\lambda \in \sigma(A)} |r(\lambda)| ||\epsilon_0||_A \\ \text{Chebyshev polynomial } C_m,\ \eta = \frac{\lambda_{\text{min}}}{\lambda_{\text{max}} - \lambda_{\text{min}}} &\to \frac{||\epsilon_0||_A}{C_m(1 + 2\eta)} \\ &\leq \frac{2||\epsilon_0||_A}{\left(1 + 2\eta + 2\sqrt{\eta(\eta+1)}\right)^m} \\ &= \frac{2||\epsilon_0||_A}{\left(\sqrt{\eta} + \sqrt{\eta+1}\right)^{2m}} \\ &= 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^m ||\epsilon_0||_A, \end{aligned}$$

where $\kappa = \lambda_{\text{max}}/\lambda_{\text{min}}$ is the condition number of (symmetric matrix) A. To sum up

Theorem 2.6. The error of the m^{th} iterate of the CG algorithm is bounded by

$$||e_m|| \leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^m ||\epsilon_0||_A, \tag{2.41}$$

where $\kappa = \lambda_{\text{max}}/\lambda_{\text{min}}$ is the condition number of (symmetric matrix) A.

During the derivation of theorem 2.6, we obtain the general expression for the error of the m^{th} iterate of the CG algorithm

$$||e_m|| \leq \min_{r \in \mathcal{P}_m, r(0)=1} \max_{\lambda \in \sigma(A)} |r(\lambda)| ||\epsilon_0||_A$$

Now define,

$$r_{\text{test}}(t) = \prod_{i=1}^{m} \frac{\lambda_i - t}{\lambda_i}.$$

Note that $r_{\text{test}} \in \mathcal{P}_m$, since it has degree m. Also, $r_{\text{test}}(0) = 1$ and $r_{\text{test}}(\lambda_i) = 0$ for $i = 1, 2, \ldots, m$. Hence, r_{test} is a polynomial that satisfies the constraints of the minimization problem. We obtain for $m = N$ that

$$||e_N||_A = ||\epsilon_0||_A \max_{\lambda \in \sigma(A)} |r_{\text{test}}(\lambda)| = 0,$$

which implies that CG converges in N iterations in exact arithmetic. Furthermore, if there are only k distinct eigenvalues, then the CG iteration terminates in at most k iterations.

Condition number estimation

During CG iterations it is possible to get an estimate of the condition number of the matrix A by using the tridiagonal Hessenberg matrix T_m and the recurrence relation for determinant of tridiagonal matrices. Suppose the characteristic polynomial of T_j is given by $p_{T_j}(\lambda)$, then

$$\begin{aligned} p_{T_0}(\lambda) &= 1, \\ p_{T_1}(\lambda) &= \lambda - \delta_1, \\ p_{T_2}(\lambda) &= (\lambda - \delta_2) p_{T_1}(\lambda) - \eta_2^2, \\ p_{T_3}(\lambda) &= (\lambda - \delta_3) p_{T_2}(\lambda) - \eta_3^2(\lambda - \delta_1), \\ &\vdots \end{aligned}$$

Hence, the characteristic polynomial of T_j can be written as

$$p_{T_j}(\lambda) = (\lambda - \delta_j) p_{T_{j-1}}(\lambda) - \eta_j^2 p_{T_{j-2}}. \tag{2.42}$$

Now the condition number of A is defined as

$$\kappa(A) = ||A||_2 ||A^{-1}||_2 = \max_{x \in \mathbb{R}^n, x \neq 0} \frac{||Ax||_2}{||x||_2} \max_{x \in \mathbb{R}^n, x \neq 0} \frac{||A^{-1}x||_2}{||x||_2}.$$

Similarly, the condition number of T_j is defined as

$$\kappa(T_j) = ||T_j||_2 ||T_j^{-1}||_2 = \max_{x \in \mathbb{R}^j, x \neq 0} \frac{||T_j x||_2}{||x||_2} \max_{x \in \mathbb{R}^j, x \neq 0} \frac{||T_j^{-1}x||_2}{||x||_2}.$$

Note that

$$\begin{aligned} \max_{x \in \mathbb{R}^j, x \neq 0} \frac{||T_j x||_2}{||x||_2} &= \max_{x \in \mathbb{R}^n, x \neq 0} \frac{||V_j^T A V_j x||_2}{||x||_2} \\ \text{orthogonality of } V_j \rightarrow &= \max_{x \in \mathbb{R}^n, x \neq 0} \frac{||V_j (V_j^T A V_j x)||_2}{||V_j x||_2} \\ &= \max_{x \in \mathbb{R}^j, x \neq 0} \frac{||A V_j x||_2}{||V_j x||_2} \\ &= \max_{\tilde{x} \in \mathcal{K}_j, \tilde{x} \neq 0} \frac{||A\tilde{x}||_2}{||\tilde{x}||_2} \\ &\approx ||A||_2, \end{aligned}$$

and similarly for the inverse. Presumably, the estimate will get better as j increases and the Krylov subspace $\mathcal{K}_j$ gets closer to the eigenspace of A. To do ►*Find proof of this*◄.

Now, combining this with the recurrence relation for the characteristic polynomial of T_j in equation (2.42), we have a way of estimating the condition number of A. In particular, at any iteration j of the CG algorithm, let $\tilde{\lambda}_{\text{max}}$ and $\tilde{\lambda}_{\text{min}}$ be the maximum and minimum zeroes of p_{T_j}. Then, the condition number of A is approximately given by

$$\kappa(A) \approx \frac{\tilde{\lambda}_{\text{max}}}{\tilde{\lambda}_{\text{min}}}. \tag{2.43}$$

