ME469A
Handout #6

Gianluca Iaccarino

Solution of Linear System

- Numerical discretization of PDEs leads to the definition of an algebraic (discrete) system of equations (ADE)
- The ADE system can be linear or non-linear depending on the original PDE
- Both FV and FD methods lead to linear systems for the solution of the convection/diffusion equation

$$A\phi = Q$$

Solution of Linear System

- The characteristics of the system are:
  - Sparse matrix - small number of non-zero elements
    $\rightarrow$ related to the size computational stencil
  - Regular pattern - the non-zero elements are organized in few diagonals
    $\rightarrow$ related to the ordering of the unknowns
  - Diagonally dominant $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$ for all i
    $\rightarrow$ depending on the specific discretization scheme: in a FV method this represents a conservation statement
  - Symmetric $a_{ij} = a_{ji}$
    $\rightarrow$ depending on the cell-to-cell relationships: in a FV method this is connected to the evaluation of the fluxes

Solution of Linear System

- The characteristics of the system are:
  - Sparse matrix - small number of non-zero elements
    $\longrightarrow$ related to the size computational stencil
  - Regular pattern - the non-zero elements are organized in few diagonals
    $\longrightarrow$ related to the ordering of the unknowns
  - Diagonally dominant $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$ for all i
    $\longrightarrow$ depending on the specific discretization scheme: in a FV method this represents a conservation statement
  - Symmetric $a_{ij} = a_{ji}$
    $\longrightarrow$ depending on the cell-to-cell relationships: in a FV method this is connected to the evaluation of the fluxes

Solution of Linear System

- The characteristics of the system are:
  - Sparse matrix - small number of non-zero elements
    $\rightarrow$ related to the size computational stencil
  - Regular pattern - the non-zero elements are organized in few diagonals
    $\rightarrow$ related to the ordering of the unknowns
  - Diagonally dominant $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$ for all i
    $\rightarrow$ depending on the specific discretization scheme: in a FV method this represents a conservation statement
  - Symmetric $a_{ij} = a_{ji}$
    $\rightarrow$ depending on the cell-to-cell relationships: in a FV method this is connected to the evaluation of the fluxes

Solution of Linear System

- The characteristics of the system are:
  - Sparse matrix - small number of non-zero elements
    $\rightarrow$ related to the size computational stencil
  - Regular pattern - the non-zero elements are organized in few diagonals
    $\rightarrow$ related to the ordering of the unknowns
  - Diagonally dominant $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$ for all i
    $\rightarrow$ depending on the specific discretization scheme: in a FV method this represents a conservation statement
  - Symmetric $a_{ij} = a_{ji}$
    $\rightarrow$ depending on the cell-to-cell relationships: in a FV method this is connected to the evaluation of the fluxes

Solution of Linear System

- The characteristics of the system are:
  - Sparse matrix - small number of non-zero elements
    $\rightarrow$ related to the size computational stencil
  - Regular pattern - the non-zero elements are organized in few diagonals
    $\rightarrow$ related to the ordering of the unknowns
  - Diagonally dominant $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$ for all i
    $\rightarrow$ depending on the specific discretization scheme: in a FV method this represents a conservation statement
  - Symmetric $a_{ij} = a_{ji}$
    $\rightarrow$ depending on the cell-to-cell relationships: in a FV method this is connected to the evaluation of the fluxes

Symmetric Linear Systems

Symmetry is an important property for a linear system

- How is it connected to the discretization scheme?
- Consider the FV discretization for the 1D convection/diffusion equation

  $A_W^i \phi_{i-1} + A_P^i \phi_i + A_E^i \phi_{i+1} = Q_i$
- Symmetry implies that $A_E^i = A_W^{i+1}$
- The convection term is discretized using linear interpolation

$$A_E^i = \rho v \lambda_{i+1/2} - \Gamma \frac{1}{x_{i+1} - x_i} = \rho v \frac{x_{i+1/2} - x_i}{x_{i+1} - x_{i-1}} - \Gamma \frac{1}{x_{i+1} - x_i}$$

$$A_W^{i+1} = \rho v (1 - \lambda_{i+1/2}) - \Gamma \frac{1}{x_{i+1} - x_i} =$$

$$\rho v \frac{x_{i+1} - x_{i+1/2}}{x_{i+1} - x_{i-1}} - \Gamma \frac{1}{x_{i+1} - x_i}$$

Symmetric Linear Systems

Symmetry is an important property for a linear system

- How is it connected to the discretization scheme?
- Consider the FV discretization for the 1D convection/diffusion equation

  $$A_W^i \phi_{i-1} + A_P^i \phi_i + A_E^i \phi_{i+1} = Q_i$$
- Symmetry implies that $A_E^i = A_W^{i+1}$
- The convection term is discretized using linear interpolation

$$A_E^i = \rho v \lambda_{i+1/2} - \Gamma \frac{1}{x_{i+1} - x_i} = \rho v \frac{x_{i+1/2} - x_i}{x_{i+1} - x_{i-1}} - \Gamma \frac{1}{x_{i+1} - x_i}$$

$$A_W^{i+1} = \rho v (1 - \lambda_{i+1/2}) - \Gamma \frac{1}{x_{i+1} - x_i} =$$

$$\rho v \frac{x_{i+1} - x_{i+1/2}}{x_{i+1} - x_{i-1}} - \Gamma \frac{1}{x_{i+1} - x_i}$$

Symmetric Linear Systems

Symmetry is an important property for a linear system

- How is it connected to the discretization scheme?
- Consider the FV discretization for the 1D convection/diffusion equation

  $A_W^i \phi_{i-1} + A_P^i \phi_i + A_E^i \phi_{i+1} = Q_i$

- Symmetry implies that $A_E^i = A_W^{i+1}$
- The convection term is discretized using linear interpolation

$$A_E^i = \rho v \lambda_{i+1/2} - \Gamma \frac{1}{x_{i+1} - x_i} = \rho v \frac{x_{i+1/2} - x_i}{x_{i+1} - x_{i-1}} - \Gamma \frac{1}{x_{i+1} - x_i}$$

$$A_W^{i+1} = \rho v (1 - \lambda_{i+1/2}) - \Gamma \frac{1}{x_{i+1} - x_i} =$$

$$\rho v \frac{x_{i+1} - x_{i+1/2}}{x_{i+1} - x_{i-1}} - \Gamma \frac{1}{x_{i+1} - x_i}$$

Symmetric Linear Systems

Symmetry is an important property for a linear system

- How is it connected to the discretization scheme?
- Consider the FV discretization for the 1D convection/diffusion equation

  $A^i_W \phi_{i-1} + A^i_P \phi_i + A^i_E \phi_{i+1} = Q_i$
- Symmetry implies that $A^i_E = A^{i+1}_W$
- The convection term is discretized using linear interpolation

$$A^i_E = \rho v \lambda_{i+1/2} - \Gamma \frac{1}{x_{i+1} - x_i} = \rho v \frac{x_{i+1/2} - x_i}{x_{i+1} - x_{i-1}} - \Gamma \frac{1}{x_{i+1} - x_i}$$

$$A^{i+1}_W = \rho v (1 - \lambda_{i+1/2}) - \Gamma \frac{1}{x_{i+1} - x_i} =$$

$$\rho v \frac{x_{i+1} - x_{i+1/2}}{x_{i+1} - x_{i-1}} - \Gamma \frac{1}{x_{i+1} - x_i}$$

Symmetric Linear Systems

Symmetry is an important property for a linear system

- How is it connected to the discretization scheme?
- Consider the FV discretization for the 1D convection/diffusion equation

  $A_W^i \phi_{i-1} + A_P^i \phi_i + A_E^i \phi_{i+1} = Q_i$
- Symmetry implies that $A_E^i = A_W^{i+1}$
- The convection term is discretized using linear interpolation

$$A_E^i = \rho v \lambda_{i+1/2} - \Gamma \frac{1}{x_{i+1} - x_i} = \rho v \frac{x_{i+1/2} - x_i}{x_{i+1} - x_{i-1}} - \Gamma \frac{1}{x_{i+1} - x_i}$$

$$A_W^{i+1} = \rho v (1 - \lambda_{i+1/2}) - \Gamma \frac{1}{x_{i+1} - x_i} =$$

$$\rho v \frac{x_{i+1} - x_{i+1/2}}{x_{i+1} - x_{i-1}} - \Gamma \frac{1}{x_{i+1} - x_i}$$

Solution of Linear System

A variety of solution methods are available - we will have a brief description of the most common and a more detailed discussion of some of the most used in CFD

- Direct methods
  - Gauss elimination
  - Tridiagonal solver
  - Tridiagonal solver for periodic domains
  - Cyclic reduction
  - LU decomposition
- Iterative methods
  - Basic methods
  - Incomplete LU decomposition (Stone)
  - Alternating Direction Implicit (ADI)
  - Conjugate Gradient Methods
  - Multigrid

