FSB

100

100 godina Fakulteta strojarstva i brodogradnje Sveučilišta u Zagrebu

100 Years of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb

Iterative linear solvers: Understanding the `fvSolution` dictionary

Tessa Uroić, Ph.D.

Faculty of Mechanical Engineering and Naval Architecture
University of Zagreb

tessa.uroic@fsb.hr
https://foam-extend.fsb.hr

OpenFOAM Workshop 17, Cambridge

July 11, 2022

FSB

IOO

Overview

The objective and learning outcomes

The objective of this training is to introduce iterative procedures for solving sparse linear systems which arise from the finite volume discretisation of partial differential equations.

The questions answered during this training session:

- What are eigenvectors and eigenvalues of a matrix?
- What are residuals and why are they useful? What is the relationship between the residual and the error?
- What is algebraic smoothness and how we exploit it?
- What is a projection operator and how to construct one?
- What is positive-definiteness of a matrix?
- What is conjugacy and how can we exploit it?
- What is a preconditioner and how to construct one?

This lecture was partially inspired by J.R. Shewchuk (An introduction to the Conjugate Gradient Method without the agonizing pain, 1994).

Introduction

There is no analytical solution of the incompressible, single-phase fluid flow equations:

$$\underbrace{\nabla \cdot \mathbf{u}}_{\text{velocity divergence}} = 0,$$

$$\underbrace{\left(\frac{\partial \mathbf{u}}{\partial t}\right)^T}_{\text{local rate of change}} + \underbrace{\nabla \cdot (\mathbf{u}\mathbf{u}^T)}_{\text{convection}} - \underbrace{\nabla \cdot (\nu \nabla \mathbf{u}^T)}_{\text{diffusion}} = - \underbrace{(\nabla p)^T}_{\text{pressure gradient}} .$$

However, we can solve a discrete representation of these partial differential equations. To employ the *finite volume discretisation* schemes, we shall divide the domain of interest into n control volumes (cells), which do not overlap (*computational mesh*). PDEs are transformed into linear(ised) equations:

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

$$\begin{bmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,n} \\ a_{1,0} & a_{1,1} & \dots & a_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,0} & a_{n,1} & \dots & a_{n,n} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}$$

and the solution $\mathbf{p}$ is calculated as a *field of values* stored in cell centroids.
Think, what is the connection between the spatial discretisation (mesh) and the coefficient matrix of the linear system?

FSB IOO

A few words about matrices

- Each equation is represented by a single matrix row. If we are solving a single PDE discretised over n **cells**, the **dimensions of the matrix are** $n \times n$.
- Matrices arising from finite volume discretisation are ***sparse***, i.e. they have many more zero than non–zero coefficients. We do not want to store zeros!
- Since there are many zeros, the **non–zero coefficients form a pattern**. The pattern can have a regular structure, e.g. the matrix can be banded.
- The upper triangle coefficients can be a mirrored image of the lower triangle coefficients - the matrix is symmetric. For example, FV discretisation of a Laplacian operator produces a ***symmetric matrix***.
- If the coefficients in the lower and upper triangle are not the same, the matrix is ***skew symmetric***.
- Linear systems can be solved by either direct or iterative methods. Direct methods are used for dense or very small matrices, while for FV matrices we employ ***iterative algorithms***.
- Iterative algorithms prefer matrices with distinct structure (optimisation and efficiency - `renumberMesh`) and ***diagonally dominant matrix rows***. A row is diagonally dominant if the magnitude of the diagonal coefficient is larger than the sum of magnitudes of off–diagonal coefficients in that row.

Can we exploit the properties of the matrix to find an optimal iterative solution algorithm?

What are eigenvectors?

- **What is an eigenvector?** An "eigenvector" $\mathbf{v}$ of a matrix $\mathbf{B}$ is a nonzero vector that does not rotate when the matrix is applied (multiplied) to it (except if it points precisely the opposite direction) - the eigenvector might change length or reverse its direction, but it won't turn sideways.
- In other words, there is some scalar constant λ such that

$$\mathbf{B}\mathbf{v} = \lambda\mathbf{v}.$$

- The value λ is an "eigenvalue" of $\mathbf{B}$.
- **Why are eigenvectors significant?** Iterative methods often depend on applying $\mathbf{B}$ to a vector over and over again. When $\mathbf{B}$ is repeatedly applied to an eigenvector $\mathbf{v}$, two things can happen:
  - If $|\lambda| < 1$, then $\mathbf{B}^i\mathbf{v} = \lambda^i\mathbf{v}$ will vanish as i approaches infinity.
  - If $|\lambda| > 1$, then $\mathbf{B}^i\mathbf{v}$ will grow to infinity.

What are eigenvectors?

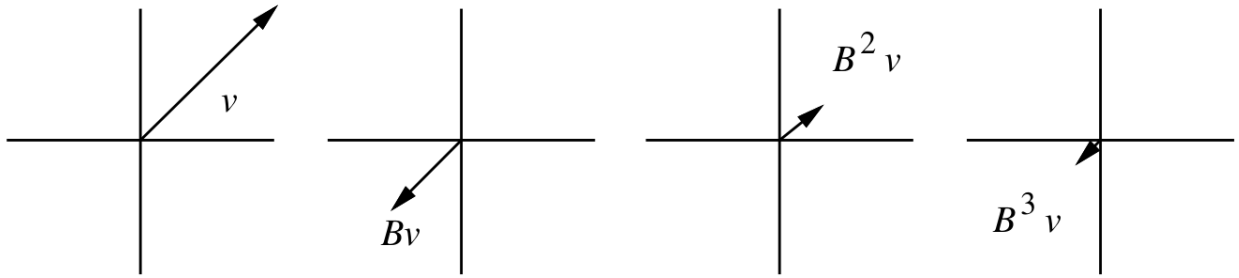


Figure 1: $\mathbf{v}$ is an eigenvector of $\mathbf{B}$ with a corresponding eigenvalue of -0.5. As i increases, $\mathbf{B}^i\mathbf{v}$ converges to zero.

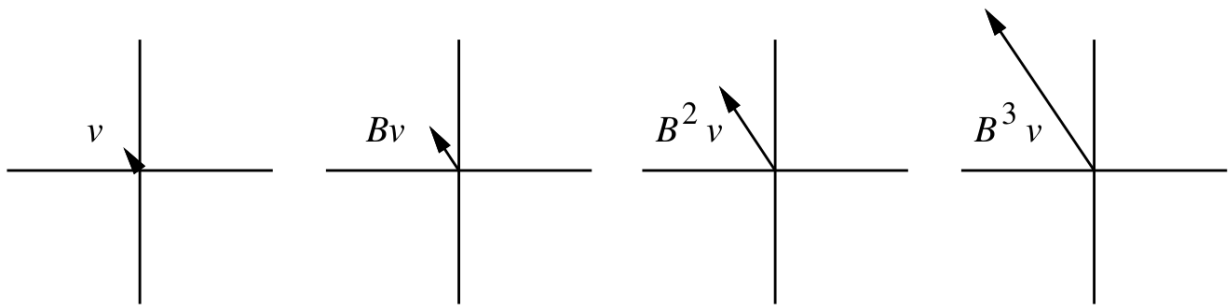


Figure 2: v is an eigenvector of $\mathbf{B}$ with a corresponding eigenvalue of 2. As i increases, $\mathbf{B}^i\mathbf{v}$ diverges to infinity.

Each time $\mathbf{B}$ is applied, the vector grows or shrinks according to the value of λ.

What are eigenvectors?

- **But, what if $\mathbf{B}$ is applied to a vector that is not an eigenvector?** Think of a vector as a sum of other vectors whose behaviour is understood.
- Consider that the set of eigenvectors $\mathbf{v}_i$ forms a basis for $\mathbb{R}^n$ (because a symmetric matrix has n eigenvectors that are linearly independent). Any n–dimensional vector can be expressed as a linear combination of these eigenvectors.
- Matrix-vector multiplication is distributive, so we can examine the effect of $\mathbf{B}$ on each eigenvector separately.
- Applying $\mathbf{B}$ to a vector $\mathbf{x} = \mathbf{v}_1 + \mathbf{v}_2$ is equivalent to applying $\mathbf{B}$ to the eigenvectors and summing the result.
- On repeated application: $\mathbf{B}^i\mathbf{x} = \mathbf{B}^i\mathbf{v}_1 + \mathbf{B}^i\mathbf{v}_2 = \lambda_1^i\mathbf{v}_1 + \lambda_2^i\mathbf{v}_2$
- If the magnitudes of all eigenvalues are smaller than one, $\mathbf{B}^i\mathbf{x}$ will converge to zero, because the eigenvectors that compose $\mathbf{x}$ converge to zero when $\mathbf{B}$ is repeatedly applied. If one of the magnitudes is greater than one, $\mathbf{x}$ will diverge into infinity.

Eigenvalues and error smoothing

Relation between the residual and error:

$$\mathbf{r}^{(i)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(i)}$$

$$\mathbf{r}^{(i)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(i)} = \mathbf{A}\mathbf{x} - \mathbf{A}\mathbf{x}^{(i)} = \mathbf{A}(\mathbf{x} - \mathbf{x}^{(i)})$$

$$\mathbf{r}^{(i)} = \mathbf{A}\mathbf{e}^{(i)}$$

- Residual is used as an indicator for the size of the error (which is not known).
- If the error is an eigenvector $\mathbf{e}^{(i)} = \mathbf{v}_i$:

$$\mathbf{A}\mathbf{v}_i = \mathbf{r}^{(i)} = \lambda \mathbf{v}_i$$

- Be careful: Large ***condition number*** (*ratio of largest to smallest eigenvalue*) means that the residual doesn't measure the error well. **Small residual, but large error!**
- It is possible to solve the residual equation and use it to compute a correction to the solution $\mathbf{x}^{(i)}$:

$$\mathbf{x}^{(i)} = \mathbf{x}^{(i-1)} + \mathbf{e}^{(i)}.$$

Gauss–Seidel method

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & & & & & & \\ a_{10} & a_{11} & & a_{13} & a_{14} & & & & \\ a_{20} & & a_{22} & & a_{24} & a_{25} & & & \\ & a_{31} & & a_{33} & & & a_{36} & & \\ & a_{41} & a_{42} & & a_{44} & & a_{46} & a_{47} & \\ & & a_{52} & & & a_{55} & & a_{57} & \\ & & & a_{63} & a_{64} & & a_{66} & & a_{68} \\ & & & & a_{74} & a_{75} & & a_{77} & a_{78} \\ & & & & & & a_{86} & a_{87} & a_{88} \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \end{bmatrix}$$

- Gauss-Seidel principle:

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

$$\mathbf{A} = \mathbf{U} + \mathbf{D} + \mathbf{L}$$

$$(\mathbf{U} + \mathbf{D} + \mathbf{L})\,\mathbf{x} = \mathbf{b}$$

$$\mathbf{U}\mathbf{x} + (\mathbf{D} + \mathbf{L})\,\mathbf{x} = \mathbf{b}$$

$$(\mathbf{D} + \mathbf{L})\,\mathbf{x}^{(i+1)} = \mathbf{b} - \mathbf{U}\,\mathbf{x}^{(i)}$$

$$\mathbf{x}^{(i+1)} = \mathbf{D}^{-1}(\mathbf{b} - \mathbf{L}\mathbf{x}^{(i+1)} - \mathbf{U}\mathbf{x}^{(i)})$$

