HUMAN CAPITAL
NATIONAL COHESION STRATEGY

EUROPEAN UNION
EUROPEAN
SOCIAL FUND

High Performance Computing

Lecture 10 - Basic Iterative Methods

Felicja Okulicka-Dłużewska

F.Okulicka@mini.pw.edu.pl

Warsaw University of Technology

Faculty of Mathematics and Information Science

WARSAW UNIVERSITY OF TECHNOLOGY
DEVELOPMENT PROGRAMME

Go to full-screen mode now by hitting CTRL-L

Overview

- **Introduction**
- Stationary iterative methods
  - Jacobi iterative method
  - Gauss-Seidel iterative method
  - Overrelaxation iterative method
- Krylov iterative methods
  - Krylov subspace
  - Conjugate gradient

We have the set of linear equations

$$Ax = b \tag{1}$$

where

A is a matrix $n \times n$,

b is a vector of right side,

$x \in R^n$ is a vector of unknowns.

An iterative method attempts to solve a problem by finding successive approximations to the solution starting from an initial guess.

Iterative methods

The popularity of the iterative methods comes from the facts that:

- in the parallel implementation the cost of communication is really lower comparing the direct methods due to the fact that only the vector of approximated solution is distributed between processes during iterations; in the case of direct methods the submatrices have to be send.
- iterative methods are used mostly for sparse matrices - they preserve the matrix structure

Iterative methods

- they do not change the structure of the matrix whereas direct methods can increase the number of nonzeros in the matrix obtained in the steps of the calculations.

The problem which arises is the convergence. The preconditioners were introduced to improve the convergence and condition coefficient of the matrix.

Iterative methods

The algorithm of iterative method :

1. Guess the start point $x^{(0)}$, $i = 0$
2. Repeat

   $x^{(i+1)} := iterative\_formula(x^{(i)})$

   calculate $\epsilon := |x^{(i+1)} - x^{(i)}|$

   $i := i + 1$

   until $\epsilon < tolerance$ or $i > Max\_iteration$

The algorithm stops with the solution when we reach the required tolerance or we end the iterations when the number of repetitions is greater then the assumed number $Max\_iteration$.

Condition number

The section is recalled after **Wiki**.

The condition number associated with a problem is **a measure of that problem's amenability to digital computation**, that is, how numerically well-conditioned the problem is.

A problem with a low condition number is said to be **well-conditioned**, while a problem with a high condition number is said to be **ill-conditioned**.

Condition number

The condition number associated with the linear equation

$$Ax = b$$

gives a bound on how inaccurate the solution x will be after approximate solution. Conditioning is **a property of the matrix**, not the algorithm or floating point accuracy of the computer used to solve the corresponding system.

Condition number

The condition number can be (very roughly) the rate at which the solution, x, will change with respect to a change in b.

Thus, if the condition number is large, even a small error in b may cause a large error in x.

On the other hand, if the condition number is small then the error in x will not be much bigger than the error in b.

Condition number

The condition number is defined more precisely to be the maximum ratio of the relative error in x divided by the relative error in b.

Let e be the error in b. Assuming that A is a square matrix, the error in the solution $A^{-1}b$ is $A^{-1}e$:

$$A(x + \delta x) = b + e$$

$$\delta x = A^{-1} \cdot e$$

The ratio of the relative error in the solution to the relative error in b is

$$\frac{\delta x/x}{e/b} = \frac{||A^{-1}e||//||A^{-1}b||}{||e||/||b||}$$

Condition number

This is easily transformed to

$$\frac{||A^{-1}e||}{||e||} \cdot \frac{||b||}{||A^{-1}b||} = \frac{||A^{-1}e||}{||e||} \cdot \frac{||b||}{||x||}$$

We have also:

$$||b|| = ||A \cdot x|| \leq ||A|| \cdot ||x||$$

Condition number

The maximum value (for nonzero b and e) is easily seen to be the product of the two operator norms:

$$\frac{||A^{-1}e||}{||e||} \cdot \frac{||b||}{||x||} \leq \frac{||A^{-1}|| \cdot ||e||}{||e||} \cdot ||A||$$

$$\kappa(A) = ||A|| \cdot ||A^{-1}||$$

The same definition is used for any consistent norm. This number arises so often in numerical linear algebra that it is given a name, **the condition number of a matrix**.

Condition number

This definition depends on the choice of norm.

- If $||\cdot||$ is the l_2 norm then

$$\kappa(A) = \frac{\sigma_{max}(A)}{\sigma_{min}(A)}$$

where $\sigma_{max}(A)$ and $\sigma_{min}(A)$ are maximal and minimal singular values of A respectively.

- If A is normal then

$$\kappa(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}$$

where $\lambda_{max}(A)$ and $\lambda_{min}(A)$ are maximal and minimal (by moduli) eigenvalues of A respectively.