Solution of Linear System

A variety of solution methods are available - we will have a brief description of the most common and a more detailed discussion of some of the most used in CFD

- Direct methods
  - Gauss elimination
  - Tridiagonal solver
  - Tridiagonal solver for periodic domains
  - Cyclic reduction
  - LU decomposition
- Iterative methods
  - Basic methods
  - Incomplete LU decomposition (Stone)
  - Alternating Direction Implicit (ADI)
  - Conjugate Gradient Methods
  - Multigrid

Solution of Linear System

A variety of solution methods are available - we will have a brief description of the most common and a more detailed discussion of some of the most used in CFD

- Direct methods
  - Gauss elimination
  - Tridiagonal solver
  - Tridiagonal solver for periodic domains
  - Cyclic reduction
  - LU decomposition
- Iterative methods
  - Basic methods
  - Incomplete LU decomposition (Stone)
  - Alternating Direction Implicit (ADI)
  - Conjugate Gradient Methods
  - Multigrid

Gauss Elimination

Recall: we want to modify in an upper triangular system

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

To make $a_{21} = 0$

- Multiply the first row by a_{21}/a_{11} and subtract from the second raw:

  $(a_{21} - a_{11}a_{21}/a_{11} \quad a_{22} - a_{12}a_{21}/a_{11} \quad \cdots)$
- resulting in $(0 \quad \tilde{a}_{22} \quad \cdots)$
- Repeat for a_{31}: multiply first row and subtract from third...

Gauss Elimination

Recall: we want to modify in an upper triangular system

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

To make $a_{21} = 0$

- Multiply the first row by a_{21}/a_{11} and subtract from the second raw:

  $(a_{21} - a_{11}a_{21}/a_{11} \quad a_{22} - a_{12}a_{21}/a_{11} \quad \cdots)$
- resulting in $(0 \quad \tilde{a}_{22} \quad \cdots)$
- Repeat for a_{31}: multiply first row and subtract from third...

Gauss Elimination

Recall: we want to modify in an upper triangular system

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

To make $a_{21} = 0$

- Multiply the first row by a_{21}/a_{11} and subtract from the second raw:

  $(a_{21} - a_{11}a_{21}/a_{11} \quad a_{22} - a_{12}a_{21}/a_{11} \quad \cdots)$
- resulting in $(0 \quad \tilde{a}_{22} \quad \cdots)$
- Repeat for a_{31}: multiply first row and subtract from third...

Gauss Elimination

Recall: we want to modify in an upper triangular system

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

To make $a_{21} = 0$

- Multiply the first row by a_{21}/a_{11} and subtract from the second raw:

  $(a_{21} - a_{11}a_{21}/a_{11} \quad a_{22} - a_{12}a_{21}/a_{11} \quad \cdots)$
- resulting in $(0 \quad \tilde{a}_{22} \quad \cdots)$
- Repeat for a_{31}: multiply first row and subtract from third...

Gauss Elimination

Recall: we want to modify in an upper triangular system

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

To make $a_{21} = 0$

- Multiply the first row by a_{21}/a_{11} and subtract from the second raw:

  $(a_{21} - a_{11}a_{21}/a_{11} \quad a_{22} - a_{12}a_{21}/a_{11} \quad \cdots)$
- resulting in $(0 \quad \tilde{a}_{22} \quad \cdots)$
- Repeat for a_{31}: multiply first row and subtract from third...

Gauss Elimination

After the first step (forward elimination) the system is

$$\tilde{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & \tilde{a}_{22} & \cdots & \tilde{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{a}_{nn} \end{pmatrix}$$

Note: the right hand side is modified as well!

It can be easily solved in a backward substitution

- $\phi_n = \tilde{b}_n / \tilde{a}_{nn}$
- $\phi_{n-1} = \cdots$

The operation count is $\approx n^3$ and the method is rarely used.

Gauss Elimination

After the first step (forward elimination) the system is

$$\tilde{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & \tilde{a}_{22} & \cdots & \tilde{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{a}_{nn} \end{pmatrix}$$

Note: the right hand side is modified as well!
It can be easily solved in a backward substitution

- $\phi_n = \tilde{b}_n / \tilde{a}_{nn}$
- $\phi_{n-1} = \cdots$

The operation count is $\approx n^3$ and the method is rarely used.

Gauss Elimination

After the first step (forward elimination) the system is

$$\tilde{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & \tilde{a}_{22} & \cdots & \tilde{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{a}_{nn} \end{pmatrix}$$

Note: the right hand side is modified as well!
It can be easily solved in a backward substitution

- $\phi_n = \tilde{b}_n / \tilde{a}_{nn}$
- $\phi_{n-1} = \cdots$

The operation count is $\approx n^3$ and the method is rarely used.

Tridiagonal Systems

The most common system for 1D problem - but also an important component in multi-dimensional problems

- $i-$th row is: $A_W^i \phi_{i-1} + A_P^i \phi_i + A_E^i \phi_{i+1} = Q_i$
- It is typically stored using three arrays, e.g. $A_E[1:N]$ with $A_E[1] = 0$, $A_W[N] = 0$
- The solution is obtained in two steps:
  - Forward elimination: transform the matrix in lower triangular

    $$i = 1, \ldots, N \; A_P^i = A_P^i - \frac{A_W^i A_E^{i-1}}{A_P^{i-1}}; \; Q_i = Q_i - \frac{A_W^i Q_{i-1}}{A_P^{i-1}}$$
  - Backward substitution: solve the triangular system

    $$i = N, \ldots, 1 \; \phi_i = \frac{Q_i - A_E^i \phi_{i+1}}{A_P^i}$$

The operation count is $\approx n$ and the method MUST be used whenever possible!

Tridiagonal Systems

The most common system for 1D problem - but also an important component in multi-dimensional problems

- $i-$th row is: $A_W^i \phi_{i-1} + A_P^i \phi_i + A_E^i \phi_{i+1} = Q_i$
- It is typically stored using three arrays, e.g. $A_E[1:N]$ with $A_E[1] = 0$, $A_W[N] = 0$
- The solution is obtained in two steps:
  - Forward elimination: transform the matrix in lower triangular

    $$i = 1, \ldots, N \; A_P^i = A_P^i - \frac{A_W^i A_E^{i-1}}{A_P^{i-1}}; \; Q_i = Q_i - \frac{A_W^i Q_{i-1}}{A_P^{i-1}}$$

  - Backward substitution: solve the triangular system

    $$i = N, \ldots, 1 \; \phi_i = \frac{Q_i - A_E^i \phi_{i+1}}{A_P^i}$$

The operation count is $\approx n$ and the method MUST be used whenever possible!

Tridiagonal Systems

The most common system for 1D problem - but also an important component in multi-dimensional problems

- $i-$th row is: $A_W^i \phi_{i-1} + A_P^i \phi_i + A_E^i \phi_{i+1} = Q_i$
- It is typically stored using three arrays, e.g. $A_E[1:N]$ with $A_E[1] = 0$, $A_W[N] = 0$
- The solution is obtained in two steps:
  - Forward elimination: transform the matrix in lower triangular

    $$i = 1, \ldots, N \; A_P^i = A_P^i - \frac{A_W^i A_E^{i-1}}{A_P^{i-1}}; \; Q_i = Q_i - \frac{A_W^i Q_{i-1}}{A_P^{i-1}}$$

  - Backward substitution: solve the triangular system

    $$i = N, \ldots, 1 \; \phi_i = \frac{Q_i - A_E^i \phi_{i+1}}{A_P^i}$$

The operation count is $\approx n$ and the method MUST be used whenever possible!

Tridiagonal Systems

The most common system for 1D problem - but also an important component in multi-dimensional problems

- $i-$th row is: $A_W^i \phi_{i-1} + A_P^i \phi_i + A_E^i \phi_{i+1} = Q_i$
- It is typically stored using three arrays, e.g. $A_E[1:N]$ with $A_E[1] = 0$, $A_W[N] = 0$
- The solution is obtained in two steps:
  - Forward elimination: transform the matrix in lower triangular

    $$i = 1, \ldots, N \; A_P^i = A_P^i - \frac{A_W^i A_E^{i-1}}{A_P^{i-1}}; \; Q_i = Q_i - \frac{A_W^i Q_{i-1}}{A_P^{i-1}}$$
  - Backward substitution: solve the triangular system

    $$i = N, \ldots, 1 \; \phi_i = \frac{Q_i - A_E^i \phi_{i+1}}{A_P^i}$$

The operation count is $\approx n$ and the method MUST be used whenever possible!

