Numerical Solutions to PDEs

Lecture Notes #23

Elliptic Equations — Steepest Descent and Conjugate Gradient

Peter Blomgren,
⟨blomgren.peter@gmail.com⟩

Department of Mathematics and Statistics
Dynamical Systems Group
Computational Sciences Research Center
San Diego State University
San Diego, CA 92182-7720

http://terminus.sdsu.edu/

Spring 2018

Outline

San Diego State University

Last Time: Linear Iterative Schemes

We looked at the Jacobi, Gauss-Seidel, SOR, and SSOR iterations applied to linear systems $A\bar{\mathbf{x}} = \bar{\mathbf{b}}$, originating from the 5-point Laplacian.

We quantified under what circumstances we can guarantee convergence of these iterations (J&GS: irreducibly diagonally dominant matrices, (S)SOR: $\omega \in (0, 2)$), and discussed the convergence rates.

The discussion was extended to general linear systems, where A may be associated with the 9-point Laplacian, or something completely different. In this discussion we introduced **preconditioning**, where we find a matrix $M \approx A$, which is much easier to invert than A itself, and we leverage this in order to generate an efficient iterative solver.

Another Point of View: Optimization

We consider a system of linear equations $A\bar{\mathbf{x}} = \bar{\mathbf{b}}$, where A is symmetric positive definite.

We define

$$F(\bar{\mathbf{y}}) = \frac{1}{2}(\bar{\mathbf{y}} - \bar{\mathbf{x}})^T A(\bar{\mathbf{y}} - \bar{\mathbf{x}}),$$

and note that since A is positive definite $F(\bar{\mathbf{y}}) \geq 0$, and $F(\bar{\mathbf{y}}) = 0 \Leftrightarrow \bar{\mathbf{y}} = \bar{\mathbf{x}}$. Further, we can define

$$E(\bar{\mathbf{y}}) = F(\bar{\mathbf{y}}) - F(\bar{\mathbf{0}}) = \frac{1}{2}\bar{\mathbf{y}}^T A\bar{\mathbf{y}} - \bar{\mathbf{y}}^T\bar{\mathbf{b}},$$

which has a unique minimum at the solution of $A\bar{\mathbf{x}} = \bar{\mathbf{b}}$.

Now the gradient of $E(\bar{\mathbf{y}})$ describes the direction of largest increase

$$G(\bar{\mathbf{y}}) = \nabla E(\bar{\mathbf{y}}) = A\bar{\mathbf{y}} - \bar{\mathbf{b}} = -\underbrace{\bar{\mathbf{r}}(\bar{\mathbf{y}})}_{\text{residual}}.$$

San Diego State University

Optimization ⇝ Steepest Descent

Since the gradient points in the direction of steepest ascent, the residual points in the direction of steepest descent.

Given an approximation (guess) $\bar{\mathbf{x}}^k$ to the solution of $A\bar{\mathbf{x}} = \bar{\mathbf{b}}$, we find a better approximation by searching in the steepest descent direction

$$\bar{\mathbf{x}}^{k+1} = \bar{\mathbf{x}}^k + \alpha_k \bar{\mathbf{r}}^k, \quad \text{where } \bar{\mathbf{r}}^k = \bar{\mathbf{b}} - A\bar{\mathbf{x}}^k,$$

and we select α_k so that $E(\bar{\mathbf{x}}^k + \alpha_k \bar{\mathbf{r}}^k)$ is minimized:

$$\begin{aligned} E(\bar{\mathbf{x}}^{k+1}) &= E(\bar{\mathbf{x}}^k + \alpha_k \bar{\mathbf{r}}^k) \\ &= \frac{1}{2}[\bar{\mathbf{x}}^k]^T A\bar{\mathbf{x}}^k + \alpha_k [\bar{\mathbf{r}}^k]^T A\bar{\mathbf{x}}^k + \frac{1}{2}\alpha_k^2 [\bar{\mathbf{r}}^k]^T A\bar{\mathbf{r}}^k - [\bar{\mathbf{x}}^k]^T \bar{\mathbf{b}} - \alpha_k [\bar{\mathbf{r}}^k]^T \bar{\mathbf{b}} \\ &= E(\bar{\mathbf{x}}^k) - \alpha_k [\bar{\mathbf{r}}^k]^T \bar{\mathbf{r}}^k + \frac{1}{2}\alpha_k^2 [\bar{\mathbf{r}}^k]^T A\bar{\mathbf{r}}^k. \end{aligned}$$