Influence of eigenvalue distribution on convergence

In the derivation of the convergence rate of the CG algorithm in 2.6, we used the Chebyshev polynomial to bound the error. However, we can find an expression of the error provided the eigendecomposition of A is available. Suppose $A = VDV^T$, then $r(A) = I - Aq(A) = V(I - Dq(D))V^T = Vr(D)V^T$. Also note that $e_0 = x^* - x_0 = A^{-1}b - x_0 = A^{-1}r_0$. As seen in section 2.4.8, the error of the m^{th} iterate of the CG algorithm is given by

$$||e_m||_A^2 = ||r_m(A)\epsilon_0||_A^2,$$

and

$$\begin{aligned}||r_m(A)\epsilon_0||_A^2 &= \epsilon_0^T r_m(A)^T A r_m(A)\epsilon_0\\ &= \epsilon_0^T V r_m(D)V^T V D V^T V r_m(D) V^T \epsilon_0\\ &= (V^T\epsilon_0)^T r_m(D) D r_m(D) V^T \epsilon_0.\end{aligned}$$

We also have

$$\begin{aligned}V^T\epsilon_0 &= V^T A^{-1} r_0\\ &= V^T V D^{-1} V^T r_0\\ &= D^{-1}\rho_0,\end{aligned}$$

where $\rho_0 = V^T r_0$ is the initial residual in the eigenvector basis of A. Therefore,

$$\begin{aligned}||r_m(A)\epsilon_0||_A^2 &= \rho_0^T D^{-1} r_m(D) D r_m(D) D^{-1}\rho_0\\ &= \rho_0^T r_m(D) D^{-1} r_m(D)\rho_0\\ &= \sum_{i=1}^n \frac{r_m(\lambda_i)^2}{\lambda_i}\rho_{0,i}^2,\end{aligned}$$

which gives

$$||e_m||_A^2 = \sum_{i=1}^n \frac{r_m(\lambda_i)^2}{\lambda_i}\rho_{0,i}^2. \tag{2.44}$$

To obtain the residual polynomial r_m, we can use the recurrence relation between the Lanczos vectors and expressions for the Hessenberg matrix coefficients in equations (2.39) and (2.40). In particular,

$$\begin{aligned}\frac{1}{\eta_{j+1}}v_j &= Av_j - \delta_j v_j - \eta_j v_{j-1}\\ &= p_{j+1}(A)v_1,\end{aligned}$$

where we define $p_{-1}(A) = 0, p_0(A) = I$. This gives

$$\begin{aligned}\eta_{j+1}p_{j+1}(A)v_1 &= Av_j - \delta_j v_j - \eta_j v_{j-1},\\ &= \left(Ap_j(A) - \delta_j p_j(A) - \eta_j p_{j-1}(A)\right)v_1,\end{aligned}$$

and therefore

$$p_{j+1}(A) = \frac{1}{\eta_{j+1}}\left((A - \delta_j)p_j(A) - \eta_j p_j(A)\right). \tag{2.45}$$

Furthermore, we have the following relation between the residual polynomial and the Lanczos polynomial [6, Section 3.2]

$$r_j(A) = (I - Aq_{j-1}(A))r_0 = \frac{p_j(A)}{p_j(0)}r_0. \tag{2.46}$$

This gives a way of calculating the residual polynomial r_m and thereby the error of the m^{th} iterate of the CG algorithm.

Additionally, the coefficients c_i of the solution polynomial q_m in equation (2.26) can be calculated. First we introduce a function that extracts the coefficients of a polynomial p

Definition 2.6. Let $p(t) = \sum_{i=0}^{n} c_i t^i$ be a polynomial of degree n. Then, the function coeff$(p; i)$ extracts the i^{th} coefficient of p such that coeff$(p; i) = c_i$.

Now using equation (2.46), we can write the solution polynomial as

$$Aq_{m-1}(A) = I - r_m(A)$$
$$r_m(\mathbf{0}) = I \implies A \sum_{i=1}^{m-1} c_{i-1} A^i = -\sum_{i=1}^{m} \text{coeff}(r_m; i) A^i,$$

