NUMERICAL LINEAR ALGEBRA

Claudio Tessa

December 25, 2025

Contents

1 Introduction

1.1 Basic matrix decomposition

1.1.1 LU factorization with (partial) pivoting

If $A \in \mathbb{R}^{n \times n}$ is nonsingular (invertible), then

$$PA = LU$$

- P is a permutation matrix
- L is unit lower matrix
- U is upper triangular

Linear system solution:

$$A\mathbf{x} = \mathbf{b}$$

1. Factor $PA = LU$ (expensive, $O(n^3)$)
2. Solve $L\mathbf{y} = P\mathbf{b}$ (lower triangular system, $O(n^2)$)
3. Solve $U\mathbf{x} = \mathbf{y}$ (upper triangular system, $O(n^2)$)

1.1.2 Cholesky decomposition

If $A \in \mathbb{R}^{n \times n}$ is symmetric ($A^T = A$) and positive definite ($\mathbf{z}^T A\mathbf{z} > 0$ for all $\mathbf{z} \neq \mathbf{0}$), then

$$A = L^T L$$

where L is lower triangular (with positive entries on the diagonal).

Linear system solution

$$A\mathbf{x} = \mathbf{b}$$

1. Factor $A = L^T L$ (expensive, $O(N^3)$)
2. Solve $L^T\mathbf{y} = \mathbf{b}$ (lower triangular system, $O(n^2)$)
3. Solve $L\mathbf{x} = \mathbf{y}$ (upper triangular system, $O(n^2)$)

1.1.3 QR decomposition

If $A \in \mathbb{R}^{n \times n}$ is nonsingular (invertible), then

$$A = QR$$

- Q is an orthogonal
- R is upper triangular

Linear system solution:

$$A\mathbf{x} = \mathbf{b}$$

1. Factor $A = QR$ (expensive, $O(n^3)$)

2. Multiply $\mathbf{c} = Q^T\mathbf{b}$ ($O(n^2)$)
3. Solve $R\mathbf{x} = \mathbf{c}$ (lower triangular system, $O(n^2)$)

1.2 Sparse matrices

A sparse matrix is a matrix in which most elements are zero. Roughly speaking, given $A \in \mathbb{R}^{n\times n}$, the number of non-zero entries of A, called nnz(A), is $O(n)$.

Many matrices arising from real applications are sparse. If we need to store A, we can exploit the sparse structure using different **storage schemes**:

Name	Easy insertion	Fast $A\mathbf{x}$
Coordinate (COO)	Yes	No
CSR	No	Yes

1.2.1 Coordinate format (COO)

The data structure consists of three arrays of length nnz(A):

- `AA`: all the values of the nonzero elements of A *in any order*
- `JR`: integer array containing their row indices
- `JC`: integer array containing their column indices

Example

$$A = \begin{pmatrix} 1. & 0. & 0. & 2. & 0. \\ 3. & 4. & 0. & 5. & 0. \\ 6. & 0. & 7. & 8. & 9. \\ 0. & 0. & 10. & 11. & 0. \\ 0. & 0. & 0. & 0. & 12. \end{pmatrix}$$

AA	12.	9.	7.	5.	1.	2.	11.	3.	6.	4.	8.	10.
JR	5	3	3	2	1	1	4	2	3	2	3	4
JC	5	5	3	4	1	4	4	1	1	2	4	3

1.2.2 Coordinate Compressed Sparse Row format (CSR)

If the elements of A are listed by row, the array `JC` might be replaced by an array that points to the beginning of each row.

- `AA`: all the values of the nonzero elements of A, *stored row by row* from $1, \ldots, n$
- `JA`: contains the column indices
- `IA`: contains the pointers to the beginning of each row in the arrays `AA` and `JA`. Thus $IA(i)$ contains the position in the arrays `AA` and `JA` where the i-th row starts. The length of `IA` is $n + 1$.

Example

$$A = \begin{pmatrix} 1. & 0. & 0. & 2. & 0. \\ 3. & 4. & 0. & 5. & 0. \\ 6. & 0. & 7. & 8. & 9. \\ 0. & 0. & 10. & 11. & 0. \\ 0. & 0. & 0. & 0. & 12. \end{pmatrix}$$

AA	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.
JA	1	4	1	2	4	1	3	4	5	3	4	5
IA	1	3	6	10	12	13						

2 Iterative methods for large and sparse linear systems

Let's consider the following linear system of equations

$$A\mathbf{x} = \mathbf{b}$$

where:

- $A \in \mathbb{R}^{n \times n}$
- $\mathbf{b} \in \mathbb{R}^n$
- $\mathbf{x} \in \mathbb{R}^n$
- $\det(A) \neq 0$

n general, direct methods (i.e., methods based on a *manipulation* of A) are not suitable whenever n is large, or A is sparse. The average cost of direct methods scales as n^3. Therefore, we use iterative methods.

We introduce a sequence $\mathbf{x}^{(k)}$ of vectors determined by a recursive relation that identifies the method. In order to initialize the iterative process, it is necessary to provide an initial vector $\mathbf{x}^{(0)}$.

For the method to make sense, it must satisfy the **convergence property**

$$\lim_{k \to +\infty} \mathbf{x}^{(k)} = \mathbf{x}$$

Convergence must not depend on the choice of $\mathbf{x}^{(0)}$.

Since the convergence is only guaranteed after an infinite number of iterations, from a practical point of view we will have to stop the iterative process after a finite number of iterations, when we believe we have arrived *sufficiently close* to the solution. For this, we use specific **stopping criteria**.

Note that, even in exact arithmetic, an iterative method will inevitably be affected by a **numerical error**.

2.1 The Jacobi method

Starting from the i-th line of the linear system:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \quad \Longrightarrow \quad a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i$$

Formally, the solution x_i for each i is given by

$$x_i = \frac{b_i - \sum_{j \neq i} a_{ij} x_j}{a_{ii}}$$

Obviously we cannot use it because we do not know x_j for $j \neq i$. We can introduce an iterative method that updates $x_i^{(k+1)}$ using the others $x_j^{(k)}$ obtained in the previous step k

$$x_i^{(k+1)} = \frac{b_i - \sum_{j \neq i} a_{ij} x_j^{(k)}}{a_{ii}} \qquad \forall i = 1, \ldots, n$$

In general, each iteration costs $\sim n^2$ operations, so the Jacobi method is competitive if the number of iterations is less that n. If A is a sparse matrix, then the cost is only $\sim n$ per iteration.

It should be noted that the solutions $x_i^{(k+1)}$ can be computed fully in parallel, which is very competitive for large scale systems.

2.2 The Gauss-Seidel method

Let's start from Jacobi's method: $x_i^{(k+1)} = \frac{b_i - \sum_{j \neq i} a_{ij} x_j^{(k)}}{a_{ii}}$

At iteration $(k+1)$, let's consider the computation of $x_i^{(k+1)}$. We observe that, for $j < i$ (for $i > 2$), $x_j^{(k+1)}$ is known (we have already calculated it). We can therefore think of using the quantities at step $(k+1)$ if $j < i$ and (as in the Jacobi method) those at the previous step k if $j > i$.

$$x_i^{(k+1)} = \frac{b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)}}{a_{ii}}$$

The computational costs are comparable to those of the Jacobi method. However, unlike the Jacobi, GS is not fully parallelizable.

2.3 Linear iterative methods

n general, we consider linear iterative methods of the following form:

$$\mathbf{x}^{(k+1)} = B\mathbf{x}^{(k)} + \mathbf{f} \qquad k \geq 0$$

where $B \in \mathbb{R}^{n \times n}$, $\mathbf{f} \in \mathbb{R}^n$

B is called iteration matrix. Its choice, together with $\mathbf{f}$, uniquely identify the method. So how to choose B and $\mathbf{f}$?

1. **Consistency**: If $\mathbf{x}^{(k)}$ is the exact solution $\mathbf{x}$, then $\mathbf{x}^{(k+1)}$ is again equal to $\mathbf{x}$

$$\mathbf{x} = B\mathbf{x} + \mathbf{f} \quad \Longrightarrow \quad \mathbf{f} = (I - B)\mathbf{x} = (I - B)A^{-1}\mathbf{b}$$

   The former identity gives a relationship between B and $\mathbf{f}$ as a function of the data.

2. **Convergence**: Let's introduce the error at step $(k+1)$: $\mathbf{e}^{(k+1)} = \mathbf{x} - \mathbf{x}^{(k+1)}$ and a suitable vector norm $\|\cdot\|$ (for example, the Euclidian norm). Then we have:

$$\|\mathbf{e}^{(k+1)}\| = \|\mathbf{x} - \mathbf{x}^{(k+1)}\| = \|\mathbf{x} - B\mathbf{x}^{(k)} - \mathbf{f}\| = \qquad \leftarrow \text{consistency}$$

$$= \|\mathbf{x} - B\mathbf{x}^{(k)} - (I - B)\mathbf{x}\| = \|B\mathbf{e}^{(k)}\|$$

$$\leq \|B\|\|\mathbf{e}^{(k)}\|$$

By recursion we obtain

$$\|\mathbf{e}^{(k-1)}\| \leq \|B\|\ \|B\|\ \|e^{(k+1)}\|$$

$$\leq \|B\|\ \|B\|\ \|B\|\ \|\mathbf{e}^{(k-2)}\| \leq \cdots \leq \|B\|^{k+1}\|\mathbf{e}^{(0)}\|$$

$$\implies \lim_{k\to\infty}\|\mathbf{e}^{(k+1)}\| \leq \left(\lim_{k\to\infty}\|B\|^{k+1}\right)\|\mathbf{e}^{(0)}\|$$

If $\|B\| < 1 \implies \lim_{k\to\infty}\|\mathbf{e}^{(k+1)}\| = 0$. Therefore, $\|B\| < 1$ is a **sufficient condition for convergence**.

