Scientific Computing WS 2017/2018

Lecture 10

Jürgen Fuhrmann

juergen.fuhrmann@wias-berlin.de

Corrigendum: there is an Perron-Frobenius theorem for general matrices

Perron-Frobenius Theorem (1912/1907)

Definition: A real n-vector $\mathbf{x}$ is

- positive ($\mathbf{x} > 0$) if all entries of $\mathbf{x}$ are positive
- nonnegative ($\mathbf{x} \geq 0$) if all entries of $\mathbf{x}$ are nonnegative

Definition: A real $n \times n$ matrix A is

- positive ($A > 0$) if all entries of A are positive
- nonnegative ($A \geq 0$) if all entries of A are nonnegative

Theorem(Varga, Th. 2.7) Let $A \geq 0$ be an irreducible $n \times n$ matrix. Then

(i) A has a positive real eigenvalue equal to its spectral radius $\rho(A)$.
(ii) To $\rho(A)$ there corresponds a positive eigenvector $\mathbf{x} > 0$.
(iii) $\rho(A)$ increases when any entry of A increases.
(iv) $\rho(A)$ is a simple eigenvalue of A.

Proof: See Varga. $\square$

Regular splittings

- $A = M - N$ is a regular splitting if
  - M is nonsingular
  - M^{-1}, N are nonnegative, i.e. have nonnegative entries
- Regard the iteration $u_{k+1} = M^{-1}Nu_k + M^{-1}b$.
- We have $I - M^{-1}A = M^{-1}N$.

Convergence theorem for regular splitting

Theorem: Assume A is nonsingular, $A^{-1} \geq 0$, and $A = M - N$ is a regular splitting. Then $\rho(M^{-1}N) < 1$.

Proof: Let $G = M^{-1}N$. Then $A = M(I - G)$, therefore $I - G$ is nonsingular.

In addition

$$A^{-1}N = (M(I - M^{-1}N))^{-1}N = (I - M^{-1}N)^{-1}M^{-1}N = (I - G)^{-1}G$$

By Perron-Frobenius (for general matrices), $\rho(G)$ is an eigenvalue with a nonnegative eigenvector $\mathbf{x}$. Thus,

$$0 \leq A^{-1}N\mathbf{x} = \frac{\rho(G)}{1 - \rho(G)}\mathbf{x}$$

Therefore $0 \leq \rho(G) \leq 1$.
As $I - G$ is nonsingular, $\rho(G) < 1$. $\square$

Perron-Frobenius for general nonnegative matrices

Each $n \times n$ matrix can be brought to the normal form

$$PAP^T = \begin{pmatrix} R_{11} & R_{12} & \dots & R_{1m} \\ 0 & R_{22} & \dots & R_{2m} \\ \vdots & & \ddots & \\ 0 & 0 & \dots & R_{mm} \end{pmatrix}$$

where for $j = 1 \dots m$, either R_{jj} irreducible or $R_{jj} = (0)$.

Theorem(Varga, Th. 2.20) Let $A \geq 0$ be an $n \times n$ matrix. Then

(i) A has a nonnegative eigenvalue equal to its spectral radius $\rho(A)$. This eigenvalue is positive unless A is reducible and its normal form is strictly upper triangular

(ii) To $\rho(A)$ there corresponds a nonzero eigenvector $\mathbf{x} \geq 0$.

(iii) $\rho(A)$ does not decrease when any entry of A increases.

Proof: See Varga; $\sigma(A) = \bigcup_{j=1}^{m} \sigma(R_{jj})$, apply irreducible Perron-Frobenius to R_{jj}. $\square$

Incomplete LU factorizations (ILU)

Idea (Varga, Buleev, 1960):

- fix a predefined zero pattern
- apply the standard LU factorization method, but calculate only those elements, which do not correspond to the given zero pattern
- Result: incomplete LU factors L, U, remainder R:

$$A = LU - R$$

- Problem: with complete LU factorization procedure, for any nonsingular matrix, the method is stable, i.e. zero pivots never occur. Is this true for the incomplete LU Factorization as well ?

