Step 7: Solve $A\vec{\varphi} = \vec{b}$: Iterative Methods

Successive Over-Relaxation (SOR)

- Idea: try to reduce largest eigenvalue of $(A_1^{-1}A_2)$ as much as possible
- Start with Gauss-Seidel ($A_1 = D - L$ and $A_2 = U$)

$$(D - L)\vec{\varphi}^{(k+1)} = U\vec{\varphi}^{(k)} + \vec{b} \qquad (i)$$

define $\vec{d} = \vec{\varphi}^{(k+1)} - \vec{\varphi}^{(k)} \quad \Rightarrow \quad \vec{\varphi}^{(k+1)} = \vec{\varphi}^{(k)} + \vec{d}$

$$\vec{\varphi}^{(k+1)} = \vec{\varphi}^{(k)} + \omega\vec{d}$$

ω > 1: over-relaxation
ω < 1: under-relaxation
ω = 1: Gauss-Seidel

Step 1: calculate $d_{i,j}$ with Gauss-Seidel (using updated values from step 2)

Step 2: calculate $\vec{\varphi}_{i,j}^{(k+1)}$ with SOR: $\vec{\varphi}_{i,j}^{(k+1)} = \vec{\varphi}_{i,j}^{(k)} + \omega d_{i,j}$

Step 7: Solve $A\vec{\varphi} = \vec{b}$: Iterative Methods

Successive Over-Relaxation (SOR)

Step 1: calculate $d_{i,j}$ with Gauss-Seidel (using updated values from step 2)

Step 2: calculate $\vec{\varphi}_{i,j}^{(k+1)}$ with SOR: $\vec{\varphi}_{i,j}^{(k+1)} = \vec{\varphi}_{i,j}^{(k)} + \omega d_{i,j}$

Step 1: Gauss Seidel

$$(D - L)\widetilde{\vec{\varphi}}^{(k+1)} = U\vec{\varphi}^{(k)} + \vec{b}$$

but use step 2 updated values where possible

$$D\widetilde{\vec{\varphi}}^{(k+1)} = L\vec{\varphi}^{(k+1)} + U\vec{\varphi}^{(k)} + \vec{b}$$

Step 2: SOR

$$\vec{\varphi}^{(k+1)} = \vec{\varphi}^{(k)} + \omega\left(\widetilde{\vec{\varphi}}^{(k+1)} - \vec{\varphi}^{(k)}\right)$$

What are the eigenvalues?

$$\vec{\varphi}^{(k+1)} = \vec{\varphi}^{(k)} + \omega\left(D^{-1}L\vec{\varphi}^{(k+1)} + D^{-1}U\vec{\varphi}^{(k)} + D^{-1}\vec{b} - \vec{\varphi}^{(k)}\right)$$

Step 7: Solve $A\vec{\varphi} = \vec{b}$: Iterative Methods

Successive Over-Relaxation (SOR)

$$\vec{\varphi}^{(k+1)} = \vec{\varphi}^{(k)} + \omega\left(D^{-1}L\vec{\varphi}^{(k+1)} + D^{-1}U\vec{\varphi}^{(k)} + D^{-1}\vec{b} - \vec{\varphi}^{(k)}\right)$$

rearrange

$$\left(I - \omega D^{-1}L\right)\vec{\varphi}^{(k+1)} = \left[\left(1-\omega\right)I + \omega D^{-1}U\right]\vec{\varphi}^{(k)} + \omega D^{-1}\vec{b}$$

$$\vec{\varphi}^{(k+1)} = \underbrace{\left(I - \omega D^{-1}L\right)^{-1}\left[\left(1-\omega\right)I + \omega D^{-1}U\right]}_{= G_{SOR}}\vec{\varphi}^{(k)} + \underbrace{\left(I - \omega D^{-1}L\right)^{-1}\omega D^{-1}}_{= B_{SOR}}\vec{b}$$

$$\vec{\varphi}^{(k+1)} = G_{SOR}\vec{\varphi}^{(k)} + B_{SOR}\vec{b}$$

- What are the eigenvalues of G_{SOR}?