which implies

$$c_i = -\text{coeff}(r_m; i+1), \quad i = 0, 1, \ldots, m-1. \tag{2.47}$$

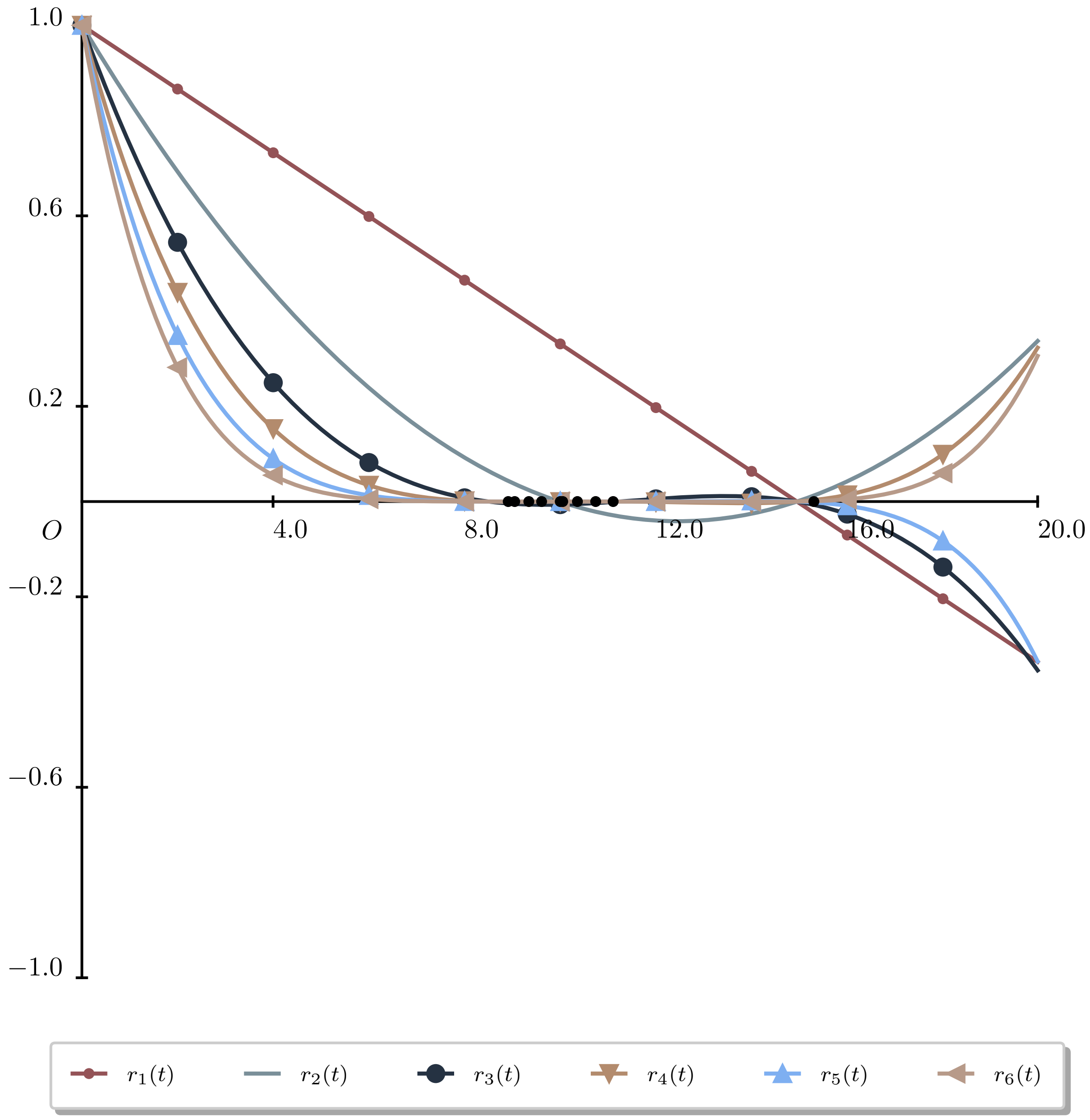


Figure 2.2: Residual polynomials resulting from successive CG iterations

The behavior of the residual polynomials is crucial for understanding the convergence properties of the CG method. In particular, the distribution of the eigenvalues of A significantly affects the convergence rate, as illustrated in figure 2.3.

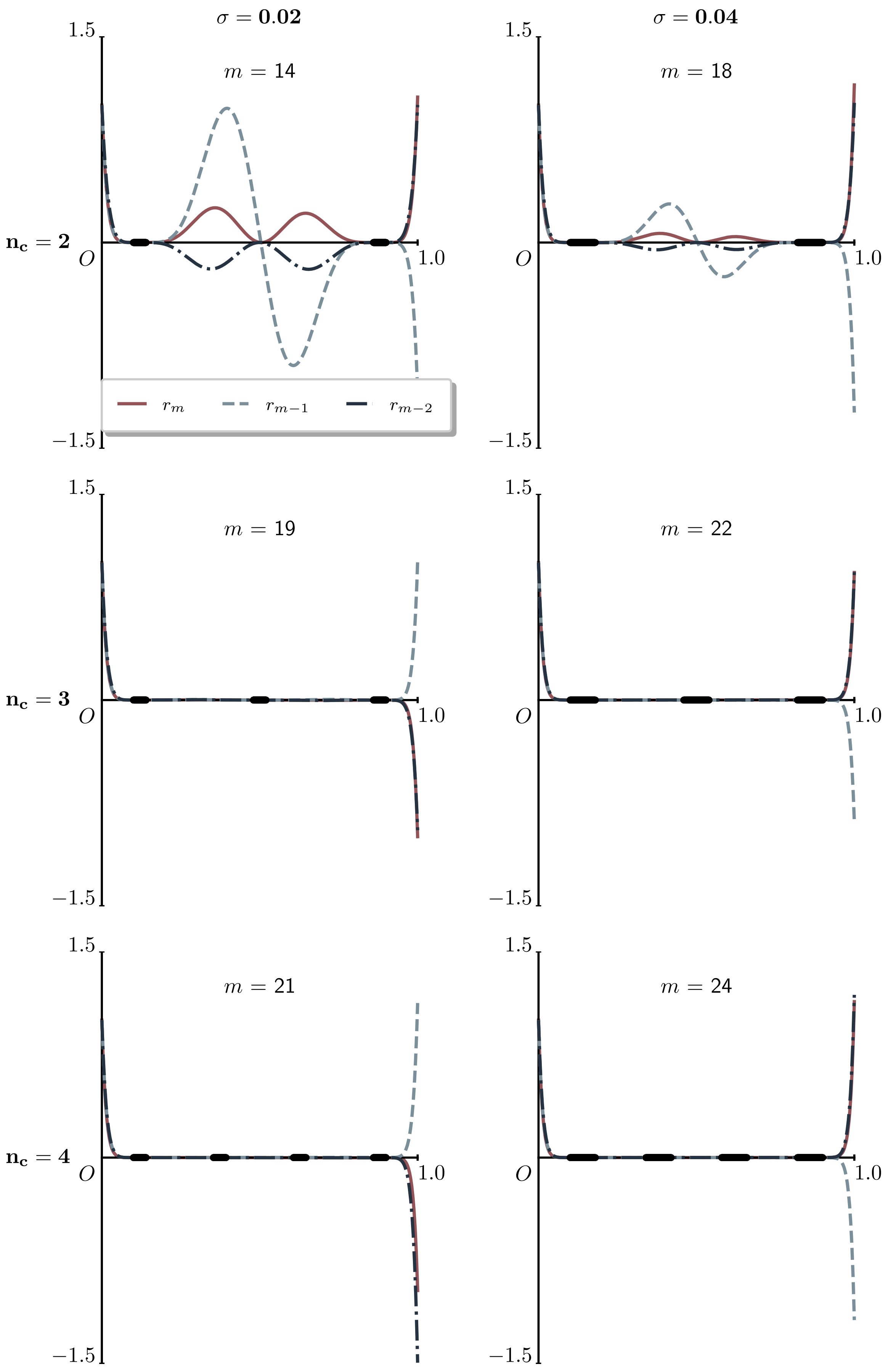


Figure 2.3: Plots of the last three CG residual polynomials for different eigenvalue distributions. n_c indicates the number of clusters and σ is the width of the cluster. The size of the system n and the condition number $\kappa(A)$ are kept constant.