Comparison of M-Matrices

Theorem(Saad, Th. 1.33): Let A, B $n \times n$ matrices such that

(i) $A \leq B$

(ii) $b_{ij} \leq 0$ for $i \neq j$.

Then, if A is an M-Matrix, so is B.

Proof: For the diagonal parts, one has $D_B \geq D_A > 0$,
$D_A - A \geq D_B - B \geq 0$ Therefore

$$I - D_A^{-1}A \geq D_A^{-1}(D_B - B) \geq D_B^{-1}(D_B - B) = I - D_B^{-1}B =: G \geq 0.$$

Perron-Frobenius $\Rightarrow \rho(G) = \rho(I - D_B^{-1}B) \leq \rho(I - D_A^{-1}A) < 1$
$\Rightarrow I - G$ is nonsingular. From the proof of the M-matrix criterion,
$D_B^{-1}B = (I - G)^{-1} = \sum_{k=0}^{\infty} G^k \geq 0$. As $D_B > 0$, we get $B \geq 0$.

□

M-Property propagation in Gaussian Elimination

Theorem:(Ky Fan; Saad Th 1.10) Let A be an M-matrix. Then the matrix A_1 obtained from the first step of Gaussian elimination is an M-matrix.

Proof: One has $a_{ij}^1 = a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}}$,
$a_{ij}, a_{i1}, a_{1j} \leq 0$, $a_{11} > 0$
$\Rightarrow a_{ij}^1 \leq 0$ for $i \neq j$

$$A = L_1 A_1 \text{ with } L_1 = \begin{pmatrix} 1 & 0 & \dots & 0 \\ \frac{-a_{12}}{a_{11}} & 1 & \dots & 0 \\ \vdots & & \ddots & 0 \\ \frac{-a_{1n}}{a_{11}} & 0 & \dots & 1 \end{pmatrix} \text{ nonsingular, nonnegative}$$

$\Rightarrow A_1$ nonsingular

Let $e_1 \dots e_n$ be the unit vectors. Then $A_1^{-1}e_1 = \frac{1}{a_11}e_1 \geq 0$. For $j > 1$,
$A_1^{-1}e_j = A^{-1}L^{-1}e_j = A^{-1}e_j \geq 0$.
$\Rightarrow A_1^{-1} \geq 0$

□

Stability of ILU

Theorem (Saad, Th. 10.2): If A is an M-Matrix, then the algorithm to compute the incomplete LU factorization with a given nonzero pattern

$$A = LU - R$$

is stable. Moreover, $A = LU - R$ is a regular splitting.

Stability of ILU decomposition II

Proof

Let $\tilde{A}_1 = A_1 + R_1 = L_1 A + R_1$ where R_1 is a nonnegative matrix which occurs from dropping some off diagonal entries from A_1. Thus, $\tilde{A}_1 \geq A_1$ and $\tilde{A}_1$ is an M-matrix. We can repeat this recursively

$$\begin{aligned}\tilde{A}_k = A_k + R_k &= L_k A_{k-1} + R_k \\ &= L_k L_{k-1} A_{k-2} + L_k R_{k-1} + R_k \\ &= L_k L_{k-1} \cdot \ldots \cdot L_1 A + L_k L_{k-1} \cdot \ldots \cdot L_2 R_1 + \cdots + R_k\end{aligned}$$

Let $L = (L_{n-1} \cdot \ldots \cdot L_1)^{-1}$, $U = \tilde{A}_{n-1}$. Then $U = L^{-1} A + S$ with

$S = L_{n-1} L_{n-2} \cdot \ldots \cdot L_2 R_1 + \cdots + R_{n-1} = L_{n-1} L_{n-2} \cdot \ldots \cdot L_2 (R_1 + R_2 + \ldots R_{n-1})$

Let $R = R_1 + R_2 + \ldots R_{n-1}$, then $A = LU - R$ where $U^{-1} L^{-1}$, R are nonnegative.