Gauss–Seidel: example

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & & & & & & \\ a_{10} & a_{11} & & a_{13} & a_{14} & & & & \\ a_{20} & & a_{22} & & a_{24} & a_{25} & & & \\ & a_{31} & & a_{33} & & & a_{36} & & \\ & a_{41} & a_{42} & & a_{44} & & a_{46} & a_{47} & \\ & & a_{52} & & & a_{55} & & a_{57} & \\ & & & a_{63} & a_{64} & & a_{66} & & a_{68} \\ & & & & a_{74} & a_{75} & & a_{77} & a_{78} \\ & & & & & & a_{86} & a_{87} & a_{88} \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \end{bmatrix}$$

Forward sweep:

$$x_0^{(i+1)} = \frac{1}{a_{00}}(b_0 - a_{01}x_1^{(i)} - a_{02}x_2^{(i)})$$

$$x_1^{(i+1)} = \frac{1}{a_{11}}(b_1 - a_{10}x_0^{(i+1)} - a_{13}x_3^{(i)} - a_{14}x_4^{(i)})$$

Backward sweep:

$$x_8^{(i+1)} = \frac{1}{a_{88}}(b_8 - a_{87}x_7^{(i)} - a_{86}x_6^{(i)})$$

$$x_7^{(i+1)} = \frac{1}{a_{77}}(b_7 - a_{78}x_8^{(i+1)} - a_{75}x_5^{(i)} - a_{74}x_4^{(i)})$$

Gauss–Seidel converges for strictly diagonally dominant or symmetric and positive–definite matrices.

Algebraic smoothness - reasoning

- In geometric multigrid, the error term on the fine mesh can be characterised as smooth if it can be well approximated on the coarse mesh (smoothness of a function).
- The definition of error smoothness in the context of algebraic multigrid is not the same, as the coarse mesh does not exist. In AMG, an error is called smooth if it cannot be eliminated, i.e. the chosen stationary point method stalls.
- The iterative solution of a linear system $\mathbf{Ax} = \mathbf{b}$ can be written in the following form:

$$\mathbf{x}^{(k+1)} = (\underbrace{\mathbf{I} - \mathbf{A}}_{\mathbf{S}})\mathbf{x}^{(k)} + \mathbf{b},$$

  where $\mathbf{I}$ is the identity matrix and $(\mathbf{I} - \mathbf{A})$ is the *iteration matrix* $\mathbf{S}$, i.e. the smoother.
- The point–fixed iteration expressed in terms of the solution error is:

$$\mathbf{e}^{(k+1)} = (\mathbf{I} - \mathbf{A})\mathbf{e}^{(k)} = \mathbf{e}^{(k)} - \mathbf{A}\mathbf{e}^{(k)}.$$

Algebraic smoothness - reasoning

- The goal is to eliminate the error to reach the correct solution, i.e. $\mathbf{e}^{(k+1)} = 0$, which gives:

$$\mathbf{e}^{(k)} - \mathbf{A}\mathbf{e}^{(k)} = 0.$$

- The error term can be represented as a linear combination of eigenvectors of the coefficient matrix $\mathbf{A}$:

$$\mathbf{e} = \sum_{i}^{n} \zeta_i \cdot \mathbf{v}_i.$$

- Thus, the components of the error which have the corresponding eigenvalue closest to 1 will be eliminated by the point–fixed method the fastest. These error components are called high–frequency errors.
- The components which have small eigenvalues will be the slowest to converge, as the component $\zeta_j \cdot \mathbf{v}_j$ will not be reduced significantly by the corresponding scaled component of the term $\mathbf{A}\mathbf{e}^{(k)}$, and are called low–frequency errors.

The idea behind AMG

- It was noticed that the global, low–frequency errors are removed at a rate inversely proportional to the size of the computational mesh.
- ***The low–frequency errors on a fine mesh become high–frequency errors on a coarse mesh and can be efficiently eliminated.***
- Multigrid algorithms help reduce the low–frequency error by successively solving the system on coarser meshes.

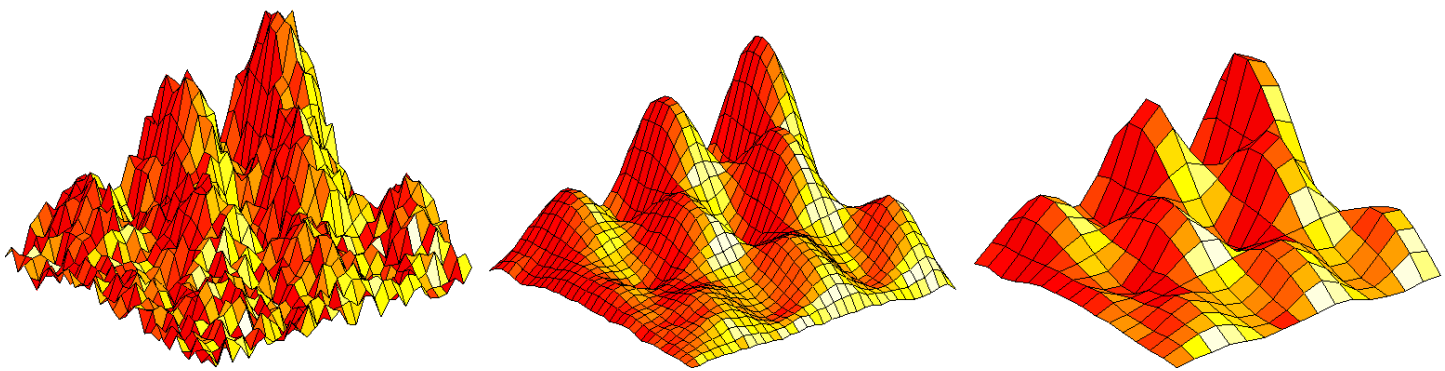
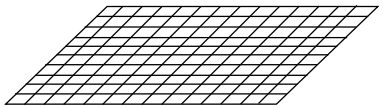


Figure 3: Effects of error smoothing and restriction. *(by Robert Falgout)*

Multigrid two-level cycle



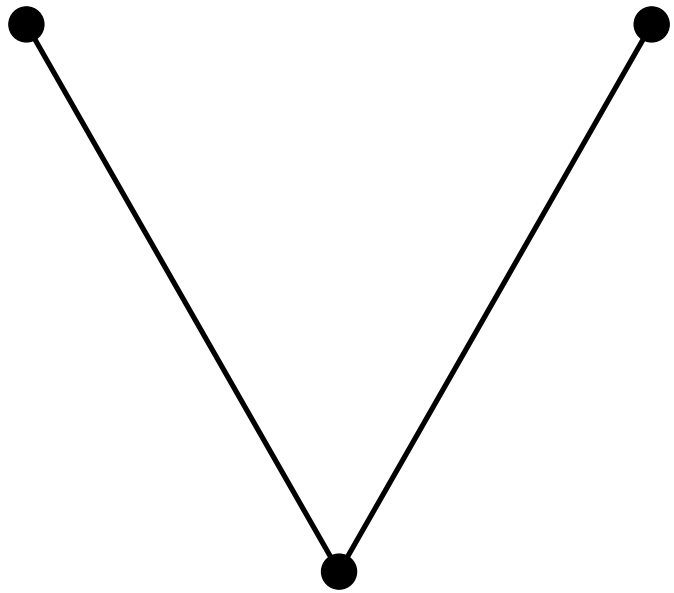
FINE LEVEL

$\mathbf{A}^{\mathrm{F}}\mathbf{x} = \mathbf{b}$

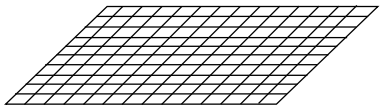
$\text{smooth} \rightarrow \bar{\mathbf{x}}^{\mathrm{F}}$

$\mathbf{r}^{\mathrm{F}} = \mathbf{b} - \mathbf{A}^{\mathrm{F}}\bar{\mathbf{x}}^{\mathrm{F}}$

$= \mathbf{A}^{\mathrm{F}}\bar{\mathbf{e}}^{\mathrm{F}}$



Multigrid two-level cycle



FINE LEVEL

$\mathbf{A}^{\mathrm{F}}\mathbf{x} = \mathbf{b}$

smooth $\rightarrow \bar{\mathbf{x}}^{\mathrm{F}}$

$\mathbf{r}^{\mathrm{F}} = \mathbf{b} - \mathbf{A}^{\mathrm{F}}\bar{\mathbf{x}}^{\mathrm{F}}$

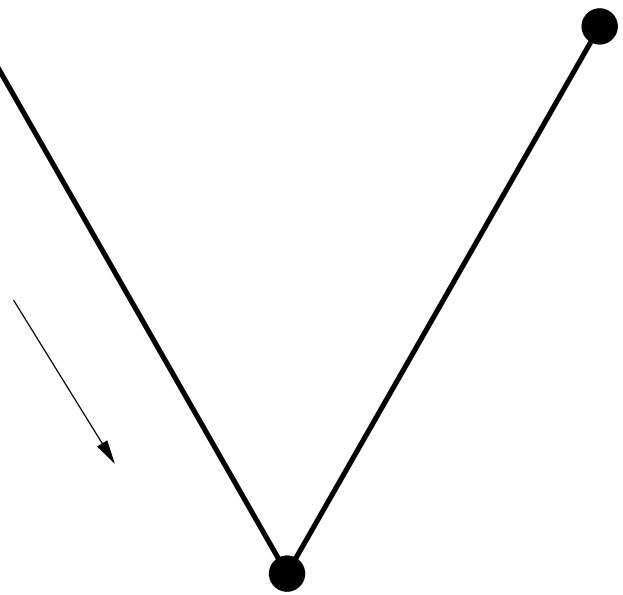
$= \mathbf{A}^{\mathrm{F}}\bar{\mathbf{e}}^{\mathrm{F}}$

restrict:

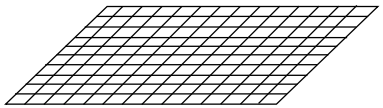
$\mathbf{r}^{\mathrm{C}} = \mathbf{R}\mathbf{r}^{\mathrm{F}}$

calculate:

$\mathbf{A}^{\mathrm{C}} = \mathbf{R}\mathbf{A}^{\mathrm{F}}\mathbf{P}$



Multigrid two-level cycle



FINE LEVEL

$\mathbf{A}^{\mathrm{F}}\mathbf{x} = \mathbf{b}$

smooth $\rightarrow \bar{\mathbf{x}}^{\mathrm{F}}$

$\mathbf{r}^{\mathrm{F}} = \mathbf{b} - \mathbf{A}^{\mathrm{F}}\bar{\mathbf{x}}^{\mathrm{F}}$