2.3.1 necessary and sufficient condition for convergence

We define $\rho(B) = \max_j|\lambda_j(B)|$, where $\lambda_j(B)$ are the eigenvalues of B. $\rho(B)$ is called **spectral radius** of B. Remember that if B is SPD, then $\rho(B) = \|B\|$.

Theorem

A consistent iterative method with iteration matrix B converges **if and only if** $\rho(B) < 1$.

2.3.2 Convergence of Jacobi (J) and Gauss-Seidel (GS) methods

Let

$$A = \begin{bmatrix} \ddots & & -F \\ & D & \\ -E & & \ddots \end{bmatrix}$$

- D: diagonal part of A
- $-E$: lower triangular part of A
- $-F$: upper triangular part of A

The Jacobi method can be rewritten as

$$D\mathbf{x}^{(k+1)} = (E + F)\mathbf{x}^{(k)} + \mathbf{b}$$

its iteration matrix is given by

$$B_J = D^{-1}(E + F) = F^{-1}(D - A) = I - D^{-1}A$$

For the Gauss-Seidel method we have

$$(D - E)\mathbf{x}^{(k+1)} = F\mathbf{x}^{(k)} + \mathbf{b}$$

The iteration matrix is given by

$$B_{GS} = (D - E)^{-1}F$$

Both methods are consistent.

- If A is strictly diagonally dominant by rows/columns, then J and GS converge.
- If A is SPD then the GS method is convergent.
- If A is tridiagonal, it can be shown that $\rho^2(B_J) = \rho(B_{GS})$. Therefore both methods converge or fail to converge at the same time. If they converge, GS is faster than J.

2.4 Stopping criteria

A practical test is needed to determine when to stop the iteration. The idea is to terminate the iterations when $\dfrac{\|\mathbf{x} - \mathbf{x}^k\|}{\|\mathbf{x}^k\|} \leq \varepsilon$, where ε is a user-defined tolerance. Unfortunately, the error is not known. We can use two criterion:

1. **Residual-based stopping criterion**.

$$\frac{\|\mathbf{x} - \mathbf{x}^{(k)}\|}{\|\mathbf{x}\|} \leq K(A)\frac{\|\mathbf{r}^{(k)}\|}{\|\mathbf{b}\|} \quad \Longrightarrow \quad \boxed{\frac{\|\mathbf{r}^k\|}{\|\mathbf{b}\|} \leq \varepsilon}$$

   This is a good stopping criterion whenever $K(A)$ is "small".

2. **Distance between consecutive iterates**. Define the relative residual $\boldsymbol{\delta}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$

$$\boxed{\|\boldsymbol{\delta}^{(k)}\| \leq \varepsilon}$$

   It can be shown that the relation between the true error and $\boldsymbol{\delta}^{(k)}$ is

$$\|\mathbf{e}^{(k)}\| \leq \frac{1}{1 - \rho(B)}\|\boldsymbol{\delta}^{(k)}\|$$

   This is a good stopping criterion only if $\rho(B) \ll 1$.

2.5 The stationary Richardson method

Given $\mathbf{x}^{(0)}$, $\alpha \in \mathbb{R}$:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha \underbrace{(\mathbf{b} - A\mathbf{x}^{(k)})}_{\text{residual } \mathbf{r}^{(k)}}$$

The idea is to update the numerical solution by adding a quantity proportional to the residual (e.g., if the residual is large, the solution at step k should be corrected a lot).

The stationary Richardson method is characterized by

$$B_\alpha = I - \alpha A \qquad \mathbf{f} = \alpha\mathbf{b}$$

2.5.1 Convergence of the stationary Richardson method

Let A be symmetric and positive definite matrix (SPD), then the stationary Richardson method is convergent if and only if

$$0 < \alpha < \frac{2}{\lambda_{max}(A)}$$

with the optimal parameter α_{opt}:

$$\alpha_{opt} = \frac{2}{\lambda_{min}(A) + \lambda_{max}(A)}$$

and optimal spectral radius ρ_{opt} (maximum convergence speed):

$$\rho_{opt} = \frac{K(A) - 1}{K(A) + 1}$$

2.6 Preconditioning techniques

The optimal value $\rho_{opt} = \frac{K(A)-1}{K(A)+1}$ expresses the maximum convergence speed that can be attained with the stationary Richardson method.

Badly conditioned matrices ($K(A) \gg 1$) are characterized by a very low convergence rate. To improve the speed of convergence, we introduce a SPD matrix P^{-1} (called *preconditioner*). Then, solving $A\mathbf{x} = \mathbf{b}$ is equivalent to the following preconditioned system:

$$P^{-\frac{1}{2}} A \underbrace{P^{-\frac{1}{2}}\mathbf{z}}_{\mathbf{x}} = P^{-\frac{1}{2}}\mathbf{b}$$

where $\mathbf{x} = P^{-\frac{1}{2}}\mathbf{z}$.

Suppose that P^{-1} has real and positive eigenvalues. We apply the stationary Richardson method to $P^{-1}A$, i.e.,

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha P^{-1}\left(\mathbf{b} - A\mathbf{x}^{(k)}\right) = \mathbf{x}^{(k)} + \alpha P^{-1}\mathbf{r}^{(k)}$$

We obtain the same results as in the non preconditioned case, provided we replace A with $P^{-1}A$:

- Convergence: $\quad 0 < \alpha < \dfrac{2}{\lambda_{max}(P^{-1}A)}$
- Optimal values:

$$\alpha_{opt} = \frac{2}{\lambda_{min}(P^{-1}A) + \lambda_{max}(P^{-1}A)} \qquad \rho_{opt} = \frac{K(P^{-1}A) - 1}{K(P^{-1}A) + 1}$$

Therefore, if $K(P^{-1}A) \ll K(A)$, we obtain a higher convergence rate with respect to the unpreconditioned case.

To have $K(P^{-1}A) \ll K(A)$ we need that the linear system in P must be "easily solvable". To this aim, P should have a special structure (e.g., diagonal, triangular, ...).

2.7 The gradient method

Let $A \in \mathbb{R}^n$ be SPD. In this case, solving the linear system $A\mathbf{x} = \mathbf{b}$ is equivalent to minimizing the quadratic function $\Phi : \mathbb{R}^n \to \mathbb{R}$

$$\Phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b}$$

As A is SPD, $\Phi(\mathbf{x})$ defines a paraboloid with global minimum in $\mathbf{x}$. Since $\nabla\Phi(\mathbf{x}) = A\mathbf{x}-\mathbf{b}$ (assuming A is symmetric), we have that the minimum ($\nabla\Phi(\mathbf{x}) = \mathbf{0}$) coincides with the solution of $A\mathbf{x} = \mathbf{b}$. The update rule is

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \nabla\Phi(\mathbf{x}^{(k)})$$

where α_k is the step size. The optimal parameter α_k is chosen by asking that

$$\frac{\mathrm{d}\Phi\left(\mathbf{x}^{(k+1)}\right)}{\mathrm{d}\alpha_k} = 0 \qquad \Longrightarrow \qquad \alpha_k = \frac{\left(\mathbf{r}^{(k)}\right)^T \mathbf{r}^{(k)}}{\left(\mathbf{r}^{(k)}\right)^T A\mathbf{r}^{(k)}}$$

Note that $\nabla\Phi(\mathbf{x}) = A\mathbf{x} - \mathbf{b} = -\mathbf{r}$. Therefore, the residual $\mathbf{r}^(k)$ at step k is

$$\mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)} = -\nabla\Phi(\mathbf{x}^{(k)})$$

Also note that the gradient method can be interpreted as a Richardson method with dynamic parameter α_k.

Algorithm 1 Gradient method (steepest descent)

Given $\mathbf{x}^{(0)}$, compute $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
while stopping criteria **do**
  $a_k = \dfrac{\left(\mathbf{r}^{(k)}\right)^T \mathbf{r}^{(k)}}{\left(\mathbf{r}^{(k)}\right)^T A\mathbf{r}^{(k)}}$ ▷ $\approx n$ flops if A is sparse
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{r}^{(k)}$ ▷ $\approx n$ flops
  $\mathbf{r}^{(k+1)} = (I - \alpha_k A)\mathbf{r}^{(k)}$ ▷ $\approx n$ flops
end while

Consider $\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}$ the error at iteration k. We have that the convergence rate of the gradient method is the same as that of stationary Richardson's method with optimal parameter:

$$\|\mathbf{e}^{(k+1)}\|_A \leq \frac{K(A)-1}{K(A)+1}\|\mathbf{e}^{(k)}\|_A \qquad \Longrightarrow \qquad \|\mathbf{e}^{(k)}\|_A \leq \left(\frac{K(A)-1}{K(A)+1}\right)^k \|\mathbf{e}^{(0)}\|_A$$

2.8 The conjugate gradient method

We introduce a new updating direction $\mathbf{d}^{(k+1)}$ in such a way that it is A-conjugate to all the previous directions $\mathbf{d}^{(j)}$, $j \leq k$ (i.e., orthogonal with respect to the scalar product induced by A)

$$\left(\mathbf{d}^{(k+1)}, \mathbf{d}^{(j)}\right)_A = \left(\mathbf{d}^{(k+1)}, A\mathbf{d}^{(j)}\right) = 0 \qquad \forall j \leq k \quad \begin{matrix}\text{conjugate}\\ \text{directions}\end{matrix}$$

Algorithm 2 Conjugate Gradient (CG) method

Given $\mathbf{x}^{(0)}$, compute $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
Set $\mathbf{d}^{(0)} = \mathbf{r}^{(0)}$
while stopping criteria **do**
  $a_k = \dfrac{\left(\mathbf{d}^{(k)}\right)^T \mathbf{r}^{(k)}}{\left(\mathbf{d}^{(k)}\right)^T A\mathbf{r}^{(k)}}$
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$ ▷ Update $\mathbf{x}^{(k+1)}$ along $\mathbf{d}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k A\mathbf{d}^{(k)}$ ▷ Update $\mathbf{r}^{(k+1)}$
  $\beta_k = \dfrac{\left(A\mathbf{d}^{(k)}\right)^T \mathbf{r}^{(k+1)}}{\left(A\mathbf{d}^{(k)}\right)^T \mathbf{d}^{(k)}}$
  $\mathbf{d}^{(k+1)} = \mathbf{r}^{(k+1)} - \beta_k \mathbf{d}^{(k)}$ ▷ Update $\mathbf{d}^{(k+1)}$
end while

In exact arithmetic, the CG method converges to the exact solution in at most n iterations. At each iteration k, the error $\mathbf{e}^{(k)} = \mathbf{x} - \mathbf{x}^{(k)}$ can be bounded by

$$\left\|\mathbf{e}^{(k)}\right\|_A \leq \frac{2c^k}{1+c^{2k}} \left\|\mathbf{e}^{(0)}\right\|_A \qquad \text{with} \qquad c = \frac{\sqrt{K(A)}-1}{\sqrt{K(A)}+1}$$

We can also use preconditioning on the CG method.