$\square$

ILU(0)

- Special case of ILU: ignore any fill-in.
- Representation:

$$M = (\tilde{D} - E)\tilde{D}^{-1}(\tilde{D} - F)$$

- $\tilde{D}$ is a diagonal matrix (wich can be stored in one vector) which is calculated by the incomplete factorization algorithm.
- Setup:

```
for(int i=0;i<n;i++)
  d(i)=a(i,i)

for(int i=0;i<n;i++)
{
  d(i)=1.0/d(i)
  for (int j=i+1;j<n;j++)
  d(j)=d(j)-a(i,j)*d(i)*a(j,i)
}
```

ILU(0)

Solve $Mu = v$

```
for(int i=0;i<n;i++)
{
  double x=0.0;
  for (int j=0;j<i;i++)
      x=x+a(i,j)*u(j)
  u(i)=d(i)*(v(i)-x)

}

for(int i=n-1;i>=0;i--)
{
   double x=0.0
   for(int j=i+1;j<n;j++)
       x=x+a(i,j)*u(j)
   u(i)=u(i)-d(i)*x
}
```

ILU(0)

- Generally better convergence properties than Jacobi, Gauss-Seidel
- One can develop block variants
- Alternatives:
  - ILUM: ("modified"): add ignored off-diagonal entries to $\tilde{D}$
  - ILUT: zero pattern calculated dynamically based on drop tolerance
- Dependence on ordering
- Can be parallelized using graph coloring
- Not much theory: experiment for particular systems
- I recommend it as the default initial guess for a sensible preconditioner
- Incomplete Cholesky: symmetric variant of ILU

Preconditioners

- Leave this topic for a while now
- Hopefully, we well be able to discuss
  - Multigrid: gives $O(n)$ complexity in optimal situations
  - Domain decomposition: Structurally well suited for large scale parallelization

Convergence theorem for regular splitting

Theorem: Assume A is nonsingular, $A^{-1} \geq 0$, and $A = M - N$ is a regular splitting. Then $\rho(M^{-1}N) < 1$.

Proof: Let $G = M^{-1}N$. Then $A = M(I - G)$, therefore $I - G$ is nonsingular.

In addition

$$A^{-1}N = (M(I - M^{-1}N))^{-1}N = (I - M^{-1}N)^{-1}M^{-1}N = (I - G)^{-1}G$$

By Perron-Frobenius (for general nonnegative matrices), $\rho(G)$ is an eigenvalue with an eigenvector $\mathbf{x} \geq 0$. Thus,

$$0 \leq A^{-1}N\mathbf{x} = \frac{\rho(G)}{1 - \rho(G)}\mathbf{x}$$

Therefore $0 \leq \rho(G) \leq 1$.
As $I - G$ is nonsingular, $\rho(G) < 1$. $\square$

Recap (ILU + proof

ILU(0)

- Special case of ILU: ignore any fill-in.
- Representation:

$$M = (\tilde{D} - E)\tilde{D}^{-1}(\tilde{D} - F)$$

- $\tilde{D}$ is a diagonal matrix (wich can be stored in one vector) which is calculated by the incomplete factorization algorithm.
- Setup:

```
for(int i=0;i<n;i++)
  d(i)=a(i,i)

for(int i=0;i<n;i++)
{
  d(i)=1.0/d(i)
  for (int j=i+1;j<n;j++)
  d(j)=d(j)-a(i,j)*d(i)*a(j,i)
}
```

ILU(0)

Solve $Mu = v$

```
for(int i=0;i<n;i++)
{
  double x=0.0;
  for (int j=0;j<i;i++)
      x=x+a(i,j)*u(j)
  u(i)=d(i)*(v(i)-x)

}

for(int i=n-1;i>=0;i--)
{
   doubl x=0.0
   for(int j=i+1;j<n;j++)
       x=x+a(i,j)*u(j)
   u(i)=u(i)-d(i)*x
}
```