Condition number

- If A is unitary then

$$\kappa(A) = 1$$

- If $||\cdot||$ is l_∞ norm and A is lower triangular non-singular (i.e., $a_{ii} \neq 0$ for each i) then

$$\kappa(A) \geq \frac{\max_i(|a_{ii}|)}{\min_i(|a_{ii}|)}$$

Overview

- Introduction
- **Stationary iterative methods**
  - Jacobi iterative method
  - Gauss-Seidel iterative method
  - Overrelaxation iterative method
- Krylov iterative methods
  - Krylov subspace
  - Conjugate gradient

Stationary iteration

Stationary iterative methods

In the matrix form the formulas are written using the following splitting of A:

$$A = D - E - F$$

D - diagonal

$-E$- strict lower part

$-F$- strict upper part

It is always assumed that diagonal entries of A are all nonzero.

Our equation can be written in the following form:

$$(D - E - F) * x = b$$

Jacobi Iteration

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

$-E$ is the strict lower part,

$-F$ is the strict upper part,

D is diagonal

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

Jacobi method converges when A is diagonally dominant.

Gauss-Seidel method

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

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

$-E$ is the strict lower part, $-F$ is the strict upper part, D is diagonal

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

We solve the system with lower triangular matrix:

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

Backward G-S

Backward Gauss-Seidel:

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

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

$-E$ is the strict lower part, $-F$ is the strict upper part,
D is diagonal

We solve the system with upper triangular matrix:

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

Splitting of matrix

The Jacobi and Gauss-Seidel iterations are both of the form:

$$Mx^{(k+1)} = N * x^{(k)} + b$$

where:

$A = M - N$	is a splitting of A
$M = D$	Jacobi
$M = D - F$	forward Gauss-Seidel
$M = D - E$	backward Gauss-Seidel

Overrelaxation

Succesive overrelaxation (SOR):

$$\omega A = (D - \omega E) - (-\omega F + (1 - \omega)D)$$

A backward SOR step can be defined analogously to the backward Gauss-Seidel.

SOR method converges when ω is properly choosen.

Symmetric SOR

$$(D - \omega E)x^{(k+1/2)} = (-\omega F + (1 - \omega D)x^{(k)} + \omega b)$$

$$(D - \omega E)x^{(k+1)} = -\omega F + (1 - \omega D)x^{(k)} + (1 - \omega D)x^{(k+1/2)} + \omega b$$

Overview

- Introduction
- Stationary iterative methods
  - Jacobi iterative method
  - Gauss-Seidel iterative method
  - Overrelaxation iterative method
- **Krylov iterative methods**
  - Krylov subspace
  - Conjugate gradient

Krylov methods

The main advantage of the Krylov subspace method is the fact that the number of iteration to solve the linear equation is bounded by the dimension of the matrix A.

We denote the characteristic polynomial of A as $p(\lambda)$:

$$p(\lambda) = det(A - \lambda I)$$

Let

$$p(\lambda) = \sum_j p_j \lambda^j$$

Cayley-Hamilton's theorem tells us that $p(A) = 0$, :

$$p_n A^n + p_{n-1} A^{n-1} + ... + p_0 I = 0$$

As the matrix is non-singular $p_n \neq 0$ than

$$A^{-1} = \frac{1}{p_0}(p_1 I + p_2 A + ... + p_n A^{n-1})$$

Krylov's space

The solution x must be

$$x = A^{-1}b = \frac{1}{p_0}(p_1 I + p_2 A + ... + p_n A^{n-1})b$$

$$= \sum_{j=0}^{n-1} x_j A^j b$$

where $x_j = -\frac{p_{j+1}}{p_0}$

Krylov's space

Thus the solution is able to be expressed as a linear combination of vectors of the form $A^j b$, for $j = 0, ..., n-1$.

Let

$$K_r = span\{b, ..., A^r b\}$$

The above space is called the **Krylov's space**.

There are several **Krylov methods**. The most popular are:

- congugate gradient (CG)
- minimum residum (MINRES)

CG algorithm

We assume that A is symmetric and positive defined (SPD).

We will call two vectors x and y A-conjugate if

$$x^T Ay = 0$$

Theorem

Let

$$\phi(x) = \frac{1}{2}x^T Ax - x^T b$$

and let x^* be an exact solution of our set of equations.

The following conditions are equivalent:

1. $\min_{x \in R^n} \phi(x) = \phi(x^*)$
2. $Ax^* = b$

One approximation strategy is to try to find approximations from the subspaces $K_0.K_1, \ldots$, which successively minimizes $\phi(x)$. After n steps we must have the solution but of course we hope that when n is very large, we might obtain a good approximation after only a small number of steps.

The obvious way to do this would be at step r to try to find coefficients q_j such that if

$$x_r = \sum_{j=0}^{r} q_j A^j b$$

then $\phi(q_0, ..., q_r)$ is minimizes:

$$\phi(q_0, ..., q_r) = \frac{1}{2} x^T A x - x^T b$$

$$= \frac{1}{2} \sum_{j=0}^{r} \sum_{k=0}^{r} q_j q_k b^T A^{j+k+1} b - \sum_{j=0}^{r} q_j b^T A^j b$$

CG cont.

The idea of conjugate gradient is to find coefficients only once in an expansion of the solution in terms of a growing set of basis vectors. To do this we need a different basis for K_r.
Let $\{d_0, ..., d_r\}$ be the base of K_r which consists of A-conjugate vectors. Then:

$$x_r = \sum_{j=0}^{r} q_j d_j$$

and

$$\phi(q_0, ..., q_r) = \frac{1}{2} \sum_{j=0}^{r} q_j^2 d_j^T A d_j - q_j d_j^T b$$

CG cont.