Tridiagonal Systems

The most common system for 1D problem - but also an important component in multi-dimensional problems

- $i-$th row is: $A_W^i \phi_{i-1} + A_P^i \phi_i + A_E^i \phi_{i+1} = Q_i$
- It is typically stored using three arrays, e.g. $A_E[1:N]$ with $A_E[1] = 0$, $A_W[N] = 0$
- The solution is obtained in two steps:
  - Forward elimination: transform the matrix in lower triangular

    $$i = 1, \ldots, N \; A_P^i = A_P^i - \frac{A_W^i A_E^{i-1}}{A_P^{i-1}}; \; Q_i = Q_i - \frac{A_W^i Q_{i-1}}{A_P^{i-1}}$$
  - Backward substitution: solve the triangular system

    $$i = N, \ldots, 1 \; \phi_i = \frac{Q_i - A_E^i \phi_{i+1}}{A_P^i}$$

The operation count is $\approx n$ and the method MUST be used whenever possible!

Tridiagonal Periodic Systems

An interesting (and useful) variant can be derived for periodic problems. The matrix is not tridiagonal:

$$\begin{pmatrix} \cdot & \cdot & & & \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W & A_P & A_E & \\ & & \cdot & \cdot & \cdot \\ & & & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix} \rightarrow \begin{pmatrix} A_P^1 & A_E^1 & & & A_W^1 \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W^i & A_P^i & A_E^i & \\ & & \cdot & \cdot & \cdot \\ A_E^N & & & A_W^N & A_P^N \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix}$$

We can use the Sherman-Morrison formula: solve two tridiagonal systems instead of one:

- Transform the system in $(\tilde{A} + uv^T)\phi = Q$ with
  $u^T = [A_P^1 \quad 0 \cdots \quad A_E^N]^T$
  $v^T = [1 \quad 0 \cdots \quad A_W^1/A_P^1]^T$
  $\tilde{A}$ is tridiagonal
- Solve $\tilde{A}y = Q$ and $\tilde{A}z = u$
- Compute $\phi = y - (v^T y)/(1 + (v^T z))z$

Tridiagonal Periodic Systems

An interesting (and useful) variant can be derived for periodic problems. The matrix is not tridiagonal:

$$\begin{pmatrix} \cdot & \cdot & & & \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W & A_P & A_E & \\ & & \cdot & \cdot & \cdot \\ & & & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix} \rightarrow \begin{pmatrix} A_P^1 & A_E^1 & & & A_W^1 \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W^i & A_P^i & A_E^i & \\ & & \cdot & \cdot & \cdot \\ A_E^N & & & A_W^N & A_P^N \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix}$$

We can use the Sherman-Morrison formula: solve two tridiagonal systems instead of one:

- Transform the system in $(\tilde{A} + uv^T)\phi = Q$ with
  $u^T = [A_P^1 \quad 0 \cdots \quad A_E^N]^T$
  $v^T = [1 \quad 0 \cdots \quad A_W^1/A_P^1]^T$
  $\tilde{A}$ is tridiagonal
- Solve $\tilde{A}y = Q$ and $\tilde{A}z = u$
- Compute $\phi = y - (v^T y)/(1 + (v^T z))z$

Tridiagonal Periodic Systems

An interesting (and useful) variant can be derived for periodic problems. The matrix is not tridiagonal:

$$\begin{pmatrix} \cdot & \cdot & & & \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W & A_P & A_E & \\ & & \cdot & \cdot & \cdot \\ & & & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix} \rightarrow \begin{pmatrix} A_P^1 & A_E^1 & & & A_W^1 \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W^i & A_P^i & A_E^i & \\ & & \cdot & \cdot & \cdot \\ A_E^N & & & A_W^N & A_P^N \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix}$$

We can use the Sherman-Morrison formula: solve two tridiagonal systems instead of one:

- Transform the system in $(\tilde{A} + uv^T)\phi = Q$ with
  $u^T = [A_P^1 \quad 0 \cdots \quad A_E^N]^T$
  $v^T = [1 \quad 0 \cdots \quad A_W^1/A_P^1]^T$
  $\tilde{A}$ is tridiagonal
- Solve $\tilde{A}y = Q$ and $\tilde{A}z = u$
- Compute $\phi = y - (v^T y)/(1 + (v^T z))z$

Tridiagonal Periodic Systems

An interesting (and useful) variant can be derived for periodic problems. The matrix is not tridiagonal:

$$\begin{pmatrix} \cdot & \cdot & & & \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W & A_P & A_E & \\ & & \cdot & \cdot & \cdot \\ & & & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix} \rightarrow \begin{pmatrix} A_P^1 & A_E^1 & & & A_W^1 \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W^i & A_P^i & A_E^i & \\ & & \cdot & \cdot & \cdot \\ A_E^N & & & A_W^N & A_P^N \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix}$$

We can use the Sherman-Morrison formula: solve two tridiagonal systems instead of one:

- Transform the system in $(\tilde{A} + uv^T)\phi = Q$ with
  $u^T = [A_P^1 \quad 0 \cdots \quad A_E^N]^T$
  $v^T = [1 \quad 0 \cdots \quad A_W^1/A_P^1]^T$
  $\tilde{A}$ is tridiagonal
- Solve $\tilde{A}y = Q$ and $\tilde{A}z = u$
- Compute $\phi = y - (v^T y)/(1 + (v^T z))z$

Tridiagonal Periodic Systems

An interesting (and useful) variant can be derived for periodic problems. The matrix is not tridiagonal:

$$\begin{pmatrix} \cdot & \cdot & & & \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W & A_P & A_E & \\ & & \cdot & \cdot & \cdot \\ & & & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix} \rightarrow \begin{pmatrix} A_P^1 & A_E^1 & & & A_W^1 \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W^i & A_P^i & A_E^i & \\ & & \cdot & \cdot & \cdot \\ A_E^N & & & A_W^N & A_P^N \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix}$$

We can use the Sherman-Morrison formula: solve two tridiagonal systems instead of one:

- Transform the system in $(\tilde{A} + uv^T)\phi = Q$ with
  $u^T = [A_P^1 \quad 0 \cdots \quad A_E^N]^T$
  $v^T = [1 \quad 0 \cdots \quad A_W^1/A_P^1]^T$
  $\tilde{A}$ is tridiagonal
- Solve $\tilde{A}y = Q$ and $\tilde{A}z = u$
- Compute $\phi = y - (v^T y)/(1 + (v^T z))z$

Cyclic reduction

In the special case of tridiagonal system with all the elements on the diagonal being equal, a even more simplified algorithm can be derived

$$\begin{pmatrix} A_P^1 & A_E^1 & & & \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W^i & A_P^i & A_E^i & \\ & & \cdot & \cdot & \cdot \\ & & & A_W^N & A_P^N \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix} \rightarrow \begin{pmatrix} A_P & A_E & & & \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W & A_P & A_E & \\ & & \cdot & \cdot & \cdot \\ & & & A_W & A_P \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix}$$

Operate on the even rows (first), say i

- multiply row $i-1$ by A_W/A_P and subtract from row i
- multiply row $i+1$ by A_E/A_P and subtract from row i
- the $i-th$ equation is
  $-(A_W^2/A_P)\phi_{i-2} + (A_P - 2A_W A_E/A_P)\phi_i - (A_E^2/A_P)\phi_{i+2}$
- Only involves even indices!
- Result: decoupled tridiagonal system of equation, size $n/2$
- The computational cost is $\approx \log_2 n$

Cyclic reduction

In the special case of tridiagonal system with all the elements on the diagonal being equal, a even more simplified algorithm can be derived

$$\begin{pmatrix} A_P^1 & A_E^1 & & & \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W^i & A_P^i & A_E^i & \\ & & \cdot & \cdot & \cdot \\ & & & A_W^N & A_P^N \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix} \rightarrow \begin{pmatrix} A_P & A_E & & & \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W & A_P & A_E & \\ & & \cdot & \cdot & \cdot \\ & & & A_W & A_P \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix}$$

Operate on the even rows (first), say i

- multiply row $i-1$ by A_W/A_P and subtract from row i
- multiply row $i+1$ by A_E/A_P and subtract from row i
- the $i-th$ equation is
  $-(A_W^2/A_P)\phi_{i-2} + (A_P - 2A_W A_E/A_P)\phi_i - (A_E^2/A_P)\phi_{i+2}$
- Only involves even indices!
- Result: decoupled tridiagonal system of equation, size $n/2$
- The computational cost is $\approx \log_2 n$

Cyclic reduction

In the special case of tridiagonal system with all the elements on the diagonal being equal, a even more simplified algorithm can be derived

$$\begin{pmatrix} A_P^1 & A_E^1 & & & & \\ \cdot & \cdot & \cdot & \cdot & & \\ & A_W^i & A_P^i & A_E^i & & \\ & & \cdot & \cdot & \cdot & \\ & & & A_W^N & A_P^N \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix} \rightarrow \begin{pmatrix} A_P & A_E & & & \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W & A_P & A_E & \\ & & \cdot & \cdot & \cdot \\ & & & A_W & A_P \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix}$$