ILU(0)

- Generally better convergence properties than Jacobi, Gauss-Seidel
- One can develop block variants
- Alternatives:
  - ILUM: ("modified"): add ignored off-diagonal entries to $\tilde{D}$
  - ILUT: zero pattern calculated dynamically based on drop tolerance
- Dependence on ordering
- Can be parallelized using graph coloring
- Not much theory: experiment for particular systems
- I recommend it as the default initial guess for a sensible preconditioner
- Incomplete Cholesky: symmetric variant of ILU

Solution of SPD system as a minimization procedure

Regard $Au = f$,where A is symmetric, positive definite. Then it defines a bilinear form $a : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$

$$a(u, v) = (Au, v) = v^T Au = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} v_i u_j$$

As A is SPD, for all $u \neq 0$ we have $(Au, u) > 0$.

For a given vector b, regard the function

$$f(u) = \frac{1}{2} a(u, u) - b^T u$$

What is the minimizer of f ?

$$f'(u) = Au - b = 0$$

- Solution of SPD system $\equiv$ minimization of f.

Method of steepest descent

- Given some vector u_i, look for a new iterate u_{i+1}.
- The direction of steepest descend is given by $-f'(u_i)$.
- So look for u_{i+1} in the direction of $-f'(u_i) = r_i = b - Au_i$ such that it minimizes f in this direction, i.e. set $u_{i+1} = u_i + \alpha r_i$ with α choosen from

$$\begin{aligned}
0 &= \frac{d}{d\alpha} f(u_i + \alpha r_i) = f'(u_i + \alpha r_i) \cdot r_i \\
&= (b - A(u_i + \alpha r_i), r_i) \\
&= (b - Au_i, r_i) - \alpha(Ar_i, r_i) \\
&= (r_i, r_i) - \alpha(Ar_i, r_i) \\
\alpha &= \frac{(r_i, r_i)}{(Ar_i, r_i)}
\end{aligned}$$

Method of steepest descent: iteration scheme

$$r_i = b - Au_i$$
$$\alpha_i = \frac{(r_i, r_i)}{(Ar_i, r_i)}$$
$$u_{i+1} = u_i + \alpha_i r_i$$

Let $\hat{u}$ the exact solution. Define $e_i = u_i - \hat{u}$, then $r_i = -Ae_i$

Let $||u||_A = (Au, u)^{\frac{1}{2}}$ be the *energy norm* wrt. A.

Theorem The convergence rate of the method is

$$||e_i||_A \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^i ||e_0||_A$$

where $\kappa = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}$ is the spectral condition number.

Method of steepest descent: advantages

- Simple Richardson iteration $u_{k+1} = u_k - \alpha(Au_k - f)$ needs good eigenvalue estimate to be optimal with $\alpha = \frac{2}{\lambda_{max}+\lambda_{min}}$
- In this case, asymptotic convergence rate is $\rho = \frac{\kappa-1}{\kappa+1}$
- Steepest descent has the same rate without need for spectral estimate

Conjugate directions

For steepest descent, there is no guarantee that a search direction $d_i = r_i = -Ae_i$ is not used several times. If all search directions would be orthogonal, or, indeed, A-orthogonal, one could control this situation.

So, let $d_0, d_1 \dots d_{n-1}$ be a series of A-orthogonal (or conjugate) search directions, i.e. $(Ad_i, d_j) = 0$, $i \neq j$.