Setting $\partial E(\bar{\mathbf{x}}^k + \alpha_k \bar{\mathbf{r}}^k)/\partial \alpha_k = 0$ gives us

$$\alpha_k = \frac{[\bar{\mathbf{r}}^k]^T \bar{\mathbf{r}}^k}{[\bar{\mathbf{r}}^k]^T A\bar{\mathbf{r}}^k} = \frac{\|\bar{\mathbf{r}}^k\|_2^2}{[\bar{\mathbf{r}}^k]^T A\bar{\mathbf{r}}^k}.$$

San Diego State University

Steepest Descent

The steepest descent algorithm is given by $\bar{\mathbf{x}}^0 = \bar{\mathbf{0}}$, $\bar{\mathbf{r}}^0 = \bar{\mathbf{b}}$:

$$\begin{aligned}
\alpha_k &= \frac{\|\bar{\mathbf{r}}^k\|^2}{[\bar{\mathbf{r}}^k]^T \mathbf{A}\bar{\mathbf{r}}^k} \\
\bar{\mathbf{x}}^{k+1} &= \bar{\mathbf{x}}^k + \alpha_k \bar{\mathbf{r}}^k \\
\bar{\mathbf{r}}^{k+1} &= \bar{\mathbf{r}}^k - \alpha_k \mathbf{A}\bar{\mathbf{r}}^k.
\end{aligned}$$

Where the update formula for the residual comes from

$$\begin{aligned}
\bar{\mathbf{x}}^{k+1} &= \bar{\mathbf{x}}^k + \alpha_k \bar{\mathbf{r}}^k \\
A\bar{\mathbf{x}}^{k+1} &= A\bar{\mathbf{x}}^k + \alpha_k A\bar{\mathbf{r}}^k \\
\bar{\mathbf{b}} - A\bar{\mathbf{x}}^{k+1} &= \bar{\mathbf{b}} - A\bar{\mathbf{x}}^k - \alpha_k A\bar{\mathbf{r}}^k \\
\bar{\mathbf{r}}^{k+1} &= \bar{\mathbf{r}}^k - \alpha_k A\bar{\mathbf{r}}^k.
\end{aligned}$$

San Diego State University

Steepest Descent

We note that the steepest descent algorithm only requires one matrix-vector product $A\bar{\mathbf{r}}^k$ and two vector-vector inner products $(\|\bar{\mathbf{r}}^k\|^2,\ [\bar{\mathbf{r}}^k]^T \mathbf{A\bar{r}^k})$ per iteration.

When A is sparse the matrix-vector product can be implemented in $\mathcal{O}(N)$ operations.

Theorem

If A is a positive definite matrix for which $A^T A^{-1}$ is also positive definite, then the steepest descent algorithm converges to the unique solution $\bar{\mathbf{x}}^* = A^{-1}\bar{\mathbf{b}}$ for any initial $\bar{\mathbf{x}}^0$.

Theorem

If A is SPD, then the steepest descent algorithm converges to the unique solution $\bar{\mathbf{x}}^* = A^{-1}\bar{\mathbf{b}}$ for any initial $\bar{\mathbf{x}}^0$.

Steepest Descent

It turns out, maybe somewhat counter-intuitively, that the steepest descent algorithm converges very slowly unless A is a (near-)multiple of the identity matrix.