Step 7: Solve $A\vec{\varphi} = \vec{b}$: Iterative Methods

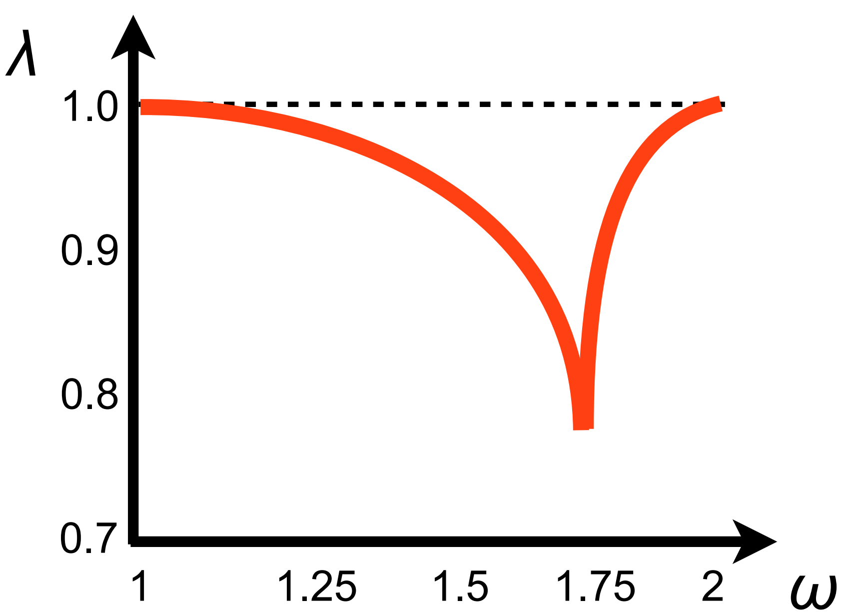
Successive Over-Relaxation (SOR)

- What are the eigenvalues of G_{SOR}?

  for the example: $$\sqrt{\lambda} = \frac{1}{2}\left(\pm|\mu|\omega \pm \sqrt{\mu^2\omega^2 - 4(\omega - 1)}\right)$$ μ: eigenvalues of Point Jacobi

➡ choose ω such that λ is minimum

$$\frac{d\lambda}{d\omega} = 0$$ unfortunately, this has no solution



⇒ minimum at $\frac{d\lambda}{d\omega} = \infty$ $$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \mu_{\max}^2}}$$

exact value depends on M,N ($\mu_{\max}$ depends on M,N)

for uniform meshes, one calculates ω_{opt} a-priori
otherwise, use numerical experiments
(usually $\omega \approx 1.7 \ldots 1.9$)

Step 7: Solve $A\vec{\varphi} = \vec{b}$: Iterative Methods

Successive Over-Relaxation (SOR)

- Is SOR always better than Gauss-Seidel, Point Jacobi?

  - from Linear Algebra:

$$\vec{\varphi}^{(k)} = c_1\lambda_1^k\vec{x}_1 + c_2\lambda_2^k\vec{x}_2 + \ldots c_n\lambda_n^k\vec{x}_n$$

$\vec{x}_i$: eigenvectors

$|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$

$$\vec{\varphi}^{(0)} = c_1\vec{x}_1 + c_2\vec{x}_2 + \ldots c_n\vec{x}_n$$

➡ even if $|\lambda_1|_{SOR} < |\lambda_1|_{GS}$, eigenvectors of SOR and GS are different

depending on initial guess, perhaps $c_i^{SOR} >> c_i^{GS}$

➡ SOR may be slower with some initial guesses
but in general, SOR will be faster

- How to code SOR?

Step 1: calculate $d_{i,j}$ with Gauss-Seidel (using updated values from step 2)

Step 2: calculate $\vec{\varphi}_{i,j}^{(k+1)}$ with SOR: $\vec{\varphi}_{i,j}^{(k+1)} = \vec{\varphi}_{i,j}^{(k)} + \omega d_{i,j}$

$$\varphi_{i,j}^{(k+1)} = \varphi_{i,j}^{(k)} + \omega \left(\widetilde{\varphi}_{i,j} - \varphi_{i,j}^{(k)} \right)$$

$$\widetilde{\varphi}_{i,j} = \frac{1}{4} \left(\varphi_{i,j-1}^{(k+1)} + \varphi_{i-1,j}^{(k+1)} + \varphi_{i+1,j}^{(k)} + \varphi_{i,j+1}^{(k)} \right) - \frac{1}{4} h^2 f_{i,j}$$

$$\varphi_{i,j}^{(k+1)} = \varphi_{i,j}^{(k)} + \omega \left(\frac{1}{4} \left(\varphi_{i,j-1}^{(k+1)} + \varphi_{i-1,j}^{(k+1)} + \varphi_{i+1,j}^{(k)} + \varphi_{i,j+1}^{(k)} \right) - \frac{1}{4} h^2 f_{i,j} - \varphi_{i,j}^{(k)} \right)$$

$$\texttt{phi(i,j)} = \texttt{phi(i,j)} +$$
$$\omega \left(\frac{1}{4} \left(\texttt{phi(i,j-1)} + \texttt{phi(i-1,j)} + \texttt{phi(i+1,j)} + \texttt{phi(i,j+1)} \right) - \frac{1}{4} h^2 \texttt{f(i,j)} - \texttt{phi(i,j)} \right)$$