- Look for u_{i+1} in the direction of d_i such that it minimizes f in this direction, i.e. set $u_{i+1} = u_i + \alpha_i d_i$ with α choosen from

$$
\begin{aligned}
0 &= \frac{d}{d\alpha} f(u_i + \alpha d_i) = f'(u_i + \alpha d_i) \cdot d_i \\
&= (b - A(u_i + \alpha d_i), d_i) \\
&= (b - Au_i, d_i) - \alpha (Ad_i, d_i) \\
&= (r_i, d_i) - \alpha (Ad_i, d_i) \\
\alpha_i &= \frac{(r_i, d_i)}{(Ad_i, d_i)}
\end{aligned}
$$

Conjugate directions II

$e_0 = u_0 - \hat{u}$ (such that $Ae_0 = -r_0$) can be represented in the basis of the search directions:

$$e_0 = \sum_{i=0}^{n-1} \delta_j d_j$$

Projecting onto d_k in the A scalar product gives

$$\begin{aligned}
(Ae_0, d_k) &= \sum_{i=0}^{n-1} \delta_j (Ad_j, d_k) \\
&= \delta_k (Ad_k, d_k) \\
\delta_k &= \frac{(Ae_0, d_k)}{(Ad_k, d_k)} = \frac{(Ae_0 + \sum_{i<k} \alpha_i d_i, d_k)}{(Ad_k, d_k)} = \frac{(Ae_k, d_k)}{(Ad_k, d_k)} \\
&= \frac{(r_k, d_k)}{(Ad_k, d_k)} \\
&= -\alpha_k
\end{aligned}$$

Conjugate directions III

Then,

$$\begin{aligned} e_i &= e_0 + \sum_{j=0}^{i-1} \alpha_j d_j = -\sum_{j=0}^{n-1} \alpha_j d_j + \sum_{j=0}^{i-1} \alpha_j d_j \\ &= -\sum_{j=i}^{n-1} \alpha_j d_j \end{aligned}$$

So, the iteration consists in component-wise suppression of the error, and it must converge after n steps. Let $k \leq i$. A-projection on d_k gives

$$(Ae_i, d_k) = -\sum_{j=i}^{n-1} \alpha_j (Ad_j, d_k) = 0$$

Therefore, $r_i = Ae_i$ is orthogonal to $d_0 \ldots d_{i-1}$.

Conjugate directions IV

Looking at the error norm $||e_i||_A$, the method yields the element with the minimum energy norm from all elements of the affine space $e_0 + \mathcal{K}_i$ where $\mathcal{K}_i = \mathrm{span}\{d_0, d_1 \ldots d_{i-1}\}$

$$
\begin{aligned}
(Ae_i, e_i) &= \left(\sum_{j=i}^{n-1} \delta_j d_j, \sum_{j=i}^{n-1} \delta_j d_j\right) = \sum_{j=i}^{n-1}\sum_{k=i}^{n-1} \delta_j \delta_k (d_j, d_k) \\
&= \sum_{j=i}^{n-1} \delta_j^2 (d_j, d_j) = \min_{e \in e_0 + \mathcal{K}_i} ||e||_A
\end{aligned}
$$

Furthermore, we have

$$
\begin{aligned}
u_{i+1} &= u_i + \alpha_i d_i \\
e_{i+1} &= e_i + \alpha_i d_i \\
Ae_{i+1} &= Ae_i + \alpha_i Ad_i \\
r_{i+1} &= r_i - \alpha_i Ad_i
\end{aligned}
$$

By what magic we can obtain these d_i?

Gram-Schmidt Orthogonalization

- Assume we have been given some linearly independent vectors $v_0, v_1 \dots v_{n-1}$.
- Set $d_0 = v_0$
- Define

$$d_i = v_i + \sum_{k=0}^{i-1} \beta_{ik} d_k$$

- For $j < i$, A-project onto d_j and require orthogonality:

$$\begin{aligned}(Ad_i, d_j) &= (Av_i, d_j) + \sum_{k=0}^{i-1} \beta_{ik}(Ad_k, d_j)\\ 0 &= (Av_i, d_j) + \beta_{ij}(Ad_j, d_j)\\ \beta_{ij} &= -\frac{(Av_i, d_j)}{(Ad_j, d_j)}\end{aligned}$$

- If v_i are the coordinate unit vectors, this is Gaussian elimination!
- If v_i are arbitrary, they all must be kept in the memory

Conjugate gradients (Hestenes, Stiefel, 1952)

As Gram-Schmidt builds up d_i from d_j, $j < i$, we can choose $v_i = r_i$, i.e. the residuals built up during the conjugate direction process.