2.9 Preconditioned Gradient and Conjugate Gradient

Both methods converge slowly with poorly conditioned matrices. We can introduce a preconditioner in order to accelerate convergence.

Let A and P be SPD. We consider the following preconditioned system.

$$\hat{A}\hat{\mathbf{x}} = \hat{\mathbf{b}}$$

$$\underbrace{P^{-\frac{1}{2}} A P^{-\frac{1}{2}}}_{\hat{A}} \underbrace{P^{\frac{1}{2}}\mathbf{x}}_{\hat{\mathbf{x}}} = \underbrace{P^{-\frac{1}{2}}\mathbf{b}}_{\hat{\mathbf{b}}}$$

The solution to the original problem can be obtained from the solution $\hat{\mathbf{x}}$ to the original problem:

$$\mathbf{x} = P^{-\frac{1}{2}}\hat{\mathbf{x}}$$

Let $\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}$ be the error at iteration k. The error bound of the PCG method is

$$\|\mathbf{e}^{(k)}\|_A \leq \frac{2c^k}{1+c^{2k}} \|\mathbf{e}^{(0)}\| \qquad \text{with} \qquad c = \frac{\sqrt{K(P^{-1}A)}-1}{\sqrt{K(P^{-1}A)}}$$

If the preconditioner is a "good" preconditioner, then

$$\frac{\sqrt{K(P^{-1}A)}-1}{\sqrt{K(P^{-1}A)}+1} < \frac{\sqrt{K(A)}-1}{\sqrt{K(A)}+1}$$

Algorithm 3 Preconditioned Conjugate Gradient (PCG) method

Given $\mathbf{x}^{(0)}$, compute $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
Set $\mathbf{d}^{(0)} = \mathbf{r}^{(0)}$
while stopping criteria **do**
  $a_k = \dfrac{\left(\mathbf{z}^{(k)}\right)^T \mathbf{r}^{(k)}}{\left(A\mathbf{d}^{(k)}\right)^T \mathbf{d}^{(k)}}$
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k A\mathbf{d}^{(k)}$
  $P\mathbf{z}^{(k+1)} = \mathbf{r}^{(k+1)}$ ▷ Compute the action of the preconditioner P on $\mathbf{r}^{(k+1)}$
  $\beta_k = \dfrac{\left(A\mathbf{d}^{(k)}\right)^T \mathbf{r}^{(k+1)}}{\left(A\mathbf{d}^{(k)}\right)^T \mathbf{d}^{(k)}}$
  $\mathbf{d}^{(k+1)} = \mathbf{r}^{(k+1)} - \beta_k \mathbf{d}^{(k)}$
end while

2.10 Krylov-space methods

For linear iterative methods (with $P = I$ and $\alpha_k = 1\ \forall k$) as those seen before, we have:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{r}^{(k)} \qquad k \geq 1 \tag{1}$$

The following recursive relation for the residuals holds

$$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - A\mathbf{r}^{(k)} \qquad k \geq 1$$

From the above identity, it follows by induction that

$$\mathbf{r}^{(k)} = p_{k-1}(A)\mathbf{r}^{(0)} \in \text{span}\ \{\mathbf{r}^{(0)},\ A\mathbf{r}^{(0)},\ \ldots,\ A^{k-1}\mathbf{r}^{(0)}\}$$

where $p_r(z) = (1-z)^r$ is a polynomial of exact degree r.

From (1) we have

$$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + \mathbf{r}^{(0)} + \cdots + \mathbf{r}^{(k-1)} = \mathbf{x}^{(0)} + p_{k-1}(A)\mathbf{r}^{(0)}$$

so $\mathbf{x}^{(k)}$ lies in the affine space $\mathbf{x}^{(0)} + \text{span}\ \{\mathbf{r}^{(0)},\ A\mathbf{r}^{(0)},\ \ldots,\ A^{k-1}\mathbf{r}^{(0)}\}$ obtained by shifting the subspace of $\mathbf{r}^{k-1}$.

Definition: Krylov subspace

Given a nonsingular $A \in \mathbb{R}^{n\times n}$ and $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{y} \neq 0$, the k-th Kyrlov subspace $\mathscr{K}_k(A, \mathbf{y})$ generated by A from $\mathbf{y}$ is

$$\mathscr{K}_k(A, \mathbf{y}) := \text{span}\ (\mathbf{y},\ A\mathbf{y},\ \ldots,\ A^{k-1}\mathbf{y})$$

Clearly, it holds

$$\mathscr{K}_1(A, \mathbf{y}) \subseteq \mathcal{K}_2(A, \mathbf{y}) \subseteq \ldots$$

We can choose the k-th approximate solution $\mathbf{x}^{(k)} \in \mathbf{x}^{(0)} + \mathscr{K}_k(A, \mathbf{r}^{(0)})$. In particular

$$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + p_{k-1}(A)\mathbf{r}^{(0)}$$

with a polynomial $p_{k-1}(A)$ of exact degree $k-1$.

When applied to large real-world problems, Kyrlov space solvers often converge very slowly, if at all. In practice, they are therefore nearly always used with preconditioning:

$$A\mathbf{x} = \mathbf{b} \qquad \Longleftrightarrow \qquad P^{-1}AP^{-1}\mathbf{x} = P^{-1}\mathbf{b} \qquad \Longleftrightarrow \qquad \hat{A}\hat{\mathbf{z}} = \hat{\mathbf{b}}, \quad \hat{P}\hat{\mathbf{z}} = \mathbf{x}$$

Applying a preconditioned Kyrlov space solver means applying the method to $\hat{A}\hat{\mathbf{z}} = \hat{b}$.

2.11 Solving nonsymmetric linear systems iteratively with Krylov space solvers

2.11.1 The biconjugate gradiend (BiCG) method

While CG (for spd A) has mutually orthogonal residuals $\mathbf{r}^{(k)}$ with

$$\mathbf{r}^{(k)} = p_k(A)\mathbf{r}^{(0)} \in \text{span}\{\mathbf{r}^{(0)}, A\mathbf{r}^{(0)}, \ldots, A^k\mathbf{r}^{(0)}\} = \mathscr{K}_{k+1}(A, \mathbf{r}^{(0)})$$

BiCG constructs in the same spaces residuals that are orthogonal to a dual Krylov space spanned by *shadow residuals*

$$\tilde{\mathbf{r}}^{(k)} = p_k(A)\tilde{\mathbf{r}}^{(0)} \in \text{span}\{\tilde{\mathbf{r}}^{(0)}, A^T\tilde{\mathbf{r}}^{(0)}, \ldots, (A^T)^k\tilde{\mathbf{r}}^{(0)}\} = \mathscr{K}_{k+1}(A^T, \tilde{\mathbf{r}}^{(0)}) \equiv \tilde{\mathscr{K}}_{k+1}$$

The initial shadow residual $\tilde{\mathbf{r}}(0)$ can be chosen freely. So, BiCG requires two matrix-vector multiplications to extend $\mathscr{K}_k$ and $\tilde{\mathscr{K}}_k$: one multiplication by A and one by A^T.

2.11.2 the BiCGSTAB method

The biconjugate gradient stabilized method (BiCGSTAB) is a variant of the biconjugate gradient method (BiCG) and has faster and smoother convergence than the original BiCG.

It is unnecessary to explicitly keep track of the residuals and search directions of BiCG. In other words, the BiCG iterations can be performed implicitly. Unlike the original BiCG method, it doed not require multiplication by A^T.

2.11.3 The GMRES method

The Generalized Minimum Residual method (GMRES) is a projection method. The method approximates the solution by the vector in a Krylov subspace with minimal residual. The arnoldi iteration is used to find this vector.

GMRES approximates the exact solution of $A\mathbf{x} = \mathbf{b}$ by the vector $\mathbf{x}^{(k)} \in \mathbf{x}^{(0)} + \mathscr{K}_k(A, \mathbf{r}^{(0)})$ that minimizes the Euclidian norm of the residual $\mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)}$

3 Solving large-scale eigenvalue problems

Eigenvalue problem

Given a matrix $A \in \mathbb{C}^{n\times n}$, find $(\lambda, \mathbf{v}) \in \mathbb{C} \times \mathbb{C}^n \setminus \{\mathbf{0}\}$ such that

$$A\mathbf{v} = \lambda\mathbf{v}$$

where λ is an eigenvalue of A, and $\mathbf{v}$ (non-zero) is the corresponding eigenvector.

The set of all the eigenvalues of a matrix A is calle the **spectrum** of A.

The maximum modulus of all the eigenvalues is called the **spectral radius** of A: $\rho(A) = \max\{|\lambda| : \lambda \in \lambda(A)\}$.

The problem $A\mathbf{v} = \lambda\mathbf{v}$ is equivalent to $(A - \lambda I)\mathbf{v} = 0$. $\det(A - \lambda I) = 0$ is a polynomial of degree n in λ: it is called the **characteristic polynomial** of A and its roots are the eigenvalues of A.

