Numerical Methods I

MATH-GA 2010.001/CSCI-GA 2420.001

Benjamin Peherstorfer
Courant Institute, NYU

Based on slides by G. Stadler and A. Donev

Today

Last time

- Singular value decomposition (SVD)
- Iterative methods for systems of linear equations

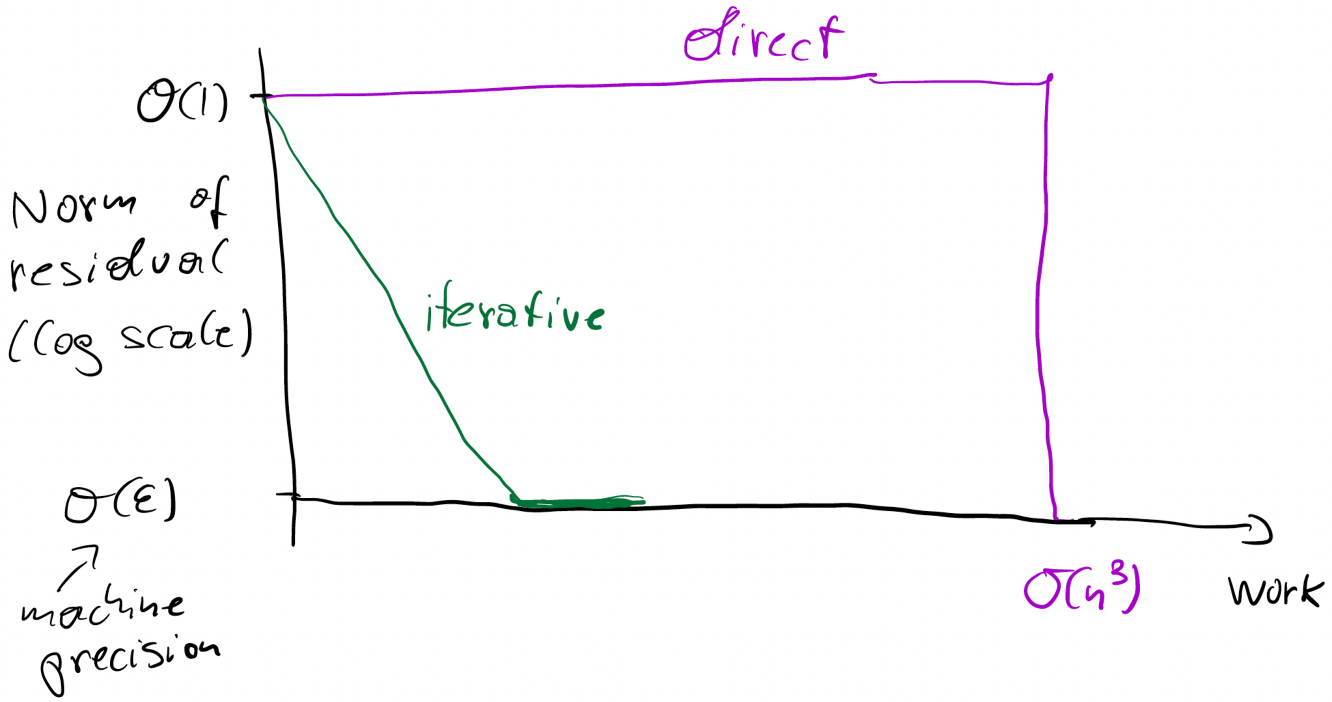
Today

- Iterative methods
- Conjugate gradients method

Announcements

- Homework 4 is due Mon, Nov 7, 2022 before class
- Midterm grades posted

Recap: Iterative methods



- Iterative methods *can* converge geometrically until residual is below machine precision
- Direct methods make no progress at all until $\mathcal{O}(n^3)$ work is done, and then lead to residual on the order of machine precision

Recap: Iterative methods

Let Q be invertible, then

$$\begin{aligned} A\boldsymbol{x} = \boldsymbol{b} &\Leftrightarrow Q^{-1}(\boldsymbol{b} - A\boldsymbol{x}) = 0 \\ &\Leftrightarrow (I - Q^{-1}A)\boldsymbol{x} + Q^{-1}\boldsymbol{b} = \boldsymbol{x} \\ &\Leftrightarrow G\boldsymbol{x} + \boldsymbol{c} = \boldsymbol{x} \end{aligned}$$

Leads to fixed-point iteration

$$\boldsymbol{x}_{k+1} = \boldsymbol{G}\boldsymbol{x}_k + \boldsymbol{c}$$

and $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$ is stationary point

$$\boldsymbol{G}\boldsymbol{x} + \boldsymbol{c} = (\boldsymbol{I} - \boldsymbol{Q}^{-1}\boldsymbol{A})\boldsymbol{x} + \boldsymbol{Q}^{-1}\boldsymbol{b} = \boldsymbol{x} - \boldsymbol{Q}^{-1}\boldsymbol{b} + \boldsymbol{Q}^{-1}\boldsymbol{b} = \boldsymbol{x}$$

Recap: Common choices for Q

Requirements on a good Q are

- Q^{-1} is representative of A^{-1}
- We can numerically solve $Qy = z$ much quicker than $Ax = b$ because in each step we solve
$$\boldsymbol{Q}\boldsymbol{x}_{k+1} = (\boldsymbol{Q} - \boldsymbol{A})\boldsymbol{x}_k + b$$

Split the matrix A as follows

$$A \quad = \quad L \quad + \quad D \quad + \quad U$$

Recap: Jacobi method

Theorem: Select now $Q = D$ ... Jacobi method. The Jacobi method converges for any starting point $\boldsymbol{x}_o$ to the solution of $A\boldsymbol{x} = \boldsymbol{b}$ if A is strictly diagonal dominant, i.e.,

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|, \qquad \text{for } i = 1, \dots, n.$$

Notice that $Q = D$ is a good choice in terms of computational costs because we can very quickly solve $Dy = z$ for the diagonal matrix D

Gauss-Seidel method

Theorem: Choose $Q = D + L$ . . . Gauss-Seidel method. The Gauss-Seidel method converges for any starting point $\boldsymbol{x}_o$ if A is symmetric positive definite (spd).
$\rightsquigarrow$ board