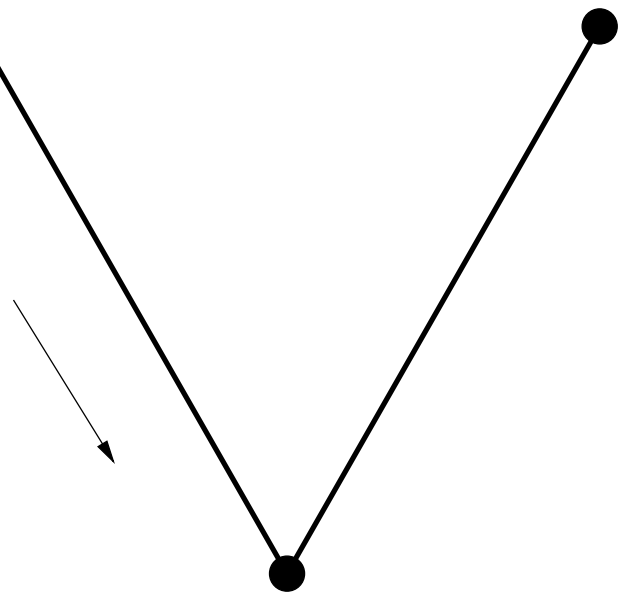
$= \mathbf{A}^{\mathrm{F}}\bar{\mathbf{e}}^{\mathrm{F}}$

restrict:

$\mathbf{r}^{\mathrm{C}} = \mathbf{R}\mathbf{r}^{\mathrm{F}}$

calculate:

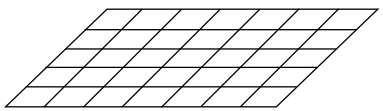
$\mathbf{A}^{\mathrm{C}} = \mathbf{R}\mathbf{A}^{\mathrm{F}}\mathbf{P}$



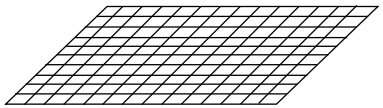
COARSE LEVEL

solve: $\mathbf{A}^{\mathrm{C}}\mathbf{e}^{\mathrm{C}} = \mathbf{r}^{\mathrm{C}}$

$\mathbf{e}^{\mathrm{C}} = (\mathbf{A}^{\mathrm{C}})^{-1}\mathbf{r}^{\mathrm{C}}$



Multigrid two-level cycle



FINE LEVEL

$\mathbf{A}^{\mathrm{F}}\mathbf{x} = \mathbf{b}$

smooth $\rightarrow \bar{\mathbf{x}}^{\mathrm{F}}$

$\mathbf{r}^{\mathrm{F}} = \mathbf{b} - \mathbf{A}^{\mathrm{F}}\bar{\mathbf{x}}^{\mathrm{F}}$

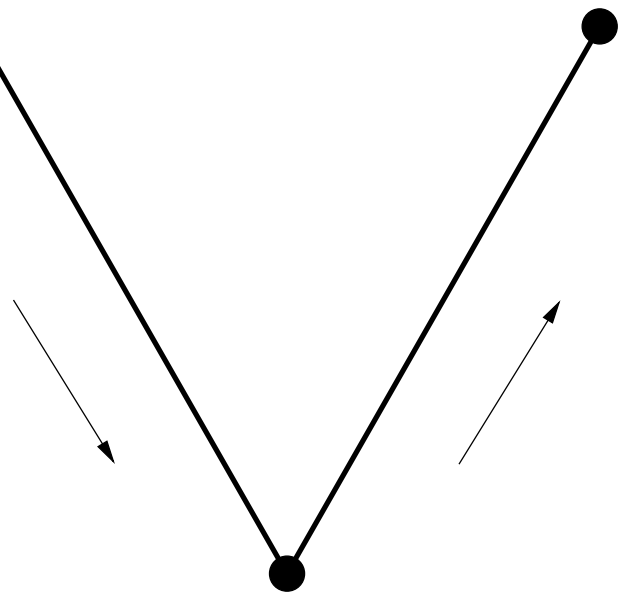
$= \mathbf{A}^{\mathrm{F}}\bar{\mathbf{e}}^{\mathrm{F}}$

restrict:

$\mathbf{r}^{\mathrm{C}} = \mathbf{R}\mathbf{r}^{\mathrm{F}}$

calculate:

$\mathbf{A}^{\mathrm{C}} = \mathbf{R}\mathbf{A}^{\mathrm{F}}\mathbf{P}$



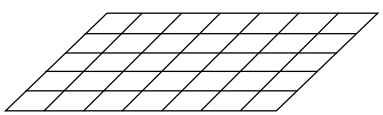
prolongate:

$\mathbf{e}^{\mathrm{F}} = \mathbf{P}\mathbf{e}^{\mathrm{C}}$

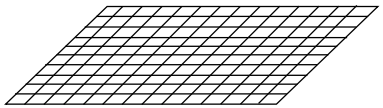
COARSE LEVEL

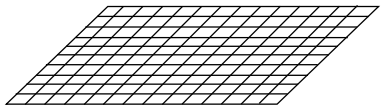
solve: $\mathbf{A}^{\mathrm{C}}\mathbf{e}^{\mathrm{C}} = \mathbf{r}^{\mathrm{C}}$

$\mathbf{e}^{\mathrm{C}} = (\mathbf{A}^{\mathrm{C}})^{-1}\mathbf{r}^{\mathrm{C}}$



Multigrid two-level cycle





FINE LEVEL

$\mathbf{A}^{\mathrm{F}}\mathbf{x} = \mathbf{b}$

smooth $\rightarrow \bar{\mathbf{x}}^{\mathrm{F}}$

$\mathbf{r}^{\mathrm{F}} = \mathbf{b} - \mathbf{A}^{\mathrm{F}}\bar{\mathbf{x}}^{\mathrm{F}}$

$= \mathbf{A}^{\mathrm{F}}\bar{\mathbf{e}}^{\mathrm{F}}$

correct: $\mathbf{x}^{\mathrm{F}} = \bar{\mathbf{x}}^{\mathrm{F}} + \mathbf{e}^{\mathrm{F}}$

smooth $\rightarrow \mathbf{x}^{\mathrm{F}}$

restrict:

$\mathbf{r}^{\mathrm{C}} = \mathbf{R}\mathbf{r}^{\mathrm{F}}$

calculate:

$\mathbf{A}^{\mathrm{C}} = \mathbf{R}\mathbf{A}^{\mathrm{F}}\mathbf{P}$

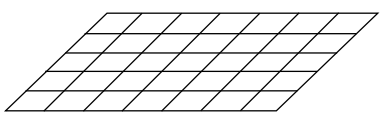
prolongate:

$\mathbf{e}^{\mathrm{F}} = \mathbf{P}\mathbf{e}^{\mathrm{C}}$

COARSE LEVEL

solve: $\mathbf{A}^{\mathrm{C}}\mathbf{e}^{\mathrm{C}} = \mathbf{r}^{\mathrm{C}}$

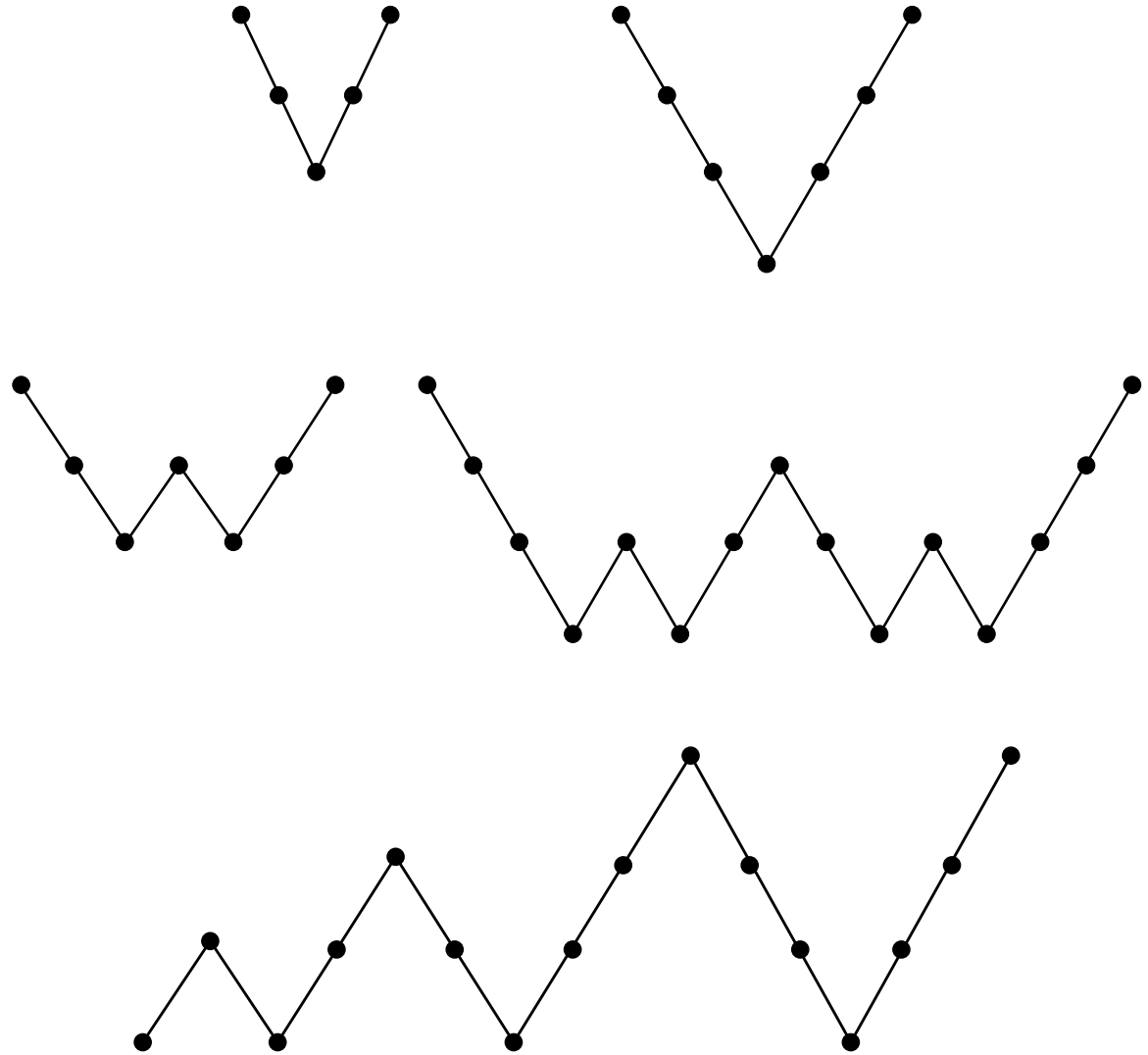
$\mathbf{e}^{\mathrm{C}} = (\mathbf{A}^{\mathrm{C}})^{-1}\mathbf{r}^{\mathrm{C}}$



Multigrid cycle

- Effects of a two level multigrid V–cycle can be represented by a matrix $\mathbf{M} = \mathbf{P}(\mathbf{R}\mathbf{A}^{\mathsf{F}}\mathbf{P})^{-1}\mathbf{R}\mathbf{A}^{\mathsf{F}}$.
- The eigenvalues of matrix $\mathbf{M}$ are $\lambda_1 = 0$ and $\lambda_2 = 1$.
- The components of the error with the eigenvalue $\lambda_2 = 1$ will be completely eliminated, while multigrid will have no effect on the components with the eigenvalue $\lambda_1 = 0$.
- Since a single two level multigrid V–cycle has absolutely no effect on some components of the error, there is no point in doing multiple cycles.
- This is where point–fixed methods complement the multigrid two level cycle. Application of these methods, which smooth out the solution, is performed on the fine level system before the calculation of the residual (pre–smoothing) and after correction of the solution (post–smoothing).
- It is uncertain how many pre– and post–smoothing sweeps are optimal for convergence (usually 2 and 2) but it was shown that it is beneficial to recursively repeat the multigrid algorithm, i.e. create multiple coarse levels and move through them while repeating all the steps of a two level algorithm.