3.1 Similarity transformations

We first need to identify what types of transformations preserve eigenvalues, and for what types of matrices the eigenvalues are easily determined.

Definition

The matrix B is similar to the matrix A if there exists a nonsingulat matrix T such taht $B = T^{-1}AT$.

With the above definition, it is trivial to show that

$$B\mathbf{y} = \lambda\mathbf{y} \implies T^{-1}AT\mathbf{y} = \lambda\mathbf{y} \implies A(T\mathbf{y}) = \lambda(T\mathbf{y})$$

so that A and B have the same eigenvalues, and if $\mathbf{y}$ is an eigenvector of B, then $\mathbf{v} = T\mathbf{y}$ is an eigenvector of A.

Similarity transformations preserve eigenvalues but do not preserve eigenvectors.

3.2 The power method

The power method is the simplest method for computing a single eigenvalue and eigenvector of a matrix.

Assume that the matrix A has a unique eigenvalue λ_1 of maximum modulus, i.e., $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$, with corresponding eigenvector $\mathbf{v}_1$.

Starting from a given nonzero vector $\mathbf{x}^{(0)}$ such that $\|\mathbf{x}^{(0)}\| = 1$, let's consider the following iteration scheme, for $k >= 0$:

$$\begin{aligned}
\mathbf{y}^{(k+1)} &\leftarrow A\mathbf{x}^{(k)} \\
\mathbf{x}^{(k+1)} &\leftarrow \frac{\mathbf{y}^{(k+1)}}{\|\mathbf{y}^{(k+1)}\|} \\
\nu^{(k+1)} &\leftarrow [\mathbf{x}^{(k+1)}]^H A\mathbf{x}^{(k+1)}
\end{aligned}$$

It can be shown that the above iteration scheme converges to a multiple of $\mathbf{v}_1$, the eigenvector corresponding to the dominant eigenvalue λ_1.

The convergence rate of the power method depends on the ratio $|\lambda_2|/|\lambda_1|$, where λ_2 is the eigenvalue having the second largest modulus. The smaller $|\lambda_2|/|\lambda_1|$, the faster the convergence is.

3.2.1 Deflation methods

Suppose that an eigenvalue λ_1 and corresponding eigenvector $\mathbf{v}_1$ for a matrix A have been computed. We can compute additional eigenvalues $\lambda_2, \ldots, \lambda_n$ of A by a process called **deflation**, which removes the known eigenvalue.

The *idea* is to construct a new matrix B with eigenvalues $\lambda_2, \ldots, \lambda_n$ that deflates the matrix A, removing λ_1. Then λ_2 can be obtained by the power method.

Let S be any nonsingular matrix such that

- $S\mathbf{v}_1 = \alpha\mathbf{e}_1$
- S is a scalar multiple of the first column $\mathbf{e}_1$ of the identity matrix I.

Then the similarity transformation determined by S transforms A into the from

$$SAS^{-1} = \begin{pmatrix} \lambda_1 & \mathbf{b}^T \\ 0 & B \end{pmatrix}$$

We use B to compute the next eigenvalue λ_2 and eigenvector $\mathbf{z}_2$. Given $\mathbf{z}_2$ eigenvector of B, we want to compute the second eigenvector $\mathbf{v}_2$ of the matrix A. We just need to add an element to the vector $\mathbf{z}_2$ (that consists of $n-1$ elements), that is

$$\mathbf{v}_2 = S^{-1}\begin{pmatrix} \alpha & \mathbf{z}_2 \end{pmatrix} \qquad \alpha = \frac{\mathbf{b}^H \mathbf{z}_2}{\lambda_1 - \lambda_2}$$

The process can be repeated to find the additional eigenvalues and eigenvectors.

3.2.2 Inverse power method

For some applications, the smallest eigenvalue of a matrix is required rather than the largest. We use the fact that the eigenvalues of A^{-1} are the reciprocals of those of A (hence the smallest eigenvalue of A is the reciprocal of the largest eigenvalue of A^{-1})

Starting form a given nonzero vector $\mathbf{q}^{(0)}$ such that $\|\mathbf{q}^{(0)}\| = 1$, let's consider the following iteration scheme, for $k \geq 0$:

$$\begin{aligned} &\text{Solve } A\mathbf{z}^{(k+1)} = \mathbf{q}^{(k)} \\ &\mathbf{q}^{(k+1)} \leftarrow \frac{\mathbf{z}^{(k+1)}}{\|\mathbf{z}^{(k+1)}\|} \\ &\sigma^{(k+1)} \leftarrow [\mathbf{q}^{k+1}]^H A\mathbf{q}^{(k+1)} \end{aligned}$$

3.2.3 Inverse power method with shift

We want to approximate the eigenvalue λ of A which is the closest to a given number $\mu \notin \sigma(A)$. We introduce $M_\mu = A - \mu I$ and observe that the eigenvalue λ of A which is closest to μ is the minimum eigenvalue of M_μ.

Starting from a given nonzero vector $\mathbf{q}^{(0)}$ such that $\|\mathbf{q}^{(0)}\| = 1$, let's consider the following iteration scheme, for $k \geq 0$:

$$\begin{aligned}
&\text{Solve } M_\mu \mathbf{z}^{(k+1)} = \mathbf{q}^{(k)} \\
&\mathbf{q}^{(k+1)} \leftarrow \frac{\mathbf{z}^{(k+1)}}{\|\mathbf{z}^{(k+1)}\|} \\
&\nu^{(k+1)} \leftarrow [\mathbf{q}^{(k+1)}]^H A \mathbf{q}^{(k+1)}
\end{aligned}$$

3.3 QR factorization

3.3.1 Projectors and complementary projectors

A projector is a square matrix $P \in \mathbb{R}^{n \times n}$ that satisfies $P^2 = P$:

- If $\mathbf{w} \in \text{range}(P)$, then $P\mathbf{w} = \mathbf{w}$. Indeed, since $\mathbf{w} \in \text{range}(P)$, then $\mathbf{w} = P\mathbf{z}$, for some $\mathbf{z}$. Therefore:
$$P\mathbf{w} = P(P\mathbf{z}) = P^2\mathbf{z} = P\mathbf{z} = \mathbf{w}$$
- The matrix $I - P$ is the complementary projector to P.
- $I - P$ projects on the nullspace of P: if $P\mathbf{w} = 0$, then $(I - P)\mathbf{w} = \mathbf{w}$, so $\text{null}(P) \subseteq \text{range}(I - P)$. But for any $\mathbf{w}$, $(I - P)\mathbf{w} = \mathbf{w} - P\mathbf{w} \in \text{null}(P)$, so $\text{range}(I - P) \subseteq \text{null}(P)$. Therefore,
$$\text{range}(I - P) = \text{null}(P)$$
and
$$\text{null}(I - P) = \text{range}(P)$$
- A projector P is *orthogonal* if $P = P^2 = P^T$.

3.3.2 The reduced QR factorization

Find orthonormal vectors $[\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n]$ that span the successive spaces spanned by the columns of $A = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$:

$$\langle \mathbf{a}_1 \rangle \subseteq \langle \mathbf{a}_1, \mathbf{a}_2 \rangle \cdots \subseteq \langle \mathbf{a}_1 \mathbf{a}_2, \ldots, \mathbf{a}_n \rangle$$

This means that (for full rank A):

$$\langle \mathbf{a}_1, \mathbf{a}_2, \ldots \mathbf{a}_j \rangle = \langle \mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_j \rangle \qquad \forall j = 1, \ldots, n$$

In matrix form, this becomes:

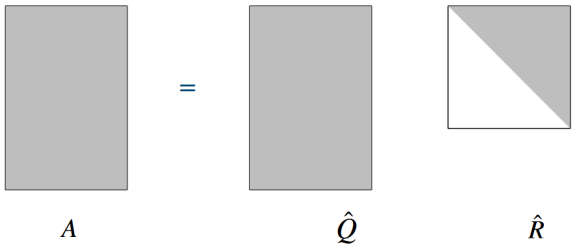
$$\left[\begin{array}{c|c|c|c} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \end{array}\right] = \left[\begin{array}{c|c|c|c} \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_n \end{array}\right] \begin{bmatrix} r_{11} & r_{12} & \ldots & r_{1n} \\ 0 & r_{22} & \ldots & \vdots \\ 0 & 0 & \ddots & r_{nn} \end{bmatrix}$$

that is

$$A = \hat{Q}\hat{R}$$

This is called the *reduced QR factorization*. In particular, the reduced QR factorization of A is the factorization $A = \hat{Q}\hat{R}$, where

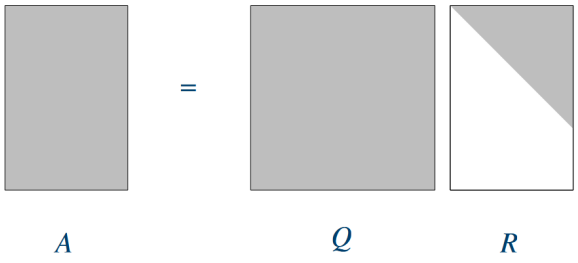
- Q is $m \times n$
- R is $n \times n$ upper-triangular



3.3.3 The full QR factorization

Let A be an $m \times n$ matrix. The *full QR factorization* of A is the factorization $A = QR$, where

- Q is $m \times n$ orthogonal ($QQ^T = I$)
- R is $m \times n$ upper-trapezial



3.3.4 Gram-Schmidt orthogonalization

Given $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$ the columns of A, find new $\mathbf{q}_j$ (the j-th column of $\hat{Q}$) orthogonal to $\mathbf{q}_1, \ldots, \mathbf{q}_{j-1}$ by subtracting components along previous vectors:

$$\mathbf{w}_j = \mathbf{a}_j - \sum_{k=1}^{j-1} (\bar{\mathbf{q}}_k^T \mathbf{a}_j)\mathbf{q}_k$$

and then normalize to get $\mathbf{q}_j = \dfrac{\mathbf{w}_j}{\|\mathbf{w}_j\|}$

We then obtain a reduced QR factorization $A = \hat{Q}\hat{R}$ with

$$r_{ij} = \bar{\mathbf{q}}_i^T \mathbf{a}_j, \quad i \neq j, \qquad \text{and} \qquad r_{jj} = \|\mathbf{a}_j - \sum_{i=1}^{j-1} r_{ij}\mathbf{q}_i\|$$

3.4 The QR algorithm

3.4.1 Schur decomposition

If $A \in \mathbb{C}^{n\times n}$, then there is a unitary matrix $U \in \mathbb{C}^{n\times n}$ such that $U^H AU = T$ is upper triangular. The diagonal elements of T are the eigenvalues of A.

