

Scientific Computing II

Krylov Methods

Exercise 6: Method of Steepest Descent

In this exercise, we want to derive and understand the method of steepest descent to solve the system of linear equations $A \cdot x = b$.

Let $A \in \mathbb{R}^{n \times n}$ be symmetric and positive definite, $b \in \mathbb{R}^n$, and $f(x) := \frac{1}{2}x^T \cdot Ax - b^T \cdot x$.

- Describe the idea of and derive the steepest descent method.
- Prove that $\nabla f(x) = Ax - b$.
- Compute the first 3 iterations of the steepest descent method to minimize

$$f(x) := \frac{1}{2}x^T Ax - b^T x \text{ for } x \in \mathbb{R}^n$$

$$\text{with } A = \begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, x^{(1)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, d^{(1)} = -\nabla f(x^{(1)}) = b - Ax.$$

What do you observe?

- Compute the exact solution of the system in (c) and compute the error after each iteration. By which factor is it reduced in each iteration?
- Write down the pseudo code for one iteration of the steepest descent method solving the two-dimensional Poisson equation on the unit interval with homogeneous Dirichlet boundary conditions:

$$\begin{aligned} \Delta u &= 0 \text{ in }]0, 1[^2, \\ u &= 0 \text{ at } \partial]0, 1[^2. \end{aligned}$$

The Laplacian is discretized in a regular grid by the 5-point-stencil

$$\Delta u(i \cdot h, j \cdot h) \approx \frac{u_{i-1,j} + u_{i,j-1} - 4u_{i,j} + u_{i+1,j} + u_{i,j+1}}{h^2}.$$

Use elementwise notation (i.e. not matrix-vector notation)!

- (f) Give the cost per iteration of the steepest descent in the form $O(N^p)$ for general systems of linear equations! In addition, consider the one-dimensional Poisson equation with homogeneous Dirichlet conditions (cf. sheet 1).
- (g) Consider the one-dimensional Poisson equation. Use the asymptotic convergence rate $\rho = \frac{\kappa-1}{\kappa+1}$ with $\kappa = \frac{\lambda_{max}}{\lambda_{min}}$ of the steepest descent method to derive the cost of the method if used as an iterative solver for this problem.
 For the Poisson problem as discretized on sheet 1, $\lambda_{max} = 1 + \cos(\pi h)$ and $\lambda_{min} = 1 - \cos(\pi h)$.

Solution:

- (a)
- Solving the system $Ax = b$ is equal to solving the residual equation $b - Ax = 0$.
 - For such a system, the quadratic form, i.e. a functional $f(x) = \frac{1}{2}x^T Ax - b^T x + c$ can be defined.
 - If A is positiv definite and symmetric, $f(x)$ is a parabolic function $\mathbb{R}^n \mapsto \mathbb{R}$ with $\nabla f(x) = Ax - b = -res$, pointing to the direction of steepest ascent.
 - In the minimum of $f(x)$ it holds that $\nabla f(x) = Ax - b = 0$.

Algorithm: From some initial guess $x^{(0)}$, walk 'downwards' along the direction of steep descent $-\nabla f = b - Ax = r$:

$$x^{(it+1)} = x^{(it)} + \alpha \cdot r^{(it)}$$

How to choose α ? \Rightarrow Find minimum along the search direction:

$$\min_{\alpha} f(x^{(it)} + \alpha \cdot r^{(it)}) \Rightarrow \text{chain rule } \nabla f(x^{(it+1)}) \cdot r^{(it)} = 0 \quad (1)$$

$$(r^{(it+1)})^T \cdot r^{(it)} = 0 \quad (2)$$

$$((b - Ax)^{(it+1)})^T \cdot r^{(it)} = 0 \quad (3)$$

$$\vdots \quad (4)$$

$$\alpha = \frac{r^T \cdot r}{(Ar)^T \cdot r} \quad (5)$$

- (b) Proof $\nabla f(v) = Ax - b$:

$$f(x) := \frac{1}{2}x^T \cdot Ax - x \cdot b, \quad f(x) = \frac{1}{2} \sum_{i=1}^n x_i \sum_{j=1}^n A_{ij}x_j - \sum_{i=1}^n x_i b_i$$

$$(\nabla f(x))_k = \frac{\partial}{\partial x_k} f(x) \quad (6)$$

$$= \frac{1}{2} \sum_{i,j} A_{ij} \frac{\partial}{\partial x_k} (x_i x_j) - \sum_i b_i \frac{\partial}{\partial x_k} x_i \quad (7)$$

$$= \frac{1}{2} \sum_{i,j} A_{ij} \left(\left(\frac{\partial}{\partial x_k} x_i \right) x_j + x_i \left(\frac{\partial}{\partial x_k} x_j \right) \right) - \sum_i b_i \frac{\partial}{\partial x_k} x_i \quad (8)$$

$$= \frac{1}{2} \sum_{i,j} A_{ij} \delta_{ik} x_j + \frac{1}{2} \sum_{i,j} A_{ij} x_i \delta_{jk} - \sum_i b_i \delta_{ik} \quad (9)$$

$$= \frac{1}{2} \sum_j A_{kj} x_j + \frac{1}{2} \sum_i A_{ik} x_i - b_k \quad (10)$$

$$\stackrel{\text{A symmetric}}{=} \sum_i A_{ki} x_i - b_k = (Ax)_k - b_k \quad (11)$$

(c) **First Iteration:**

$$\alpha_1 = \frac{d^T d}{d^T A d} = \frac{2}{7}, \quad \text{as } d^{(1)} = -\nabla f(0) = b$$

$$x^{(2)} = x^{(1)} + \alpha_1 d^{(1)} = \frac{2}{7} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Second Iteration:

$$d^{(2)} = d^{(1)} - \alpha_1 A d^{(1)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \frac{2}{7} \begin{pmatrix} 4 \\ 3 \end{pmatrix}$$

$$= \frac{1}{7} \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

$$\alpha_2 = \frac{2}{3}$$

$$x^{(3)} = x^{(2)} + \alpha_2 d^{(2)} = \frac{4}{21} \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

Third Iteration:

$$d^{(3)} = d^{(2)} - \alpha_2 A d^{(2)} = \frac{1}{21} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\alpha_3 = \frac{2}{7}$$

$$x^{(4)} = x^{(3)} + \alpha_3 d^{(3)} = \frac{2}{147} \begin{pmatrix} 15 \\ 29 \end{pmatrix}$$

Observation: The initial search direction is searched for the second time, the exact solution is still not reached.

(d) Exact solution:

$$\begin{aligned}\bar{x} &= \frac{1}{5} \begin{pmatrix} 2 & -1 \\ -1 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ &= \frac{1}{5} \begin{pmatrix} 1 \\ 2 \end{pmatrix}\end{aligned}$$

Error $e^{(k)} = x^{(k)} - \bar{x}$ or $e^{(k+1)} = e^{(k)} + \alpha_k \bar{d}^{(k)}$.

$$\begin{array}{llllll} e^{(1)} = \begin{pmatrix} -0.2 \\ -0.4 \end{pmatrix} & f(x^{(1)}) = 0 & r^{(1)} \approx 1.41 & \|e^{(1)}\|_A \approx 0.775 & \|e^{(1)}\|_2 \approx 0.447 \\ e^{(2)} = \begin{pmatrix} \frac{3}{35} \\ -\frac{1}{35} \end{pmatrix} \approx \begin{pmatrix} 0.09 \\ -0.11 \end{pmatrix} & f(x^{(2)}) \approx -0.286 & r^{(2)} \approx 0.20 & \|e^{(2)}\|_A \approx 0.169 & \|e^{(2)}\|_2 \approx 0.143 \\ e^{(3)} = \begin{pmatrix} \frac{-1}{105} \\ \frac{2}{105} \end{pmatrix} \approx \begin{pmatrix} -0.01 \\ 0.02 \end{pmatrix} & f(x^{(3)}) \approx -0.299 & r^{(3)} \approx 0.07 & \|e^{(3)}\|_A \approx 0.037 & \|e^{(1)}\|_2 \approx 0.021 \\ e^{(4)} = \begin{pmatrix} 0.004 \\ -0.005 \end{pmatrix} & f(x^{(4)}) \approx -0.3 & r^{(4)} \approx 0.009 & \|e^{(4)}\|_A \approx 0.008 & \|e^{(1)}\|_2 \approx 0.007 \end{array}$$