Step 7: Solve $A\vec{\varphi} = \vec{b}$: Iterative Methods

Pre-conditioning

- Idea: pre-multiply system of equations by carefully constructed matrix, to get smaller eigenvalues of the iteration matrix
- Can be combined with any iterative method

Comment on implementation and boundary conditions

- Example: 1D Poisson equation

$$\frac{\partial^2 \phi}{\partial x^2} = f(x)$$

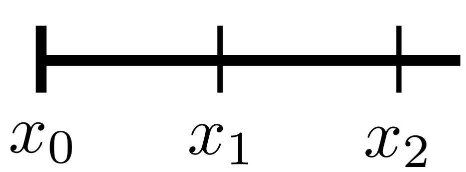
  - using 2nd-order central finite differences

$$\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{h^2} = f_i$$

  - iterate, for example using Point-Jacobi

$$\phi_i^{(k+1)} = \frac{1}{2}\left(\phi_{i+1}^{(k)} + \phi_{i-1}^{(k)}\right) - \frac{1}{2}h^2 f_i$$

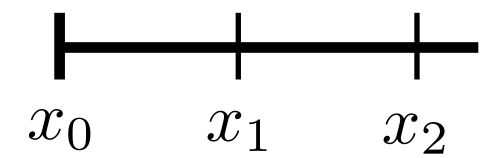
    - Problem: this won't work for points on the boundary (i=0 or i=M)



    - Need to apply appropriate boundary conditions!

Comment on implementation and boundary conditions

$$\phi_i^{(k+1)} = \frac{1}{2}\left(\phi_{i+1}^{(k)} + \phi_{i-1}^{(k)}\right) - \frac{1}{2}h^2 f_i$$



- Need to apply appropriate boundary conditions!
  - Dirichlet boundary condition:
    - value on boundary is known, so no need to solve the iterative equation for the boundary points
    - modifies the loop extend (instead of from 0 to M, 1 to M-1)
    - must have the boundary value stored in the initial guess
  - Neumann boundary condition:
    - approximate Neumann gradient by one-sided finite differences
    - solve finite difference equation for boundary value
    - use boundary value as in Dirichlet boundary condition
    - modifies the loop extend (instead of from 0 to M, 1 to M-1)
    - must update boundary value after each iteration (this is not fully consistent with Gauss Seidel, though)

Step 7: Solve $A\vec{\varphi} = \vec{b}$: Iterative Methods

Multigrid Acceleration

- we had: $A\vec{\varphi} = \vec{b} = \Delta^2 \vec{f}$
- from now on, bring Δ^2 from right-hand-side to left-hand-side and include in A

$$\left(\frac{1}{\Delta^2} A\right) \vec{\varphi} = \vec{f} \quad \rightarrow \quad A\vec{\varphi} = \vec{f}$$

A now includes $1/\Delta^2$

- exact: $A\vec{\varphi} = \vec{f}$ *(i)*
- but we only know an estimate: $A\vec{\varphi}^{(n)} = \vec{f} - \vec{r}^{(n)}$ *(ii)*

$\vec{r}$: residual

- take *(i)* - *(ii)*:

$$A \underbrace{\left(\vec{\varphi} - \vec{\varphi}^{(n)}\right)}_{\vec{\epsilon}^{(n)}} = \vec{r}^{(n)} \quad \Rightarrow \quad A\vec{\epsilon}^{(n)} = \vec{r}^{(n)}$$

error

➡ as $\vec{\epsilon}^{(n)} \rightarrow 0 \quad \Rightarrow \quad \vec{r}^{(n)} \rightarrow 0$

➡ reducing the error is equivalent to reducing the residual

Example #1:

$$\frac{d^2\varphi}{dx^2} = \sin(k\pi x) \qquad 0 \le x \le 1 \qquad \varphi(0) = \varphi(1) = 0$$

- exact solution is: $\varphi(x) = -\frac{1}{k^2\pi^2}\sin(k\pi x)$

- define mesh: $M+1$ points

$$h = \frac{1}{M} \qquad x_i = ih$$

- use 2nd-order central differences

$$\frac{1}{h^2}\varphi_{i+1} - \frac{2}{h^2}\varphi_i + \frac{1}{h^2}\varphi_{i-1} = f_i = \sin(k\pi ih)$$