$U = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n]$ are called **Schur vectors** (They are in general *not* eigenvectors).

The k-th column of $U^H AU = T$ reads:

$$A\mathbf{u}_k = \lambda_k \mathbf{u}_k + \sum_{i=1}^{k-1} t_{ik}\mathbf{u}_i \qquad \Longrightarrow \qquad Au_k \in \text{span}\{\mathbf{u}_1, \ldots, \mathbf{u}_k\} \quad \forall k$$

- The first Schur vector $\mathbf{u}_1$ is an eigenvector of A
- The first k Schur vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ for an invariant subspace for A
- The Schur decomposition is not unique

3.4.2 Basic QR algorithm

Let $A \in \mathbb{C}^{n \times n}$. The QR algorithm computes an upper triangular matrix T and a unitary matrix U such that $A = UTU^H$ is the Schur decomposition of A.

1: Set $A^{(0)} = A, U^{(0)} = I$
2: **while** stopping criteria **do**
3: $\quad A^{(k-1)} = Q^{(k)}R^{(k)}$ ▷ QR factorization of $A^{(k-1)}$
4: $\quad A^{(k)} = R^{(k)}Q^{(k)}$
5: $\quad U^{(k)} = U^{(k-1)}Q^{(k)}$ ▷ Update transformation matrix
6: **end while**
7: **return** $T = A^{(k)}, U = U^{(k)}$

3.4.3 Convergence

Let's assume that all the eigenvalues are isolated, i.e., $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$. Then, the elements of $A^{(k)}$ below the diagonal converge to zero, i.e.,

$$\lim_{k \to \infty} a_{ij}^{(k)} = 0 \quad \forall i > j$$

Moreover, it can be shown that

$$a_{ij}^{(k)} = O\left(\left|\frac{\lambda_i}{\lambda_j}\right|^k\right) \quad i > j$$

The basic QR algorithm can be used to compute eigenvalues, but it is computationally expensive ($O(n^3)$ operations per iteration). It can have very slow convergence if the eigenvalues of A are close.

To improve the situation, we can reduce the matrix A to a similar matrix taht is upper Hessenberg, which reduces the cost to $O(n^2)$ operations per iteration.

3.4.4 Hessenberg QR-method

A matrix $H \in \mathbb{C}^{n \times n}$ is called a Hessenberg matrix if its elements below the lower off-diagonal are zero:

$$h_{ij} = 0, \quad i > j + 1$$

To improve the QR-method we make use of an algorithm consisting of two phases:

- **Phase 1** - Compute a Hessenberg matrix H and an orthogonal matrix U such that $A = UHU^H$. Such a reduction can be done with a finite number of operations.
- **Phase 2** - Apply the basic QR-method to the matrix H.

3.5 The Lanczos algorithm

The Lanczos algorithm can be used to efficiently find the extremal eigenvalues (maximum and minimum)) of a symmetric matrix A of size $n \times n$.

It is based on computing the following decomposition of A:

$$A = QTQ^T$$

where Q is an orthonormal basis of vectors $\mathbf{q}_1, \ldots, \mathbf{q}_n$ and T is tri-diagonal:

$$Q = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n] \quad T = \begin{bmatrix} \alpha_1 & \beta_1 & 0 & \ldots & 0 \\ \beta_1 & \alpha_2 & \beta_2 & \ldots & 0 \\ 0 & \ddots & \ddots & \vdots & 0 \\ 0 & \ddots & \ddots & \vdots & \beta_{n-1} \\ 0 & \ldots & 0 & \beta_{n-1} & \alpha_n \end{bmatrix}$$

The decomposition always exists and is unique provided that $\mathbf{q}_1$ has been specified.

We know that $T = Q^T AQ$ which gives

$$\alpha_k = \mathbf{q}_k^T A \mathbf{q}_k, \qquad \beta_k = \mathbf{q}_{k+1}^T A \mathbf{q}_k$$

the full decomposition is obtained by imposing $AQ = QT$:

$$[A\mathbf{q}_1, A\mathbf{q}_2, \ldots, A\mathbf{q}_n] = [\alpha_1, \mathbf{q}_1 + \beta_1 \mathbf{q}_2, \beta_1 \mathbf{q}_1 + \alpha_2 \mathbf{q}_2 + \beta_2 \mathbf{q}_3, \ldots, \beta_{n-1} \mathbf{q}_{n-1} + \alpha_n \mathbf{q}_n]$$

At every iteration k, the algorithm generates intermediate matrices Q_k and T_k that satisfy $T_k = Q_k^T A Q_k$.

Remarks: $\mathbf{q}_1$ is set randomly; the orthonormal vectors $\mathbf{q}_k$ are called Lanczos vectors.

4 Overdetermined linear systems and SVD

An overdetermined linear system is a set of linear equations where you have more equations than unknowns.

The mathematical problem

Given $A \in \mathbb{R}^{m \times n}$, $m \geq n$, and $\mathbf{b} \in \mathbb{R}^m$ find $\mathbf{x} \in \mathbb{R}^n$ such that

$$A\mathbf{x} = \mathbf{b}$$

Note that generally the above problem has no solution (in the classical sense) unless the right side $\mathbf{b}$ is an element of range(A). We need a "new" concept of solution. The basic approach is to look for an $\mathbf{x}$ that makes $A\mathbf{x}$ "close" to $\mathbf{b}$.

4.1 Solution in the least-squares sense

Given $A \in \mathbb{R}^{m \times n}$, $m \geq n$, we say that $\mathbf{x}^* \in \mathbb{R}^n$ is a solution of the linear system $A\mathbf{x} = \mathbf{b}$ in the least-squares sense if

$$\Phi(\mathbf{x}^*) = \min_{\mathbf{y} \in \mathbb{R}^n} \Phi(\mathbf{y})$$

where

$$\Phi(\mathbf{y}) = \|A\mathbf{y} - \mathbf{b}\|_2^2$$

The problem thus consists of minimizing the Euclidean norm of the residual.

The solution $\mathbf{x}^*$ can be found by imposing the condition that the gradient of the function $\Phi(\cdot)$ must be equal to zero at $\mathbf{x}^*$.

From the definition we have

$$\Phi(\mathbf{y}) = (A\mathbf{y} - \mathbf{b})^T(A\mathbf{y} - \mathbf{b}) \tag{2}$$
$$= \mathbf{y}^T A^T A\mathbf{y} - 2\mathbf{y}^T A\mathbf{b} + \mathbf{b}^T\mathbf{b} \tag{3}$$

Therefore:

$$\nabla\Phi(\mathbf{y}) = 2A^T\mathbf{y} - 2A^T\mathbf{b}$$

from which it follows that $\mathbf{x}^*$ must be the solution of the square system of normal equations

$$A^T A\mathbf{x}^* = A^T\mathbf{b}$$

However, instead of considering the system of normal equations, we can use the QR factorization:

Theorem

Let $A \in \mathbb{R}^{m\times n}$, with $m \geq n$ be a full rank matrix. Then the unique solution in the least-square sense $\mathbf{x}^*$ of $A\mathbf{x}^* = \mathbf{b}$ is given by $\mathbf{x}^* = \hat{R}^{-1}\hat{Q}^T\mathbf{b}$, where $\hat{R} \in \mathbb{R}^{n\times n}$ and $\hat{Q} \in \mathbb{R}^{m\times n}$ are the matrices of the reduced QR factorization of A. Moreover, the minimum of $\Phi(\cdot)$ is given by $\Phi(\mathbf{x}^*) = \sum_{i=n+1}^{m} [(Q^T\mathbf{b})_i]^2$

4.2 Singular value decomposition (SVD)

Any matrix can be reduced in diagonal form by a suitable pre and post-multiplication by unitary matrices.

Theorem

Let $A \in \mathbb{R}^{m\times n}$. There exists two **orthogonal** matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ such that

$$U^T AV = \Sigma = \text{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m\times n}$$

with $p = \min_(m, n)$ and $\sigma_1 \geq \cdots \geq \sigma_p \geq 0$. The previous formula is called **Singular value decomposition** (**SVD**) of A, and the numbers σ_i are called singular values of A.

4.2.1 Generalized inverse of A

Suppose that $A \in \mathbb{R}^{m\times n}$ has rank equal to r and that is admits a SVD of the type $U^T AV = \Sigma$. The matrix

$$A^\dagger = V\Sigma^\dagger U^T$$

is called the **generalized inverse of** A, with $\Sigma^\dagger = \text{diag}\left\{\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_p}, 0, \ldots, 0\right\}$

If $n = m = \text{rank}(A)$, then $A^\dagger = A^{-1}$.

Going back to our problem, that is find $\mathbf{x}^* \in \mathbb{R}^n$ **with minimal Euclidian** norm such that $\|A\mathbf{x}^* - \mathbf{b}\|_2^2 \leq \min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|_2^2$, we have the following:

Theorem

Let $A \in \mathbb{R}^{m\times n}$ with SVD given by $A = U\Sigma V^T$. Then the unique solution to the previous problem is

$$\mathbf{x}^* = A^\dagger \mathbf{b}$$

where $A^\dagger$ is the pseudo-inverse of A.

