LIÈGE université
Sciences Appliquées

Part II

Linear Algebra Complements

LIÈGE université
Sciences Appliquées

Lecture 10 Conjugate Gradients

Mathématiques appliquées (MATH0504-1)
B. Dewals, C. Geuzaine

Learning objectives of this lecture

Learn about iterative methods for solving systems of linear equations

Understand the conjugate gradients iteration

Understand the principle of preconditioning

Outline

1. Overview of iterative methods
2. Conjugate Gradients
3. Preconditioning

LIÈGE université
Sciences Appliquées

1 – Overview of iterative methods

(Lecture 32 in Trefethen & Bau, 1997)

Why iterate?

We wish to solve a nonsingular system of linear equations

$$Ax = b$$

with $A \in \mathbb{C}^{m \times m}$.

For general (dense) matrices A, direct methods (Gaussian elimination, LU factorization) take $\mathcal{O}(m^3)$ operations – which rapidly becomes intractable for large m.

Sparsity and black boxes

Can this bottleneck be beaten?

For sparse matrices such as those resulting from PDE approximations, direct methods suffer from fill-in and iterative methods provide a path forward.

Iterative methods require nothing more than the ability to determine Ax for any x, which on a computer can be done by a “black box”.
For a sparse matrix it is e.g. easy to design a procedure that will compute Ax in $\mathcal{O}(vm)$ instead of $\mathcal{O}(m^2)$ operations.

Projection into Krylov subspaces

Compared to the stationary iterative methods that we recalled last week, modern iterative methods are based on the idea of projecting an *m*-dimensional problem into a lower-dimensional Krylov subspace, spanned by the vectors b, Ab, A^2b, A^3b, ...

These vectors can be computed by the "black box" in the form b, Ab, $A(Ab)$, $A(A(Ab))$, ...

Why use Krylov subspaces?
Recall the Cayley-Hamilton theorem

Let $A \in \mathbb{C}^{m \times m}$ with characteristic polynomial

$$p(\lambda) = \det(\lambda I - A)$$
$$= \lambda^m + c_{m-1}\lambda^{m-1} + \cdots + c_1\lambda + c_0$$

Then

$$p(A) = A^m + c_{m-1}A^{m-1} + \cdots + c_1 A + c_0 I = 0$$

Why use Krylov subspaces?
Recall the Cayley-Hamilton theorem

Multiplying $p(A) = 0$ by x , we get:

$$A^m x + c_{m-1} A^{m-1} x + \cdots + c_1 A x + c_0 x = 0$$

If $Ax = b,$ this entails

$$A^{m-1} b + c_{m-1} A^{m-2} b + \cdots + c_1 b + c_0 x = 0$$

$$\Rightarrow x = -c_0^{-1} (c_1 b + c_2 A b + \cdots + c_{m-1} A^{m-2} b + A^{m-1} b)$$

and thus

$$x \in \langle b, Ab, \cdots, A^{m-2} b, A^{m-1} b \rangle$$

Projection into Krylov subspaces

There exist many Krylov subspace methods, for either solving linear systems or eigenvalue problems

	$Ax = b$	$Ax = \lambda x$
$A = A^*$	CG	Lanczos
$A \neq A^*$	GMRES, CGN, BCG, …	Arnoldi

LIÈGE université
Sciences Appliquées

2 – Conjugate Gradients

(Lecture 38 in Trefethen & Bau, 1997)

Conjugate Gradients

Conjugate Gradients (CG) is the “original” Krylov subspace method, discovered by Hestenes and Stiefel in 1952.

- It is probably the most famous Krylov subspace method, and one of the mainstays of scientific computing.
- Other Krylov subspace methods include: Arnoldi, Lanczos, GMRES, BiCGSTAB, QMR, MINRES…

CG solves symmetric definite systems amazingly quickly if the eigenvalues are well distributed

Some definitions

Let $A \in \mathbb{R}^{m \times m}$ be a real symmetric positive definite (SPD) matrix.