Choice of cycle



FSB IOO

Algebraic smoothness - pointer for formulating the interpolation

- In the context of multigrid, methods which efficiently reduce the high–frequency components of the error are called *smoothers*.
- Residual ($\mathbf{r}^{(k)} = \mathbf{A}\mathbf{e}^{(k)}$) does not have to be a good indication of the correct solution since a small residual doesn't necessarily imply a small error: error scaled by the coefficient matrix $\mathbf{A}$ is much smaller than the error itself.
- The role of the algebraic multigrid algorithm is to identify the direction of low–frequency components of the error, turn them into high–frequency errors on the coarse mesh, and use the smoother to efficiently eliminate them.
- Smooth (low–frequency) components of the error are characterised by slow reduction using the point–fixed method, i.e. the error does not significantly change between two iterations. From that, an equation which describes the algebraically smooth error is derived:

$$a_{ii}e_i + \sum_{j \in \mathbb{N}_i} a_{ij}e_j = 0.$$

- The equation originates from the limitations of the point–fixed methods, thus it is natural to exploit it for the definition of coarse levels, i.e. for the interpolation of the correction term from coarse level equations into the fine level equations.

AMG coarsening algorithms

Options for forming interpolation by injection or weighted residual method:

- **Aggregative multigrid - AAMG**
  - Equations are grouped into clusters (multiple fine cells are joined to form a single coarse cell).
  - Groups of cells are formed based on the size of off-diagonal coefficients: $\frac{a_{ij}^2}{a_{ii}a_{jj}}$ should exceed a certain size.
  - The interpolation operator is piecewise constant (injection).
  - The coarse level matrix coefficients are obtained as a sum of fine level matrix coefficients.
- **Selective multigrid - SAMG**
  - The set of coarse equations is a subset of fine level equations.
  - The choice of the coarse equations is based on the number of equations which depend on a single equation.
  - The interpolation operator is obtained from approximation of the algebraically smooth error.
  - Coarse level matrix is obtained by a triple matrix product of restriction, fine level matrix and prolongation operator (Galerkin weighted residuals method).

Choice of multigrid smoothers

- Classical stationary–point methods: Jacobi, Gauss–Seidel, symmetrical Gauss–Seidel...
- More effective (but also more computationally expensive) are matrix factorizations such as incomplete LU factorization (**ILU**).
- Smoothing is done before the restriction of the residual (*pre-smoothing*) and after the correction of the solution (*post-smoothing*). Post-smoothing should be more efficient.
- The fine level solver can be a Krylov subspace method. For obtaining a solution on the coarsest level, a direct solver can be used (eg. Gaussian elimination).

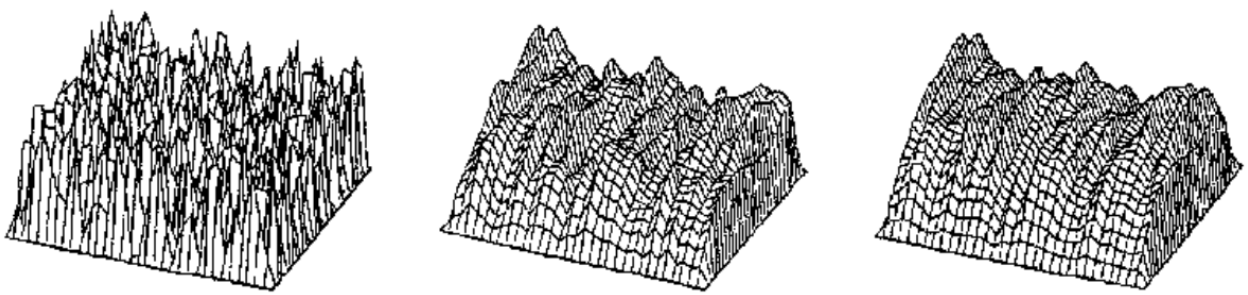


Figure 4: Effect of the Gauss–Seidel smoother

What is a positive definite matrix?

- A matrix is positive-definite if, for every nonzero vector $\mathbf{x}$:

$$\mathbf{x}^T \mathbf{A} \mathbf{x} > 0$$

- It is not a very intuitive idea, and it's hard to imagine how a matrix that is positive-definite might look differently from one that isn't. We will get a feeling for what positive-definiteness is about when we see how it affects the shape of quadratic forms.
- A quadratic form is simply a scalar, quadratic function of a vector with the form

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

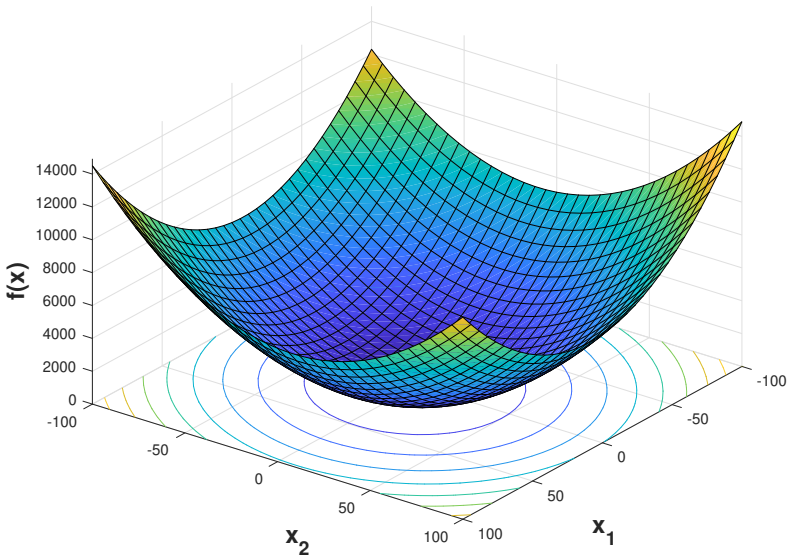
where $\mathbf{A}$ is a matrix, $\mathbf{x}$ and $\mathbf{b}$ are vectors, and c is a scalar constant.

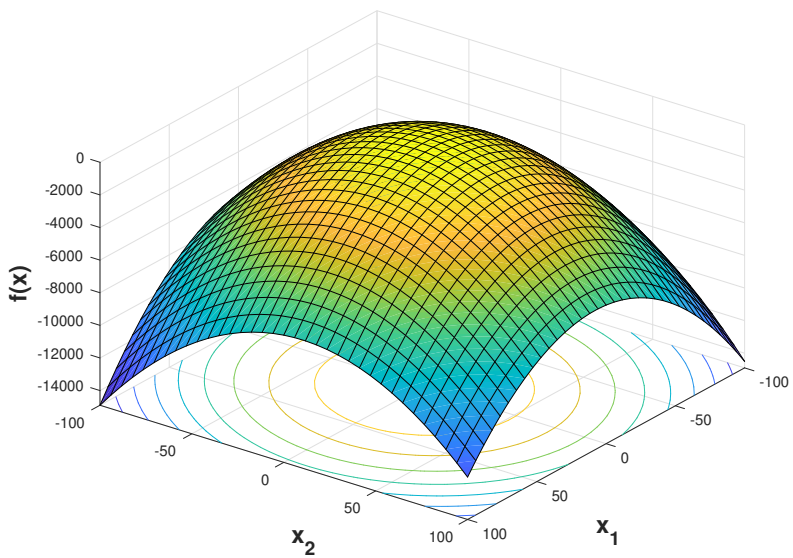
What is a positive definite matrix?

A linear system, $\mathbf{Ax} = \mathbf{b}$, can be reformulated into a minimisation problem, where the objective is to find an extreme value (minimum or maximum) of the quadratic function:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{Ax} - \mathbf{b}^T\mathbf{x} + c.$$

For a system consisting of two equations, it is easy to draw the quadratic function: see example below. $f(\mathbf{x})$ in the shape of a convex paraboloid belongs to a positive definite matrix, while a concave paraboloid corresponds to a quadratic function of a negative definite matrix. Both of these types of matrices have a quadratic function with a clear extremum, a minimum for a positive definite matrix and maximum for a negative definite matrix.





What is a positive definite matrix?

The quadratic function can be represented in two dimensions, using the isocontours which connect the points with the same value of $f(\mathbf{x})$. Black arrows represent the eigenvectors of the matrix with the corresponding eigenvalues. The eigenvector with the larger eigenvalue coincides with the minor axis of the elliptic isocontour, while the eigenvector with the smaller eigenvalue coincides with the major axis.

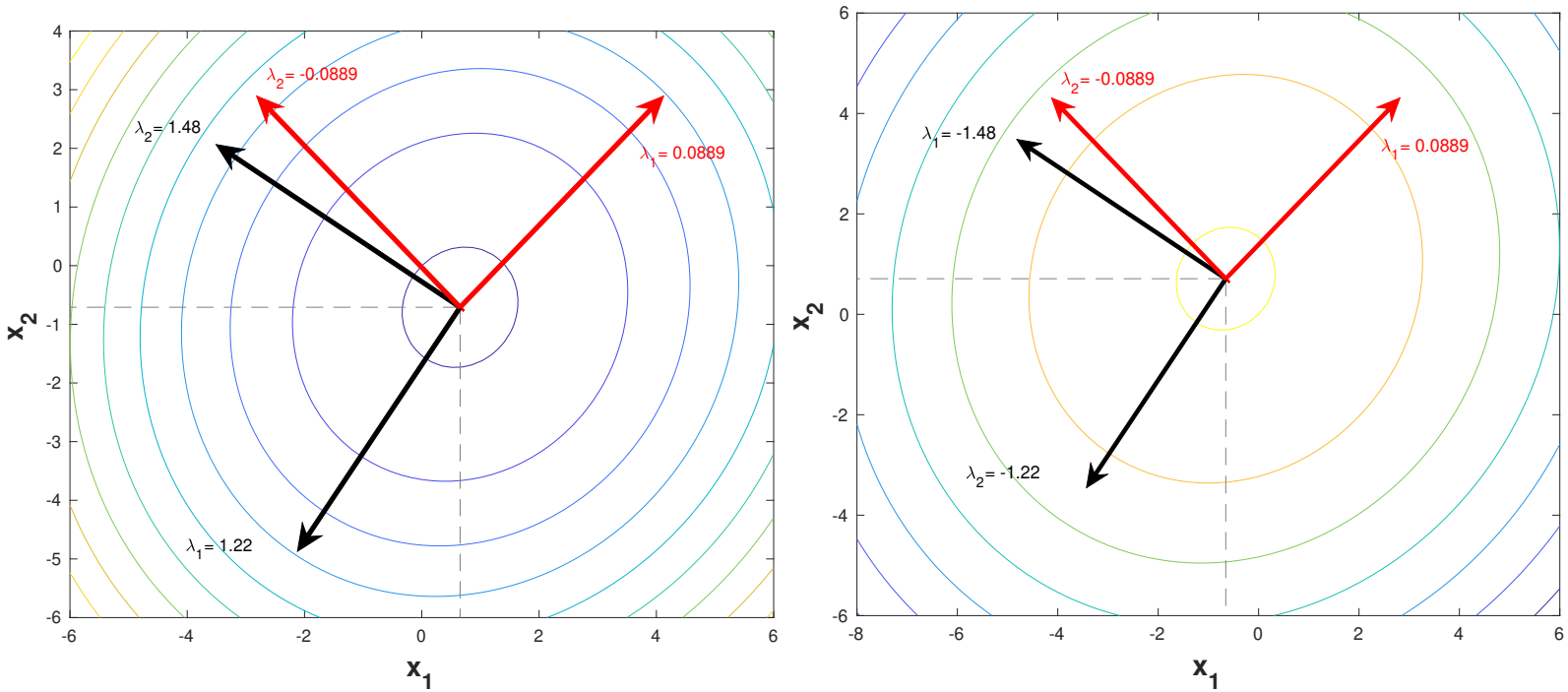


Figure 5: Isocontours of a quadratic function corresponding to a positive definite matrix (left) and a negative definite matrix (right). The red arrows represent the eigenvectors of a Jacobi preconditioned matrix $\mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$.