Now the minimization problem is almost trivial, each component is independent of the others and at each step, the existing components do not change. So we only calculate the components in each direction once. Of course we do have to construct the basis but since it has to span and at the step we already have an A-conjugate basis for K_r , all we have to do is compute one additional basis vector at each step.

CG cont.

Theorem

Let x_{r-1} minimilizes ϕ w K_{r-1} and let $r_{r-1} = b - Ax_{r-1}$.
Then, if for the vector d_r we have $d_r^T Ay = 0$ for all $y \in K_{r-1}$, the vector

$$x_r = x_{r-1} + \alpha_r d_r$$

minimilizases ϕ w K_r, where α is calculated as follows:

$$\alpha_r = \frac{d_r^T b}{d_r^T A d_r}$$

CG cont.

If

$$\alpha_r = \frac{d_r^T b}{d_r^T A d_r}$$

then

$$x_r = x_{r-1} + \alpha_r d_r$$

minimalizases ϕ w K_r and

$$r_{r-1}^T d_s = r_{r-1}^T r_s = d_r^T A d_s = 0$$

for $s = 0, ..., r-1$.

CG cont.

We can show that:

$$\alpha_r = \frac{d_r^T r_{r-1}}{d_r^T A d_r} = \frac{r_{r-1}^T r_{r-1}}{d_r^T A d_r}$$

$$\beta_r = -\frac{r_{r-1}^T A d_{r-1}}{d_{r-1}^T A d_{r-1}} = -\frac{r_{r-1}^T r_{r-1}}{d_{r-1}^T A d_{r-1}}$$

CG cont.

Only four vectors should be remembered:

$$x_{r-1}, d_{r-1}, r_{r-1}, Ad_{r-1}$$

It is worth observing that we only ever need to store four vectors, each of which can be replaced as the iteration proceeds and other than the vector, there are only inner products to evaluate. The algorithm is very compact.

Convergence

$$\kappa_{CG}(A) = \frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} < 1$$

Theoretically CG converges always!!!

CG algorithm

1. Compute $r_0 := b - Ax_0$, $d_0 := r_0$
2. For j=0,1,2,... until convergence , Do
3. $\alpha_j := (r_j, r_j)/(Ad_j, d_j)$
4. $x_{j+1} := x_j + \alpha_j d_j$
5. $r_{j+1} := r_j - \alpha_j A d_j$
6. $\beta_j := (r_{j+1}, r_{j+1})/(r_j, r_j)$
7. $d_{j+1} := r_{j+1} + \beta_j d_j$
8. EndDo

CG implementation

1. Compute $r_0 := b - Ax_0$, $d_0 := r_0$
   ; vector-matrix, vector-vector
2. For j=0,1,2,... until convergence , Do
3. $\alpha_j := (r_j, r_j)/(Ad_j, d_j)$; vector-vector, vector-matrix
4. $x_{j+1} := x_j + \alpha_j d_j$; vector-vector, scalar-vector
5. $r_{j+1} := r_j - \alpha_j A d_j$; vector-vector, vector-matrix
6. $\beta_j := (r_{j+1}, r_{j+1})/(r_j, r_j)$; vector-vector
7. $d_{j+1} := r_{j+1} + \beta_j d_j$; vector-vector, scalar-vector
8. EndDo

CG Example

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \cdot x = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

The initial aproximations are:

$$x_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad r_0 = b - A \cdot x_0 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad d_0 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\alpha_0 := (r_0, r_0)/(Ad_0, d_0) = (\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix})/(\begin{bmatrix} 2 \\ -2 \\ 2 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}) = 0,5$$

CG Example cont

$$x_1 := x_0 + \alpha_0 d_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + 0,5 \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0,5 \\ 0 \\ 0,5 \end{bmatrix}$$

$$x_1 = \begin{bmatrix} 0,5 \\ 0 \\ 0,5 \end{bmatrix} \quad r_1 = r_0 - \alpha_0 A d_0 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\beta_0 := (r_0, r_0)/(r_1, r_1) = (\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix})/(\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}) = 0,5$$

CG Example cont

$$d_1 = r_1 + \beta \cdot d_0 = \begin{bmatrix} 0{,}5 \\ 1 \\ 0{,}5 \end{bmatrix}$$

$$\alpha_1 := (r_1, r_1)/(Ad_1, d_1) = (\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix})/(\begin{bmatrix} 2 \\ -2 \\ 2 \end{bmatrix} \cdot \begin{bmatrix} 0{,}5 \\ 1 \\ 0{,}5 \end{bmatrix}) = 1$$

CG Example cont

The solution is obtained in the second step of iteration:

$$x_2 := x_1 + \alpha_1 d_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$r_2 = r_1 - \alpha_1 A d_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Implementation

The parallel implementations of the iterative methods base on the parallel vector-vector and vector-matrix operations.

Thank you for your attention!

Any questions are welcome.