Gauss-Seidel

$$x_{k+1} = (\underline{I} - (D+L)^{-1}A)\, x_k + (D+L)^{-1} b$$

$$= \underbrace{\big((D+L)^{-1}(D+L) - (D+L)^{-1}(L+D+R)\big)}\, x_k + (D+L)^{-1} b$$

$$= -(D+L)^{-1} R\, x_k + (D+L)^{-1} b$$

For any spd A, we have a scalar product

$$\langle x, y \rangle_A = \langle x, Ay \rangle \quad \text{on } \mathbb{R}^n$$

And for any matrix $B \in \mathbb{R}^{n\times n}$, we have adjoint w.r.t. $\langle\cdot,\cdot\rangle_A$ given by

$$B^* = A^{-1} B^T A$$

so that

$$\langle Bx, y \rangle_A = \langle x, B^* y \rangle_A \quad \forall x, y \in \mathbb{R}^n$$

B is positive w.r.t. $\langle\cdot,\cdot\rangle_A$ if $\langle Bx, x\rangle_A > 0 \quad \forall x \neq 0$

—

Let $G \in \mathbb{R}^{n\times n}$ with adjoint G^* w.r.t. $\langle\cdot,\cdot\rangle$.
Then, if $B = I - G^*G$ is positive w.r.t. $\langle\cdot,\cdot\rangle$, it follows $\rho(G) < 1$.

In GS, we have

$$G = I - (D+L)^{-1}A$$

$B = I - G^*G$ is a positive matrix w.r.t. $\langle \cdot, \cdot \rangle_A$.

$$G^* = A^{-1}G^TA =$$

$$= I - A^{-1}A^T\underbrace{(D+R)^{-1}}_{(D+L)^{-T}}A$$

$$= I - (D+R)^{-1}A$$

$$B = I - G^*G = \dots = (D+R)^{-1}D(D+L)^{-1}A$$

show that B is pos.

$$\langle Bx, x\rangle_A = \langle D^{1/2}(D+L)^{-1}Ax, D^{1/2}(D+L)^{-1}Ax\rangle$$

$$= \|\underbrace{D^{1/2}}\underbrace{(D+L)^{-1}}\underbrace{Ax}\|_2^2 > 0 \quad \forall x \neq 0$$

$$\Rightarrow \rho(G) < 1$$

Gauss-Seidel method

Theorem: Choose $Q = D + L$ ... Gauss-Seidel method. The Gauss-Seidel method converges for any starting point $\boldsymbol{x}_o$ if A is symmetric positive definite (spd).
$\leadsto$ board

Notice that $Q = D + L$ is a good choice in terms of computational costs because we can very quickly solve $(D + L)y = z$ for the lower triangular matrix $D + L$ (forward substitution)

Relaxation methods:

Use linear combination between new and previous iterate:

$$\boldsymbol{x}_{k+1} = \omega \underbrace{(G\boldsymbol{x}_k + c)}_{\boldsymbol{x}'_{k+1}} + (1-\omega)\boldsymbol{x}_k = G_\omega \boldsymbol{x}_k + \omega c,$$

where $\omega > 0$ is a damping/relaxation parameter (sometimes, $\omega > 1$ is used, leading to overrelaxation). Target is to choose ω such that $\rho(G_\omega)$ is minimal.

Def: A fixed point method $\boldsymbol{x}_{k+1} = G\boldsymbol{x}_k + \boldsymbol{c}$ with $G = G(A)$ is called *symmetrizable* if for any symmetric positive definite (spd) matrix A, the matrix $I - G$ is similar to an spd matrix, i.e., there is a regular W such that $W(I-G)W^{-1}$ spd.

board

Symmetrizable:

↳ Richardson: $G = I - A$

$\Rightarrow I - G = A$ is spd, if A spd

↳ Jacobi: $G = I - D^{-1}A$, set $W = D^{1/2}$

$$D^{1/2}(I - G)D^{-1/2} = I - I + D^{1/2}D^{-1}AD^{-1/2}$$

$$= D^{-1/2}AD^{-1/2}$$

For sym. schemes

$$x_{k+1} = Gx_k + c \quad , \quad G = G(A) \quad , \quad A \text{ spd}$$

$$\sigma(G) \subset (-\infty, 1)$$

$$x_{k+1} = w(Gx_k + c) + (1-w)x_k$$

$$= G_w x_k + wc$$

with $G_w = wG + (1-w)I$

want find $0 < w \le 1$ with minimal $\rho(G_w)$

EV of G_w are

$$\lambda_i(G_w) = w\lambda_i(G) + 1 - w$$
$$= 1 - w(1 - \lambda_i(G))$$

$$\lambda_{min}(G) \le \lambda_{max}(G) < 1$$

$$\lambda_i(G_w) < 1$$

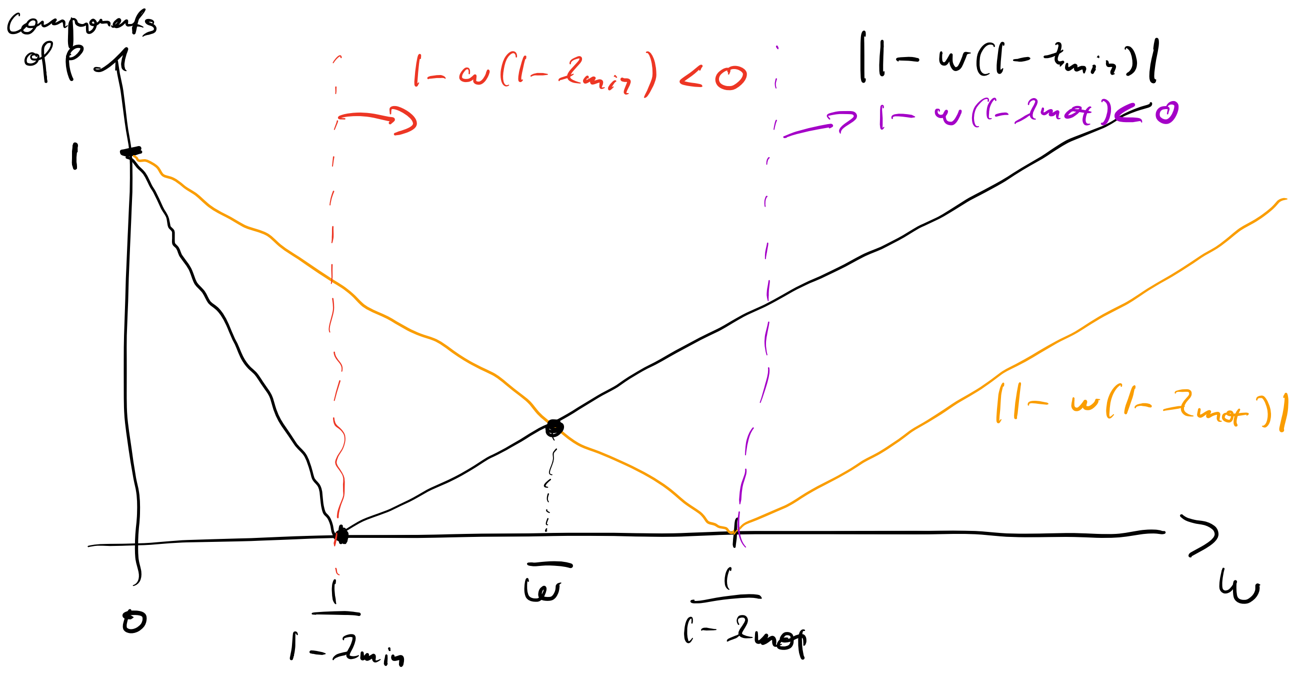
$$\rho(G_w) = \max\{|1 - w(1 - \lambda_{min}(G))|, \; |1 - w(1 - \lambda_{max}(G))|\}$$

Let's look at $w_{min} = \frac{1}{1 - \lambda_{min}(G)} > 0$

$$|1 - w_{min}(1 - \lambda_{min}(G))| = 0$$

Now $w_{max} = \frac{1}{1 - \lambda_{max}(G)}$

$$|1 - w_{max}(1 - \lambda_{max}(G))| = 0$$



optimal $\bar{\omega}$ satisfies

$$-(1-\bar{\omega}(1-\lambda_{min}(G))) = 1-\bar{\omega}(1-\lambda_{max}(G))$$

$$\bar{\omega} = \frac{2}{2-\lambda_{max}(G)-\lambda_{min}(G)}$$

Richardson: spd matrix A , $G = I - A$

$$\lambda_{min}(G) = 1 - \lambda_{max}(A)$$

$$\lambda_{max}(G) = 1 - \lambda_{min}(A)$$

$$\bar{\omega} = \frac{2}{\lambda_{max}(A) + \lambda_{min}(A)}$$

$$\rho(G_{\bar{\omega}}) = \dots = \frac{\lambda_{max}(A) - \lambda_{min}(A)}{\lambda_{max}(A) + \lambda_{min}(A)}$$

$$= \dots = \frac{k_2(A) - 1}{k_2(A) + 1} < 1$$

Finding the optimal damping parameter: $\rightsquigarrow$ board

Finding the optimal damping parameter: $\rightsquigarrow$ board

We obtain that

$$\bar{\omega} = \frac{2}{2 - \lambda_{\text{max}}(G) - \lambda_{\text{min}}(G)}$$

is the optimal damping parameter for symmetrizable iteration methods that minimizes the spectral radius. The spectral radius is

$$\rho(G_{\bar{\omega}}) < 1$$

This means that for a suitable choice $\bar{\omega}$ we can make *any* symmetrizable iteration method convergent for an spd A!

Finding the optimal damping parameter: $\rightsquigarrow$ board

We obtain that

$$\bar{\omega} = \frac{2}{2 - \lambda_{\text{max}}(G) - \lambda_{\text{min}}(G)}$$

is the optimal damping parameter for symmetrizable iteration methods that minimizes the spectral radius. The spectral radius is

$$\rho(G_{\bar{\omega}}) < 1$$

This means that for a suitable choice $\bar{\omega}$ we can make *any* symmetrizable iteration method convergent for an spd A!

Optimal damping parameter for Richardson iteration $\rightsquigarrow$ board

Algorithmic perspective on iterative methods

Richardson iterations

$$\boldsymbol{x}_{k+1} = (I - A)\boldsymbol{x}_k + b$$

```
for k = 0, 1, ...
  for i = 0, 1, ..., n-1:   $x_{k+1}[i] = x_k[i] + r_k[i]$
```

where the residual $\boldsymbol{r}_k$ at iteration k is given by

$$\boldsymbol{r}_k = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k$$

What would we like to update $\boldsymbol{x}_k$ with?

Algorithmic perspective on iterative methods

Richardson iterations

$$\boldsymbol{x}_{k+1} = (I - A)\boldsymbol{x}_k + b$$

```
for k = 0, 1, ...
  for i = 0, 1, ..., n-1:   $x_{k+1}[i] = x_k[i] + r_k[i]$
```

where the residual $\boldsymbol{r}_k$ at iteration k is given by

$$\boldsymbol{r}_k = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k$$

What would we like to update $\boldsymbol{x}_k$ with? We would like to update $\boldsymbol{x}_k$ in the direction of the error $\boldsymbol{e}_k = \boldsymbol{x} - \boldsymbol{x}_k$ because then

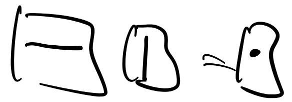
$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{e}_k = \boldsymbol{x}$$

However, we don't have the error $\boldsymbol{e}_k$ and therefore it is reasonable to use the next best thing which is the residual in many situations

$$\boldsymbol{r}_k = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}\boldsymbol{x}_k = \boldsymbol{A}(\boldsymbol{x} - \boldsymbol{x}_k) = \boldsymbol{A}\boldsymbol{e}_k$$

$\rightsquigarrow$ Richardson iteration updates $\boldsymbol{x}_k$ in the direction of the residual $\boldsymbol{r}_k$

Jacobi iterations



$$\boldsymbol{x}_{k+1} = (I - D^{-1}A)\boldsymbol{x}_k + D^{-1}b$$

$r_k[i] = b_i - a_{i,:} x_k$

$a_{i,:} x_{k+1} = b_i$

```
for k = 0, 1, ...
 for i = 0, 1, ..., n-1:   $y[i] = 1/a_{ii} r_k[i]$
 for i = 0, 1, ..., n-1:   $x_{k+1}[i] = x_k[i] + y[i]$
```