Error reduction:

$$\begin{array}{ll} \frac{\|e^{(2)}\|_A}{\|e^{(1)}\|_A} \approx 0.22 & \frac{\|e^{(2)}\|_2}{\|e^{(1)}\|_2} \approx 0.32 \\ \frac{\|e^{(3)}\|_A}{\|e^{(2)}\|_A} \approx 0.22 & \frac{\|e^{(3)}\|_2}{\|e^{(2)}\|_2} \approx 0.15 \\ \frac{\|e^{(4)}\|_A}{\|e^{(3)}\|_A} \approx 0.22 & \frac{\|e^{(4)}\|_2}{\|e^{(3)}\|_2} \approx 0.33 \end{array}$$

(e) Pseudo Code:

$$r_i = b - Ax_i \quad (12)$$

$$\alpha_i = \frac{r_i^T r_i}{r_i^T A r_i} \quad (13)$$

$$x_{i+1} = x_i + \alpha_i r_i \quad (14)$$

in Matlab Pseudo-Code :

```
for j=1:N do
    r_i^j = b_i^j;
    for k=1:N do
        r_i^j = r_i^j - A(j,k) * x_i^k;
    end
end
for j=1:N do
```

```

    a = a + r_i^j * r_i^j;
    for k=1:N do
        tmp^j = A(j,k) * r_i^k;
    end
    b = b + r_i^j * tmp^j;
end
alpha = a / b;

for j=1:N do
    x_{i+1}^j = x_i^j + alpha * r_i^j
end

```

(f) Number of operations in general:

- $r^{(i)} = b - Ax^{(i)}$: $2 \cdot N^2 + N$ Operations.
- $\alpha_i = \frac{(r^{(i)})^T r^{(i)}}{(r^{(i)})^T A r^{(i)}}$: $2 \cdot N^2 + 4N$ Operations.
- $x^{(i+1)} = x^{(i)} + \alpha_i r^{(i)}$: $2 \cdot N$ Operations.

In total: $4N^2 + 7N$ Operations $\Rightarrow O(N^2)$.

For Poisson-Equation in 3d, with N degrees of freedom per dimension: $O(N^6)$.

Considering the sparsity of the resulting matrix:

- $\vec{v} = A\vec{u} - \vec{b}$
 - $A\vec{u}$: taking into account the sparsity of our matrix, we get 7 multiplications and 6 additions per line of A , A has N^3 lines
 $\Rightarrow 13N^3$ operations,
 - $A\vec{u} - \vec{b}$: N^3 operations

Together: $14N^3$ operations.

- $\frac{\vec{v}^T \vec{v}}{\vec{v}^T A \vec{v}}$
 - $A\vec{v}$: $13N^3$ operations (see above),
 - $\vec{v}^T A\vec{v}$: $2N^3$ operations,
 - $\vec{v}^T \vec{v}$: $2N^3$ operations,

Together: $17N^3$ operations.

- $\vec{u} = \vec{u} + c\vec{v}$:

$2N^3$ operations.

Sum: $33N^3 = O(N^3)$ operations.

- (g) The convergence rate is given as $\rho = \frac{\kappa-1}{\kappa+1}$ with $\kappa = \frac{\lambda_{max}}{\lambda_{min}}$. We know that the eigen values λ_{min} and λ_{max} for the 1d Poisson equation read: $\lambda_{max} = 1 + \cos(\pi h)$ and $\lambda_{min} = 1 - \cos(\pi h)$.