5 Multigrid methods

A multigrid (MG) method is an iterative algorithm of the form

$$\mathbf{x}^{(k+1)} = MG\left(\mathbf{x}^{(k)}\right), \quad k \geq 0$$

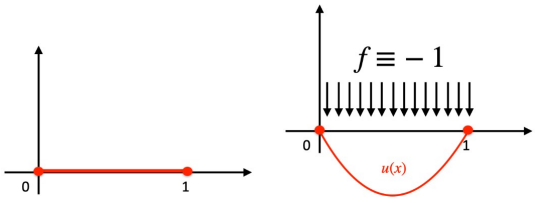
for solving the sparse linear systems of equations stemming from the numerical discretization of differential equations. MG methods are based on a hierarchy of levels (associated with a hierarchy of discretizations)

The main idea of multigrid is to accelerate the convergence of a basic iterative method by a gloval correction of the fine grid solution approximation accomplpished by solving a coarse problem. This coarse-level problem should be "similar" to the fine grid problem.

5.1 A simple 1D example

One-dimensional boundary value problem describing the displacement $u(x)$ of a uniform rod subject to an external load $f(x)$ and fixed at the extrema.

$$-u''(x) = f(x), \quad x \in (0,1)$$
$$u(0) = 0, \quad u(1) = 0$$
$$\text{Grid-size:} h = \frac{1}{n+1}, \quad n \geq 1$$
$$\text{Grid:} x_j = jn, \quad j = 0, \ldots, n+1$$



5.2 approximation with the finite element method

Let u_j be the approximate solution of $u(x_j), j = 1, \ldots, n$ on a uniform grid of mesh size h (u_0 and u_{n+1} are known). Then,

$$-u_{j-1} + 2u_j - u_{j+1} = hf(x_j), \quad j = 1, \ldots, n$$

$$u_0 = 0 \qquad u_{n+1} = 0$$

5.3 Matrix form

Define $\mathbf{x}_h = (u_1, u_2, \ldots, u_n)^T$ and $\mathbf{b}_h = (b_1, b_2, \ldots, b_n)^T$. The (sparse) linear system of equations is

$$A_h \mathbf{x}_h = \mathbf{b}_h$$

where

$$\begin{bmatrix} 2 & -1 & \ldots & \ldots & 0 \\ -1 & 2 & -1 & \ldots & 0 \\ 0 & \ldots & \ldots & \ldots & -1 \\ 0 & \ldots & \ldots & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

5.4 Weighted Jacobi iteration

We consider the weighted Jacobi iteration applied to the one-dimensional model problem. Recalling that $B_\omega = (1-\omega)I + \omega B_J$, where B_J is the iteration matrix of the Jacobi method, we have

$$B_\omega = I - \frac{\omega}{2} \begin{bmatrix} 2 & -1 & \ldots & \ldots & 0 \\ -1 & 2 & -1 & \ldots & 0 \\ 0 & \ldots & \ldots & \ldots & 0 \\ 0 & \ldots & \ldots & -1 & 2 \end{bmatrix}$$

Therefore, teh eigenvalues of B_J satisfy $\lambda(B_\omega) = 1 - \frac{\omega}{2}\lambda(A_h)$

5.5 On the eigenpairs of the matrix A_h

The eigenvalues of A_h are

$$\lambda_j(A_h) = 4\sin^2\left(\frac{j\pi}{2(n+1)}\right) \quad j = 1, \ldots, n$$

and therefore

$$\lambda_j(B_\omega) = 1 - \frac{\omega}{2} 4\sin^2\left(\frac{j\pi}{2(n+1)}\right) \quad j = 1, \ldots n$$

The corresponding eigenvector $\mathbf{w}_j(A_h)$ of A_h (and of B_ω) has components

$$(\mathbf{w}_j(A))_i = \sin\left(\frac{ij\pi}{n+1}\right)$$

Note that if $0 < \omega \leq 1$, then $|\lambda_j(B_\omega)| < 1 \quad \forall j$ and the weighted Jacobi iteration converges.

5.6 Effect of k iterations of weighted Jacobi

Let $\mathbf{x}^{(0)}$ be an initial guess, we have $\mathbf{e}^{(0)} = \mathbf{x} - \mathbf{x}^{(0)} = \mathbf{x}^{(0)}$, we represent $\mathbf{e}^{(0)}$ using the eigenvectors of A_h in the form

$$\mathbf{e}^{(0)} = \sum_{j=1}^{n} c_j \mathbf{w}_j$$

for suitable coefficients $c_j, j = 1, \ldots, j$.

It can be shown that after k steps of the iteration, the error is given by

$$\mathbf{e}^{(k)} = B_\omega^k \mathbf{e}^{(0)} = \sum_{j=1}^{n} c_j B_\omega^k \mathbf{w}_k = \sum_{j=1}^{n} c_j \lambda_j^k(B_\omega)\mathbf{w}_k$$

That is, after k iterations, the j-th mode of the initial error has been reduced by a factor of $\lambda_j^k(B_\omega)$.

5.7 Coarse grids

In passing from the fine grid to the coarse grid, a mode becomes more oscillatory. This is true provided that $1 \leq j < n/2$.

The important point is that smooth modes on a fine grid look less smooth on a coarse grid. This suggests that when relaxation begins to stall, signaling the predominance of smooth error modes, it is a good idea to move to a coarser grid, where the smooth error modes appear more oscillatory and relaxation will be more effective. So how do we move to a coarser gris and relax on the more oscillatory error modes?

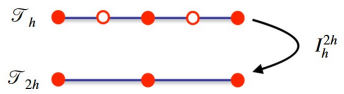
Relaxation on the original equation $A_h\mathbf{x}_h = \mathbf{b}_h$ with an arbitrary initial guess $\mathbf{x}^{(0)}$ is equivalent to relaxing on the residual equation $A_h\mathbf{e}_h = \mathbf{r}_h^{(k)}$ with the specific initial guess $\mathbf{e}_h^{(0)} = \mathbf{0}$.

Recall that $\mathbf{r}_h^{(k)} = \mathbf{b}_h - A_h\mathbf{x}_h^{(k)}$, $\mathbf{e}^{(k)} = \mathbf{x}_h - \mathbf{x}_h^{(k)}$. This connection between the original and the residual equations motivates the use of the residual equations motivates the use of the residual equation.

5.8 Correction scheme $\mathbf{x}_h^{(k)} \to \mathbf{x}_h^{(k+1)}$

The idea is to use the residual equation to relax on the error:

1. Relax ν_1 times on $A_h\mathbf{x}_h = \mathbf{b}_h$ to obtain an approximation $\mathbf{x}_h^{(k+\nu_1)}$
2. Compute the residual $\mathbf{r}_h^{(k+\nu_1)} = \mathbf{b}_h - A_h\mathbf{x}_h^{(k+\nu_1)}$
3. Move the residual $\mathbf{r}_h^{(k+\nu_1)}$ from $\mathscr{T}_h$ to $\mathscr{T}_{2h}$ to obtain $\mathbf{r}_{2h}^{(k+\nu_1)}$
4. Solve the residual equations $A_{2h}\mathbf{e}_{2h} = \mathbf{r}_{2h}^{(k+\nu_1)}$ on $\mathscr{T}_{2h}$
5. Move the error $\mathbf{e}_{2h}$ from $\mathscr{T}_{2h}$ to $\mathscr{T}_h$ to obtain $\mathbf{e}_h$
6. Correct the approximation obtained on $\mathscr{T}_h$ with the error estimate obtained on $\mathscr{T}_{2h}$, i.e., $\mathbf{x}_h^{(k+1)} = \mathbf{x}_h^{(k+\nu_1)} + \mathbf{e}_h$



5.9 Restriction operator (fine-to-coarse, 1D)

The second class of intergrid transfer operations incolves moving vectors from a fine grid to a coarse grid. They are generally called **restriction** operators and are denoted by

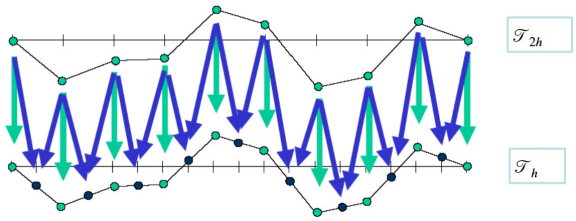
$$I_h^{2h}, \quad I_h^{2h} : \mathscr{T}_h \to \mathscr{T}_{2h}$$

The most obvious restriction operator is injection, defined by $I_h^{2h}\mathbf{w}_h = \mathbf{w}_{2h}$

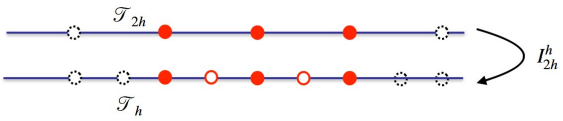
5.10 Interpolation operator (coarse-to-fine, 1D)

Mapping $I_{2h}^h : \mathscr{T}_{2h} \to \mathscr{T}_h$, i.e., $I_{2h}^h \mathbf{v}_{2h} = \mathbf{v}_h$ where

$$\mathbf{v}_{h,i} = \begin{cases} \mathbf{v}_{2h,i} & \text{if the node } i \text{ is common node of both } \mathscr{T}_h \text{ and } \mathscr{T}_{2h} \\ \dfrac{\mathbf{v}_{2h,i}^+ + \mathbf{v}_{2h,i}^-}{2} & \text{if the node } i \text{ in } \mathscr{T}_h \text{ is not a node in } \mathscr{T}_{2h} \end{cases}$$



$I_{2h}^h : \mathscr{T}_{2h} \to \mathscr{T}_h$ is a linear operator from $\mathbb{R}^n \to \mathbb{R}^m$



$$I_{2h}^h \mathbf{v}_{2h} = \begin{bmatrix} 1 & & \\ 1/2 & 1/2 & \\ & 1 & \\ & 1/2 & 1/2 \\ & & 1 \end{bmatrix} \begin{bmatrix} v_1^{2h} \\ v_2^{2h} \\ v_3^{2h} \end{bmatrix} = \begin{bmatrix} v_1^{2h} \\ (v_1^{2h} + v_2^{2h})/2 \\ v_2^{2h} \\ (v_2^{2h} + v_3^{2h})/2 \\ v_3^{2h} \end{bmatrix} = \begin{bmatrix} v_1^h \\ v_2^h \\ v_3^h \\ v_4^h \\ v_5^h \end{bmatrix}$$

- If the exact error of $\mathscr{T}_h$ is **smooth**, an interpolant of the coarse-grid error $\mathbf{e}_{2h}$ should give a good representation of the exact error.
- If the exact error of $\mathscr{T}_h$ is **oscillatory**, an interpolant of the coarse-grid error $\mathbf{e}_{2h}$ may give a poor representation of the exact error.