- In every substep i of iteration k, an update $y[i]$ is computed and stored
- Applied immediately, this would lead to the (momentary) disappearance of the i-th component of the residual r_k
- Thus, with this current approximation, equation i would be solved exactly—an improvement that would be lost immediately in the following substep for the equation $i + 1$
- However, the updates of a component are not applied immediately but only at the end of an iteration step (second i-loop)

Gauss-Seidel iteration

$$\boldsymbol{x}_{k+1} = (I - (L + D)^{-1}A)\boldsymbol{x}_k + (L + D)^{-1}b$$

```
for k = 0, 1, ...
 for i = 0, 1, ..., n-1:   $r_k[i] = b[i] - \sum_{j=1}^{i-1} a_{ij}x_{k+1}[j] - \sum_{j=i}^{n} a_{ij}x_k[j]$
                           $y[i] = 1/a_{ii}r_k[i], \quad x_{k+1}[i] = x_k[i] + y[i]$
```

- In contrast to Jacobi method, the update is performed immediately
- Therefore the new modified values for components $1, ..., i - 1$ are already available for updating component i

Damping (mostly Jacobi) or over-relaxation (mostly Gauss-Seidel) means to take step lengths different from 1 in the direction of the residual

The spectral radius of typical iterative matrices

- The spectral radius ρ determines convergence and speed; the smaller ρ, the faster the error decays. In practice, ρ is often very close to 1 so that even though $\rho < 1$ it takes an unreasonable amount of iterations to get a reasonable answer

- An important sample scenario is the discretization of PDEs: It is typical that ρ depends on the dimension n of the matrix A, and thus in terms of PDE discretization it depends on the mesh width h of the underlying grid. For example

$$\rho \in \mathcal{O}(1 - h_l^2) = \mathcal{O}(1 - \frac{1}{4^l})$$

  with mesh width $h_l = 2^{-l}$ in one dimension ($\mathcal{O}(1 - 16^{-l})$ in two spatial dimensions)

- This is a huge disadvantage: Why?

The spectral radius of typical iterative matrices

- The spectral radius ρ determines convergence and speed; the smaller ρ, the faster the error decays. In practice, ρ is often very close to 1 so that even though $\rho < 1$ it takes an unreasonable amount of iterations to get a reasonable answer

- An important sample scenario is the discretization of PDEs: It is typical that ρ depends on the dimension n of the matrix A, and thus in terms of PDE discretization it depends on the mesh width h of the underlying grid. For example

$$\rho \in \mathcal{O}(1 - h_l^2) = \mathcal{O}(1 - \frac{1}{4^l})$$

  with mesh width $h_l = 2^{-l}$ in one dimension ($\mathcal{O}(1 - 16^{-l})$ in two spatial dimensions)

- This is a huge disadvantage: Why? the finer the grid is (and therefore the more accurate our approximation of the PDE solution should be), the slower these iterative methods get.