Let $\mathcal{K}_i = \mathrm{span}\{d_0 \dots d_{i-1}\}$. Then, $r_i \perp \mathcal{K}_i$

But d_i are built by Gram-Schmidt from the residuals, so we also have $\mathcal{K}_i = \mathrm{span}\{r_0 \dots r_{i-1}\}$ and $(r_i, r_j) = 0$ for $j < i$.

From $r_i = r_{i-1} - \alpha_{i-1} A d_{i-1}$ we obtain

$\mathcal{K}_i = \mathcal{K}_{i-1} \cup \mathrm{span}\{A d_{i-1}\}$

This gives two other representations of $\mathcal{K}_i$:

$$\begin{aligned}\mathcal{K}_i &= \mathrm{span}\{d_0, Ad_0, A^2 d_0, \dots, A^{i-1} d_0\}\\ &= \mathrm{span}\{r_0, Ar_0, A^2 r_0, \dots, A^{i-1} r_0\}\end{aligned}$$

Such type of subspace of $\mathbb{R}^n$ is called *Krylov subspace*, and orthogonalization methods are more often called *Krylov subspace methods*.

Conjugate gradients II

Look at Gram-Schmidt under these conditions. The essential data are (setting $v_i = r_i$ and using $j < i$) $\beta_{ij} = -\frac{(Ar_i, d_j)}{(Ad_j, d_j)} = -\frac{(Ad_j, r_i)}{(Ad_j, d_j)}$.

Then, for $j \le i$:

$$\begin{aligned}
r_{j+1} &= r_j - \alpha_j A d_j \\
(r_{j+1}, r_i) &= (r_j, r_i) - \alpha_j (A d_j, r_i) \\
\alpha_j (A d_j, r_i) &= (r_j, r_i) - (r_{j+1}, r_i) \\
(A d_j, r_i) &= \begin{cases} -\frac{1}{\alpha_j}(r_{j+1}, r_i), & j+1 = i \\ \frac{1}{\alpha_j}(r_j, r_i), & j = i \\ 0, & \text{else} \end{cases} = \begin{cases} -\frac{1}{\alpha_{i-1}}(r_i, r_i), & j+1 = i \\ \frac{1}{\alpha_i}(r_i, r_i), & j = i \\ 0, & \text{else} \end{cases}
\end{aligned}$$

For $j < i$:

$$\beta_{ij} = \begin{cases} \frac{1}{\alpha_{i-1}} \frac{(r_i, r_i)}{(A d_{i-1}, d_{i-1})}, & j+1 = i \\ 0, & \text{else} \end{cases}$$

Conjugate gradients III

For Gram-Schmidt we defined (replacing v_i by r_i):

$$\begin{aligned} d_i &= r_i + \sum_{k=0}^{i-1} \beta_{ik} d_k \\ &= r_i + \beta_{i,i-1} d_{i-1} \end{aligned}$$

So, the new orthogonal direction depends only on the previous orthogonal direction and the current residual. We don't have to store old residuals or search directions. In the sequel, set $\beta_i := \beta_{i,i-1}$.