The residuals tend to oscillate so that $\bar{\mathbf{r}}^{k+2}$ points in the same direction as $\bar{\mathbf{r}}^{k}$, and very little progress is made.

Next we quantify this convergence rate, and discuss the **conjugate gradient method** which is an "accelerated version of steepest descent."

Convergence Rate for Steepest Descent

Theorem (Convergence Rate for Steepest Descent)

If A is a symmetric positive definite matrix whose eigenvalues lie in the interval $[a, b]$, then the error vector $\bar{\mathbf{e}}^k$ for the steepest descent method satisfies

$$[\bar{\mathbf{e}}^k]^T A \bar{\mathbf{e}}^k \leq \left[\frac{b-a}{b+a}\right]^{2k} [\bar{\mathbf{e}}^0]^T A \bar{\mathbf{e}}^0 \equiv \left[\frac{\kappa-1}{\kappa+1}\right]^{2k} [\bar{\mathbf{e}}^0]^T A \bar{\mathbf{e}}^0$$

The larger the interval $[a, b]$, *i.e.* the more ill-conditioned A is, the slower the convergence rate we get.

The **condition number** κ of a matrix is defined as

$$\kappa = \frac{b}{a} = \frac{|\lambda|_{\text{max}}}{|\lambda|_{\text{min}}},$$

it is an intrinsic measure of difficult the matrix is to invert.

The Steepest Descent Method: Zig-Zagging

The "zig-zagging" $(\bar{\mathbf{p}}^{k+2} \approx \bar{\mathbf{p}}^{k})$ is what causes the steepest descent method to slow down. The amount of zig-zagging is directly proportional to the ratio $|\lambda|_{\text{max}}/|\lambda|_{\text{min}}$, or more generally for a non-square matrix A, $\sigma_{\text{max}}/\sigma_{\text{min}}$, where σ_ν are the singular values of A.

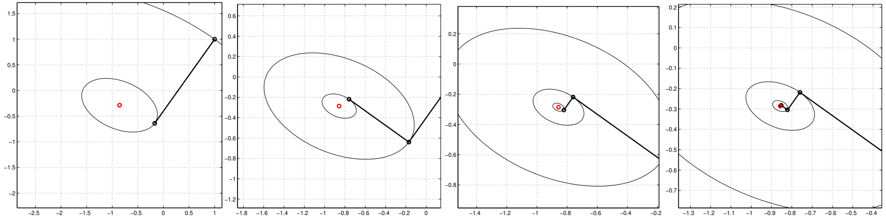

Figure: Illustration of the "zig-zagging" of the search directions in the steepest descent algorithm. If $\kappa = 1$, then all the level curves of $\|A\bar{\mathbf{x}} - \bar{\mathbf{b}}\| = c$ are circles (hyper-spheres in $\mathbb{R}^n$) and the steepest descent direction points straight in toward the central point. The more elongated the ellipse becomes, the more zig-zagging we get...

The Conjugate Gradient Method

The Conjugate Gradient method can be viewed as an acceleration of the steepest descent method, in which we by adding a little bit of "memory" to the algorithm can avoid the zig-zagging.

We consider

$$\bar{\mathbf{x}}^{k+1} = \bar{\mathbf{x}}^k + \alpha_k \underbrace{\left[\bar{\mathbf{r}}^k + \gamma_k \underbrace{\left(\bar{\mathbf{x}}^k - \bar{\mathbf{x}}^{k-1}\right)}_{\alpha_{k-1}\bar{\mathbf{p}}^{k-1}}\right]}_{\bar{\mathbf{p}}^k},$$

clearly, if $\gamma_k \equiv 0$, we can recover the steepest descent algorithm.