Hence, the number of iterations required for convergence depends on the specific clustering of the eigenvalues, as pointed out for example in Kelley, Section 2.3.

From the behavior exhibited in figure figure 2.3 as well as from theorem 2.3 we can reason what the best and worst possible spectra for CG convergence are. That is, the best possible spectrum is one where eigenvalues are tightly clustered around distinct values, while the worst possible spectrum is one where the eigenvalues are evenly distributed across the whole range of the spectrum. This is illustrated in figure 2.4.

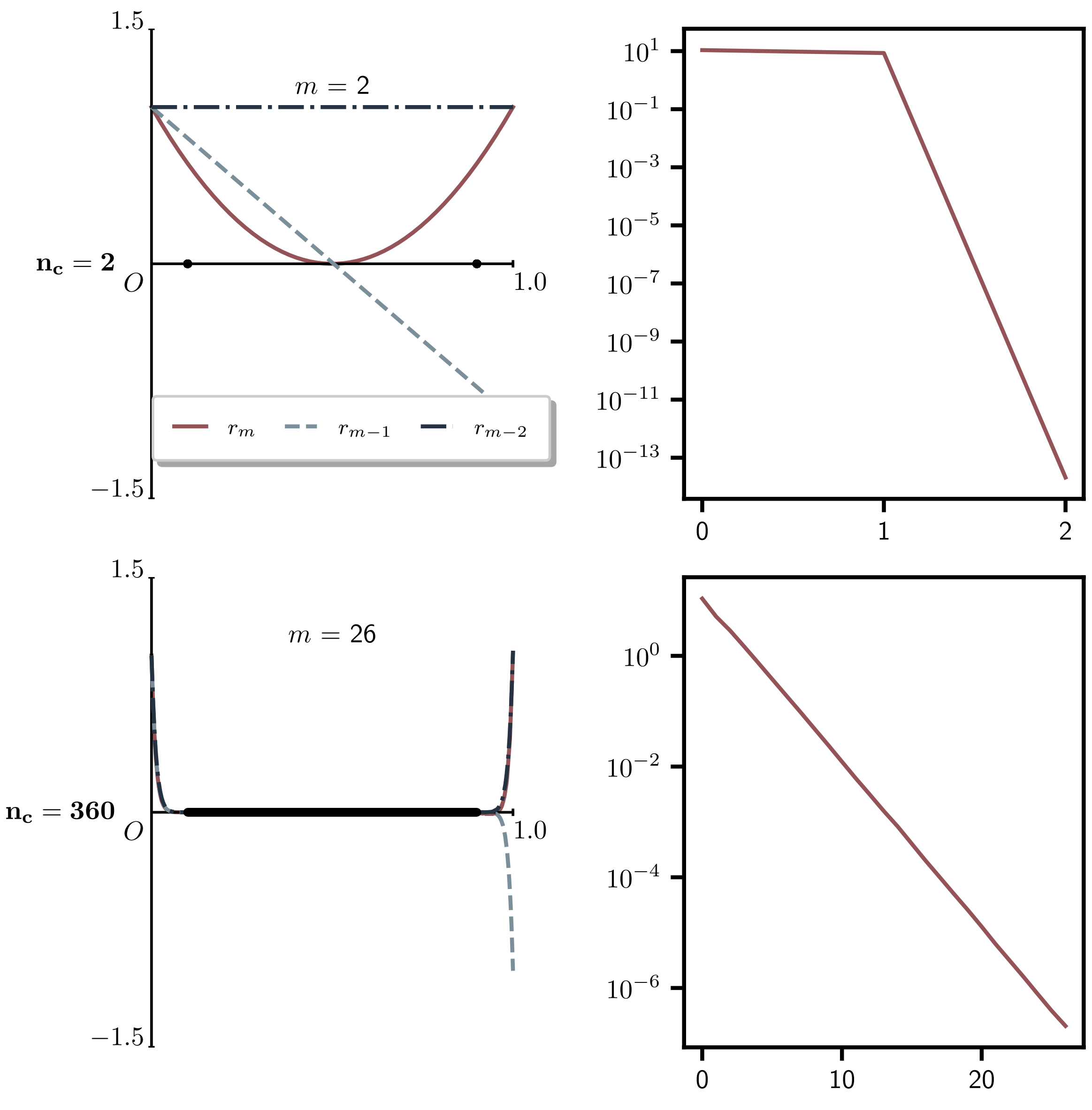


Figure 2.4: Best and worst possible spectra for CG convergence

2.4.9. Preconditioning

Suppose M is some SPD preconditioner, then variants of CG can be derived by applying M to the system of equations. The three main approaches are

i left

$$M^{-1}Ax = M^{-1}b$$

ii right

$$\begin{aligned} AM^{-1}u &= M^{-1}b \\ x &= M^{-1}u; \end{aligned}$$

iii symmetric or split

$$\begin{aligned} M &= LL^T \\ x &= L^{-T}u \\ L^{-1}AL^{-T}u &= L^{-1}b. \end{aligned}$$

Furthermore, all these variants are mathematically equivalent in some sense. Indeed, for the cases item preconditioner-type i and item preconditioner-type ii, we can rewrite the CG algorithm using the $M-$ or $M^{-1}-$inner products, respectively. In either case the iterates are the same. For instance for the left preconditioned CG, we define $z_j = M^{-1}r_j$. Note that $M^{-1}A$ is self-adjoint with respect to the $M-$inner product, that is

$$(M^{-1}Ax, y)_M = (Ax, y) = (x, Ay) = (x, M^{-1}Ay)_M.$$

We use this to get a new expression for α_j. To that end, we write

$$\begin{aligned} 0 &= (r_{j+1}, r_j)_M \\ &= (z_{j+1}, r_j) \\ &= (z_j - \alpha_j M^{-1}Ap_j, M^{-1}r_j)_M \\ &= (z_j, M^{-1}r_j)_M - \alpha_j(M^{-1}Ap_j, M^{-1}r_j)_M \\ &= (z_j, z_j)_M - \alpha_j(M^{-1}Ap_j, z_j)_M \end{aligned}$$

and therefore

$$\alpha_j = \frac{(z_j, z_j)_M}{(M^{-1}Ap_j, z_j)_M}.$$

Using $p_{j+1} = z_{j+1} + \beta_j p_j$ and A-orthogonality of the search directions with respect to $M-$norm $(Ap_j, p_k)_M = 0$ (), we can write

$$\alpha_j = \frac{(z_j, z_j)_M}{(M^{-1}Ap_j, p_j)_M}.$$

Similarly, we can derive the equivalent expression of equation (2.38) as

$$\beta_j = \frac{(z_{j+1}, z_{j+1})_M}{(z_j, z_j)_M}.$$

This gives the left preconditioned CG algorithm in algorithm 12.