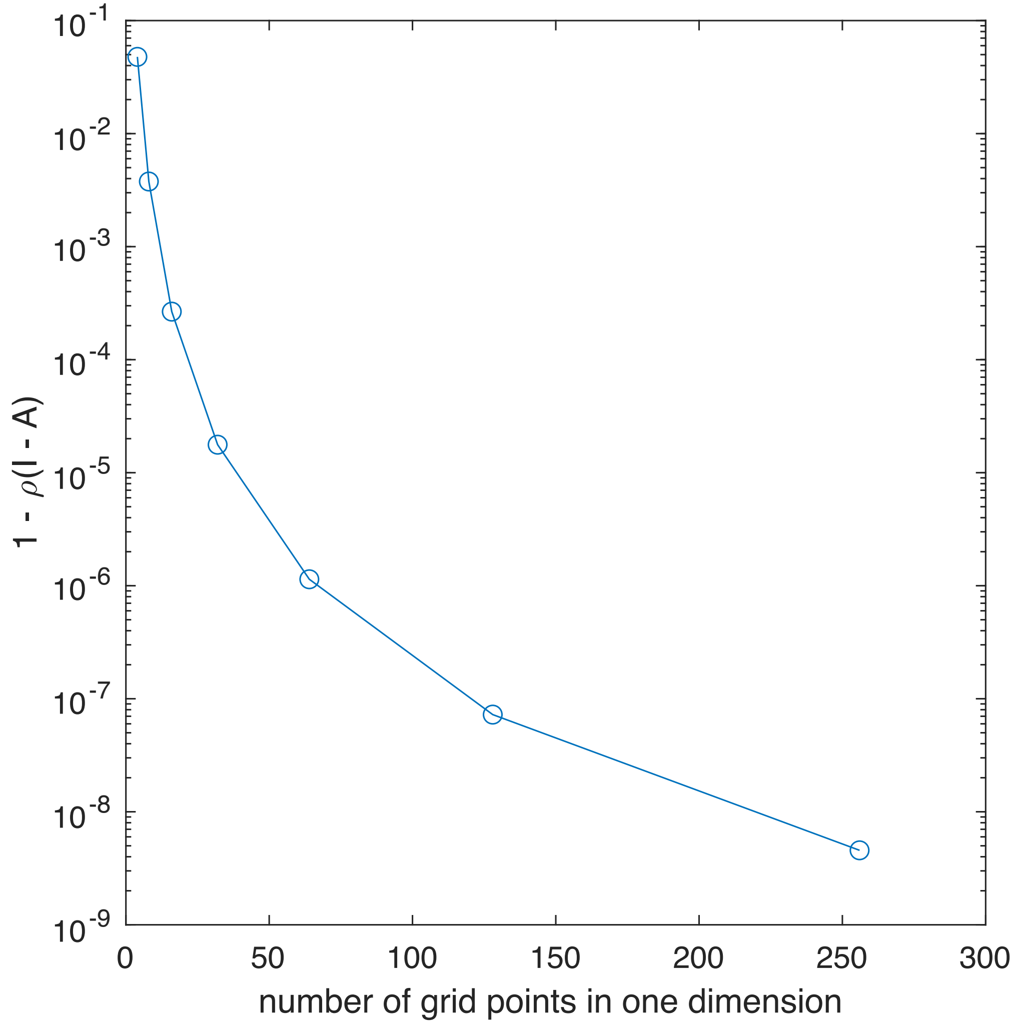
Consider the Laplace equation

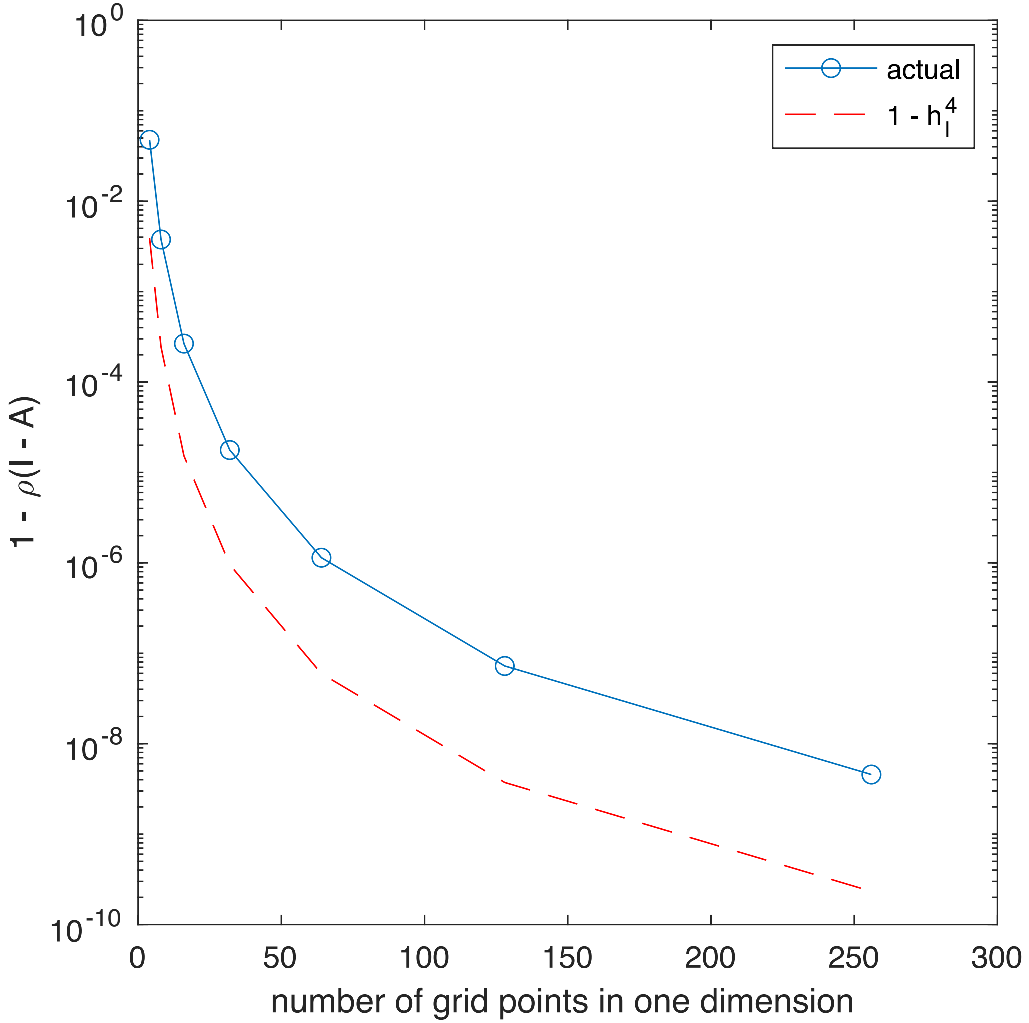
$$\Delta u(x_1, x_2) = 0\,,$$

in two dimensions and discretize with five-point second-order finite-difference stencil on $N \times N$ grid points ($\rightsquigarrow$ NM2)

Plot the spectral radius $\rho(I - A)$ (Richardson interpolation) w.r.t. number of grid points

```
Nlist = 2.^(2:8);
eList = [];
for N=Nlist
    X = gallery('poisson', N)*(1/N^2);
    eList(end + 1) = eigs(speye(N^2) - X, 1);
end
```


$1 - \rho(I - A)$
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{-9}
0
50
100
150
200
250
300
number of grid points in one dimension

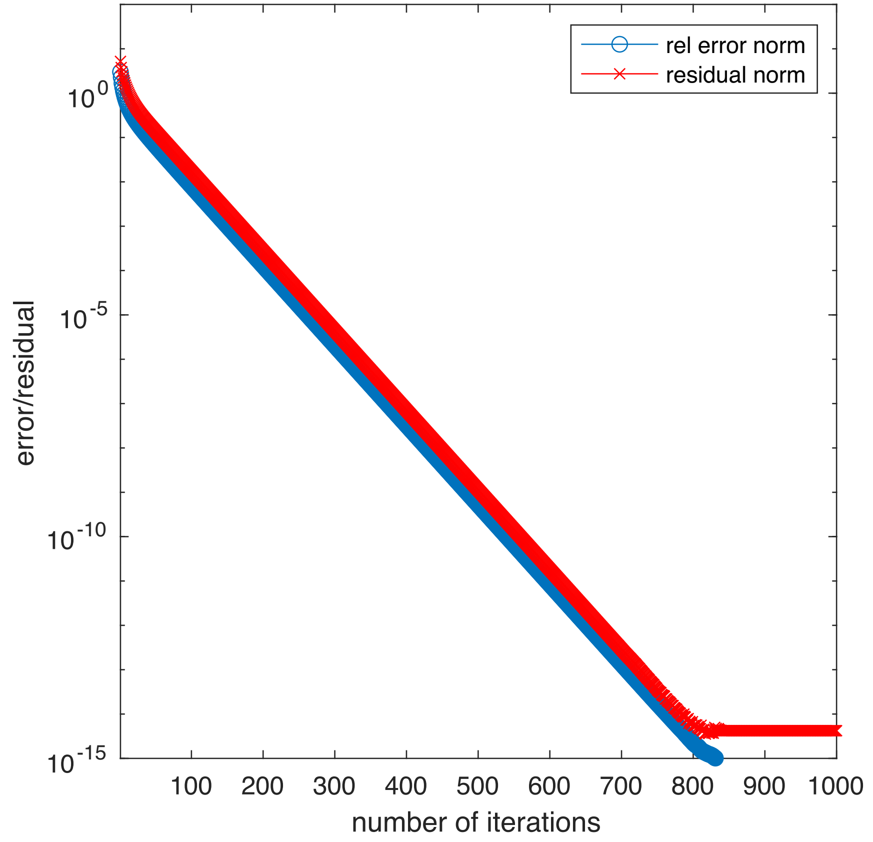


actual
$1 - h_l^4$
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
$1 - \rho(I - A)$
0
50
100
150
200
250
300
number of grid points in one dimension