$$\begin{aligned} \rho &= \frac{\kappa-1}{\kappa+1} = \frac{\frac{1+\cos(\pi h)}{1-\cos(\pi h)} - 1}{\frac{1+\cos(\pi h)}{1-\cos(\pi h)} + 1} \\ &= \frac{1 + \cos(\pi h) - (1 - \cos(\pi h))}{1 + \cos(\pi h) + 1 - \cos(\pi h)} \\ &= \cos(\pi h) \stackrel{\text{Taylor series}}{=} 1 - \frac{h^2}{2} + O(h^4) \end{aligned}$$

If we halve the mesh width h and neglect higher order terms, the application of two iterations gives:

$$\left(1 - \left(\frac{h}{2}\right)^2\right) \cdot \left(1 - \left(\frac{h}{2}\right)^2\right) = \left(1 - \left(\frac{h}{2}\right)^2 + \frac{\left(\frac{h}{2}\right)^4}{4}\right) \doteq 1 - \left(\frac{h}{2}\right)^2$$

The application of two more iterations gives the initial error reduction of $1 - \frac{h^2}{2}$.

That means, if we double the number of grid points (degrees of freedom N)
 \Rightarrow we have to perform 4 times as many iterations
 \Rightarrow number of iterations behaves like $O(N^2)$

Exercise 7: Conjugate Gradients Method

- Describe the idea of and derive the Conjugate Gradients method.
- Solve the system from 6 c) with CG. What do you observe?
- Give the cost per iteration of CG in the form $O(N^p)$ for general systems of linear equations! In addition, consider the one-dimensional Poisson equation with homogeneous Dirichlet conditions (cf. sheet 1).
- Give the overall costs of CG if used as a direct solver! Compare to the results for Gaussian elimination.
- Consider the one-dimensional Poisson equation. Use the asymptotic convergence rate of CG $\rho = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$ to derive the cost of the method if used as an iterative solver. For the Poisson problem discretized as on sheet 1, $\lambda_{max} = 1 + \cos(\pi h)$ and $\lambda_{min} = 1 - \cos(\pi h)$.

Solution:

- The problem with Steepest Descent is that it searches same directions several times.
Idea:

- Take a set of orthogonal vectors $\{d_0, d_1, \dots, d_n\}$ as search directions.
- Walk so far along that direction, that the error e^{i+1} is orthogonal to all previous search directions $\{d_0, \dots, d_i\}$.
- $d_i^T \cdot e_{i+1} = 0$
- $d_i^T \cdot (e_i + \alpha d_i) = 0 \Rightarrow \alpha = -\frac{d_i^T \cdot e_i}{d_i^T \cdot d_i}$

Problem: Error is not known \Rightarrow

- Use A-orthogonal search directions: $d_i^T A d_j = 0$;
- $\Rightarrow \alpha = -\frac{d_i^T A \cdot e_i}{d_i^T \cdot d_i} = -\frac{d_i^T r_i}{d_i^T \cdot d_i}$

Conjugate Gradients - Algorithm

$$\text{How far to go along } d_i? \quad \alpha = \frac{r_i^T r_i}{d_i^T A d_i}$$

$$\text{Compute new } u_i \quad u_{i+1} = u_i + \alpha d_i$$

$$\text{Compute new residual} \quad r_{i+1} = r_i - \alpha A d_i$$

$$\text{Compute } \beta, \text{ so that} \quad \beta = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}$$

$$d_i \text{ and } d_{i+1} \text{ are orthogonal} \quad d_{i+1} = r_{i+1} + \beta d_i$$

(b) **First Iteration:**

$$d_1 = r_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\alpha_1 = \frac{r_1^T r_1}{d_1^T A d_1} = \frac{(1 \ 1) \begin{pmatrix} 1 \\ 1 \end{pmatrix}}{(1 \ 1) \begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}} = \frac{2}{7}$$

$$u_2 = u_1 + \alpha_1 d_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \frac{2}{7} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{2}{7} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$r_2 = r_1 - \alpha_1 A d_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \frac{2}{7} \begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{7} \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

$$\beta_2 = \frac{r_2^T r_2}{r_1^T r_1} = \frac{1}{49}$$

$$d_2 = r_2 + \beta_2 d_1 = \frac{1}{7} \begin{pmatrix} -1 \\ 1 \end{pmatrix} + \frac{1}{49} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{49} \begin{pmatrix} -6 \\ 8 \end{pmatrix}$$