Algorithm 12 Left preconditioned CG [8, Algorithm 9.1]

```
$r_0 = b - Ax_0$, $z_0 = M^{-1}r_0$, $p_0 = z_0$, $\beta_0 = 0$
for $j = 0, 1, 2, \ldots, m$ do
    $\alpha_j = (z_j, z_j)_M/(M^{-1}Ap_j, p_j)_M = (r_j, z_j)/(Ap_j, p_j)$
    $x_{j+1} = x_j + \alpha_j p_j$
    $r_{j+1} = r_j - \alpha_j Ap_j$
    $z_{j+1} = M^{-1}r_{j+1}$
    $\beta_j = (z_{j+1}, z_{j+1})_M/(z_j, z_j)_M = (r_{j+1}, z_{j+1})/(r_j, z_j)$
    $p_{j+1} = z_{j+1} + \beta_j p_j$
end for
```

Furthermore it can be shown that the iterates of CG applied to the system with item preconditioner-type iii results in identical iterates [8, Algorithm 9.2].

Examples of preconditioners

$q(A)$ **polynomial preconditioner** If the spectrum of A is approximately known, and we have a way of approximating the solution polynomial from equation (2.47), then we can use this polynomial as a preconditioner. This is known as the $q(A)$ polynomial preconditioner. That is

$$\begin{aligned} M^{-1} &= q(A), \\ M^{-1}A &= M^{-1}b. \end{aligned}$$

2.5. Generalized Minimal Residual Method

This section is largely based on Section 6.5 from the book by Saad about iterative methods.

CG is limited to symmetric positive definite matrices, but the Generalized Minimal Residual Method (GMRES) can be used for general matrices.

GMRES is a Krylov subspace projection method with $\mathcal{K} = \mathcal{K}_m$ and $\mathcal{L} = A\mathcal{K}_m$. This method minimizes the residual norm, as is described in section 2.2.2 with equation (2.6) and later in section 2.4.1 with the second instance of item CG-type i.

Furthermore, GMRES is similar to Arnoldi's method for linear systems (see sections 2.4.3 and 2.4.4). The only difference being that instead of performing a system solve after determining the Hessenberg matrix H_m, GMRES performs a least squares solve to find vector that minimizes the residual norm. To show this note that $x_m = x_0 + V_m y$ and consider

$$\begin{aligned} b - Ax_m &= b - A(x_0 + V_m y) \\ &= r_0 - AV_m y \\ \text{equation (2.28b)} \rightarrow &= \beta v_1 - V_{m+1}\bar{H}_m y \\ &= V_{m+1}(\beta e_1 - \bar{H}_m y) \end{aligned}$$

Now, using orthonormality of V_{m+1}, we can write

$$\begin{aligned} y &= \underset{y}{\text{argmin}}||b - Ax_m|| \\ &= \underset{y}{\text{argmin}}||V_{m+1}(\beta e_1 - \bar{H}_m y)|| \\ &= \underset{y}{\text{argmin}}||\beta e_1 - \bar{H}_m y||, \end{aligned}$$

which is a least squares problem.

Even though GMRES is the same as algorithm 7 with a least squares solve instead of a system solve, the algorithm is presented in algorithm 13 for completeness.

Algorithm 13 GMRES [8, Algorithm 6.9]

```
1: Compute $r_0 = b - Ax_0$, $\beta = ||r_0||_2$ and $v_1 = r_0/\beta$
2: Define $H_m = \{0\}$
3: Define $V_1 = \{v_1\}$
4: for $j = 1, 2, \ldots, m$ do
5:     $w_j = Av_j$
6:     for $i = 1, 2, \ldots, j$ do
7:         $h_{ij} = (w_j, v_i)$ and store $h_{ij}$ in $H_m$
8:         $w_j = w_j - h_{ij}v_i$
9:     end for
10:    $h_{j+1,j} = ||w_j||_2$
11:    if $h_{j+1,j} = 0$ then
12:        $m = j$
13:        break
14:    end if
15:    $v_{j+1} = w_j/h_{j+1,j}$ and store $v_{j+1}$ into $V_{j+1}$
16: end for
17: Construct Hessenberg matrix $\bar{H}_m \in \mathbb{R}^{(m+1)\times m}$
18: Solve the least squares problem $\min_{y\in\mathbb{R}^m} ||\beta e_1 - \bar{H}_m y||_2$
19: $x_m = x_0 + V_m y_m$
```

2.5.1. GMRES implementation details

Algorithm 13 does not explicitly produce successive approximations x_k as in algorithm 7. Hence, it is cumbersome to check for convergence. Moreover, employing a monolithic least squares solve at the end of the algorithm does not allow the use of intermediary results like in algorithm 7 or algorithm 10.