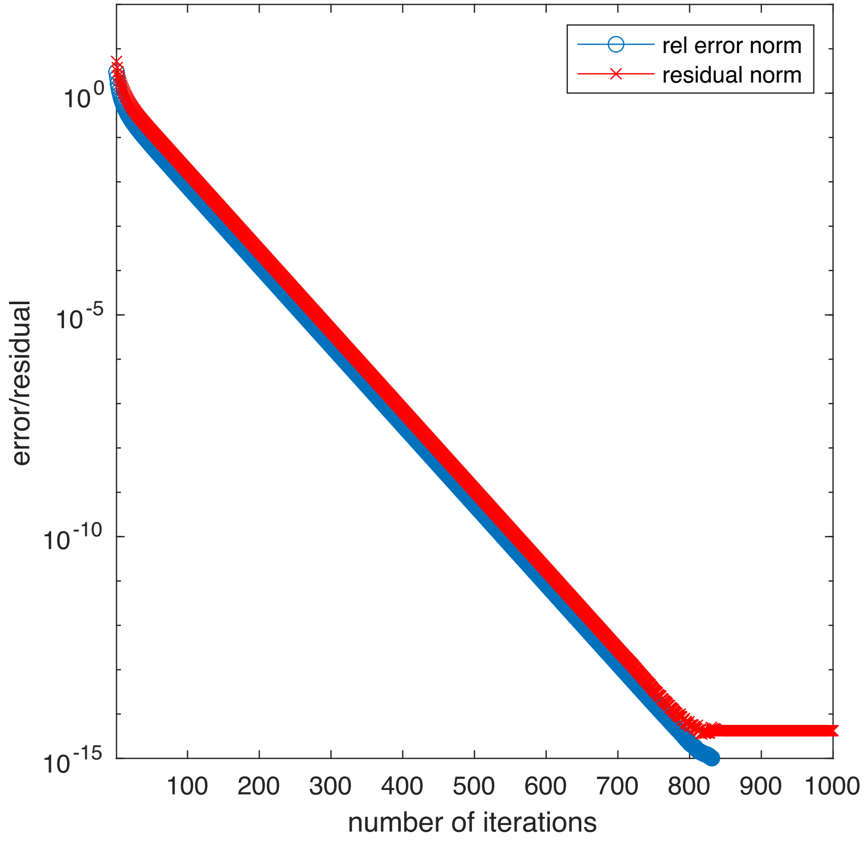
Running Jacobi on a Poisson matrix with $N = 10$ and $N = 100$ grid points in each dimension:

```
A = gallery('poisson', N)*(1/N^2);
b = randn(N^2, 1);
x = randn(N^2, 1);
xTrue = A\b;
M = speye(N^2) - spdiags(1./diag(A), 0, N^2, N^2)*A;
c = spdiags(1./diag(A), 0, N^2, N^2)*b;

hist = [];
for iter=1:1000
    x = M*x + c;
    hist(iter, 1) = norm(x - xTrue)/norm(x);
    hist(iter, 2) = norm(A*x - b);
end
```