Operate on the even rows (first), say i

- multiply row $i-1$ by A_W/A_P and subtract from row i
- multiply row $i+1$ by A_E/A_P and subtract from row i
- the $i-th$ equation is
  $-(A_W^2/A_P)\phi_{i-2} + (A_P - 2A_W A_E/A_P)\phi_i - (A_E^2/A_P)\phi_{i+2}$
- Only involves even indices!
- Result: decoupled tridiagonal system of equation, size $n/2$
- The computational cost is $\approx \log_2 n$

Cyclic reduction

In the special case of tridiagonal system with all the elements on the diagonal being equal, a even more simplified algorithm can be derived

$$\begin{pmatrix} A_P^1 & A_E^1 & & & & \\ \cdot & \cdot & \cdot & \cdot & & \\ & A_W^i & A_P^i & A_E^i & & \\ & & \cdot & \cdot & \cdot & \\ & & & A_W^N & A_P^N \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix} \rightarrow \begin{pmatrix} A_P & A_E & & & & \\ \cdot & \cdot & \cdot & \cdot & & \\ & A_W & A_P & A_E & & \\ & & \cdot & \cdot & \cdot & \\ & & & A_W & A_P \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix}$$

Operate on the even rows (first), say i

- multiply row $i-1$ by A_W/A_P and subtract from row i
- multiply row $i+1$ by A_E/A_P and subtract from row i
- the $i-th$ equation is
  $-(A_W^2/A_P)\phi_{i-2} + (A_P - 2A_W A_E/A_P)\phi_i - (A_E^2/A_P)\phi_{i+2}$
- Only involves even indices!
- Result: decoupled tridiagonal system of equation, size $n/2$
- The computational cost is $\approx \log_2 n$

Cyclic reduction

In the special case of tridiagonal system with all the elements on the diagonal being equal, a even more simplified algorithm can be derived

$$\begin{pmatrix} A_P^1 & A_E^1 & & & \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W^i & A_P^i & A_E^i & \\ & & \cdot & \cdot & \cdot \\ & & & A_W^N & A_P^N \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix} \rightarrow \begin{pmatrix} A_P & A_E & & & \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W & A_P & A_E & \\ & & \cdot & \cdot & \cdot \\ & & & A_W & A_P \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix}$$

Operate on the even rows (first), say i

- multiply row $i-1$ by A_W/A_P and subtract from row i
- multiply row $i+1$ by A_E/A_P and subtract from row i
- the $i-th$ equation is
  $-(A_W^2/A_P)\phi_{i-2} + (A_P - 2A_W A_E/A_P)\phi_i - (A_E^2/A_P)\phi_{i+2}$
- Only involves even indices!
- Result: decoupled tridiagonal system of equation, size $n/2$
- The computational cost is $\approx \log_2 n$

Cyclic reduction

In the special case of tridiagonal system with all the elements on the diagonal being equal, a even more simplified algorithm can be derived

$$\begin{pmatrix} A_P^1 & A_E^1 & & & \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W^i & A_P^i & A_E^i & \\ & & \cdot & \cdot & \cdot \\ & & & A_W^N & A_P^N \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix} \rightarrow \begin{pmatrix} A_P & A_E & & & \\ \cdot & \cdot & \cdot & \cdot & \\ & A_W & A_P & A_E & \\ & & \cdot & \cdot & \cdot \\ & & & A_W & A_P \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix}$$

Operate on the even rows (first), say i

- multiply row $i-1$ by A_W/A_P and subtract from row i
- multiply row $i+1$ by A_E/A_P and subtract from row i
- the $i-th$ equation is
  $-(A_W^2/A_P)\phi_{i-2} + (A_P - 2A_W A_E/A_P)\phi_i - (A_E^2/A_P)\phi_{i+2}$
- Only involves even indices!
- Result: decoupled tridiagonal system of equation, size $n/2$
- The computational cost is $\approx \log_2 n$

Cyclic reduction

In the special case of tridiagonal system with all the elements on the diagonal being equal, a even more simplified algorithm can be derived

$$\begin{pmatrix} A_P^1 & A_E^1 & & & & \\ \cdot & \cdot & \cdot & \cdot & & \\ & A_W^i & A_P^i & A_E^i & & \\ & & \cdot & \cdot & \cdot & \\ & & & A_W^N & A_P^N \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix} \rightarrow \begin{pmatrix} A_P & A_E & & & & \\ \cdot & \cdot & \cdot & \cdot & & \\ & A_W & A_P & A_E & & \\ & & \cdot & \cdot & \cdot & \\ & & & A_W & A_P \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix}$$

Operate on the even rows (first), say i

- multiply row $i-1$ by A_W/A_P and subtract from row i
- multiply row $i+1$ by A_E/A_P and subtract from row i
- the $i-th$ equation is
  $-(A_W^2/A_P)\phi_{i-2} + (A_P - 2A_W A_E/A_P)\phi_i - (A_E^2/A_P)\phi_{i+2}$
- Only involves even indices!
- Result: decoupled tridiagonal system of equation, size $n/2$
- The computational cost is $\approx \log_2 n$

Cyclic reduction

In the special case of tridiagonal system with all the elements on the diagonal being equal, a even more simplified algorithm can be derived

$$\begin{pmatrix} A_P^1 & A_E^1 & & & & \\ \cdot & \cdot & \cdot & \cdot & & \\ & A_W^i & A_P^i & A_E^i & & \\ & & \cdot & \cdot & \cdot & \\ & & & A_W^N & A_P^N \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix} \rightarrow \begin{pmatrix} A_P & A_E & & & & \\ \cdot & \cdot & \cdot & \cdot & & \\ & A_W & A_P & A_E & & \\ & & \cdot & \cdot & \cdot & \\ & & & A_W & A_P \end{pmatrix} \begin{pmatrix} \cdot \\ \cdot \\ \phi_i \\ \cdot \\ \cdot \end{pmatrix}$$

Operate on the even rows (first), say i

- multiply row $i-1$ by A_W/A_P and subtract from row i
- multiply row $i+1$ by A_E/A_P and subtract from row i
- the $i-th$ equation is
  $-(A_W^2/A_P)\phi_{i-2} + (A_P - 2A_W A_E/A_P)\phi_i - (A_E^2/A_P)\phi_{i+2}$
- Only involves even indices!
- Result: decoupled tridiagonal system of equation, size $n/2$
- The computational cost is $\approx \log_2 n$

LU decomposition

A variant of Gauss elimination that is useful in CFD is the LU decomposition

- Formally $A = LU$ with L and U lower and upper triangular matrices (with some limitations and details to make the transformation unique)
- A byproduct of the Gauss elimination

  U is generated during the forward phase

  L is generated during the backward phase
- The solution to $A\phi = Q$ is obtained in two steps:

  $Ly = Q$ and $U\phi = y$
- The LU factorization can be obtained without knowing Q - this is not true for Gauss elimination
- Useful when need to solve several systems with same A but different Q.

LU decomposition

A variant of Gauss elimination that is useful in CFD is the LU decomposition

- Formally $A = LU$ with L and U lower and upper triangular matrices (with some limitations and details to make the transformation unique)
- A byproduct of the Gauss elimination

  U is generated during the forward phase

  L is generated during the backward phase
- The solution to $A\phi = Q$ is obtained in two steps:

  $Ly = Q$ and $U\phi = y$
- The LU factorization can be obtained without knowing Q - this is not true for Gauss elimination
- Useful when need to solve several systems with same A but different Q.

LU decomposition

A variant of Gauss elimination that is useful in CFD is the LU decomposition

- Formally $A = LU$ with L and U lower and upper triangular matrices (with some limitations and details to make the transformation unique)
- A byproduct of the Gauss elimination

  U is generated during the forward phase

  L is generated during the backward phase
- The solution to $A\phi = Q$ is obtained in two steps:

  $Ly = Q$ and $U\phi = y$
- The LU factorization can be obtained without knowing Q - this is not true for Gauss elimination
- Useful when need to solve several systems with same A but different Q.

LU decomposition

A variant of Gauss elimination that is useful in CFD is the LU decomposition

- Formally $A = LU$ with L and U lower and upper triangular matrices (with some limitations and details to make the transformation unique)
- A byproduct of the Gauss elimination

  U is generated during the forward phase

  L is generated during the backward phase
- The solution to $A\phi = Q$ is obtained in two steps:

  $Ly = Q$ and $U\phi = y$
- The LU factorization can be obtained without knowing Q - this is not true for Gauss elimination
- Useful when need to solve several systems with same A but different Q.