To tackle these issues, one can apply a rotation to the Hessenberg matrix at each iteration in order to make it upper triangular. The i^{th} rotation matrix is of the form [8, Equation 6.34]

$$\Omega_i = \begin{pmatrix} 1 &&&&&&& \\ & \ddots &&&&&& \\ && 1 &&&&& \\ &&& c_i & s_i &&& \\ &&& -s_i & c_i &&& \\ &&&&& 1 && \\ &&&&&& \ddots & \\ &&&&&&& 1 \end{pmatrix}, \tag{2.48}$$

where [8, Equation 6.37]

$$\begin{aligned} c_i &= \frac{h^(i-1)_{ii}}{\sqrt{h^(i-1)^2_{ii} + h^2_{i+1,i}}}, \\ s_i &= \frac{h_{i+1,i}}{\sqrt{h^(i-1)^2_{ii} + h^2_{i+1,i}}}, \end{aligned}$$

and $h^(i-1)_i i$ is the i^{th} diagonal element of the Hessenberg matrix after $i-1$ iterations and rotations $\bar{H}^(i-1)_{i-1}$. That is

$$\bar{H}^(k)_k = \Omega_k \dots \Omega_1 \bar{H}_k = Q_k \bar{H}_k,$$

where

$$Q_k = \Omega_k \dots \Omega_1.$$

Define

$$\bar{R}_k = \bar{H}^(k)_k, \tag{2.49a}$$

$$\bar{g}_k = Q_k(\beta e_1) = (\gamma_1, \dots, \gamma_{k+1})^T. \tag{2.49b}$$

Note that

$$\det(\Omega_i) = 1, \quad \forall i,$$

and, therefore, $\det(Q_k) = 1$. This implies that Q_k is unitary. Consequently,

$$\min \|\beta e_1 - \bar{H}_m y\| = \min \|Q_m(\beta e_1 - \bar{H}_m y)\| = \min \|\bar{g}_m - \bar{R}_m y\|.$$

The following theorem is then obtained [8, Proposition 6.9].

Theorem 2.7. Let $m \leq n$ and $\Omega_i, i = 1, \dots, m$ be the rotation matrices used to transform $\bar{H}_m$ into an upper triangular form. Denote by $\bar{R}_m, \bar{g}_m = (\gamma_1, \dots, \gamma_{m+1})^T$ the resulting matrix and right-hand side, as defined by equations (2.49a) and (2.49b) and by R_m, g_m the $m \times m$ upper triangular matrix and m-dimensional vector obtained from $\bar{R}_m, \bar{g}_m$ by deleting their last row and component respectively. Then,

i The rank of AV_m is equal to the rank of R_m. In particular, if $r_{mm} = 0$ then A must be singular.

ii The vector y_m which minimizes $\|\beta e_1 - \bar{H}_m y\|_2$ is given by

$$y_m = R_m^{-1} g_m.$$

iii The residual vector at step m satisfies

$$b - Ax_m = V_{m+1}(\beta e_1 - \bar{H}_m y_m) = V_{m+1} Q_m^T (\gamma_{m+1} e_{m+1}),$$

and, as a result,

$$\|b - Ax_m\|_2 = |\gamma_{m+1}|.$$

Item rotated GMRESiii is particularly useful for checking convergence, as [8, Equation 6.47]

$$\gamma_{k+1} = -s_j \gamma_j.$$

Therefore,

$$\|r_k\| = |s_k||\gamma_k|. \tag{2.50}$$

2.5.2. Relation between GMRES and CG

To do ►*Talk about how the iterates of the (rotated) GMRES method are related to the iterates of the CG method.*◄

2.6. Nonlinear systems

We start out with a general formulation of a nonlinear system of equations. Define the non-linear operator $F : \mathbb{R}^N \to \mathbb{R}^N$ as

$$F(x) = 0, \tag{2.51}$$

Some general restrictions are placed on F, which can be weakened depending on the method

i Existence: 2.51 has a solution x^*.

ii Lipschitz continuity: $F' : \Omega \to \mathbb{R}^{N\times N}$ is Lipschitz continuous with constant γ.

iii Non-singularity: $F'(x^*)$ is nonsingular.

Define $\mathbb{B}(r) = \{x \in \mathbb{R}^N | ||e|| < r\}$ to be the ball of radius r centered at the one of the roots of F with $e = x - x^*$. The following theorem then follows from the restrictions above [5, lemma 4.3.1].

Theorem 2.8. Assume that the standard assumptions items F-criterium i to F-criterium iii hold. Then there is $\delta > 0$ so that for all $x \in \mathcal{B}(\delta)$

$$\|F'(x)\| \leq 2\,\|F'(x^*)\| \tag{2.52}$$

$$\left\|F'(x)^{-1}\right\| \leq 2\left\|F'(x^*)^{-1}\right\| \tag{2.53}$$

and

$$\frac{\|e\|}{2\left\|F'(x^*)^{-1}\right\|} \leq \|F(x)\| \leq 2\,\|F'(x^*)\|\,\|e\| \tag{2.54}$$

Various kinds of convergence known as ‘q-type’ for iteration techniques can be defined for nonlinear systems. The following definitions are taken from [5, Section 4.3]. If $x_k \to x^*$, then for k sufficiently large, there exists convergence that is:

i q-linear with q-factor $\nu \in (0, 1)$

$$||x_{k+1} - x^*|| \leq \nu ||x_k - x^*||,$$

ii q-superlinear

$$\lim_{k\to\infty} \frac{||x_{k+1} - x^*||}{||x_k - x^*||} = 0,$$

iii q-quadratic with constant $C > 0$

$$||x_{k+1} - x^*|| \leq C||x_k - x^*||^2,$$

iv q-quadratic with q-order $\mu > 1$ and constant $C > 0$

$$||x_{k+1} - x^*|| \leq C||x_k - x^*||^{\mu}.$$

For instance, Fixed Point (FP) or Richardson iteration method is q-linear with constant equal to the Lipschitz constant of the operator [5, Theorem 4.2.1]. Moreover, assuming items F-criterium i to F-criterium iii hold, then Newton’s method is q-quadratic with $C = \gamma\|F'(x^*)\|$ provided that the initial guess is close enough to the solution [5, Theorem 5.1.2]. Additionally, The chord method is q-linear with q-factor $\nu = C_C\|e_0\|$ [5, Theorem 5.4.2], where $C_C = \bar{C}(1+2\gamma)$ and $\bar{C} = C+16\|F(x^*)^{-1}\|^2\|F(x^*)\|+4\|F(x^*)^{-1}\|$ [5, Theorem 5.4.1].