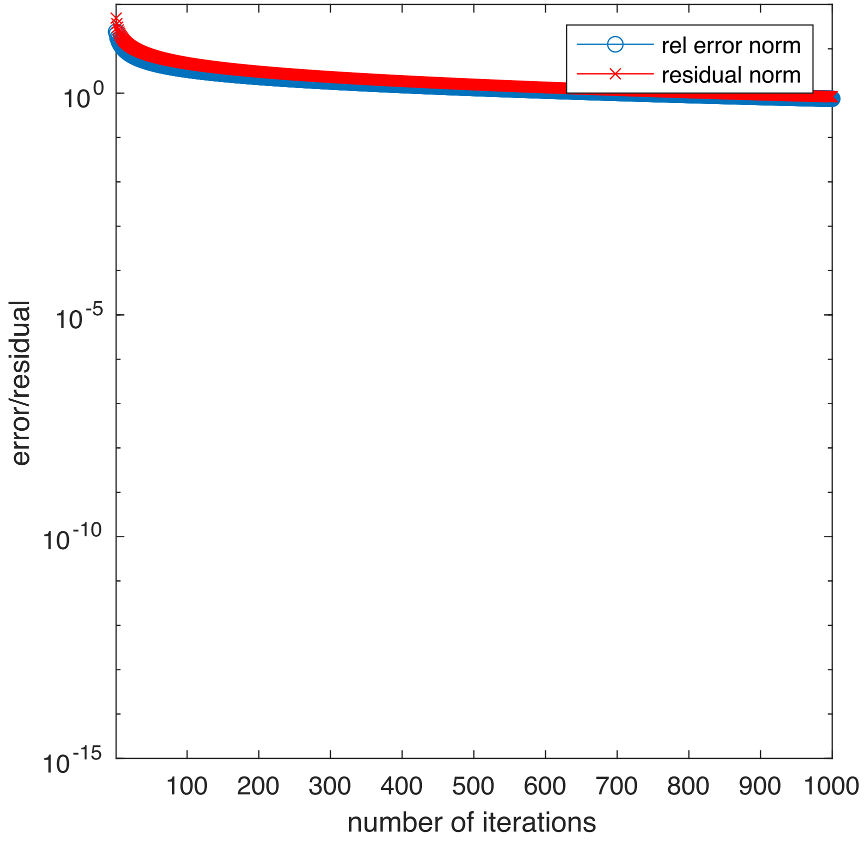
$N = 10$ $\qquad$ $N = 100$



$N = 10$

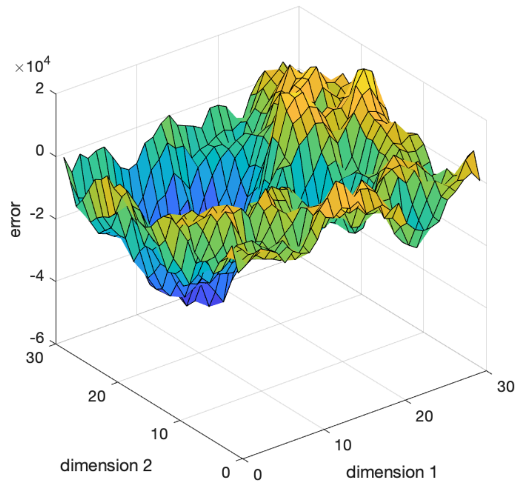

rel error norm
residual norm
error/residual
10^{0}
10^{-5}
10^{-10}
10^{-15}
100 200 300 400 500 600 700 800 900 1000
number of iterations

$N = 100$

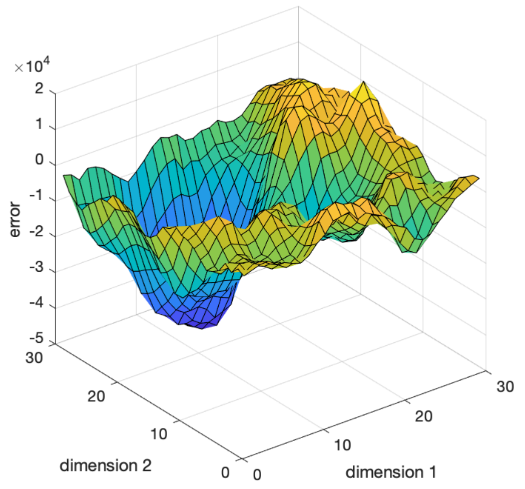

rel error norm
residual norm
error/residual
10^{0}
10^{-5}
10^{-10}
10^{-15}
100 200 300 400 500 600 700 800 900 1000
number of iterations

First few iterations of Jacobi relaxation

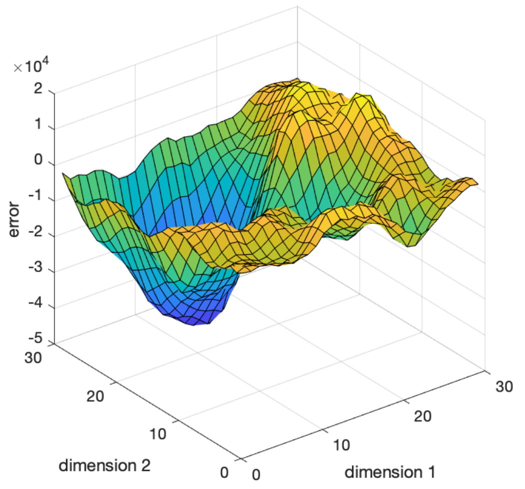
```

N = 30; A = gallery('poisson', N)*(1/N^2);
[X, Y] = meshgrid(linspace(1, N, N), linspace(1, N, N));
b = 10*randn(N^2, 1);
x = randn(N^2, 1);
xTrue = A\b;

for i=1:5
    x = (speye(N^2) - spdiags(1./diag(A), 0, N^2, N^2)*A)*x + ...
                spdiags(1./diag(A), 0, N^2, N^2)*b;
end
```