We wish to solve a nonsingular system of equations

$$Ax = b$$

with exact solution $x_* = A^{-1}b$.

Let $\mathscr{K}_n = \mathscr{K}_n(A, b)$ denote the n^{th} Krylov subspace generated by b, defined as

$$\mathscr{K}_n = \langle b, Ab, \ldots, A^{n-1}b \rangle$$

$\mathscr{K}_n$ is thus spanned by the images of b under the first n powers of A, starting with $n = 0$.

Minimizing the A-norm of the error

Since A is SPD, all its eigenvalues are positive, or equivalently $x^T Ax > 0$ for every nonzero $x \in \mathbb{R}^m$.

Then the function $||.||_A$ defined by

$$||x||_A = \sqrt{x^T Ax}$$

is a norm in $\mathbb{R}^m$.

Indeed: $||x||_A \geq 0,$ and $||x||_A = 0$ only if $x = 0$

$$||x+y||_A \leq ||x||_A + ||y||_A$$

$$||\alpha x||_A = |\alpha| ||x||_A$$

The norm $||.||_A$ is called the "A-norm".

Minimizing the A-norm of the error

What is the conjugate gradient iteration?

It is a system of recurrence formulas that generates the unique sequence of iterates $x_n \in \mathscr{K}_n$ with the property that at each step n the A-norm $||e_n||_A$ of the error

$$e_n = x_* - x_n$$

is minimized.

The Conjugate Gradient iteration

The CG iteration is the following:

$x_0 = 0, r_0 = b, p_0 = r_0$
for $n = 1, 2, 3, \ldots$ **do**

$\alpha_n = \frac{r_{n-1}^T r_{n-1}}{p_{n-1}^T A p_{n-1}}$	step length
$x_n = x_{n-1} + \alpha_n p_{n-1}$	approximate solution
$r_n = r_{n-1} - \alpha_n A p_{n-1}$	residual
$\beta_n = \frac{r_n^T r_n}{r_{n-1}^T r_{n-1}}$	
$p_n = r_n + \beta_n p_{n-1}$	search direction

end

We leave aside for now how and when to stop.

The Conjugate Gradient iteration

- The CG iteration is very simple: you can program it in a few lines of Matlab or Python
- It involves several vector manipulations and one matrix-vector product ($Ap_{n\text{-}1}$)
- If A is dense and unstructured, this matrix-vector product dominates the operation count, which is $\mathcal{O}(m^2)$ per step
- If A is sparse, the operation count may be as low as $\mathcal{O}(m)$ per step

The Conjugate Gradient iteration

Let's first explore three properties of the CG iteration:

1. Identity of subspaces

$$\mathscr{K}_n = \langle x_1, x_2, \ldots, x_n \rangle = \langle p_0, p_1, \ldots, p_{n-1} \rangle$$
$$= \langle r_0, r_1, \ldots, r_{n-1} \rangle = \langle b, Ab, \ldots, A^{n-1}b \rangle$$

2. Orthogonality of residuals

$$r_n^T r_j = 0 \quad (j < n)$$

3. A-conjugateness of search directions

$$p_n^T A p_j = 0 \quad (j < n)$$

First property

Prop 1: $\mathcal{K}_n = \langle x_1, x_2, \ldots, x_n \rangle = \langle p_0, p_1, \ldots, p_{n-1} \rangle$
$= \langle r_0, r_1, \ldots, r_{n-1} \rangle = \langle b, Ab, \ldots, A^{n-1}b \rangle$

By induction on n:

- $x_0 = 0$ and $x_n = x_{n-1} + \alpha_n p_{n-1}$
$\Rightarrow x_n \in \langle p_0, p_1, \ldots, p_{n-1} \rangle$

- $p_0 = r_0$ and $p_n = r_n + \beta_n p_{n-1}$
$\Rightarrow \langle p_0, p_1, \ldots, p_{n-1} \rangle = \langle r_0, r_1, \ldots, r_{n-1} \rangle$