6 Algebraic multigrid methods

Algebraic multigrid (AMG) is based on MG principles but uses matrix coefficients instead of geometric information.

Classical AMG is based on the observation that the algebraic smooth error varies slowly in the direction of the matrix's relatively large (negative) coefficients. This gives us an algebraic way to track smooth errors (however, we still need to define "large").

Definition: strong connection

Given a threshold $\theta \in (0, 1)$, we say that i is **strongly connected** with j if

$$-a_{i,j} \geq \theta \max_{k \neq i}(-a_{i,k})$$

Let us denote with S_i the set of vertices that i is strongly connected to by

$$S_i = \{j \in N_i : i \text{ strongly connects to } j\}$$

where $N_i = \{j \neq i : a_{i,j} \neq 0\}$. This gives us a strength matrix S, with S_i as its i-th row.

6.1 Standard coarsening

Since algebraic smooth error varies slowly in the direction of strong connections, we should essentially coarsen in the direction of strong connections.

This results in a C/F-splitting such that F-vertices strongly connect to neighbouring C-vertices, and then the idea is to represent the values of a smooth vector at F-vertices as a linear combination of the values at C-vertices.

Given a strength matrix S, a general coarsening procedure consists of two steps.

Algorithm 4 General coarsening algorithm

1: Choose an independent set of C-vertices based on the graph of S.
2: Choose additional C-vertices in order to sarisfy the interpolation requirements.

For standard coarsening, the strength matrix S is constructed based on A directly, and the general coarsening algorithm is applied.

In step `1` of the algorithm, the independent set can be chosen in the following way:

1. Start with a first vertex i to become a C-vertex
2. Then, all vertices j which i strongly connect to become F-vertices.
3. Next, another vertex from undecided vertices becomes a C-vertex, and all vertices which are strongly connected to it become F-vertices.
4. This procedure is repeated until all vertices become C or F-vertices.

Remark: this procedure highly depends on the order of the vertices. To obtain a relatively uniform distribution of C/F vertices, a suitable measure of importance is introduced (C-AMG)

6.2 Direct interpolation

Let $\mathbf{e} = (e_1, e_2, \dots, e_i, \dots)$ be the error. A simple characterization of algebraic smooth error is $A\mathbf{e} \approx 0$. In other words,

$$a_{i,i}e + \sum_{j \in N_i} a_{i,j}e_j \approx 0 \quad i \in F \tag{4}$$

The idea is that we want to choose proper $w_{i,j}$ such that for any algebraic smooth errors

$$e_i \approx \sum_{j \in C} w_{i,j}e_j \quad j \in F$$

For $i \in F$, we define:

- $C_i = C \cap N_i$: C-points strongly connected to i
- $C_i^S = C \cap S_i$
- F_i^S : F-points strongly connected to i ($F_i = F \cap N_i$
- $N_i^W = \dfrac{N_i}{C_i \cup F_i^S}$: all points weakly connected to i

We can then rewrite (4) as follows (replace $\approx$ with $=$):

$$a_{i,i}e_i + \alpha \sum_{j \in C_i^S} a_{i,j}e_j = 0 \qquad \alpha = \frac{\sum_{j \in N} a_{i,j}}{\sum_{j \in C_i^S} a_{i,j}}$$

Therefore, the formula of direct interpolation is

$$w_{i,j} = \alpha \frac{a_{i,j}}{a_{i,i}}, \quad i \in F, j \in C_i^S, \qquad \alpha = \frac{\sum_{j \in N_i} a_{i,j}}{\sum_{j \in C_i^S} a_{i,j}}$$

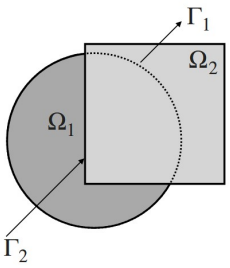
7 Domain decomposition methods

7.1 Alternating Schwarz method

Consider an elliptic partial differential equation of the form:

$$Lu = f \quad \text{in } \Omega = \Omega_1 \cup \Omega_2$$

with boundary condition $u = g$ on $\partial\Omega$



Given $u^{(0)}$,

1. On Ω_1, solve $\begin{cases} Lu_1^{\left(k+\frac{1}{2}\right)} = f & \text{in } \Omega_1 \\ u_1^{\left(k+\frac{1}{2}\right)} = g & \text{in } \partial\Omega_1 \setminus \Gamma_1 \\ u_1^{\left(k+\frac{1}{2}\right)} = u_2^{(k)} & \text{in } \Gamma_1 \end{cases}$

2. On Ω_2, solve $\begin{cases} Lu_2^{(k+1)} = f & \text{in } \Omega_2 \\ u_2^{(k+1)} = g & \text{in } \partial\Omega_2 \setminus \Gamma_2 \\ u_2^{(k+1)} = u_1^{\left(k+\frac{1}{2}\right)} & \text{in } \Gamma_2 \end{cases}$

3. Define $u^{(k+1)} = \begin{cases} u_1^{\left(k+\frac{1}{2}\right)} & \text{in } \Omega \setminus \Omega_2 \\ u_2^{(k+1)} & \text{in } \Omega_2 \end{cases}$

Alternating iterations continue until convergence to the solution u on the entire domain Ω.

7.2 Discretized Schwarz methods

Discretization yields a $n \times n$ symmetric positive definite linear algebraic system of the form $A\mathbf{x} = \mathbf{b}$.

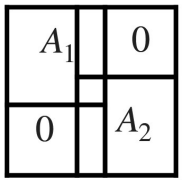
For $i = 1, 2$, let S_i be the set of n_i indices of grid points in the interior of Ω_i, where $n_i = |S_i|$. Because subdomains overlap, $S_1 \cap S_2 \neq \emptyset$ and $n_1 + n_2 > n$.

For $i = 1, 2$, let R_i be $n_i \times n$ the Boolean restriction matrix such that for any vector $\mathbf{v} \in \mathbb{R}^n$, $\mathbf{v}_i = R_i\mathbf{v} \in \mathbb{R}^{n_i}$ contains precisely those components of $\mathbf{v}$ corresponding to indices in S_i (i.e., those components associated to nodes in Ω_i).

Conversely, let $R^T \in \mathbb{R}^{n \times n_i}$ be the extension matrix that expands the vector $\mathbf{v}_i \in \mathbb{R}^{n_i}$ into a vector $\mathbf{v} = R_i^T\mathbf{v}_i \in \mathbb{R}^n$, whose components correspond to indices in S_i and are the same as those of $\mathbf{v}_i$. The remaining components are all zero.

The principal submatrices $A_i \in \mathbb{R}^{n_i \times n_i}, i = 1, 2$ of A correpsonding to two subdomains are given by

$$A_1 = R_1 A R_1^T \qquad A_2 = R_2 A R_2^T$$



For discretized problem, the alternating Schwarz iteration takes the following form

$$\mathbf{x}^{\left(k+\frac{1}{2}\right)} = \mathbf{x}^{(k)} + R_1^T A_1^{-1} R_1 \left(\mathbf{b} - A\mathbf{x}^{(k)}\right)$$
$$\mathbf{x}^{(k+1)} = \mathbf{x}^{\left(k+\frac{1}{2}\right)} + R_2^T A_2^{-1} R_2 \left(\mathbf{b} - A\mathbf{x}^{\left(k+\frac{1}{2}\right)}\right)$$

This method is called **multiplicative Schwarz method**.

Overall, the error $\mathbf{e}^{(k)} = \mathbf{x} - \mathbf{x}^{(k)}$ updated as $\mathbf{e}^{(k+1)} = B_{MS}\mathbf{e}^{(k)}$, where

$$B_{MS} = (I - R_2^T A_2^{-1} R_2 A)(I - R_1^T A_1^{-1} R_1 A)$$

As of yet, we have no parallelism, since the two subproblems must be solved sequentially for each iteration. But instead of Gauss-Seidel, we can use the block Jacobi approach, whose subproblems can be solved simultaneously. This method is called **Additive Schwarz method**.

Overall, the error $\mathbf{e}^{(k)} = \mathbf{x} - \mathbf{x}^{(k)}$ updated as $\mathbf{e}^{(k+1)} = B_{AS}\mathbf{e}^{(k)}$, where

$$B_{AS} = (R_2^T A_2^{-1} R_2 + R_1^T A_1^{-1} R_1)A$$

With either Gaauss-Seidel or Jacobi version, it can be shown that iteration converges at a rate independent of mesh size, provided overlap area between subdomains is sufficiently large.

7.3 Additive Schwarz preconditioner

Consider the additive Schwarz method

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + P_{ad}^{-1}\mathbf{r}^{(k)} \quad k \geq 0$$

with $P_{ad}^{-1} = (R_1^T A_1^{-1} R_1 + R_2^T A_2^{-1} R_2)$

Remember that the symmetry of the preconditioner means that it can be used also in conjunction with PCG, with preconditioner P_{ad} to accelerate convergence

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k P_{ad}^{-1}\mathbf{p}^{(k)} \quad k \geq 0$$

7.4 Symmetrized multiplicative Schwarz preconditioner

the multiplicative Schwarz iteration matrix is not symmetric, but can be made symmetric by additional step with A_1^{-1} each iteration.