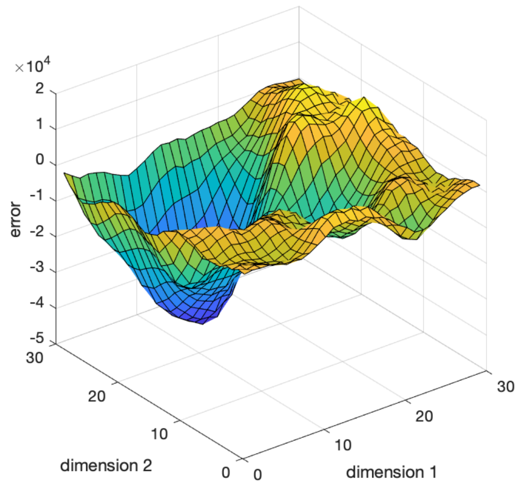
iteration 0



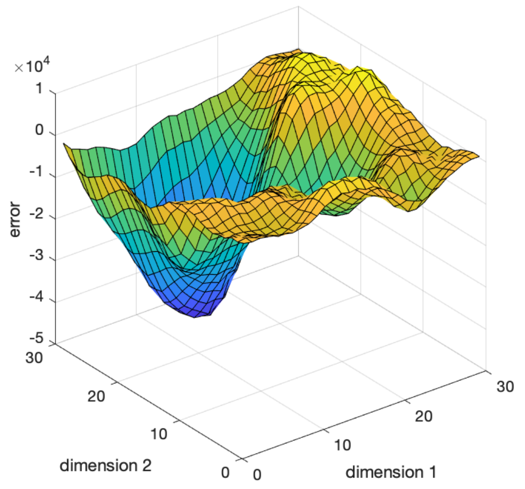
iteration 1



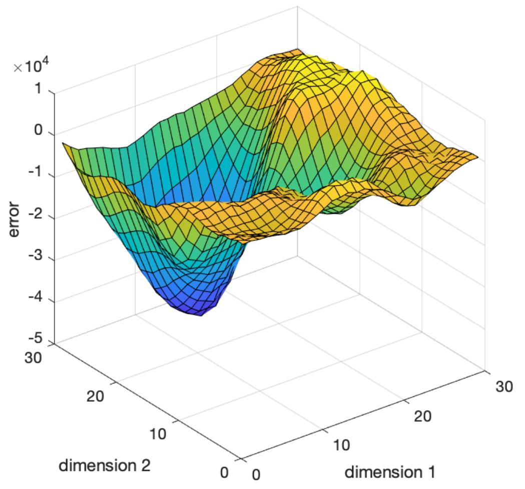
iteration 2



iteration 3

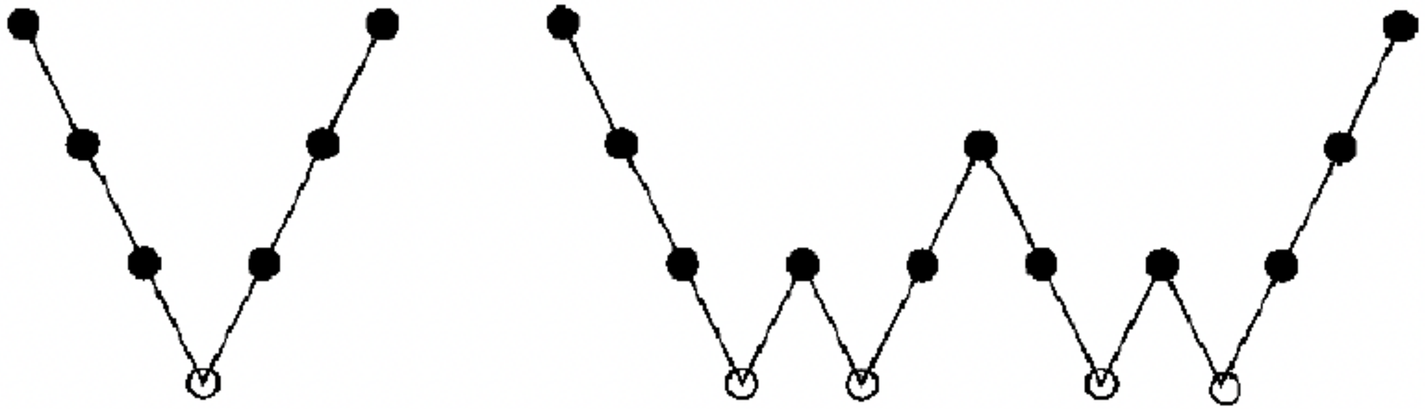


iteration 4



iteration 5

- Relaxation methods such as Jacobi and Gauss Seidel have a smoothing effect on the error
- Even if $\rho \approx 1$, only a few iterations are necessary to obtain a smooth error; this means that high-frequency error is reduced very quickly whereas the low-frequency error is reduced slowly
- Multigrid methods exploit this effect and represent the smoothed error on a coarse grid, where it becomes high-frequency error again, which can be smoothed quickly



$\rightsquigarrow$ Multigrid methods are among the most efficient solvers for PDE problems $\rightsquigarrow$ NM 2 in Spring

Conjugate gradient method

In the following A is symmetric positive definite.

Formulate solving $Ax = b$ as an optimization problem: Define

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

and minimize

$$\min_{x \in \mathbb{R}^n} f(x)$$

Because A is positive definite, the function f is convex. It is sufficient to look at the gradient

$$\nabla f(x) = \frac{1}{2}A^T x + \frac{1}{2}Ax - b = Ax - b = -r(x) = 0 \iff Ax = b$$

What is the benefit of this point of view?

In the following A is symmetric positive definite.

Formulate solving $Ax = b$ as an optimization problem: Define

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

and minimize

$$\min_{x \in \mathbb{R}^n} f(x)$$

Because A is positive definite, the function f is convex. It is sufficient to look at the gradient

$$\nabla f(x) = \frac{1}{2} A^T x + \frac{1}{2} A x - b = Ax - b = -r(x) = 0 \iff Ax = b$$

What is the benefit of this point of view? We now can let loose all what we know about optimization to solve $Ax = b$

Our first try is applying the method of steepest descent in the direction of the negative gradient

$$-\nabla f = r$$

which happens to be the residual

$$
\begin{aligned}
r_k &= b - Ax_k \\
\alpha_k &= \frac{r_k^T r_k}{r_k^T A r_k} \\
x_{k+1} &= x_k + \alpha_k r_k
\end{aligned}
$$

The step length α_k minimizes $f(x_k + \alpha_k r_k)$ as a function of α_k $\rightsquigarrow$ board

$$f(x) = \frac{1}{2} x^T A x - b^T x, \quad \nabla f(x) = Ax - b$$

$$\min_{\alpha_k} f(x_k + \alpha_k r_k) = \frac{1}{2}(x_k + \alpha_k r_k)^T A (x_k + \alpha_k r_k) - b^T (x_k + \alpha_k r_k)$$

$$\frac{\partial}{\partial \alpha_k} f(x_k + \alpha_k r_k) = r_k^T \left[\nabla f(x_k + \alpha_k r_k) \right]$$

$$= r_k^T \left[A x_k + A \alpha_k r_k - b \right] \overset{!}{=} 0$$

$$r_k^T A x_k + \alpha_k r_k^T A r_k - r_k^T b \overset{!}{=} 0$$

$$r_k^T (\underbrace{A x_k - b}_{-r_k}) + \alpha_k r_k^T A r_k \overset{!}{=} 0$$

$$\Rightarrow \alpha_k^* = \frac{r_k^T r_k}{r_k^T A r_k}$$

For steepest descent, if A is spd, we obtain

$$\|x^* - x_k\|_A \leq \left(\frac{\kappa_2(A) - 1}{\kappa_2(A) + 1} \right)^k \|x^* - x_0\|_A \, ,$$

where $\langle x, y \rangle_A = x^T A y$ and $\| \cdot \|_A = \sqrt{\langle \cdot, \cdot \rangle_A}$.

Proof $\leadsto$ board

Spd A:

$$\|x_k - x^*\|_A \le \left(\frac{k_2(A)-1}{k_2(A)+1}\right)^k \|x_k - x_0\|_A$$

$$x_{k+1}(\alpha) = x_k + \alpha\, r_k \qquad \left(r_k = b - Ax_k = Ax^* - Ax_k\right)$$

$$= x_k + \alpha \underbrace{A(x^* - x_k)}$$

$$\underbrace{x^* - x_{k+1}(\alpha)}_{\substack{\| \\ e_{k+1}(\alpha)}} = x^* - x_k - \alpha A(x^* - x_k) =$$

$$= (I - \alpha A)\underbrace{(x^* - x_k)}_{e_k}$$

$$e_{k+1}(\alpha) = (I - \alpha A)e_k$$

$$\|e_{k+1}(\alpha)\|_A^2 = e_k^T (I - \alpha A)^T A (I - \alpha A) e_k$$

$$\boxed{e_k = \sum_{i=1}^{n} \sigma_i z_i}$$