We have

$$\begin{aligned} d_{i-1} &= r_{i-1} + \beta_{i-1} d_{i-2} \\ (d_{i-1}, r_{i-1}) &= (r_{i-1}, r_{i-1}) + \beta_{i-1}(d_{i-2}, r_{i-1}) \\ &= (r_{i-1}, r_{i-1}) \\ \beta_i &= \frac{1}{\alpha_{i-1}} \frac{(r_i, r_i)}{(Ad_{i-1}, d_{i-1})} = \frac{(r_i, r_i)}{(d_{i-1}, r_{i-1})} \\ &= \frac{(r_i, r_i)}{(r_{i-1}, r_{i-1})} \end{aligned}$$

Conjugate gradients IV - The algorithm

Given initial value u_0, spd matrix A, right hand side b.

$$\begin{aligned} d_0 &= r_0 = b - Au_0 \\ \alpha_i &= \frac{(r_i, r_i)}{(Ad_i, d_i)} \\ u_{i+1} &= u_i + \alpha_i d_i \\ r_{i+1} &= r_i - \alpha_i Ad_i \\ \beta_{i+1} &= \frac{(r_{i+1}, r_{i+1})}{(r_i, r_i)} \\ d_{i+1} &= r_{i+1} + \beta_{i+1} d_i \end{aligned}$$

At the i-th step, the algorithm yields the element from $e_0 + \mathcal{K}_i$ with the minimum energy error.

Theorem The convergence rate of the method is

$$||e_i||_A \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^i ||e_0||_A$$

where $\kappa = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}$ is the spectral condition number.

Preconditioning

Let M be spd, and spectrally equivalent to A, and assume that $\kappa(M^{-1}A) << \kappa(A)$.

Let E be such that $M = EE^T$, e.g. its Cholesky factorization. Then, $\sigma(M^{-1}A) = \sigma(E^{-1}AE^{-T})$:

Assume $M^{-1}Au = \lambda u$. We have

$$(E^{-1}AE^{-T})(E^Tu) = (E^TE^{-T})E^{-1}Au = E^TM^{-1}Au = \lambda E^Tu$$

$\Leftrightarrow$ E^Tu is an eigenvector of $E^{-1}AE^{-T}$ with eigenvalue λ.

Preconditioned CG I

Now we can use the CG algorithm for the preconditioned system

$$E^{-1}AE^{-T}\tilde{x} = E^{-1}b$$

with $\tilde{u} = E^T u$

$$\begin{aligned}
\tilde{d}_0 &= \tilde{r}_0 = E^{-1}b - E^{-1}AE^{-T}u_0 \\
\alpha_i &= \frac{(\tilde{r}_i, \tilde{r}_i)}{(E^{-1}AE^{-T}\tilde{d}_i, \tilde{d}_i)} \\
\tilde{u}_{i+1} &= \tilde{u}_i + \alpha_i \tilde{d}_i \\
\tilde{r}_{i+1} &= \tilde{r}_i - \alpha_i E^{-1}AE^{-T}\tilde{d}_i \\
\beta_{i+1} &= \frac{(\tilde{r}_{i+1}, \tilde{r}_{i+1})}{(\tilde{r}_i, \tilde{r}_i)} \\
\tilde{d}_{i+1} &= \tilde{r}_{i+1} + \beta_{i+1}\tilde{d}_i
\end{aligned}$$

Not very practical as we need E

Preconditioned CG II

Assume $\tilde{r}_i = E^{-1} r_i$, $\tilde{d}_i = E^T d_i$, we get the equivalent algorithm

$$
\begin{aligned}
r_0 &= b - Au_0 \\
d_0 &= M^{-1} r_0 \\
\alpha_i &= \frac{(M^{-1} r_i, r_i)}{(Ad_i, d_i)} \\
u_{i+1} &= u_i + \alpha_i d_i \\
r_{i+1} &= r_i - \alpha_i A d_i \\
\beta_{i+1} &= \frac{(M^{-1} r_{i+1}, r_{i+1})}{(r_i, r_i)} \\
d_{i+1} &= M^{-1} r_{i+1} + \beta_{i+1} d_i
\end{aligned}
$$

It relies on the solution of the preconditioning system, the calculation of the matrix vector product and the calculation of the scalar product.