- $r_0 = b$ and $r_n = r_{n-1} - \alpha_n A p_{n-1}$
$\Rightarrow \langle r_0, r_1, \ldots, r_{n-1} \rangle = \langle b, Ab, \ldots, A^{n-1}b \rangle$

Prop 2: $r_n^T r_j = 0 \quad (j < n)$

Prop 3: $p_n^T A p_j = 0 \quad (j < n)$

Second property

$$r_n = r_{n-1} - \alpha_n A p_{n-1} \Rightarrow r_n^T r_j = r_{n-1}^T r_j - \alpha_n (A p_{n-1})^T r_j$$
$$= r_{n-1}^T r_j - \alpha_n p_{n-1}^T A r_j$$

① If $j < n-1$ both terms on the right are zero by induction:

- Induction hypotheses:

$$r_{n-1}^T r_j = 0 \;\; \& \;\; p_{n-1}^T A p_j = 0 \text{ for } j < n-1$$

- First term: direct by first hypothesis
- Second term: substitute $r_j = p_j - \beta_j p_{j-1}$, then use second hypothesis twice

Prop 2: $r_n^T r_j = 0 \quad (j < n)$

Prop 3: $p_n^T A p_j = 0 \quad (j < n)$

Second property

$$r_n = r_{n-1} - \alpha_n A p_{n-1} \Rightarrow r_n^T r_j = r_{n-1}^T r_j - \alpha_n (A p_{n-1})^T r_j$$
$$= r_{n-1}^T r_j - \alpha_n p_{n-1}^T A r_j$$

② If $j = n-1$ the difference on the right is zero if

$$\alpha_n = \frac{r_{n-1}^T r_{n-1}}{p_{n-1}^T A r_{n-1}}$$

By the induction hypothesis we have $p_{n-1}^T A p_{n-2} = 0$ and thus:

$$\alpha_n = \frac{r_{n-1}^T r_{n-1}}{p_{n-1}^T A r_{n-1} + \beta_{n-1} p_{n-1}^T A p_{n-2}} = \frac{r_{n-1}^T r_{n-1}}{p_{n-1}^T A (r_{n-1} + \beta_{n-1} p_{n-2})}$$
$$= \frac{r_{n-1}^T r_{n-1}}{p_{n-1}^T A p_{n-1}}$$

Prop 2: $r_n^T r_j = 0 \quad (j < n)$

Prop 3: $p_n^T A p_j = 0 \quad (j < n)$

Third property

$$p_n = r_n + \beta_n p_{n-1} \Rightarrow p_n^T A p_j = r_n^T A p_j + \beta_n p_{n-1}^T A p_j$$

① If $j < n-1$ both terms on the right are again zero by induction:

- Induction hypotheses:

$$r_{n-1}^T r_j = 0 \ \ \& \ \ p_{n-1}^T A p_j = 0 \text{ for } j < n-1$$

- Second term: direct by second hypothesis
- First term: $r_j = r_{j-1} - \alpha_j A p_{j-1}$, i.e. $r_{j+1} = r_j - \alpha_{j+1} A p_j$
  Hence

$$r_n^T A p_j = \frac{1}{\alpha_{j+1}} r_n^T (r_j - r_{j+1})$$

  which is zero by prop 2

Prop 2: $r_n^T r_j = 0 \quad (j < n)$

Prop 3: $p_n^T A p_j = 0 \quad (j < n)$

Third property

$$p_n = r_n + \beta_n p_{n-1} \Rightarrow p_n^T A p_j = r_n^T A p_j + \beta_n p_{n-1}^T A p_j$$

② If $j = n-1$ the sum on the right is zero provided that

$$\beta_n = -\frac{r_n^T A p_{n-1}}{p_{n-1}^T A p_{n-1}} = -\frac{\alpha_n r_n^T A p_{n-1}}{\alpha_n p_{n-1}^T A p_{n-1}} = \frac{r_n^T(-\alpha_n A p_{n-1})}{p_{n-1}^T(\alpha_n A p_{n-1})}$$