We form the new search direction $\bar{\mathbf{p}}^k$ as a linear combination of the steepest descent direction $\bar{\mathbf{r}}^k$ and the previous search direction $\bar{\mathbf{p}}^{k-1}$, *i.e*

$$\bar{\mathbf{p}}^k = \bar{\mathbf{r}}^k + \beta_{k-1}\bar{\mathbf{p}}^{k-1}.$$

San Diego State University

The Conjugate Gradient Method

The conjugate gradient iteration involves updates for the approximate solution $\bar{\mathbf{x}}$, the residual $\bar{\mathbf{r}}$, and the search direction $\bar{\mathbf{p}}$:

$$\begin{aligned} \bar{\mathbf{x}}^{k+1} &= \bar{\mathbf{x}}^{k} + \alpha_k \bar{\mathbf{p}}^{k}, \\ \bar{\mathbf{r}}^{k+1} &= \bar{\mathbf{r}}^{k} - \alpha_k A \bar{\mathbf{p}}^{k}, \\ \bar{\mathbf{p}}^{k+1} &= \bar{\mathbf{r}}^{k+1} + \beta_k \bar{\mathbf{p}}^{k}. \end{aligned}$$

Where we want to select α_k and β_k in an optimal way. A minimization of the error $E(\bar{\mathbf{x}}^{k+1})$ with respect to α (just as in the steepest descent case), and a similar analysis of $E(\bar{\mathbf{x}}^{k+1})$ with respect to β gives

$$\alpha_k = \frac{\|\bar{\mathbf{r}}^k\|_2^2}{[\bar{\mathbf{p}}^k]^T A \bar{\mathbf{p}}^k}, \quad \beta_k = -\frac{[\bar{\mathbf{r}}^{k+1}]^T A \bar{\mathbf{p}}^k}{[\bar{\mathbf{p}}^k]^T A \bar{\mathbf{p}}^k} \equiv \frac{\|\bar{\mathbf{r}}^{k+1}\|_2^2}{\|\bar{\mathbf{r}}^k\|_2^2}.$$

The Conjugate Gradient Method

Algorithm: The Conjugate Gradient Method

```
$\bar{\mathbf{p}}^0 = \bar{\mathbf{r}}^0 = \bar{\mathbf{b}} - A\bar{\mathbf{x}}^0,\ k = 0$
while ( $\|\bar{\mathbf{r}}^k\| > \epsilon_{\text{tol}}\|\bar{\mathbf{r}}^0\|$ )
    $\alpha_k = \frac{\|\bar{\mathbf{r}}^k\|_2^2}{[\bar{\mathbf{p}}^k]^T A\bar{\mathbf{p}}^k}$
    $\bar{\mathbf{x}}^{k+1} = \bar{\mathbf{x}}^k + \alpha_k\bar{\mathbf{p}}^k$
    $\bar{\mathbf{r}}^{k+1} = \bar{\mathbf{r}}^k - \alpha_k A\bar{\mathbf{p}}^k$
    $\beta_k = \frac{\|\bar{\mathbf{r}}^{k+1}\|_2^2}{\|\bar{\mathbf{r}}^k\|_2^2}$
    $\bar{\mathbf{p}}^{k+1} = \bar{\mathbf{r}}^{k+1} + \beta_k\bar{\mathbf{p}}^k$
endwhile ( $k := k + 1$ )
```

The Conjugate Gradient Method

The CG method only requires one matrix-vector product $A\bar{\mathbf{p}}^k$, and two vector-vector inner products $[\bar{\mathbf{p}}^k]^T A\bar{\mathbf{p}}^k$ and $\|\bar{\mathbf{r}}^k\|_2^2$ per iteration, hence if A has $\mathcal{O}(N)$ non-zero entries, the work/iteration is $\mathcal{O}(N)$.

The CG gets its name (somewhat incorrectly, it should be **"the *A*-conjugate search-direction method"**) from the fact that the generated residuals are orthogonal, and the search directions are A-conjugate, *i.e.*

$$[\bar{\mathbf{r}}^k]^T \bar{\mathbf{r}}^j = [\bar{\mathbf{p}}^k]^T A\bar{\mathbf{p}}^j = 0, \quad \text{for } k \neq j.$$

A direct corollary of these (easily checked) facts, is

Corollary

If A is an $N \times N$ symmetric positive definite matrix, then the CG algorithm converges in at most N steps.