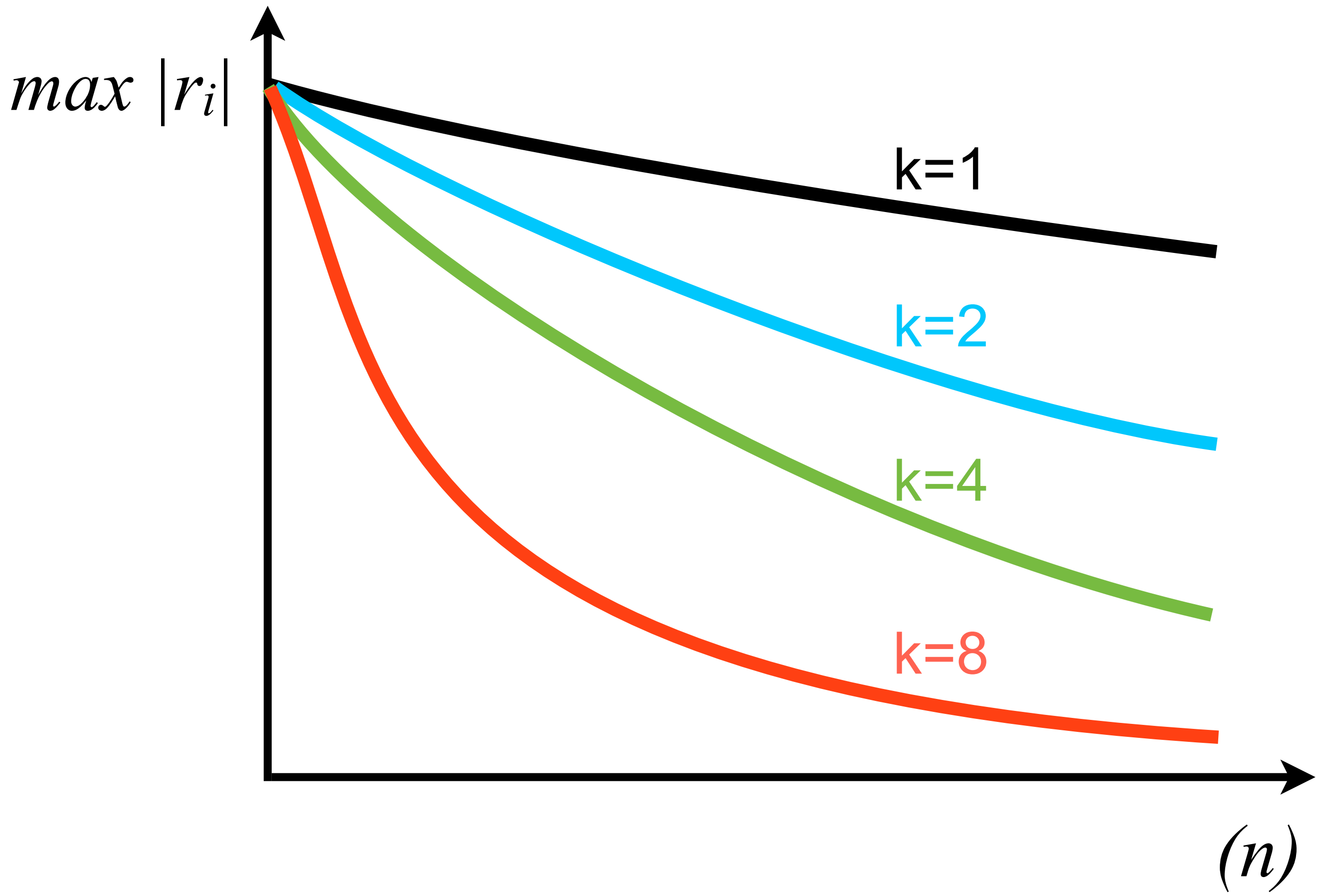
- use $\vec{\varphi} = \vec{0}$ as initial guess $\Rightarrow \quad \vec{r}^{(0)} = \vec{f} - A\vec{\varphi}^{(0)} = \vec{f}$

- solve with Gauss-Seidel

code example MG1

Example #1:

$$\frac{d^2\varphi}{dx^2} = \sin\left(k\pi x\right) \qquad 0 \le x \le 1 \qquad \varphi(0) = \varphi(1) = 0$$



$\Rightarrow$ larger k converge faster

Example #2:

code example MG2 & MG3

$$\frac{d^2\varphi}{dx^2} = \frac{1}{2}\left[\sin\left(\pi x\right) + \sin\left(16\pi x\right)\right] \qquad 0 \leq x \leq 1 \qquad \varphi(0) = \varphi(1) = 0$$

- Key observation:
  - rapidly varying parts (large k) converge much faster than slowly varying parts (small k)

- BUT: a slowly varying function on a fine mesh is a rapidly varying function on a coarse mesh!