This is the same as the β_n in the CG iteration, since

- $r_n^T(-\alpha_n A p_{n-1}) = r_n^T(r_n - r_{n-1}) = r_n^T r_n$ by prop 2
- $p_{n-1}^T(\alpha_n A p_{n-1}) = (r_{n-1}^T + \beta_{n-1} p_{n-2}^T)\alpha_n A p_{n-1}$

$$= r_{n-1}^T \alpha_n A p_{n-1} \quad \text{by induction}$$

$$= r_{n-1}^T(r_{n-1} - r_n)$$

$$= r_{n-1}^T r_{n-1} \quad \text{by prop 2}$$

Optimality of CG: theorem

Let the CG iteration be applied to a symmetric positive definite matrix problem $Ax = b$. If the iteration has not already converged ($r_{n-1} \neq 0$), then x_n is the unique point in $\mathscr{K}_n$ that minimizes $||e_n||_A$.

The convergence is monotonic

$$||e_n||_A \leq ||e_{n-1}||_A$$

and $e_n = 0$ is achieved for some $n \leq m$.

Optimality of CG: proof

From the first property we know that $x_n \in \mathscr{K}_n$.

To show that it is the only point in $\mathscr{K}_n$ that minimizes $||e_n||_A$, consider an arbitrary point

$$x = x_n - \Delta x \in \mathscr{K}_n$$

with error $e = x_* - x = e_n + \Delta x$. We calculate

$$\begin{aligned}
||e||_A^2 &= (e_n + \Delta x)^T A (e_n + \Delta x) \\
&= e_n^T A e_n + e_n^T A(\Delta x) + (\Delta x)^T A e_n + (\Delta x)^T A(\Delta x) \\
&= e_n^T A e_n + e_n^T A^T (\Delta x) + (\Delta x)^T A e_n + (\Delta x)^T A(\Delta x) \\
&= e_n^T A e_n + (A e_n)^T (\Delta x) + (\Delta x)^T A e_n + (\Delta x)^T A(\Delta x) \\
&= e_n^T A e_n + (\Delta x)^T A(\Delta x) + 2(\Delta x)^T A e_n
\end{aligned}$$

Optimality of CG: proof

The last term $2(\Delta x)^T A e_n = 2(\Delta x)^T A(x_* - x_n)$

$$= 2(\Delta x)^T (b - A x_n)$$

$$= 2(\Delta x)^T r_n$$

is an inner product of r_n with a vector in $\mathscr{K}_n$ (since $(\Delta x) = (x - x_n) \in \mathscr{K}^n$, as both x and $x_n \in \mathscr{K}_n$), which by the second property is 0.

Thus:

$$||e||_A^2 = e_n^T A e_n + (\Delta x)^T A (\Delta x)$$

Optimality of CG: proof

Only the second term in $||e||_A^2 = e_n^T A e_n + (\Delta x)^T A (\Delta x)$ depends on Δx, and since A is positive definite, it is ≥ 0, attaining the value 0 if and only if $x_n = x$, i.e. when $\Delta x = 0$. Thus $||e_n||_A$ is minimal if and only if $x_n = x$, as claimed.

The monotonicity is a consequence of the inclusion of Krylov subspaces:

$e_n = x_* - x_n$ is minimal for all x_n in $\mathcal{K}_n$

$e_{n+1} = x_* - x_{n+1}$ is minimal for all x_{n+1} in $\mathcal{K}_{n+1}$

Since $\mathcal{K}_n \subset \mathcal{K}_{n+1}$, $||e_{n+1}||$ cannot be $> ||e_n||$.

Optimality of CG: proof

Finally, since $\mathscr{K}_n$ is a subset of $\mathbb{R}^m$ of dimension n as long as convergence has not been achieved, convergence must be achieved in at most m steps.

Note that the guarantee that CG converges in at most m steps is void in floating point arithmetic.