The Conjugate Gradient Method

The N-step termination theorem tells us that for the 5-point Laplacian on an $N \times N$ grid we need at most

$$W_{\text{CG}} = \underbrace{5(N \times N)}_{\text{Matrix Entries}} \cdot \underbrace{N \times N}_{\text{iterations}} = \mathcal{O}\left(N^4\right),$$

operations to compute the exact solution to $A\bar{\mathbf{x}} = \bar{\mathbf{b}}$. This may not seem so impressive, since optimal SOR does a better job

$$W^*_{\text{SOR}} \approx \frac{N^3}{\pi^2} \log(\epsilon^{-1}) = \mathcal{O}\left(N^3\right).$$

However, in practice the iterates $\bar{\mathbf{x}}^k$ generated by the CG-iteration converge to $\bar{\mathbf{x}}$ very rapidly, and the iteration can be stopped for $k \ll N \times N$ iterations. Applied to the 5-point Laplacian, the CG iteration and optimal SOR both require $\sim N \log(\epsilon^{-1})$ iterations to reach a specified tolerance. CG has the advantage over SOR in that *(i) there is no parameter (ω) which must be optimally chosen*; further *(ii) the CG-iteration can be accelerated further by preconditioning PCG(M)*.

Convergence Rate for the Conjugate Gradient Method

Theorem (Convergence Rate for Conjugate Gradient)

If A is a symmetric positive definite matrix whose eigenvalues lie in the interval $[a,b]$, then the error vector $\bar{\mathbf{e}}^k$ for the steepest descent method satisfies

$$[\bar{\mathbf{e}}^k]^T A \bar{\mathbf{e}}^k \leq \left[\frac{\sqrt{b}-\sqrt{a}}{\sqrt{b}+\sqrt{a}}\right]^{2k} [\bar{\mathbf{e}}^0]^T A \bar{\mathbf{e}}^0 \equiv \left[\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right]^{2k} [\bar{\mathbf{e}}^0]^T A \bar{\mathbf{e}}^0.$$

SAN DIEGO STATE UNIVERSITY

Conjugate Gradient vs. Steepest Descent

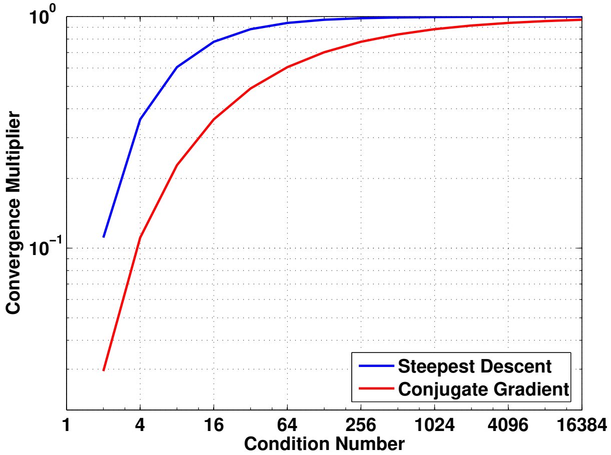

Figure: The convergence multipliers $m_{\text{SD}} = \left[\frac{\kappa-1}{\kappa+1}\right]^2$, and $m_{\text{CG}} = \left[\frac{\sqrt{\kappa}-\sqrt{1}}{\sqrt{\kappa}+\sqrt{1}}\right]^2$.