Quadratic form

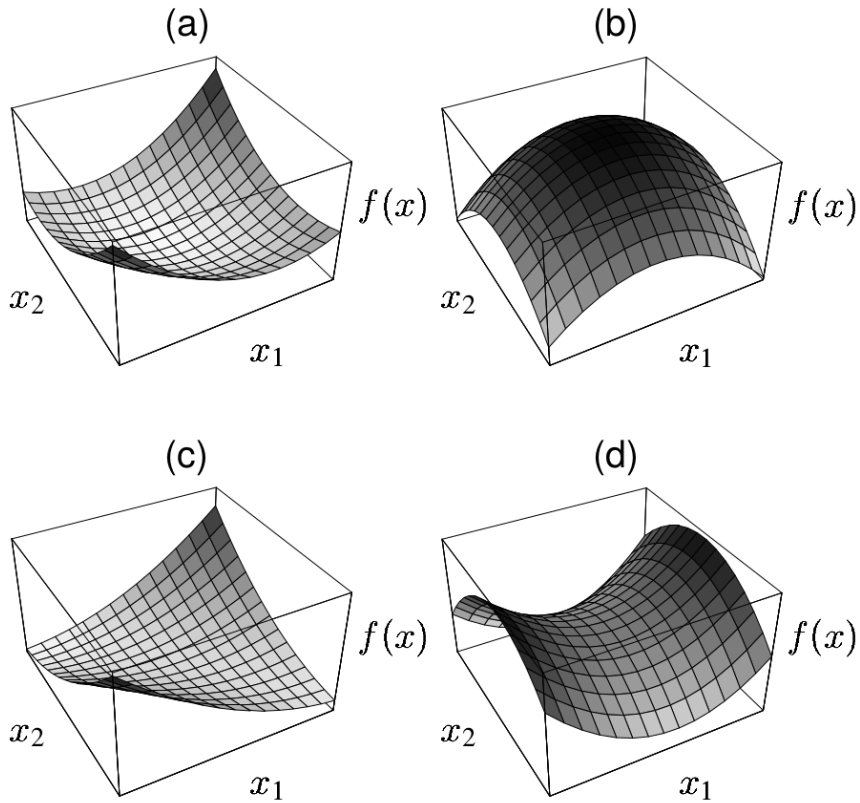


Figure 6:
(a) Quadratic form for a positive-definite matrix.
(b) For a negative-definite matrix.
(c) For a singular (and positive-indefinite) matrix. A line that runs through the bottom of the valley is the set of solutions.
(d) For an indefinite matrix. Because the solution is a saddle point, Steepest Descent and CG will not work. In three dimensions or higher, a singular matrix can also have a saddle.

Method of Steepest descent

At the extremum of the quadratic function, the gradient of the function is equal to 0:

$$f'(\mathbf{x}) = \frac{1}{2}\mathbf{A}^T\mathbf{x} + \frac{1}{2}\mathbf{A}\mathbf{x} - \mathbf{b} = 0.$$

For a symmetric ($\mathbf{A} = \mathbf{A}^T$) positive definite matrix, it reduces to:

$$f'(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b} = 0,$$

which is equal to the initial linear system. Thus, the solution of the system is a minimum of the quadratic function. For every point

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

it is possible to calculate the value of the gradient of the quadratic function, which is a vector and it can be seen that for a symmetric positive definite matrix, it is equal to the negative residual:

$$f'(\mathbf{x}) = -(\mathbf{b} - \mathbf{A}\mathbf{x}) = -\mathbf{r}.$$

FSB IOO

Some definitions

- Let's recap:
  - Error is a vector which indicates how far we are from the solution:

$$\mathbf{e}^{(i)} = \mathbf{x}^{(i)} - \mathbf{x}$$

  - Residual indicates how far we are from the correct value of the right hand side of the system:

$$\mathbf{r}^{(i)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(i)}$$

  - The residual is actually the error which is transformed by $\mathbf{A}$ into the space of $\mathbf{b}$:

$$\mathbf{r}^{(i)} = -\mathbf{A}\mathbf{e}^{(i)}$$

  - More importantly:

$$\mathbf{r}^{(i)} = -f'(\mathbf{x}^{(i)})$$

- In the method of the Steepest Descent, we start at an arbitrary point x_0 and take a series of steps $(x_1, x_2, ...)$ sliding down the paraboloid, until we are satisfied that we are close enough to the solution $\mathbf{x}$. When we take a step, we choose the direction in which f decreases most quickly, which is the direction opposite to $f'(\mathbf{x}^{(i)})$. **So the residual is the *direction of the steepest descent.***
- But, how big a step should we take?

FSB IOO

Method of Steepest descent

The gradient of the quadratic function points in the direction of the largest increase of the function, and the residual points in the opposite direction, that is, in the direction of the largest decrease of the function. The iterative method for solving a linear system as a minimisation problem, which uses the residual as a search direction is called *steepest descent*. Each iteration of steepest descent has the following form:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha \mathbf{r}^{(k)},$$

where α is the length of the step in the direction of the residual, calculated using a *line search* procedure.

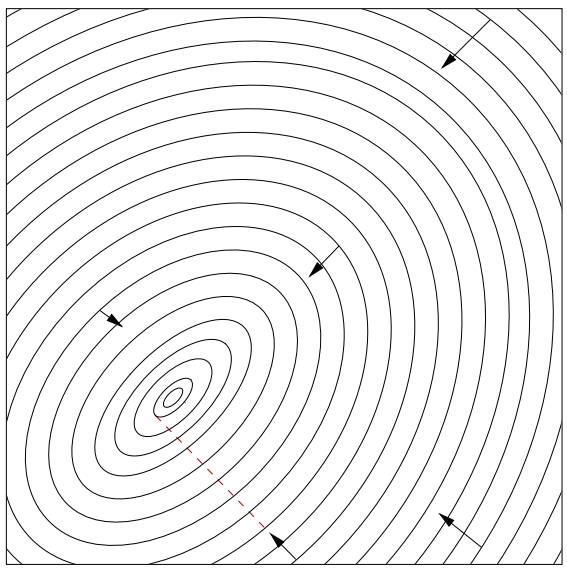


Figure 7: Residual vectors plotted on the isocontours of the quadratic function, pointing in the direction of the greatest decrease of the function.

Method of steepest descent

The method needs a large number of iterations, compared to the size of the system, to reach the correct solution. The reason can be found in the eigenspectrum of the coefficient matrix. Similar to fixed–point methods, the problem appears if the magnitudes of the eigenvalues are different in size.