Moreover, it can occur that the operator F introduces errors that are independent of the approximation error. In this case a different kind of convergence ‘r-type’ is used to describe the behavior of the method [5, Definition 4.1.3]

Definition 2.7. Let $\{x_n\} \subset R^N$ and $x^* \in R^N$. Then $\{x_n\}$ converges to x^* r-(quadratically, super-linearly, linearly) if there is a sequence $\{\xi_n\} \subset R$ converging q-(quadratically, super-linearly, linearly) to zero such that

$$\|x_n - x^*\| \leq \xi_n$$

We say that $\{x_n\}$ converges r-super-linearly with r-order $\mu > 1$ if $\xi_n \to 0$ q-super-linearly with q -order μ.

2.6.1. Inexact Newton-Raphson method

The Newton-Raphson (NR) iteration or step s for the k^{th} iterate such that $x_{k+1} = x_k + s$ given is exactly defined as

$$F'(x_k)s = -F(x_k).$$

However, for large sparse systems it is computationally expensive to solve the linear system exactly. Instead, we can solve the linear system approximately by using an iterative method, like CG or GMRES (see sections 2.4 and 2.5). However, this leads to an approximate newton step, and thus the Inexact Newton-Raphson (INR) method is defined as follows [5, Equation 6.1]:

$$\|F'(x_k)s + F(x_k)\| \leq \eta_k \|F(x_k)\|,$$

where η_k is an inexactness parameter.

Convergence of INR

The following theorem is taken from [5, Theorem 6.1.2] and uses theorem 2.8 to state that the inexact Newton method converges q-linearly or faster

Theorem 2.9. Let items F-criterium i to F-criterium iii hold. Then there are δ and $\bar{\eta}$ such that if $x_0 \in \mathcal{B}(\delta), \{\eta_k\} \subset [0, \bar{\eta}]$ and $\bar{\eta}$ is such that $C_I < 1$ then the inexact Newton iteration converges q-linearly to x^* with constant $C_I = (C + 4\kappa(F'(x^*)))\,(\bar{\eta} + \delta)$. Moreover,

1. if $\eta_k \to 0$ the convergence is q-superlinear,
2. if $\eta_n \leq C_\eta \left\|F\left(x_n\right)\right\|^p$ for some $C_\eta > 0$ the convergence is q-superlinear with q -order $1 + p$.

Additionally, if the constraint on $\bar{\eta}$ is relaxed to $0 \leq \eta_k \leq \bar{\eta} < 1$ a result similar to theorem 2.9 can be obtained [5, Theorem 6.1.4]. In this case, the convergence is with respect to the norm $\|\cdot\|_*$, which is defined as $\|\cdot\|_* = \|F'(x^*)\cdot\|$.

Theorem 2.10. Let items F-criterium i to F-criterium iii hold. Then there is δ such that if $x_0 \in \mathcal{B}(\delta), \{\eta_k\} \subset [0, \eta]$ with $\eta \leq \bar{\eta} < 1$, then the inexact Newton iteration converges q-linearly with respect to $\|\cdot\|_*$ to x^* with constant $\bar{\eta}$. Moreover,

1. if $\eta_k \to 0$ the convergence is q-superlinear,
2. if $\eta_n \leq C_\eta \left\|F\left(x_n\right)\right\|^p$ for some $C_\eta > 0$ the convergence is q-superlinear with q -order $1 + p$.

Furthermore, errors made in the calculation of F do not directly slow down the convergence of the method. However, they do introduce a separate error term resulting in r- instead q-type convergence [5, Theorems 5.4.6, 6.1.5, 6.1.6].

Implementation of INR

As mentioned in the beginning of this section, the linear system in the INR method is solved approximately. If one of the Krylov subspace method is used, then for each iteration thereof the action of $F'(x_k)$ on a vector is required. That is, $F'(x_k)v$, where v is a basis vector of the Krylov subspace (see algorithms 7 and 13), or provided $F'(x_k)$ is SPD, $F'(x_k)p$ where p is the search direction in the CG method (see algorithm 11). This action of $F'(x_k)$ on a vector v can be approximated in two ways

1. By using a finite difference approximation to get an approximation of the Jacobian matrix $F'(x_k)$ and, subsequently applying said matrix directly.
2. By using a directional derivative of F at x_k in the direction of the vector v. For instance, the first-order accurate forward difference approximation of the directional derivative is given by

$$D_h F(x; v) = \begin{cases} 0, & \text{v = 0,} \\ \|v\| \dfrac{F(x + h\|x\| v/\|v\|) - F(x)}{h\|x\|}, & \text{v,x} \neq 0, \\ \|v\| \dfrac{F(hv/\|v\|) - F(x)}{h}, & \text{v} \neq 0\text{, x = 0.} \end{cases} \tag{2.55}$$

In the remainder of this section a general Krylov subspace method with corresponding operator G is denoted by Krylov(r_0, G). For instance, Krylov$(r_0, F'(x_k))$ is the Krylov subspace method with the exact jacobian $F'(x_k)$ and initial residual $r_0 = F(x_k)$, or Krylov$(r_0, D_h F(x_k; \cdot))$ is the Krylov subspace method with the directional derivative $D_h F(x_k; \cdot)$ and initial residual r_0. Additionally, note that any of the Krylov methods from algorithms 7, 11 and 13 produce intermediate residuals ρ_j (see equations (2.30), (2.35) and (2.50)), which can be used to check for convergence.

Algorithm 14 describes a general recipe for solving nonlinear systems with INR

Algorithm 14 INR-Krylov[5, Algorithms 6.2.1 and 6.3.1]

```
$r_0$ = $\|F(x_0)\|/\sqrt{N}$, $k = 0$
while $\|F(x_k)\| >$ tol do
    $k = k + 1$
    Select $\eta$ (see section 2.6.1)
    Perform Krylov$(-F(x_k), G)$ until $\rho_j \leq \eta\|F(x_k)\|$ or until $j = j_{\text{max}}$ to get $s_k$
    $x_{k+1} = x_k + s_k$
end while
```

where the tolerance tol $= \tau_a + \tau_r r_0$ (see section 2.6.1) and j_{max} is the maximum number of iterations for the Krylov method. The convergence of the INR-Krylov method is discussed in the next section.