Second Iteration:

$$\alpha_2 = \frac{r_2^T r_2}{d_2^T A d_2} = \frac{\frac{1}{49} \begin{pmatrix} -1 & 1 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \end{pmatrix}}{\frac{1}{49} \begin{pmatrix} -6 & 8 \end{pmatrix} \begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix} \frac{1}{49} \begin{pmatrix} -6 \\ 8 \end{pmatrix}} = \frac{7}{10}$$

$$u_3 = u_2 + \alpha_2 d_2 = \frac{2}{7} \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \frac{7}{10} \cdot \frac{1}{49} \begin{pmatrix} -6 \\ 8 \end{pmatrix} = \frac{2}{7} \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \frac{1}{70} \begin{pmatrix} -6 \\ 8 \end{pmatrix} = \frac{1}{70} \begin{pmatrix} 14 \\ 28 \end{pmatrix}$$

$$= \frac{1}{5} \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

(c) Let N be the number of unknowns. From the algorithm shown in the lecture:

In general:

- three scalar products $\approx 6N$ operations
- two matrix \times vector products $\approx 4N^2$ operations
- three scalar \times vector products $\approx 3N$ operations
- three vector additions/subtractions $\approx 3N$ operations

Sum: $4N^2 + 12N = O(N^2)$ operations.

Taking the sparsity of the matrix for the 1d poisson problem into account:

- three scalar products $\approx 6N$ operations
- two matrix \times vector products $\approx 10N$ operations
- three scalar \times vector products $\approx 3N$ operations
- three vector additions/subtractions $\approx 3N$ operations

Sum: $28N = O(N)$ operations.

(d) If CG is used as a direct solver, it performs N iterations (one for each dimension). Thus the overall complexity is $O(N^3)$ in general, or $O(N^2)$ for sparse matrices.

Gaussian elimination has same complexities for 1d. Think about higher dimensional problems ;).

(e) The convergence rate is given as $\rho = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$ with $\kappa = \frac{\lambda_{max}}{\lambda_{min}}$. We know that the eigen values λ_{min} and λ_{max} for the 1d Poisson equation read: $\lambda_{max} = 1 + \cos(\pi h)$ and $\lambda_{min} = 1 - \cos(\pi h)$.

$$\begin{aligned}
\rho &= \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} = \frac{\sqrt{\frac{1+\cos(\pi h)}{1-\cos(\pi h)}} - 1}{\sqrt{\frac{1+\cos(\pi h)}{1-\cos(\pi h)}} + 1} \\
&= \frac{\sqrt{1+\cos(\pi h)} - \sqrt{1-\cos(\pi h)}}{\sqrt{1+\cos(\pi h)} + \sqrt{1-\cos(\pi h)}} \\
&= 1 - \frac{2\sqrt{1-\cos(\pi h)}}{\sqrt{1+\cos(\pi h)} + \sqrt{1-\cos(\pi h)}} \\
&\stackrel{\text{Taylor series}}{=} 1 - \frac{2\sqrt{\pi^2 h^2 + O(h^4)}}{\sqrt{2 + O(h^2)} + \sqrt{\pi^2 h^2 + O(h^4)}} \\
&= 1 - (2\pi + c)h + O(h^2).
\end{aligned}$$

If we halve the mesh width h and neglect higher order terms, the application of two iterations gives:

$$\begin{aligned}
\left(1 - (2\pi + c)\frac{h}{2}\right) \cdot \left(1 - (2\pi + c)\frac{h}{2}\right) &= 1 - (2\pi + c)h + \left((2\pi + c)\frac{h}{2}\right)^2 \\
&\doteq 1 - (2\pi + c)h
\end{aligned}$$

I.e. the application of two iterations gives the initial error reduction of $1 - (2\pi + c)h$.

That means, if we double the number of grid points (degrees of freedom N):

⇒ we have to perform 2 times as many iterations

⇒ number of iterations behaves like $O(N)$