In practice, when CG is applied to matrices whose spectra (perhaps thanks to preconditioning) are well-enough behaved, convergence to a desired accuracy is achieved for $n \ll m$.

CG as an optimization algorithm

The CG iteration can be interpreted as an algorithm for minimizing a nonlinear function of $x \in \mathbb{R}^m$.

At the heart of the algorithm is

$$x_n = x_{n-1} + \alpha_n p_{n-1}$$

where a current approximation $x_{n\text{-}1}$ is updated to a new approximation x_n by moving a distance α_n (the step length) in the direction $p_{n\text{-}1}$.

Which function?

CG as an optimization algorithm

Let's examine the function

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

A short computation reveals that

$$\begin{aligned} ||e_n||_A^2 &= e_n^T A e_n \\ &= (x_* - x_n)^T A (x_* - x_n) \\ &= x_n^T A x_n - 2x_n^T A x_* + x_*^T A x_* \\ &= x_n^T A x_n - 2x_n^T b + x_*^T b \\ &= 2\varphi(x_n) + \text{constant} \end{aligned}$$

Thus $\varphi(x)$ is the same as $||e_n||_A^2$ except for a factor of 2 and the (unknown) constant $x_*^T b$.

CG as an optimization algorithm

Like $||e_n||_A^2$, $\varphi(x)$ must achieve its minimum (namely the value $-x_*^T b/2$) uniquely at $x = x_*$.

The CG iteration can thus be interpreted as an iterative process for minimizing the quadratic function $\varphi(x)$ of $x \in \mathbb{R}^m$. At each step, an iterate

$$x_n = x_{n-1} + \alpha_n p_{n-1}$$

is computed that minimizes $\varphi(x)$ over all x in the one-dimensional space $x_{n-1} + \langle p_{n-1} \rangle$.

The formula $\alpha_n = \dfrac{r_{n-1}^T r_{n-1}}{p_{n-1}^T A p_{n-1}}$ provides the optimal step length.