INR-Krylov convergence

The approximation of the jacobian introduces error. The next theorems are the equivalents of theorems 2.9 and 2.10 [5, Theorem 6.2.1].

Theorem 2.11. Let items F-criterium i to F-criterium iii hold. Then there are δ, σ such that if $x_0 \in \mathcal{B}(\delta)$ and the sequences $\{\eta_k\}$ and $\{\eta_k\}$ satisfy

$$\sigma_k = \eta_k + C_{\text{INR-K}} h_k \leq \sigma,$$

then the forward difference INR-Krylov with iteration $G = D_{h_k}(x_k; \cdot)$ converges q-linearly with constant $C_{\text{INR-K}} = 4\gamma(\|x^*\| + \delta)(1 + \eta)\|F'(x^*)^{-1}\|$. Moreover,

1. if $\sigma_k \to 0$ the convergence is q-superlinear,
2. if $\sigma_k \leq C_\eta \|F(x_n)\|^p$ for some $C_\eta > 0$ the convergence is q-superlinear with q -order $1 + p$.

Similarly, the constraint on the sequence σ_k can be somewhat relaxed resulting in the next theorem

Theorem 2.12. Let items F-criterium i to F-criterium iii hold. Then there are δ, σ such that if $x_0 \in \mathcal{B}(\delta)$ and the sequences $\{\eta_k\}$ and $\{\eta_k\}$ satisfy

$$\sigma_k = \eta_k + C_{\text{INR-K}} h_k \leq \sigma,$$

then the forward difference inexact NR iteration converges q-linearly with respect $\|\cdot\|_*$. Moreover,

1. if $\sigma_k \to 0$ the convergence is q-superlinear,
2. if $\sigma_k \leq C_\eta \|F(x_n)\|^p$ for some $C_\eta > 0$ the convergence is q-superlinear with q -order $1 + p$.

Note that in the case that h_k is around machine precision $\sigma_k \approx \eta_k$, provided that η_k is not too small.

INR-Krylov inexactness parameter

The inexactness parameter η_k can be chosen in various ways. For instance, it can be chosen to be a constant, a function of the residual, or a function of the iteration number. The following is taken from [5, Equations 6.19 and 6.20] and provides a guideline for choosing η_k. Let

$$\eta_k^A = \gamma \frac{\|F(x_k)\|^2}{\|F(x_{k-1})\|^2},$$

then define

$$\eta_n^B = \begin{cases} \eta_{\max}, & k = 0, \\ \min\left(\eta_{\max}, \eta_k^A\right), & k > 0, \gamma\eta_{k-1}^2 \leq .1 \\ \min\left(\eta_{\max}, \max\left(\eta_k^A, \gamma\eta_{k-1}^2\right)\right), & k > 0, \gamma\eta_{k-1}^2 > .1 \end{cases}$$

and finally set

$$\eta_k = \min\left(\eta_{\max}, \max\left(\eta_k^B, \frac{1}{2}\tau_t/\|F(x_k)\|\right)\right), \tag{2.56}$$

where

$$\tau_t = \tau_a + \tau_r\|F(x_0)\|.$$

The parameters $\eta_{\max}, \tau_a, \tau_r$ are user-defined constants. The parameter γ is the Lipschitz constant of the operator F. For instance, for a 2D finite difference problem one can set $\tau_a = \tau_r = h^2$, where h is the grid constant.

This choice of η_k is based on the assumption that the error in the approximation of the jacobian is proportional to the error in the approximation of the residual. The parameter η_k is then chosen to be the minimum of the maximum of the two errors and a user-defined constant. The parameter $\eta_{\max}$ is used to prevent the inexactness parameter from becoming too large. The parameter τ_t is used to prevent the inexactness parameter from becoming too small. The parameter τ_a is used to prevent the inexactness parameter from becoming too small in the beginning of the iteration, while τ_r does the same for the end of the iteration. This prevents so-called ‘safeguarding’ during and ‘over-solving’ at the end of the iteration [5, Section 6.3].

Bibliography

Alves, F. A. C. S., Heinlein, A., & Hajibeygi, H. (2024). A computational study of algebraic coarse spaces for two-level overlapping additive schwarz preconditioners. URL.

Dohrmann, C. R., Klawonn, A., & Widlund, O. B. (2008). A family of energy minimizing coarse spaces for overlapping schwarz preconditioners. *Domain Decomposition Methods in Science and Engineering XVII*, *60*, 247–254.

Dolean, V., Jolivet, P., & Nataf, F. (2015). *An introduction to domain decomposition methods*. Society for Industrial; Applied Mathematics. https://doi.org/10.1137/1.9781611974065

Graham, I. G., Lechner, P. O., & Scheichl, R. (2007). Domain decomposition for multiscale pdes. *Numerische Mathematik*, *106*, 589–626. https://doi.org/10.1007/s00211-007-0074-1

Kelley, C. T. (1995). *Iterative methods for linear and nonlinear equations*. Society for Industrial; Applied Mathematics. https://doi.org/10.1137/1.9781611970944

Meurant, G., & Strakoš, Z. (2006). The lanczos and conjugate gradient algorithms in finite precision arithmetic. *Acta Numerica*, *15*, 471–542. https://doi.org/10.1017/S096249290626001X

Rheinbach, O. (2018). Multiscale coarse spaces for overlapping schwarz methods based on the acms space in 2d (L. R. (Ronny Ramlau, Ed.) [Online available: https://epub.oeaw.ac.at/?arp=0x0038c0cb - Last access:7.2.2025], 156–182. URL.

Saad, Y. (2003). *Iterative methods for sparse linear systems* (Second). Society for Industrial; Applied Mathematics. https://doi.org/10.1137/1.9780898718003