LU decomposition

A variant of Gauss elimination that is useful in CFD is the LU decomposition

- Formally $A = LU$ with L and U lower and upper triangular matrices (with some limitations and details to make the transformation unique)
- A byproduct of the Gauss elimination

  U is generated during the forward phase

  L is generated during the backward phase
- The solution to $A\phi = Q$ is obtained in two steps:

  $Ly = Q$ and $U\phi = y$
- The LU factorization can be obtained without knowing Q - this is not true for Gauss elimination
- Useful when need to solve several systems with same A but different Q.

LU decomposition

A variant of Gauss elimination that is useful in CFD is the LU decomposition

- Formally $A = LU$ with L and U lower and upper triangular matrices (with some limitations and details to make the transformation unique)
- A byproduct of the Gauss elimination

  U is generated during the forward phase

  L is generated during the backward phase
- The solution to $A\phi = Q$ is obtained in two steps:

  $Ly = Q$ and $U\phi = y$
- The LU factorization can be obtained without knowing Q - this is not true for Gauss elimination
- Useful when need to solve several systems with same A but different Q.

Iterative methods

- Direct solvers
  - preferable when possible, but both Gauss elimination and LU decomposition do not take into account the sparsity of the matrices
  - Parallelization of direct solvers is typically not possible, or at least not efficient
  - Discretization errors are typically large and solving the system *exactly* is of little advantage
  - Non-linear PDEs require iterative methods and the linear system is only an intermediate step
- How do we build an iterative method for $A\phi = Q$?
  - Guess a solution ϕ^0
  - Construct an updating procedure $\phi^{p+1} = g(\phi^p, A, Q)$
  - Check if the current iterate ϕ^{p+1} is close to ϕ

Iterative methods

- Direct solvers
  - preferable when possible, but both Gauss elimination and LU decomposition do not take into account the sparsity of the matrices
  - Parallelization of direct solvers is typically not possible, or at least not efficient
  - Discretization errors are typically large and solving the system *exactly* is of little advantage
  - Non-linear PDEs require iterative methods and the linear system is only an intermediate step
- How do we build an iterative method for $A\phi = Q$?
  - Guess a solution ϕ^0
  - Construct an updating procedure $\phi^{p+1} = g(\phi^p, A, Q)$
  - Check if the current iterate ϕ^{p+1} is close to ϕ

Iterative methods

- Direct solvers
  - preferable when possible, but both Gauss elimination and LU decomposition do not take into account the sparsity of the matrices
  - Parallelization of direct solvers is typically not possible, or at least not efficient
  - Discretization errors are typically large and solving the system *exactly* is of little advantage
  - Non-linear PDEs require iterative methods and the linear system is only an intermediate step
- How do we build an iterative method for $A\phi = Q$?
  - Guess a solution ϕ^0
  - Construct an updating procedure $\phi^{p+1} = g(\phi^p, A, Q)$
  - Check if the current iterate ϕ^{p+1} is close to ϕ

Iterative methods

- Direct solvers
  - preferable when possible, but both Gauss elimination and LU decomposition do not take into account the sparsity of the matrices
  - Parallelization of direct solvers is typically not possible, or at least not efficient
  - Discretization errors are typically large and solving the system *exactly* is of little advantage
  - Non-linear PDEs require iterative methods and the linear system is only an intermediate step
- How do we build an iterative method for $A\phi = Q$?
  - Guess a solution ϕ^0
  - Construct an updating procedure $\phi^{p+1} = g(\phi^p, A, Q)$
  - Check if the current iterate ϕ^{p+1} is close to ϕ

Iterative methods

- Direct solvers
  - preferable when possible, but both Gauss elimination and LU decomposition do not take into account the sparsity of the matrices
  - Parallelization of direct solvers is typically not possible, or at least not efficient
  - Discretization errors are typically large and solving the system *exactly* is of little advantage
  - Non-linear PDEs require iterative methods and the linear system is only an intermediate step
- How do we build an iterative method for $A\phi = Q$?
  - Guess a solution ϕ^0
  - Construct an updating procedure $\phi^{p+1} = g(\phi^p, A, Q)$
  - Check if the current iterate ϕ^{p+1} is close to ϕ

Iterative methods

- Direct solvers
  - preferable when possible, but both Gauss elimination and LU decomposition do not take into account the sparsity of the matrices
  - Parallelization of direct solvers is typically not possible, or at least not efficient
  - Discretization errors are typically large and solving the system *exactly* is of little advantage
  - Non-linear PDEs require iterative methods and the linear system is only an intermediate step
- How do we build an iterative method for $A\phi = Q$?
  - Guess a solution ϕ^0
  - Construct an updating procedure $\phi^{p+1} = g(\phi^p, A, Q)$
  - Check if the current iterate ϕ^{p+1} is close to ϕ

Iterative methods

- Direct solvers
  - preferable when possible, but both Gauss elimination and LU decomposition do not take into account the sparsity of the matrices
  - Parallelization of direct solvers is typically not possible, or at least not efficient
  - Discretization errors are typically large and solving the system *exactly* is of little advantage
  - Non-linear PDEs require iterative methods and the linear system is only an intermediate step
- How do we build an iterative method for $A\phi = Q$?
  - Guess a solution ϕ^0
  - Construct an updating procedure $\phi^{p+1} = g(\phi^p, A, Q)$
  - Check if the current iterate ϕ^{p+1} is close to ϕ

Iterative methods

- Direct solvers
  - preferable when possible, but both Gauss elimination and LU decomposition do not take into account the sparsity of the matrices
  - Parallelization of direct solvers is typically not possible, or at least not efficient
  - Discretization errors are typically large and solving the system *exactly* is of little advantage
  - Non-linear PDEs require iterative methods and the linear system is only an intermediate step
- How do we build an iterative method for $A\phi = Q$?
  - Guess a solution ϕ^0
  - Construct an updating procedure $\phi^{p+1} = g(\phi^p, A, Q)$
  - Check if the current iterate ϕ^{p+1} is close to ϕ

Iterative methods

- Direct solvers
  - preferable when possible, but both Gauss elimination and LU decomposition do not take into account the sparsity of the matrices
  - Parallelization of direct solvers is typically not possible, or at least not efficient
  - Discretization errors are typically large and solving the system *exactly* is of little advantage
  - Non-linear PDEs require iterative methods and the linear system is only an intermediate step
- How do we build an iterative method for $A\phi = Q$?
  - Guess a solution ϕ^0
  - Construct an updating procedure $\phi^{p+1} = g(\phi^p, A, Q)$
  - Check if the current iterate ϕ^{p+1} is close to ϕ

Iterative methods

We need to introduce some *basic* concepts:

- Define the residual ρ^p as $\rho^p = Q - A\phi^p$
- The error is $\epsilon^p = \phi - \phi^p$
- The relationship is: $A\epsilon^p = \rho^p$

A generic iterative method can be written as:

$$M\phi^{p+1} = N\phi^p + B$$

- At convergence $\phi^{p+1} = \phi^p = \phi \rightarrow A = M - N \quad Q = B$
- A more general constraint is: $PA = M - N \quad PQ = B$
  P is called a preconditioner matrix

Iterative methods

We need to introduce some *basic* concepts:

- Define the residual ρ^p as $\rho^p = Q - A\phi^p$
- The error is $\epsilon^p = \phi - \phi^p$
- The relationship is: $A\epsilon^p = \rho^p$

A generic iterative method can be written as:

$$M\phi^{p+1} = N\phi^p + B$$

- At convergence $\phi^{p+1} = \phi^p = \phi \rightarrow A = M - N \quad Q = B$
- A more general constraint is: $PA = M - N \quad PQ = B$
  P is called a preconditioner matrix

Iterative methods

We need to introduce some *basic* concepts:

- Define the residual ρ^p as $\rho^p = Q - A\phi^p$
- The error is $\epsilon^p = \phi - \phi^p$
- The relationship is: $A\epsilon^p = \rho^p$

A generic iterative method can be written as:

$$M\phi^{p+1} = N\phi^p + B$$

- At convergence $\phi^{p+1} = \phi^p = \phi \rightarrow A = M - N \quad Q = B$
- A more general constraint is: $PA = M - N \quad PQ = B$
  P is called a preconditioner matrix

Iterative methods

We need to introduce some *basic* concepts:

- Define the residual ρ^p as $\rho^p = Q - A\phi^p$
- The error is $\epsilon^p = \phi - \phi^p$
- The relationship is: $A\epsilon^p = \rho^p$

A generic iterative method can be written as:

$$M\phi^{p+1} = N\phi^p + B$$

- At convergence $\phi^{p+1} = \phi^p = \phi \rightarrow A = M - N \quad Q = B$
- A more general constraint is: $PA = M - N \quad PQ = B$
  P is called a preconditioner matrix

Iterative methods

We need to introduce some *basic* concepts:

- Define the residual ρ^p as $\rho^p = Q - A\phi^p$
- The error is $\epsilon^p = \phi - \phi^p$
- The relationship is: $A\epsilon^p = \rho^p$