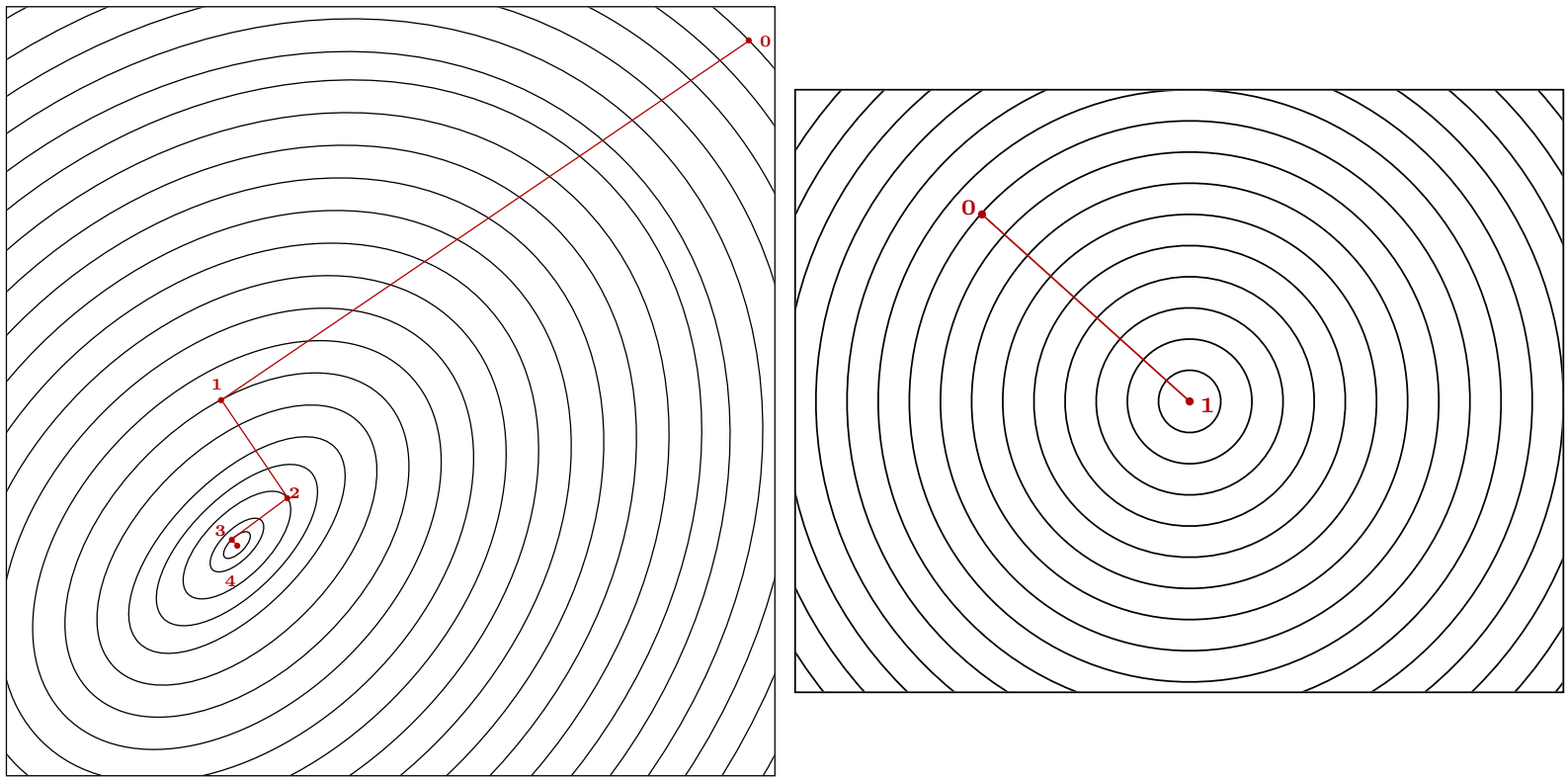
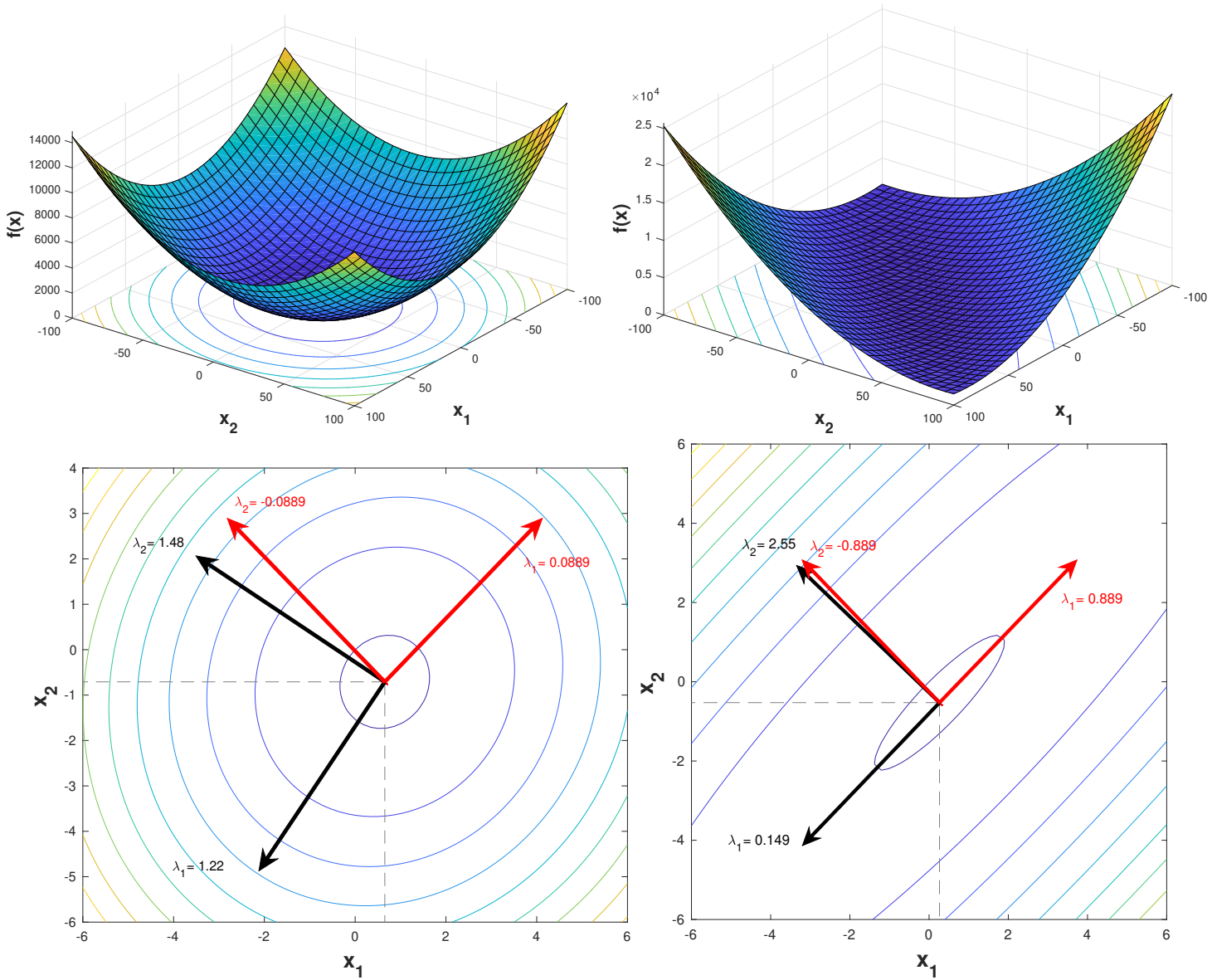


Figure 8: Convergence of steepest descent for a 2x2 matrix: left – matrix with two distinct eigenvalues, right – matrix with duplicate eigenvalues.

Method of steepest descent

Quadratic functions of two symmetric positive definite matrices with eigenvalues of different magnitudes. The black arrows correspond to the eigenvectors of matrix $\mathbf{A}$, while the red arrows represent the eigenvectors of a Jacobi preconditioned matrix $\mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$.



Method of steepest descent

The quadratic function on the left belongs to the matrix $\begin{bmatrix} 1.4 & -0.12 \\ -0.12 & 1.3 \end{bmatrix}$, and it has two eigenvalues with similar magnitudes and both are positive (positive definite matrix). The shape of the isocontours is almost circular.

The quadratic function on the right corresponds to matrix $\begin{bmatrix} 1.4 & -1.2 \\ -1.2 & 1.3 \end{bmatrix}$, and its shape is more stretched. The eigenvalues are of different order of magnitude and it can be seen that the larger eigenvalue belongs to the eigenvector which coincides with the minor axis of the elliptic isocontour.

Steepest descent will quickly converge for the first shape of the quadratic function, since the residual at every point is directed towards the centre of the circle.

Method of conjugate directions

Using the residual in steepest descent as the search direction does not yield fast convergence, since it is neccesary to correct the solution in a single direction multiple times.

The idea is to use n mutually orthogonal search directions and eliminate each of the n components of the error in one step. This is achieved by locating the point on the line corresponding to the search direction which minimises the quadratic function, which is similar to line search, but using mutually orthogonal search directions means that the final solution will be reached in maximally n steps, because we cannot step in the same direction more than once.

Using the directional derivative and setting it to zero yields:

$$\begin{aligned}\frac{d}{d\alpha} f(\mathbf{x}^{(1)}) &= f'(\mathbf{x}^{(1)})^T \frac{d}{d\alpha}(\mathbf{x}^{(1)}) = f'(\mathbf{x}^{(1)})^T \frac{d}{d\alpha}(\mathbf{x}^0 + \alpha \mathbf{d}^0) \\ &= f'(\mathbf{x}^{(1)})^T \mathbf{d}^{(0)} = (\mathbf{r}^{(1)})^T \mathbf{d}^{(0)} = 0.\end{aligned}$$

Thus, the minimum is found where the residual is orthogonal to the search direction.

Conjugate directions

If the residual is written in terms of the scaled error, the definition of *conjugate* or $\mathbf{A}$*–orthogonal* vectors appears:

$$\mathbf{d}^{(k)}\mathbf{A}\mathbf{e}^{(k+1)} = 0.$$

The search direction is orthogonal to the residual, and it is $\mathbf{A}$–orthogonal to the error.

If the function is minimised at some point along the direction $\mathbf{d}^{(0)}$, the gradient of the function at that point is equal to zero. The search in the new direction should not affect the gradient of the previous step, since the function is already minimised in that previous direction. The change of the gradient of the function can be expressed as:

$$\begin{aligned}\Delta f'(\mathbf{x}) &= f'(\mathbf{x}^{(1)}) - f'(\mathbf{x}^{(0)}) = (\mathbf{A}\mathbf{x}^{(1)} - \mathbf{b}) - (\mathbf{A}\mathbf{x}^{(0)} - \mathbf{b}) \\ &= \mathbf{A}(\mathbf{x}^{(0)} + \mathbf{d}^{(0)}) - \mathbf{A}\mathbf{x}^{(0)} = \mathbf{A}\mathbf{d}^{(0)}.\end{aligned}$$

Conjugate directions

Since the step in the next direction $\mathbf{d}^{(1)}$ shouldn't affect the gradient in direction $\mathbf{d}^{(0)}$, the change of the gradient $\Delta f'(\mathbf{x})$ should be orthogonal to the new direction:

$$\mathbf{d}^{(1)}\mathbf{A}\mathbf{d}^{(0)} = 0,$$

which is identical to (36). Thus, defining the search directions to be $\mathbf{A}$–orthogonal, produces an iteration which will minimise the function (find the solution of the system) in n steps, since the minimum in one direction won't be compromised by the minimum in another direction.

To ensure that the method steps in a certain direction only once, the component of the error in that direction should be eliminated, i.e. equal to zero. Thus, the remaining error is $\mathbf{A}$–orthogonal to the search direction, and the length of the next step can be obtained:

$$(\mathbf{d}^{(k)})^T\mathbf{A}\mathbf{e}^{(k+1)} = 0,$$

$$(\mathbf{d}^{(k)})^T\mathbf{A}(\mathbf{e}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}) = 0,$$

$$\alpha^{(k)} = -\frac{(\mathbf{d}^{(k)})^T\mathbf{A}\mathbf{e}^{(k)}}{(\mathbf{d}^{(k)})^T\mathbf{A}\mathbf{d}^{(k)}} = -\frac{(\mathbf{d}^{(k)})^T\mathbf{r}^{(k)}}{(\mathbf{d}^{(k)})^T\mathbf{A}\mathbf{d}^{(k)}}.$$

Conjugate gradients

In order to calculate the projection of the new search direction onto the old ones, it is necessary to store all the previous vectors in memory, and the complexity of the entire procedure is estimated at $\mathcal{O}(n^3)$. In order to improve the method, similar to steepest descent, use the residuals for the starting set of linearly independent vectors: $\mathbf{u}^{(i)} = \mathbf{r}^{(i)}$.

The residuals have several useful properties:

- Residuals are orthogonal to previous search directions.

$$(\mathbf{d}^{(j)})^T \mathbf{r}^{(i)} = 0 \quad j < i.$$

- Residuals are orthogonal to the initial set of linearly independent vectors.

$$\mathbf{u}^{(i)} \mathbf{r}^{(j)} = 0 \quad i < j.$$

- Residual is a linear combination of the previous residual and $\mathbf{A}\mathbf{d}^{(k)}$:

$$\mathbf{r}^{(k+1)} = \mathbf{A}\mathbf{e}^{(k+1)} = \mathbf{A}(\mathbf{e}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}) = \mathbf{r}^{(k)} + \alpha^{(k)}\mathbf{A}\mathbf{d}^{(k)}.$$

Conjugate gradients

For each new direction, the approximation of the solution comes closer to the actual solution, i.e. the method always finds the optimal point in the space (combination of vectors) where it is possible to explore (new dimensions are added to the space with each new direction).

Each new iteration (step in new direction) adds a new dimension to the solution space (hyperplane spanned by search directions $\mathbf{d}$).

The solution of the linear system, i.e. the minimum of the quadratic function is a linear combination of the vectors in space $\mathcal{D}^{(i)}$. The subspace has a property which makes storing all the search directions redundant: since the residual $\mathbf{r}^{(k+1)}$ is orthogonal to all the previous search directions, it is orthogonal to the whole space $\mathcal{D}^{(k+1)}$ spanned by them.