A few issues

Usually we stop the iteration when the residual r becomes small. However during the iteration, floating point errors occur which distort the calculations and lead to the fact that the accumulated residuals

$$r_{i+1} = r_i - \alpha_i A d_i$$

give a much more optimistic picture on the state of the iteration than the real residual

$$r_{i+1} = b - A u_{i+1}$$

C++ implementation

```
template < class Matrix, class Vector, class Preconditioner, class Real >
int  CG(const Matrix &A, Vector &x, const Vector &b,
   const Preconditioner &M, int &max_iter, Real &tol)
{ Real resid;
  Vector p, z, q;
  Vector alpha(1), beta(1), rho(1), rho_1(1);
  Real normb = norm(b);
  Vector r = b - A*x;
  if (normb == 0.0)   normb = 1;
  if ((resid = norm(r) / normb) <= tol) {
    tol = resid;
    max_iter = 0;
    return 0;
  }
  for (int i = 1; i <= max_iter; i++) {
    z = M.solve(r);
    rho(0) = dot(r, z);
    if (i == 1)
      p = z;
    else {
      beta(0) = rho(0) / rho_1(0);
      p = z + beta(0) * p;
    }
    q = A*p;
    alpha(0) = rho(0) / dot(p, q);
    x += alpha(0) * p;
    r -= alpha(0) * q;
    if ((resid = norm(r) / normb) <= tol) {
      tol = resid;
      max_iter = i;
      return 0;
    }
    rho_1(0) = rho(0);
  }
  tol = resid;   return 1;
}
```

C++ implementation II

- Available from http://www.netlib.org/templates/cpp//cg.h
- Slightly adapted for numcxx
- Available in numxx in the namespace netlib.

Unsymmetric problems

- By definition, CG is only applicable to symmetric problems.
- The biconjugate gradient (BICG) method provides a generalization:

Choose initial guess x_0, perform

$$\begin{aligned}
r_0 &= b - Ax_0 & \hat{r}_0 &= \hat{b} - \hat{x}_0 A^T \\
p_0 &= r_0 & \hat{p}_0 &= \hat{r}_0 \\
\alpha_i &= \frac{(\hat{r}_i, r_i)}{(\hat{p}_i, Ap_i)} & & \\
x_{i+1} &= x_i + \alpha_i p_i & \hat{x}_{i+1} &= \hat{x}_i + \alpha_i \hat{p}_i \\
r_{i+1} &= r_i - \alpha_i A p_i & \hat{r}_{i+1} &= \hat{r}_i - \alpha_i \hat{p}_i A^T \\
\beta_i &= \frac{(\hat{r}_{i+1}, r_{i+1})}{(\hat{r}_i, r_i)} & & \\
p_{i+1} &= r_{i+1} + \beta_i p_i & \hat{p}_{i+1} &= \hat{r}_{i+1} + \beta_i \hat{p}_i
\end{aligned}$$

The two sequences produced by the algorithm are biorthogonal, i.e., $(\hat{p}_i, Ap_j) = (\hat{r}_i, r_j) = 0$ for $i \neq j$.

Unsymmetric problems II

- BiCG is very unstable and additionally needs the transposed matrix vector product, it is seldomly used in practice
- There is as well a preconditioned variant of BiCG which also needs the transposed preconditioner.
- Main practical approaches to fix the situation:
  - "Stabilize" BiCG $\rightarrow$ BiCGstab (H. Van der Vorst, 1992)
  - tweak CG $\rightarrow$ "Conjugate gradients squared" (CGS, Sonneveld, 1989)
  - Error minimization in Krylov subspace $\rightarrow$ "Generalized Minimum Residual" (GMRES, Saad/Schulz, 1986)
- Both CGS and BiCGstab can show rather erratic convergence behavior
- For GMRES one has to keep the full Krylov subspace, which is not possible in practice $\Rightarrow$ restart strategy.
- From my experience, BiCGstab is a good first guess