A generic iterative method can be written as:

$$M\phi^{p+1} = N\phi^p + B$$

- At convergence $\phi^{p+1} = \phi^p = \phi \rightarrow A = M - N \quad Q = B$
- A more general constraint is: $PA = M - N \quad PQ = B$
  P is called a preconditioner matrix

Iterative methods

We need to introduce some *basic* concepts:

- Define the residual ρ^p as $\rho^p = Q - A\phi^p$
- The error is $\epsilon^p = \phi - \phi^p$
- The relationship is: $A\epsilon^p = \rho^p$

A generic iterative method can be written as:

$$M\phi^{p+1} = N\phi^p + B$$

- At convergence $\phi^{p+1} = \phi^p = \phi \rightarrow A = M - N \quad Q = B$
- A more general constraint is: $PA = M - N \quad PQ = B$
  P is called a preconditioner matrix

Iterative methods

We need to introduce some *basic* concepts:

- Define the residual ρ^p as $\rho^p = Q - A\phi^p$
- The error is $\epsilon^p = \phi - \phi^p$
- The relationship is: $A\epsilon^p = \rho^p$

A generic iterative method can be written as:

$$M\phi^{p+1} = N\phi^p + B$$

- At convergence $\phi^{p+1} = \phi^p = \phi \rightarrow A = M - N \quad Q = B$
- A more general constraint is: $PA = M - N \quad PQ = B$
  P is called a preconditioner matrix

Iterative methods

We need to introduce some *basic* concepts:

- Define the residual ρ^p as $\rho^p = Q - A\phi^p$
- The error is $\epsilon^p = \phi - \phi^p$
- The relationship is: $A\epsilon^p = \rho^p$

A generic iterative method can be written as:

$$M\phi^{p+1} = N\phi^p + B$$

- At convergence $\phi^{p+1} = \phi^p = \phi \rightarrow A = M - N \quad Q = B$
- A more general constraint is: $PA = M - N \quad PQ = B$
  P is called a preconditioner matrix

Iterative methods

Desired properties of an iterative method:

- Solving $M\phi^{p+1} = N\phi^p + B$ should be inexpensive
- Examples: M diagonal, tridiagonal, lower/upper triangular
- $M \approx A$ to ensure fast convergence (if $M = A$ the method is direct!)

More precisely we can define convergence:

- The exact solution satisfies $M\phi = N\phi + B$
- the error satisfies $M\epsilon^{p+1} = N\epsilon^p \rightarrow \epsilon^{p+1} = M^{-1}N\epsilon^p$
- The method converges if $\lim_{p\rightarrow\infty} \epsilon^p = 0$
- If all eigenvalues of $(M^{-1}N)$, $\lambda_k < 1$ the method converges

Iterative methods

Desired properties of an iterative method:

- Solving $M\phi^{p+1} = N\phi^p + B$ should be inexpensive
- Examples: M diagonal, tridiagonal, lower/upper triangular
- $M \approx A$ to ensure fast convergence (if $M = A$ the method is direct!)

More precisely we can define convergence:

- The exact solution satisfies $M\phi = N\phi + B$
- the error satisfies $M\epsilon^{p+1} = N\epsilon^p \rightarrow \epsilon^{p+1} = M^{-1}N\epsilon^p$
- The method converges if $\lim_{p\rightarrow\infty} \epsilon^p = 0$
- If all eigenvalues of $(M^{-1}N)$, $\lambda_k < 1$ the method converges

Iterative methods

Desired properties of an iterative method:

- Solving $M\phi^{p+1} = N\phi^p + B$ should be inexpensive
- Examples: M diagonal, tridiagonal, lower/upper triangular
- $M \approx A$ to ensure fast convergence (if $M = A$ the method is direct!)

More precisely we can define convergence:

- The exact solution satisfies $M\phi = N\phi + B$
- the error satisfies $M\epsilon^{p+1} = N\epsilon^p \rightarrow \epsilon^{p+1} = M^{-1}N\epsilon^p$
- The method converges if $\lim_{p\rightarrow\infty} \epsilon^p = 0$
- If all eigenvalues of $(M^{-1}N)$, $\lambda_k < 1$ the method converges

Iterative methods

Desired properties of an iterative method:

- Solving $M\phi^{p+1} = N\phi^p + B$ should be inexpensive
- Examples: M diagonal, tridiagonal, lower/upper triangular
- $M \approx A$ to ensure fast convergence (if $M = A$ the method is direct!)

More precisely we can define convergence:

- The exact solution satisfies $M\phi = N\phi + B$
- the error satisfies $M\epsilon^{p+1} = N\epsilon^p \rightarrow \epsilon^{p+1} = M^{-1}N\epsilon^p$
- The method converges if $\lim_{p\rightarrow\infty} \epsilon^p = 0$
- If all eigenvalues of $(M^{-1}N)$, $\lambda_k < 1$ the method converges

Iterative methods

Desired properties of an iterative method:

- Solving $M\phi^{p+1} = N\phi^p + B$ should be inexpensive
- Examples: M diagonal, tridiagonal, lower/upper triangular
- $M \approx A$ to ensure fast convergence (if $M = A$ the method is direct!)

More precisely we can define convergence:

- The exact solution satisfies $M\phi = N\phi + B$
- the error satisfies $M\epsilon^{p+1} = N\epsilon^p \rightarrow \epsilon^{p+1} = M^{-1}N\epsilon^p$
- The method converges if $\lim_{p\rightarrow\infty} \epsilon^p = 0$
- If all eigenvalues of $(M^{-1}N)$, $\lambda_k < 1$ the method converges

Iterative methods

Desired properties of an iterative method:

- Solving $M\phi^{p+1} = N\phi^p + B$ should be inexpensive
- Examples: M diagonal, tridiagonal, lower/upper triangular
- $M \approx A$ to ensure fast convergence (if $M = A$ the method is direct!)

More precisely we can define convergence:

- The exact solution satisfies $M\phi = N\phi + B$
- the error satisfies $M\epsilon^{p+1} = N\epsilon^p \rightarrow \epsilon^{p+1} = M^{-1}N\epsilon^p$
- The method converges if $\lim_{p\rightarrow\infty} \epsilon^p = 0$
- If all eigenvalues of $(M^{-1}N)$, $\lambda_k < 1$ the method converges

Iterative methods

Desired properties of an iterative method:

- Solving $M\phi^{p+1} = N\phi^p + B$ should be inexpensive
- Examples: M diagonal, tridiagonal, lower/upper triangular
- $M \approx A$ to ensure fast convergence (if $M = A$ the method is direct!)

More precisely we can define convergence:

- The exact solution satisfies $M\phi = N\phi + B$
- the error satisfies $M\epsilon^{p+1} = N\epsilon^p \rightarrow \epsilon^{p+1} = M^{-1}N\epsilon^p$
- The method converges if $\lim_{p\to\infty} \epsilon^p = 0$
- If all eigenvalues of $(M^{-1}N)$, $\lambda_k < 1$ the method converges

Iterative methods

Desired properties of an iterative method:

- Solving $M\phi^{p+1} = N\phi^p + B$ should be inexpensive
- Examples: M diagonal, tridiagonal, lower/upper triangular
- $M \approx A$ to ensure fast convergence (if $M = A$ the method is direct!)

More precisely we can define convergence:

- The exact solution satisfies $M\phi = N\phi + B$
- the error satisfies $M\epsilon^{p+1} = N\epsilon^p \rightarrow \epsilon^{p+1} = M^{-1}N\epsilon^p$
- The method converges if $\lim_{p\rightarrow\infty} \epsilon^p = 0$
- If all eigenvalues of $(M^{-1}N)$, $\lambda_k < 1$ the method converges

Iterative methods

Desired properties of an iterative method:

- Solving $M\phi^{p+1} = N\phi^p + B$ should be inexpensive
- Examples: M diagonal, tridiagonal, lower/upper triangular
- $M \approx A$ to ensure fast convergence (if $M = A$ the method is direct!)

More precisely we can define convergence:

- The exact solution satisfies $M\phi = N\phi + B$
- the error satisfies $M\epsilon^{p+1} = N\epsilon^p \rightarrow \epsilon^{p+1} = M^{-1}N\epsilon^p$
- The method converges if $\lim_{p\rightarrow\infty} \epsilon^p = 0$
- If all eigenvalues of $(M^{-1}N)$, $\lambda_k < 1$ the method converges

Iterative methods

Desired properties of an iterative method:

- Solving $M\phi^{p+1} = N\phi^p + B$ should be inexpensive
- Examples: M diagonal, tridiagonal, lower/upper triangular
- $M \approx A$ to ensure fast convergence (if $M = A$ the method is direct!)