The space $\mathcal{D}^{(k+1)}$ contains:

$$\mathcal{D}^{(k+1)} = \text{span}\{\mathcal{D}^{(k)}, \mathbf{A}\mathcal{D}^{(k)}\}.$$

$\mathbf{r}^{(k+1)}$ is conjugate to $\mathcal{D}^{(k)}$ and all the search directions $\mathbf{d}^0, \ldots, \mathbf{d}^{(k-1)}$ it contains. It is only necessary to make $\mathbf{d}^{(k)}$ conjugate to $\mathbf{r}^{(k+1)}$, thus the projection is done onto only one vector.

Conjugate gradients

1. In the beginning assume $\mathbf{x}^{(0)} = 0$, and then $\mathbf{d}^{(0)} = \mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$.
2. Calculate the length of the step in direction $\mathbf{d}^{(k)}$:

$$\alpha^{(k)} = \frac{(\mathbf{r}^{(k)})^T \mathbf{r}^{(k)}}{(\mathbf{d}^{(k)})^T \mathbf{A}\mathbf{d}^{(k)}}.$$

3. Calculate the new solution:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)} \mathbf{d}^{(k)}.$$

4. Calculate the new residual:

$$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} + \alpha^{(k)} \mathbf{A}\mathbf{d}^{(k)}.$$

5. Calculate the projection operator of the new residual onto the previous search direction:

$$\beta^{(k)} = \frac{(\mathbf{r}^{(k+1)})^T \mathbf{r}^{(k+1)}}{(\mathbf{r}^{(k)})^T \mathbf{r}^{(k)}} \mathbf{d}^{(k)}.$$

6. Calculate the new search direction by making it $\mathbf{A}$–orthogonal to the new residual:

$$\mathbf{d}^{(k+1)} = \mathbf{r}^{(k+1)} + \beta^{(k)} \mathbf{d}^{(k)}.$$

7. Return to step 2 if the convergence criterion is not satisfied.

Conjugate gradients

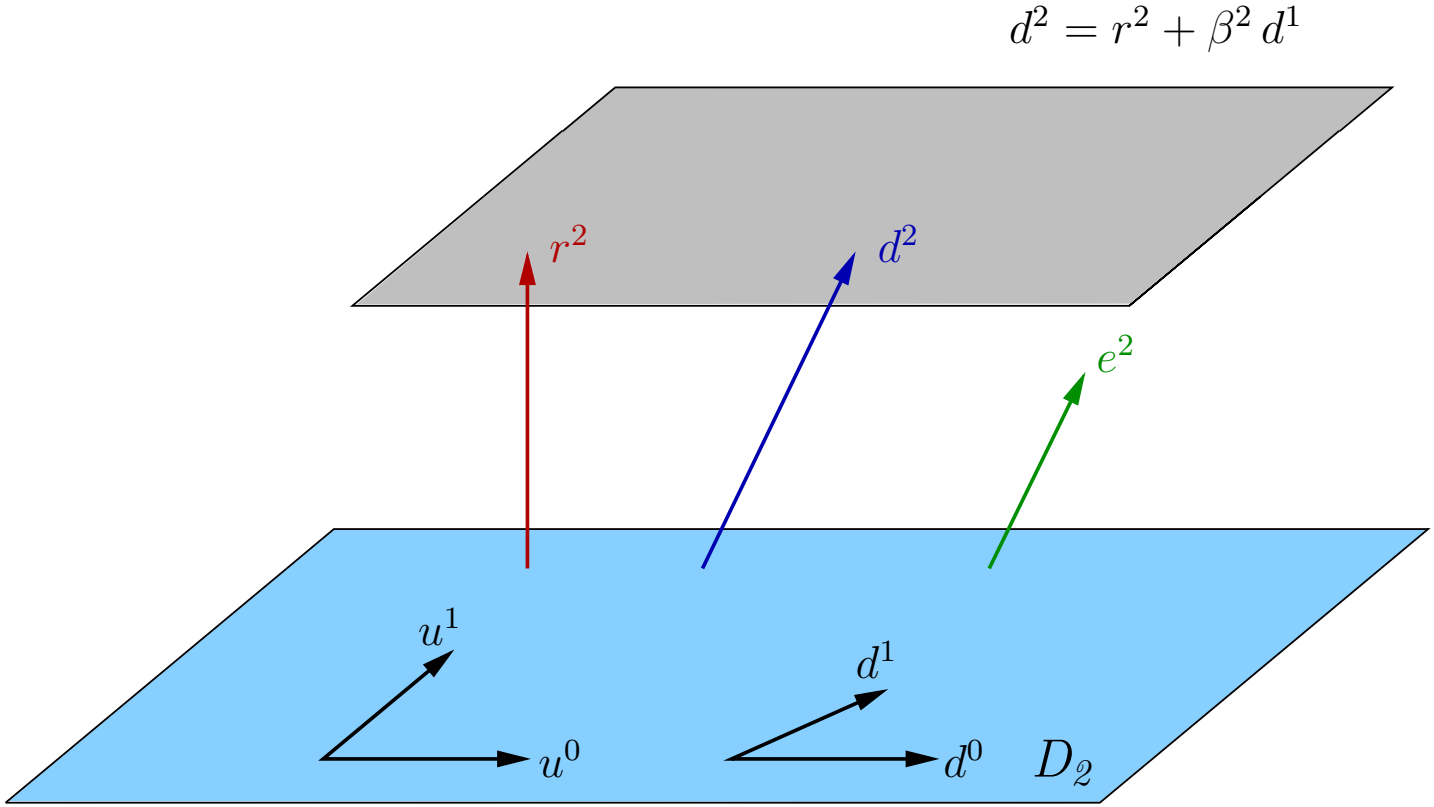


Figure 9: Subspace $\mathcal{D}^{(2)}$ for the approximation of the solution in the second iteration is spanned by the initial vectors $\mathbf{u}^{(0)}$ and $\mathbf{u}^{(1)}$. It is also spanned by $\mathbf{A}$–orthogonal vectors $\mathbf{d}^{(0)}$ and $\mathbf{d}^{(1)}$. The error $\mathbf{e}^{(2)}$ is $\mathbf{A}$–orthogonal to $\mathcal{D}^{(2)}$, while the residual $\mathbf{r}^{(2)}$ is orthogonal. The new search direction is a linear combination of $\mathbf{r}^{(2)}$ and $\mathbf{d}^{(1)}$ and it is $\mathbf{A}$–orthogonal to $\mathcal{D}^{(2)}$.

Other variants of Krylov subspace solvers

Generalized Minimal Residual - GMRES

- Applicable to unsymmetric systems.
- Generates a sequence of orthogonal vectors, but in absence of symmetry all previously computed vectors in the orthogonal sequence have to be retained.
- Minimizes the residual norm.
- Restarted versions of the method are used to save memory.

Biconjugate Gradient - BiCG

- Replaces the orthogonal sequence of residuals by two mutually orthogonal sequences, but at a price of no longer providing a minimization.
- Update relations for residuals in CG are augmented in BiCG by relations that are similar but based on $\mathbf{A}^T$ instead of $\mathbf{A}$.

Biconjugate Gradient Stabilized - BiCGStab

- Can be interpreted as a product of BiCG and repeatedly applied GMRES. At least locally, a residual vector is minimized which leads to smoother convergence.

Preconditioning

The conjugate gradient method **will converge to the correct solution in n iterations**, where n is the dimension of the system. However, for applications in computational fluid dynamics, even this number is inadequate, since the dimension of the system depends on the number of cells and, for implicitly coupled systems, the number of physical variables.

Usually, it is not necessary to conduct the maximum number of iterations, since, as it was seen with steepest descent, **the largest components of the error will be eliminated first**. A satisfactory approximation of the solution can be achieved in significantly less than n iterations.

The convergence of the method can also be drastically improved by **manipulating the shape of the quadratic function** to be more spherical, i.e. closer to the quadratic function of the identity matrix. This transformation of the coefficient matrix $\mathbf{A}$ is called preconditioning.

Preconditioning

- In practice, for large scale problems, CG methods are always used with a preconditioner.
- The idea is to find an operator $\mathbf{M}$ such that $\mathbf{M}^{-1}\mathbf{A}$ has better (but still unknown) spectral properties.

Where does the idea of preconditioning come from?

- For $\mathbf{M} = \mathbf{A}$:

$$\mathbf{M}^{-1}\mathbf{A}x = \mathbf{A}^{-1}\mathbf{A}x = \mathbf{I}x = \mathbf{M}^{-1}b$$

- The system would become ideal, since the matrix would reduce to an identity matrix and all subspace methods would deliver the true solution in one single step.
- But, inverting $\mathbf{A}$ is too expensive!

- The hope is that for $\mathbf{M}$ in some sense close to $\mathbf{A}$, Krylov method applied to

$$\mathbf{M}^{-1}\mathbf{A}x = \mathbf{M}^{-1}b$$

would need only a few iterations to yield a close enough approximation of the solution.

Preconditioning

- To find an efficient linear operator $\mathbf{M}$ (called a *preconditioner*), its properties should satisfy:
  - $\mathbf{M}$ is a good approximation of $\mathbf{A}$ in some sense.
  - The cost of construction of $\mathbf{M}$ is not prohibitive.
  - The system $\mathbf{M}y = z$ is much easier to solve than the original system.
- Except for some trivial situations, the matrix $\mathbf{M}^{-1}\mathbf{A}$ is never formed explicitly! (We would usually get a dense matrix which would kill the efficiency.)
- Instead, for each required application of $\mathbf{M}^{-1}\mathbf{A}$ to some vector y:

$$\mathbf{M}^{-1}\mathbf{A}y \rightarrow \mathbf{A}y = w \rightarrow \mathbf{M}^{-1}w = z$$

  - apply $\mathbf{A}$ to y and calculate w.
  - apply $\mathbf{M}^{-1}$ to w and calculate z (done by solving $\mathbf{M}z = w$).
- Only very special and simple preconditioners can be applied explicitly to $\mathbf{A}$ (for example, a diagonal preconditioner).

Preconditioning

The same preconditioner applied to the matrix will give the same eigenvalues, however, the convergence depends on the eigenvectors: specifically, on the components of the starting residual in eigenvector directions. Different implementations of preconditioners can have quite different eigenvectors, and thus different convergence behaviour.

Left preconditioning

- Apply the iterative method to

$$\mathbf{M}^{-1}\mathbf{A}x = \mathbf{M}^{-1}b.$$

- Note that with left preconditioning we are minimizing the *preconditioned residual* $\mathbf{M}^{-1}(b - \mathbf{A}x_k)$, which may be quite different from the residual $b - \mathbf{A}x_k$. This could have consequences for stopping criteria based on the norm of the residual.

Right preconditioning

- Apply the iterative method to

$$\mathbf{A}\mathbf{M}^{-1}y = b,$$
$$\mathbf{M}x = y.$$

- The advantage of right preconditioning is that it affects only the operator and not the right hand side.
- Note that the error norm in y may be much smaller than the error norm in x.