The multiplicative Schwarz iteration matrix is not symmetric, but can be made symmetric by additional step with A_1^{-1} each iteration:

$$\begin{aligned}
\mathbf{x}^{\left(k+\frac{1}{3}\right)} &= \mathbf{x}^{(k)} + R_1^T A_1^{-1} R_1(\mathbf{b} - A\mathbf{x}^{(k)}) \\
\mathbf{x}^{\left(k+\frac{2}{3}\right)} &= \mathbf{x}^{\left(k+\frac{1}{3}\right)} + R_2^T A_2^{-1} R_2 \left(\mathbf{b} - A\mathbf{x}^{\left(k+\frac{1}{3}\right)}\right) \\
\mathbf{x}^{(k+1)} &= \mathbf{x}^{\left(k+\frac{2}{3}\right)} + R_1^T A_1^{-1} R_1 \left(\mathbf{b} - A\mathbf{x}^{\left(k+\frac{2}{3}\right)}\right)
\end{aligned}$$

which yields to a symmetric preconditioner that can be used in conjunction with PCG to accelerate convergence

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k P_{mus}^{-1}\mathbf{r}^{(k)} \quad k \geq 0$$

7.5 Many overlapping subdomains

To achieve a higher degree of parallelism with the Schwarz method, we can apply the two-domain algorithm recursively or use many subdomains. If there are p overlapping subdomains, then define matrices R_i and A_i as before, $i = 1, \ldots, p$.

The additive Schwarz preconditioner then takes the form

$$P_{ad}^{-1} = \sum_{i=1,\ldots,p} R_i^T A_i^{-1} R_i$$

The resulting generalization of block-Jacobi iteration is highly parallel, but not algorithmically scalable because the convergence rate degreades as p grows. The convergence rate can be restored by using a coarse grid correction to provide global coupling.

The additive Schwarz preconditioner then takes the form

$$P_{ad} = \sum_{i=0,\ldots,p} R_i^T A_i^{-1} R_i$$

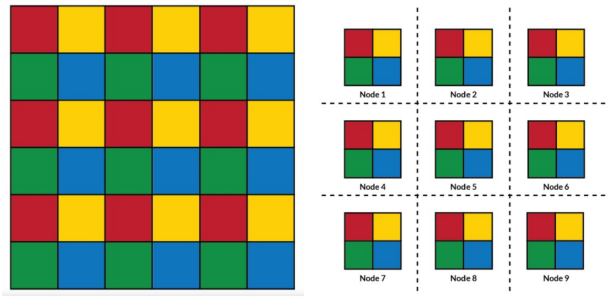
Multiplicative Schwarz iteration for p domains is defined analogously. As with classical Gauss-Seidel vs Jacobi, multiplicative Schwarz has a faster convergence rate than corresponging additive Schwarz (though it still requires coarse grid correction to remain scalable).

Unfortunatly, multiplicative Schwarz appears to provide no parallelism, as p subproblems per iteration must be solved sequentially. As with glassical Gauss-Seidel, parallelism can be introduced by **coloring** subdomains to identify independent subproblems that can be solved simultaneously.

7.6 Coloring techniques

The multiplicative Schwarz preconditioner is inherently serial. We must use a subdomain coloring mechanism in order to identify a set of subdomains that can be processed concurrently. This may limit the degree of parallelism if there is a low number of subdomains per color.

In general, the multiplicative Schwarz method converges faster than teh additive Schwarz method, while the latter can result in better parallel speedup.

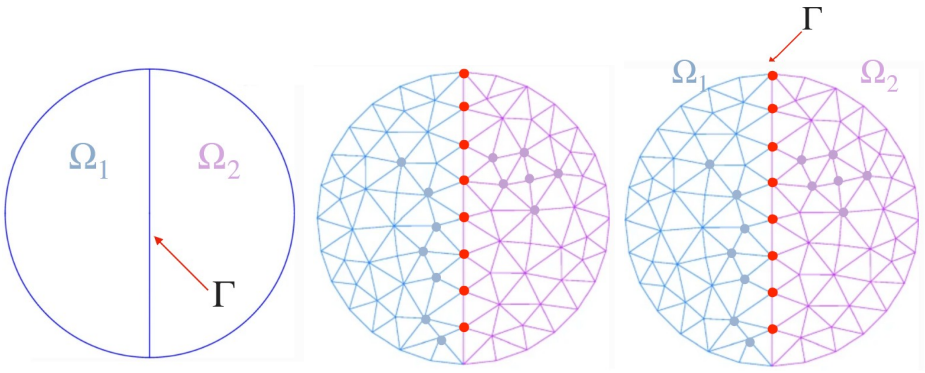


7.7 Non-overlapping subdomains

We now consider adjacent subdomains whose only points in common are along their mutual boundary Γ. We partition indices of unknowns in the corresponding discrete linear system into three sets:

- S_1 corresponding to interior nodes in Ω_1

- S_2 corresponding to interior nodes in Ω_2
- S_Γ corresponding to interface nodes in Γ



Partitioning matrix and right-hand-side vector accordingly, we obtain a symmetric block linear system

$$\begin{bmatrix} A_{11} & 0 & A_{1\Gamma} \\ 0 & A_{22} & A_{2\Gamma} \\ A_{1\Gamma}^T & A_{2\Gamma}^T & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_\Gamma \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_\Gamma \end{bmatrix}$$

Zero blocks result from assumption that nodes in Ω_1 are not directly connected to nodes in Ω_2, but ony through interface nodes in Ω_2, but only through interface nodes in Γ.

7.8 The Shur complement system

Definition: Shur complement

Consider the block LU factorization of matrix A

$$\begin{bmatrix} A_{11} & 0 & A_{1\Gamma} \\ 0 & A_{22} & A_{2\Gamma} \\ A_{1\Gamma}^T & A_{2\Gamma}^T & A_{\Gamma\Gamma} \end{bmatrix} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ A_{1\Gamma}^T & A_{11}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{11} & 0 & A_{1\Gamma} \\ 0 & A_{22} & A_{2\Gamma} \\ 0 & 0 & S \end{bmatrix}$$

where S is the **Shur complement**

$$S = A_{\Gamma\Gamma} - A_{1\Gamma}^T A_{11}^{-1} A_{1\Gamma} - A_{2\Gamma}^T A_{22}^{-1} A_{2\Gamma}$$

We can now determine interface unknowns $\mathbf{u}_\Gamma$ by solving the system

$$S\mathbf{x}_\Gamma = \tilde{\mathbf{b}}_\Gamma$$

where

$$\tilde{\mathbf{b}}_\Gamma = \mathbf{b}_\Gamma - A_{1\Gamma}^T A_{11}^{-1} \mathbf{b}_1 - A_{2\Gamma}^T A_{22}^{-1} \mathbf{b}_2$$

The remaining unknowns (which can be computed simultaneously) are then given by

$$\mathbf{x}_1 = A_{11}^{-1}(\mathbf{b}_1 - A_{1\Gamma}\mathbf{x}_\Gamma)$$
$$\mathbf{x}_2 = A_{22}^{-1}(\mathbf{b}_2 - A_{2\Gamma}\mathbf{x}_\Gamma)$$

The Shur complement matrix S is expensive to compute and is generally dense even if A is sparse. If the Schur complement system $S\mathbf{x}_\Gamma = \tilde{\mathbf{b}}_\Gamma$ is solved iteratively, tehn S doesn't need to be formed explicitly.

Matrix-vector multiplication by S requires teh solution in each subdomain, implicitly involving A_{11}^{-1} and A_{22}^{-1}, which can be done in parallel. The condition number of S is generally better than that of A, typically $O(h^{-1})$ instead of $O(h^{-1})$ for mesh size h. In practice, suitable interface preconditioners are still needed to accelertate convergence.

7.9 Many non-overlapping subdomains

To impropve parallelism with the Schur method, we can use many subdomains.

If there are p non-overlapping subdomains, let I be the set of indices of interior nodes of subdomains and, as before, let Γ be the set of indices of interface nodes. Then the discrete linear system has the following block form

$$\begin{bmatrix} A_{II} & A_{I\Gamma} \\ A_{II\Gamma}^T & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} \mathbf{x}_I \\ \mathbf{x}_\Gamma \end{bmatrix} = \begin{bmatrix} \mathbf{b}_I \\ \mathbf{b}_\Gamma \end{bmatrix}$$

The matrix A_{II} is block diagonal and has the following structure

$$A_{11} = \begin{bmatrix} A_{11} & 0 & \dots & 0 \\ 0 & A_{22} & \dots & 0 \\ 0 & \vdots & \ddots & 0 \\ 0 & \dots & 0 & A_{pp} \end{bmatrix}$$

As before, block LU factorization of matrix A yields a system

$$S\mathbf{x}_\Gamma = \tilde{\mathbf{b}}_\Gamma$$

where the Schur complement matrix S is given by

$$S = A_{\Gamma\Gamma} - A_{I\Gamma}^T A_{II}^{-1} A_{I\Gamma}$$

and $\tilde{\mathbf{b}}_\Gamma = \mathbf{b}_\Gamma - A_{I\Gamma}^T A_{II}^{-1} \mathbf{b}_I$

This system can be solved again iteratively without forming S explicitly. Suitable interface preconditioners can be used to accelerate convergence.

Interioi unknowns are then given by

$$\mathbf{x}_I = A_{II}^{-1}(\mathbf{b}_I - A_{I\Gamma}\mathbf{x}_\Gamma)$$

All the occurrences of A_{II}^{-1} can be performed on all subdomains in parallel because A_{II} is block diagonal.