More precisely we can define convergence:

- The exact solution satisfies $M\phi = N\phi + B$
- the error satisfies $M\epsilon^{p+1} = N\epsilon^p \rightarrow \epsilon^{p+1} = M^{-1}N\epsilon^p$
- The method converges if $\lim_{p\rightarrow\infty} \epsilon^p = 0$
- If all eigenvalues of $(M^{-1}N)$, $\lambda_k < 1$ the method converges

Iterative methods

A trivial example

- Suppose we want to solve: $ax = b$ (one equation!)
- Build an iterative method $mx^{p+1} = nx^p + b$ with $m - n = a$
- The error equation is $\epsilon^{n+1} = \frac{n}{m}\epsilon^n$
- The method converges quickly (the error decreases fast) if n/m is small or if n is small

  $\rightarrow m \approx a$

Iterative methods

A trivial example

- Suppose we want to solve: $ax = b$ (one equation!)
- Build an iterative method $mx^{p+1} = nx^p + b$ with $m - n = a$
- The error equation is $\epsilon^{n+1} = \frac{n}{m}\epsilon^n$
- The method converges quickly (the error decreases fast) if n/m is small or if n is small

  $\rightarrow m \approx a$

Iterative methods

A trivial example

- Suppose we want to solve: $ax = b$ (one equation!)
- Build an iterative method $mx^{p+1} = nx^p + b$ with $m - n = a$
- The error equation is $\epsilon^{n+1} = \frac{n}{m}\epsilon^n$
- The method converges quickly (the error decreases fast) if n/m is small or if n is small

  $\rightarrow m \approx a$

Iterative methods

A trivial example

- Suppose we want to solve: $ax = b$ (one equation!)
- Build an iterative method $mx^{p+1} = nx^p + b$ with $m - n = a$
- The error equation is $\epsilon^{n+1} = \frac{n}{m}\epsilon^n$
- The method converges quickly (the error decreases fast) if n/m is small or if n is small

  $\rightarrow m \approx a$

Iterative methods

A trivial example

- Suppose we want to solve: $ax = b$ (one equation!)
- Build an iterative method $mx^{p+1} = nx^p + b$ with $m - n = a$
- The error equation is $\epsilon^{n+1} = \frac{n}{m}\epsilon^n$
- The method converges quickly (the error decreases fast) if n/m is small or if n is small

  $\rightarrow m \approx a$

Iterative methods

A trivial example

- Suppose we want to solve: $ax = b$ (one equation!)
- Build an iterative method $mx^{p+1} = nx^p + b$ with $m - n = a$
- The error equation is $\epsilon^{n+1} = \frac{n}{m}\epsilon^n$
- The method converges quickly (the error decreases fast) if n/m is small or if n is small

  $\rightarrow m \approx a$

Classic iterative methods

Consider the 5-point discretization we introduced to solve the steady convection/diffusion problem in 2D

$$A_S\phi_S + A_W\phi_W + A_P\phi_P + A_N\phi_N + A_E\phi_E = Q_P$$

- Jacobi: $M = diag(A)$

$$\phi_P^{p+1} = \frac{1}{A_P}\left(Q_P - A_S\phi_S^p - A_W\phi_W^p - A_N\phi_N^p - A_E\phi_E^p\right)$$

- Gauss-Seidel: $M = Lower(A)$

$$\phi_P^{p+1} = \frac{1}{A_P}\left(Q_P - A_S\phi_S^{p+1} - A_W\phi_W^{p+1} - A_N\phi_N^p - A_E\phi_E^p\right)$$

- Successive Over-Relaxation

$$\phi_P^{p+1} = \omega\frac{1}{A_P}\left(Q_P - A_S\phi_S^{p+1} - A_W\phi_W^{p+1} - A_N\phi_N^p - A_E\phi_E^p\right) + (1-\omega)\phi_P^p$$

Classic iterative methods

Consider the 5-point discretization we introduced to solve the steady convection/diffusion problem in 2D

$$A_S\phi_S + A_W\phi_W + A_P\phi_P + A_N\phi_N + A_E\phi_E = Q_P$$

- Jacobi: $M = diag(A)$

$$\phi_P^{p+1} = \frac{1}{A_P}\left(Q_P - A_S\phi_S^p - A_W\phi_W^p - A_N\phi_N^p - A_E\phi_E^p\right)$$

- Gauss-Seidel: $M = Lower(A)$

$$\phi_P^{p+1} = \frac{1}{A_P}\left(Q_P - A_S\phi_S^{p+1} - A_W\phi_W^{p+1} - A_N\phi_N^p - A_E\phi_E^p\right)$$

- Successive Over-Relaxation

$$\phi_P^{p+1} = \omega\frac{1}{A_P}\left(Q_P - A_S\phi_S^{p+1} - A_W\phi_W^{p+1} - A_N\phi_N^p - A_E\phi_E^p\right) + (1-\omega)\phi_P^p$$

Classic iterative methods

Consider the 5-point discretization we introduced to solve the steady convection/diffusion problem in 2D

$$A_S\phi_S + A_W\phi_W + A_P\phi_P + A_N\phi_N + A_E\phi_E = Q_P$$

- Jacobi: $M = diag(A)$

$$\phi_P^{p+1} = \frac{1}{A_P}\left(Q_P - A_S\phi_S^p - A_W\phi_W^p - A_N\phi_N^p - A_E\phi_E^p\right)$$

- Gauss-Seidel: $M = Lower(A)$

$$\phi_P^{p+1} = \frac{1}{A_P}\left(Q_P - A_S\phi_S^{p+1} - A_W\phi_W^{p+1} - A_N\phi_N^p - A_E\phi_E^p\right)$$

- Successive Over-Relaxation

$$\phi_P^{p+1} = \omega\frac{1}{A_P}\left(Q_P - A_S\phi_S^{p+1} - A_W\phi_W^{p+1} - A_N\phi_N^p - A_E\phi_E^p\right) + (1-\omega)\phi_P^p$$

Classic iterative methods

Consider the 5-point discretization we introduced to solve the steady convection/diffusion problem in 2D

$$A_S\phi_S + A_W\phi_W + A_P\phi_P + A_N\phi_N + A_E\phi_E = Q_P$$

- Jacobi: $M = diag(A)$

$$\phi_P^{p+1} = \frac{1}{A_P}\left(Q_P - A_S\phi_S^p - A_W\phi_W^p - A_N\phi_N^p - A_E\phi_E^p\right)$$

- Gauss-Seidel: $M = Lower(A)$

$$\phi_P^{p+1} = \frac{1}{A_P}\left(Q_P - A_S\phi_S^{p+1} - A_W\phi_W^{p+1} - A_N\phi_N^p - A_E\phi_E^p\right)$$

- Successive Over-Relaxation

$$\phi_P^{p+1} = \omega\frac{1}{A_P}\left(Q_P - A_S\phi_S^{p+1} - A_W\phi_W^{p+1} - A_N\phi_N^p - A_E\phi_E^p\right) + (1-\omega)\phi_P^p$$

SOR method

SOR has an adjustable parameter ω

- How do we select it?
- For symmetric matrices it can be proven than $0 < \omega < 2$ leads to convergence!
- $\omega = 1$ is Gauss-Seidel, and $\omega < 1$ leads to slower convergence
- How do we select ω?
- No general rule, but case-specific results are available: not much better than "use $\omega = 1.5$"

SOR method

SOR has an adjustable parameter ω

- How do we select it?
- For symmetric matrices it can be proven than $0 < \omega < 2$ leads to convergence!
- $\omega = 1$ is Gauss-Seidel, and $\omega < 1$ leads to slower convergence
- How do we select ω?
- No general rule, but case-specific results are available: not much better than "use $\omega = 1.5$"

SOR method

SOR has an adjustable parameter ω

- How do we select it?
- For symmetric matrices it can be proven than $0 < \omega < 2$ leads to convergence!
- $\omega = 1$ is Gauss-Seidel, and $\omega < 1$ leads to slower convergence
- How do we select ω?
- No general rule, but case-specific results are available: not much better than "use $\omega = 1.5$"

SOR method

SOR has an adjustable parameter ω

- How do we select it?
- For symmetric matrices it can be proven than $0 < \omega < 2$ leads to convergence!
- $\omega = 1$ is Gauss-Seidel, and $\omega < 1$ leads to slower convergence
- How do we select ω?
- No general rule, but case-specific results are available: not much better than "use $\omega = 1.5$"

SOR method

SOR has an adjustable parameter ω

- How do we select it?
- For symmetric matrices it can be proven than $0 < \omega < 2$ leads to convergence!
- $\omega = 1$ is Gauss-Seidel, and $\omega < 1$ leads to slower convergence
- How do we select ω?
- No general rule, but case-specific results are available: not much better than "use $\omega = 1.5$"

SOR method

SOR has an adjustable parameter ω

- How do we select it?
- For symmetric matrices it can be proven than $0 < \omega < 2$ leads to convergence!
- $\omega = 1$ is Gauss-Seidel, and $\omega < 1$ leads to slower convergence
- How do we select ω?
- No general rule, but case-specific results are available: not much better than "use $\omega = 1.5$"

Iterative methods

We can use the modified equation analysis to gain an understanding of the iteration scheme

- We are solving:
  $A_S\phi_S + A_W\phi_W + A_P\phi_P + A_N\phi_N + A_E\phi_E = Q_P$
  which corresponds to:
  $\sum_f \int_{S_f} \rho\phi\vec{v}\cdot\hat{n}_f dS = \sum_f \int_{S_f} \Gamma\nabla\phi\cdot\hat{n}_f dS$
- An iterative scheme (GS) is solving:
  $$\phi_P^{p+1} = \frac{1}{A_P}\left(Q_P - A_S\phi_S^{p+1} - A_W\phi_W^{p+1} - A_N\phi_N^{p} - A_E\phi_E^{p}\right)$$
  which corresponds to:
  $\frac{d}{dt}\int_{\Omega_i}\rho\phi dV + \sum_f \int_{S_f} \rho\phi\vec{v}\cdot\hat{n}_f dS = \sum_f \int_{S_f} \Gamma\nabla\phi\cdot\hat{n}_f dS$
- Iterative schemes are introducing a pseudo time-stepping!