Conjugate Gradient vs. Steepest Descent

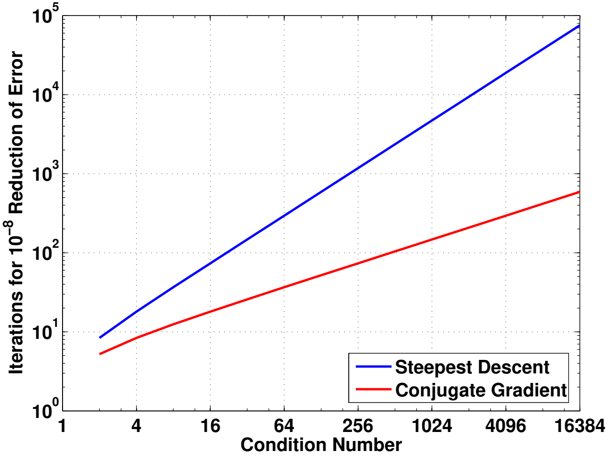

Figure: The number of iterations necessary to reduce the initial error by a factor of 10^{-8}.

GS vs. SOR vs. CG — 1 of 2

n	n^2	$\kappa(A)$	GS	SOR*	CG
8	64	47	252	65	11
16	256	169	837	121	27
32	1,024	641	2,870	223	52
64	4,096	2,489	9,983	414	98
128	16,384	9,807	34,706	777	192
256	65,536	38,926	—	1,473	370
512	262,144	155,103	—	2,813	715

Table: Number of iterations needed to achieve 10^{-8} **relative update.** 5-point Laplacian ∇^2_{5pt} in 2D discretized on an $n \times n$ grid $\rightsquigarrow$ $n^2 \times n^2$ matrix, with $\sim 5n^2$ non-zero elements.

GS vs. SOR vs. CG

n	n^2	$\kappa(A)$	GS	SOR*	CG
8	64	47	—	71	10
16	256	169	—	136	28
32	1,024	641	—	261	59
64	4,096	2,489	—	504	119
128	16,384	9,807	—	984	239
256	65,536	38,926	—	1,938	470
512	262,144	155,103	—	3,844	941

Table: Number of iterations needed to achieve 10^{-8} **residual reduction.** 5-point Laplacian ∇^2_{5pt} in 2D discretized on an $n \times n$ grid $\rightsquigarrow$ $n^2 \times n^2$ matrix, with $\sim 5n^2$ non-zero elements.

Bottom Line: Even in the “homework case” where the optimal SOR parameter is known, the Conjugate Gradient approach is better.

San Diego State University

Speeding Up Conjugate Gradient — PCG(M)

The conjugate gradient algorithm is not the end of the story (it is just barely the end of the beginning). By combining the `CG`-algorithm with the idea of preconditioning ($M \approx A$, and M easily invertible) the Preconditioned `CG` algorithm can be derived.

Further, the `CG`-method can be extended to work for non-symmetric matrices as well:

Symmetry	**Linear System** $\mathbf{A}\bar{\mathbf{x}} = \bar{\mathbf{b}}$	**Eigenvalue Problem** $\mathbf{A}\bar{\mathbf{x}} = \lambda\bar{\mathbf{x}}$
$\mathbf{A} = \mathbf{A}^*$	**CG**	**Lanczos**
$\mathbf{A} \neq \mathbf{A}^*$	**GMRES** **CGNE** / **CGNR** **BiCG**, etc...	**Arnoldi**

Finite Differences vs. Finite Elements

This ends our overview of finite difference schemes for hyperbolic, parabolic, and elliptic problems. We have seen quite a few tools useful for both analysis and implementation of these schemes...

More Topics...

- Spectral Methods
- Mimetic Methods (a different view of the Finite Difference problem)
- Finite Element Methods — a different approach to approximation.
  - The FEM formulation is better suited for complex domains, and includes local error estimates which help us locally improve the solution exactly where these errors are large.
  - The biggest disadvantage, from a pedagogical point of view, is that whereas FD methods are quite straight-forward to implement, setting up a meaningful FEM-solver requires more "technology." There are some nice ($$$) commercial packages available (*e.g.* Comsol Multiphysics: **http://www.comsol.com/**).

San Diego State University