Preconditioning

- Preconditioning can be considered as an attempt to stretch the quadratic form to make it appear more spherical (eigenvalues are more clustered, scaling along eigenvector axes). A perfect preconditioner is $\mathbf{M} = \mathbf{A}$ because $\mathbf{M}^{-1}\mathbf{A}$ is a unitary matrix and the system can be solved in a single iteration. But, computing $\mathbf{A}^{-1}$ is equivalent to solving the system.
- The simplest preconditioner is the *Jacobi* or *diagonal* preconditioner (which scales the quadratic form along the coordinate axes). It's fast because diagonal matrix is easy to invert.
- A better preconditioner is *Cholesky preconditioning* which is a technique for factoring a matrix into the form $\mathbf{L}\mathbf{L}^T$, where $\mathbf{L}$ is a lower triangular matrix. It is a version of the *LU factorization* for symmetric matrices.
- The LU factorization is performed by a modified Gaussian elimination: eliminate the entries below the main diagonal to obtain the upper triangular matrix $\mathbf{U}$ and remember the multiple of row j that is subtracted from row i to zero the coefficient a_{ij}. This multiple is the subdiagonal entry of $\mathbf{L}$.

$$\forall_{i,j>k} a_{ij} = a_{ij} - a_{ik} a_{kk}^{-1} a_{kj}$$

Preconditioning

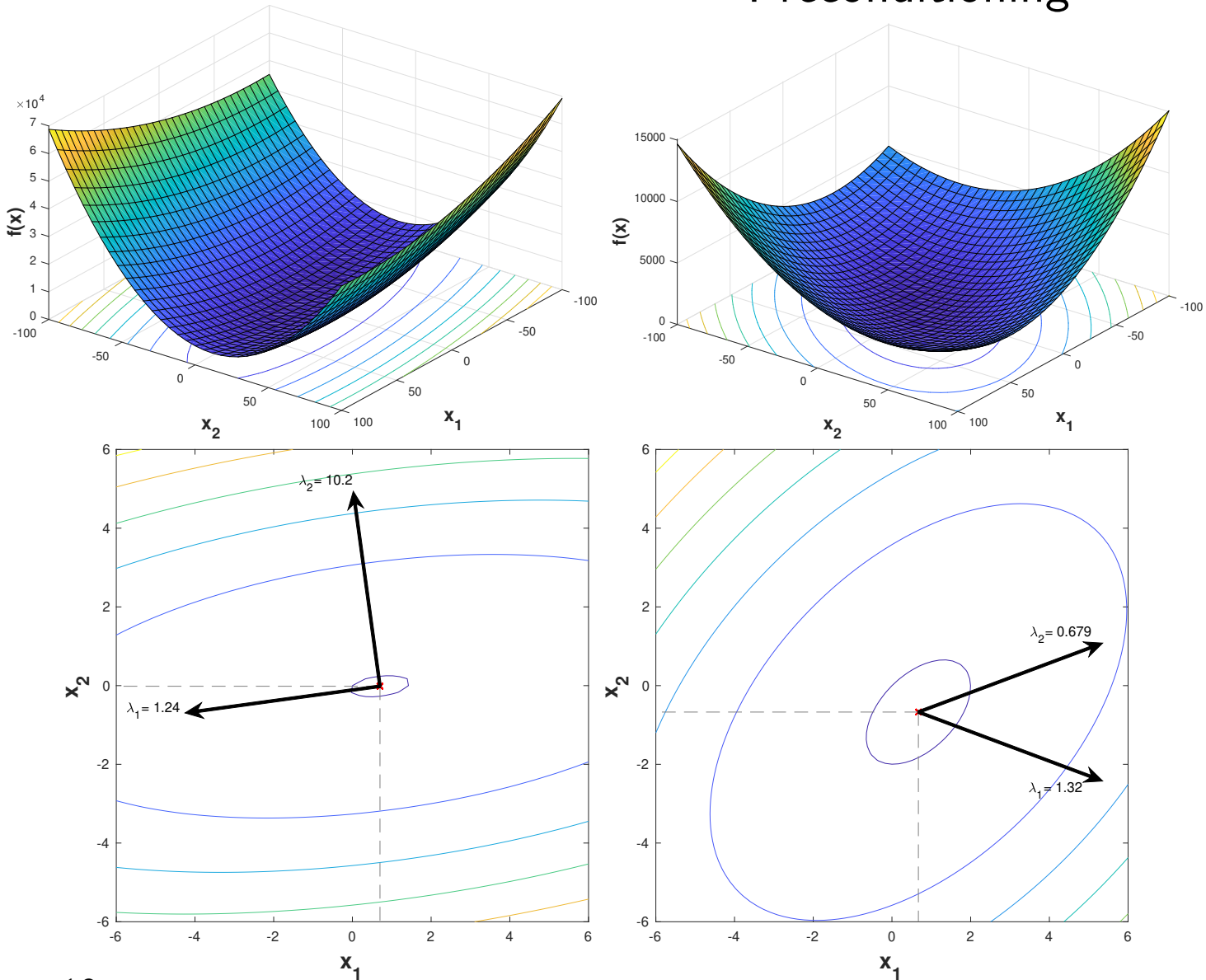


Figure 10: On the left: quadratic function of a symmetric positive definite matrix and eigenvectors with the corresponding eigenvalues. On the right: diagonally preconditioned matrix with the corresponding eigenvectors and eigenvalues, no longer symmetric.

Preconditioning

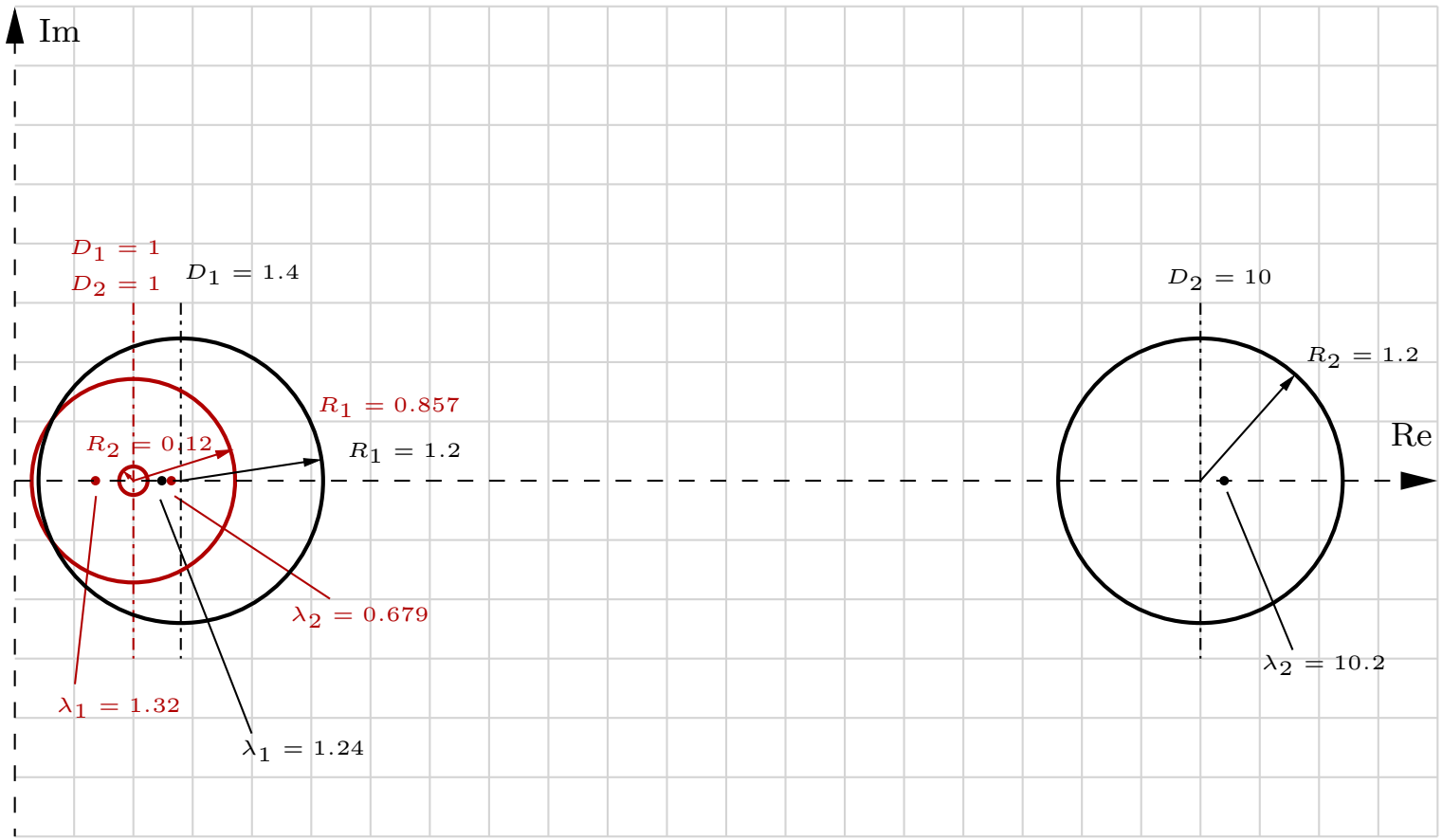


Figure 11: Demonstration of explicit diagonal preconditioning using the Gershgorin theorem: the eigenvalues of the preconditioned matrix (red) are clustered closer together in comparison to the eigenvalues of the original matrix (black).

Convergence criteria in `fvSolution`

- Normalisation of residuals:
  To be able to compare convergence of different cases, the residual $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$ is normalised by dividing it with a normalisation factor ξ:

$$\xi = \sum_i^n (|\mathbf{A}\mathbf{x} - \mathbf{A}\overline{\mathbf{x}}| + |\mathbf{b} - \mathbf{A}\overline{\mathbf{x}}|),$$

  where $\mathbf{x}$ is the current value of the solution vector and every component of $\overline{\mathbf{x}}$ is equal to an average value of $\mathbf{x}$ over n (all) equations:

$$\overline{\mathbf{x}} = \frac{\sum_i^n x_i}{n}.$$

- Relative tolerance:

$$\texttt{relTol} = \frac{\text{initial residual}}{\text{final residual}}.$$

- Tolerance (set $\texttt{relTol} = 0$):

$$\texttt{tolerance} = \text{final residual}.$$

FSB IOO

Summary

- Matrices arising from Finite Volume Method discretisation are sparse. The structure and number of non–zero coefficients is a consequence of the connectivity from the computational mesh.
- The choice of a linear solution algorithm depends on the properties of the coefficient matrix.
- Classical iterative methods, e.g. Gauss–Seidel are not very effective in solving the system as they require matrices with special properties and converge very slowly.
- Multigrid solvers are extremely efficient and rely on the ability of the stationary–point solvers to smooth the error.
- CG solver is used ONLY for symmetric matrices, while for unsymmetric matrices some variants are used: BiCG, BiCGStab, GMRES.
- Always use preconditioners with Krylov subspace solvers (diagonal or some variant of ILU).
- Multigrid can be used as a stand–alone solver or as a preconditioner with the Krylov subspace solvers.

Used and recommended literature:

- Y. Saad: Iterative methods for sparse linear systems, 2000
- J.R. Shewchuk: An introduction to the Conjugate Gradient Method without the agonizing pain, 1994
- H.A. van der Vorst: Iterative Krylov methods for large linear systems, 2003
- T. Uroić: Implicitly coupled finite volume algorithms, PhD Thesis, 2019
- U. Trottenberg, C. Oosterlee, A. Schüller: Multigrid, 2001