  R.J. LeVeque L.N. Trefethen, “Fourier Analysis of the SOR Iteration”, IMA J. Num. Analysis (1988)

Iterative methods

We can use the modified equation analysis to gain an understanding of the iteration scheme

- We are solving:
  $A_S\phi_S + A_W\phi_W + A_P\phi_P + A_N\phi_N + A_E\phi_E = Q_P$
  which corresponds to:
  $\sum_f \int_{S_f} \rho\phi\vec{v}\cdot\hat{n}_f dS = \sum_f \int_{S_f} \Gamma\nabla\phi\cdot\hat{n}_f dS$
- An iterative scheme (GS) is solving:
  $$\phi_P^{p+1} = \frac{1}{A_P}\left(Q_P - A_S\phi_S^{p+1} - A_W\phi_W^{p+1} - A_N\phi_N^{p} - A_E\phi_E^{p}\right)$$
  which corresponds to:
  $\frac{d}{dt}\int_{\Omega_i}\rho\phi dV + \sum_f \int_{S_f} \rho\phi\vec{v}\cdot\hat{n}_f dS = \sum_f \int_{S_f} \Gamma\nabla\phi\cdot\hat{n}_f dS$
- Iterative schemes are introducing a pseudo time-stepping!

  R.J. LeVeque L.N. Trefethen, “Fourier Analysis of the SOR Iteration”, IMA J. Num. Analysis (1988)

Iterative methods

We can use the modified equation analysis to gain an understanding of the iteration scheme

- We are solving:
  $A_S\phi_S + A_W\phi_W + A_P\phi_P + A_N\phi_N + A_E\phi_E = Q_P$
  which corresponds to:
  $\sum_f \int_{S_f} \rho\phi\vec{v}\cdot\hat{n}_f dS = \sum_f \int_{S_f} \Gamma\nabla\phi\cdot\hat{n}_f dS$
- An iterative scheme (GS) is solving:
  $$\phi_P^{p+1} = \frac{1}{A_P}\left(Q_P - A_S\phi_S^{p+1} - A_W\phi_W^{p+1} - A_N\phi_N^{p} - A_E\phi_E^{p}\right)$$
  which corresponds to:
  $\frac{d}{dt}\int_{\Omega_i}\rho\phi dV + \sum_f \int_{S_f} \rho\phi\vec{v}\cdot\hat{n}_f dS = \sum_f \int_{S_f} \Gamma\nabla\phi\cdot\hat{n}_f dS$
- Iterative schemes are introducing a pseudo time-stepping!

  R.J. LeVeque L.N. Trefethen, “Fourier Analysis of the SOR Iteration”, IMA J. Num. Analysis (1988)

Iterative methods

We can use the modified equation analysis to gain an understanding of the iteration scheme

- We are solving:
  $A_S\phi_S + A_W\phi_W + A_P\phi_P + A_N\phi_N + A_E\phi_E = Q_P$
  which corresponds to:
  $\sum_f \int_{S_f} \rho\phi\vec{v}\cdot\hat{n}_f dS = \sum_f \int_{S_f} \Gamma\nabla\phi\cdot\hat{n}_f dS$
- An iterative scheme (GS) is solving:
  $$\phi_P^{p+1} = \frac{1}{A_P}\left(Q_P - A_S\phi_S^{p+1} - A_W\phi_W^{p+1} - A_N\phi_N^{p} - A_E\phi_E^{p}\right)$$
  which corresponds to:
  $\frac{d}{dt}\int_{\Omega_i}\rho\phi dV + \sum_f \int_{S_f}\rho\phi\vec{v}\cdot\hat{n}_f dS = \sum_f \int_{S_f}\Gamma\nabla\phi\cdot\hat{n}_f dS$
- Iterative schemes are introducing a pseudo time-stepping!

  R.J. LeVeque L.N. Trefethen, “Fourier Analysis of the SOR Iteration”, IMA J. Num. Analysis (1988)

Iterative methods

We can use the modified equation analysis to gain an understanding of the iteration scheme

- We are solving:
  $A_S\phi_S + A_W\phi_W + A_P\phi_P + A_N\phi_N + A_E\phi_E = Q_P$
  which corresponds to:
  $\sum_f \int_{S_f} \rho\phi\vec{v}\cdot\hat{n}_f dS = \sum_f \int_{S_f} \Gamma\nabla\phi\cdot\hat{n}_f dS$
- An iterative scheme (GS) is solving:
  $$\phi_P^{p+1} = \frac{1}{A_P}\left(Q_P - A_S\phi_S^{p+1} - A_W\phi_W^{p+1} - A_N\phi_N^{p} - A_E\phi_E^{p}\right)$$
  which corresponds to:
  $\frac{d}{dt}\int_{\Omega_i} \rho\phi dV + \sum_f \int_{S_f} \rho\phi\vec{v}\cdot\hat{n}_f dS = \sum_f \int_{S_f} \Gamma\nabla\phi\cdot\hat{n}_f dS$
- Iterative schemes are introducing a pseudo time-stepping!

  R.J. LeVeque L.N. Trefethen, “Fourier Analysis of the SOR Iteration”, IMA J. Num. Analysis (1988)

Iterative methods

We can use the modified equation analysis to gain an understanding of the iteration scheme

- We are solving:
  $A_S\phi_S + A_W\phi_W + A_P\phi_P + A_N\phi_N + A_E\phi_E = Q_P$
  which corresponds to:
  $\sum_f \int_{S_f} \rho\phi\vec{v}\cdot\hat{n}_f dS = \sum_f \int_{S_f} \Gamma\nabla\phi\cdot\hat{n}_f dS$
- An iterative scheme (GS) is solving:
  $$\phi_P^{p+1} = \frac{1}{A_P}\left(Q_P - A_S\phi_S^{p+1} - A_W\phi_W^{p+1} - A_N\phi_N^{p} - A_E\phi_E^{p}\right)$$
  which corresponds to:
  $\frac{d}{dt}\int_{\Omega_i} \rho\phi dV + \sum_f \int_{S_f} \rho\phi\vec{v}\cdot\hat{n}_f dS = \sum_f \int_{S_f} \Gamma\nabla\phi\cdot\hat{n}_f dS$
- Iterative schemes are introducing a pseudo time-stepping!

R.J. LeVeque L.N. Trefethen, “Fourier Analysis of the SOR Iteration”, IMA J. Num. Analysis (1988)

Iterative methods

We can use the modified equation analysis to gain an understanding of the iteration scheme

- We are solving:
  $A_S\phi_S + A_W\phi_W + A_P\phi_P + A_N\phi_N + A_E\phi_E = Q_P$
  which corresponds to:
  $\sum_f \int_{S_f} \rho\phi\vec{v}\cdot\hat{n}_f dS = \sum_f \int_{S_f} \Gamma\nabla\phi\cdot\hat{n}_f dS$
- An iterative scheme (GS) is solving:
  $$\phi_P^{p+1} = \frac{1}{A_P}\left(Q_P - A_S\phi_S^{p+1} - A_W\phi_W^{p+1} - A_N\phi_N^{p} - A_E\phi_E^{p}\right)$$
  which corresponds to:
  $\frac{d}{dt}\int_{\Omega_i} \rho\phi dV + \sum_f \int_{S_f} \rho\phi\vec{v}\cdot\hat{n}_f dS = \sum_f \int_{S_f} \Gamma\nabla\phi\cdot\hat{n}_f dS$
- Iterative schemes are introducing a pseudo time-stepping!

  R.J. LeVeque L.N. Trefethen, “Fourier Analysis of the SOR Iteration”, IMA J. Num. Analysis (1988)