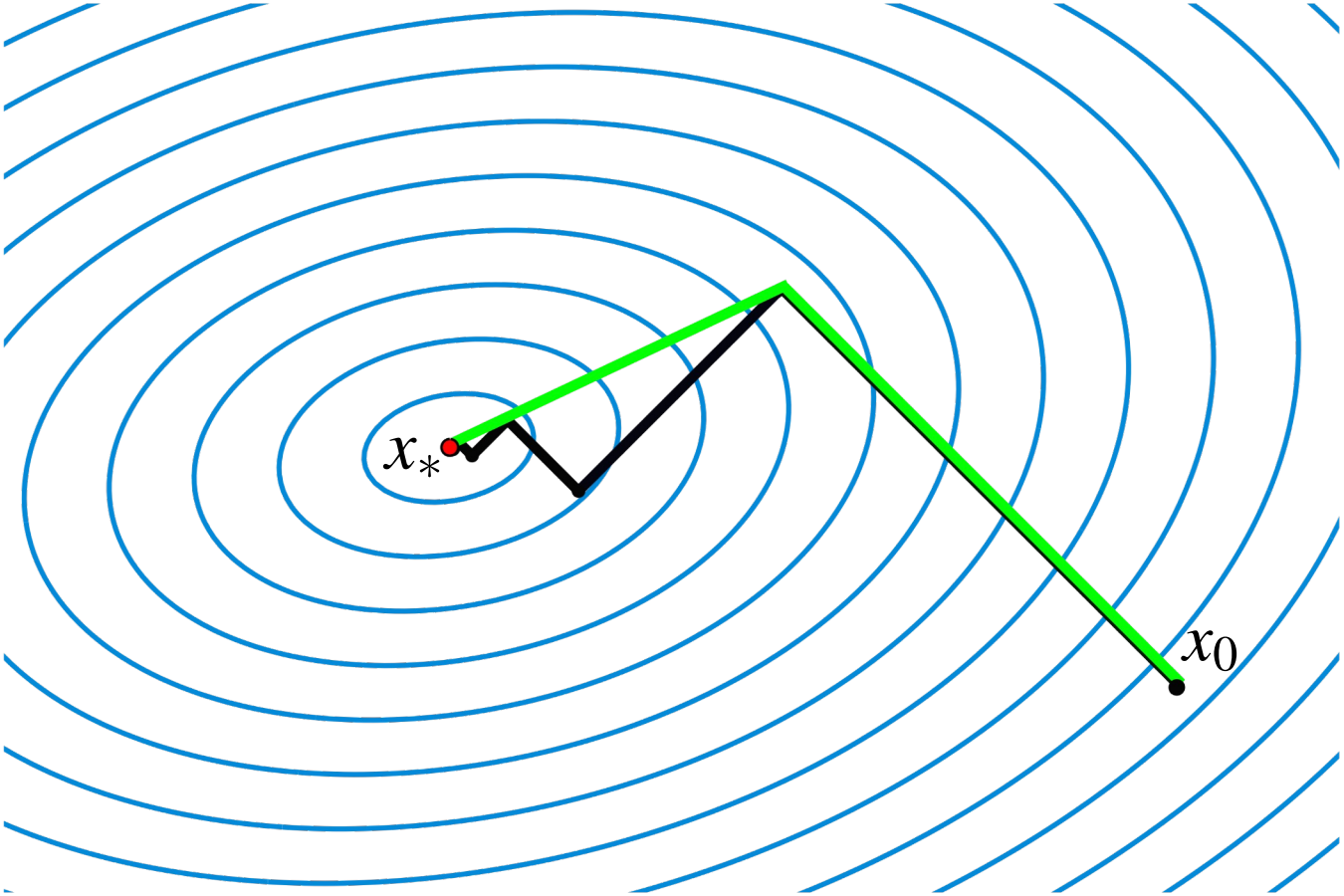
CG as an optimization algorithm

What makes the CG iteration remarkable is the choice of the search directions $p_{n\text{-}1}$, which has the special property that minimizing $\varphi(x)$ over the

$$x_{n-1} + \langle p_{n-1} \rangle$$

actually minimizes it over all of $\mathscr{K}_n$!

CG as an optimization algorithm



CG (green) vs. Steepest Descent (black) in $\mathbb{R}^2$

Rate of convergence

Two results, without proof:

1. If A has only n distinct eigenvalues, the CG iteration converges in at most n steps
2. If A has a condition number κ, then

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

Since $\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \sim 1-\frac{2}{\sqrt{\kappa}}$ for $\kappa \to \infty$, it implies that for κ large but not too large, convergence to a specified tolerance can be expected in $\mathcal{O}(\sqrt{\kappa})$ iterations.

LIÈGE université
Sciences Appliquées

3 – Preconditioning

(Lecture 40 in Trefethen & Bau, 1997)

Preconditioning

For any nonsingular matrix $M \in \mathbb{R}^{m \times m}$, the system

$$M^{-1}Ax = M^{-1}b$$

has the same solution as the system $Ax = b$.

The convergence of iterative methods will however now depend on the spectral properties of $M^{-1}A$ instead of A.

So if this *preconditioner* M is chosen wisely, the preconditioned system can be solved much faster than the original system.

Preconditioning

In practice one of course needs to find a matrix M such that it must be possible to compute the operations represented by the product $M^{-1}A$ efficiently.

As usual, this will not mean an explicit construction of the inverse M^{-1}, but the solution of systems of the form $My = c$.

Finding good preconditioners (e.g. M “close to” A) is an active research area, in particular linked to the approximation of various PDEs.

Preconditioning

A non-exhaustive list of some types of preconditioners includes

- Diagonal scaling (Jacobi)
- Incomplete LU or Cholesky factorization
- Coarse grid approximations, multigrid iteration, low-order discretization
- Block preconditioners and domain decomposition
- …

Next week

- Singular value decomposition