AEE471/MAE561 Computational Fluid Dynamics

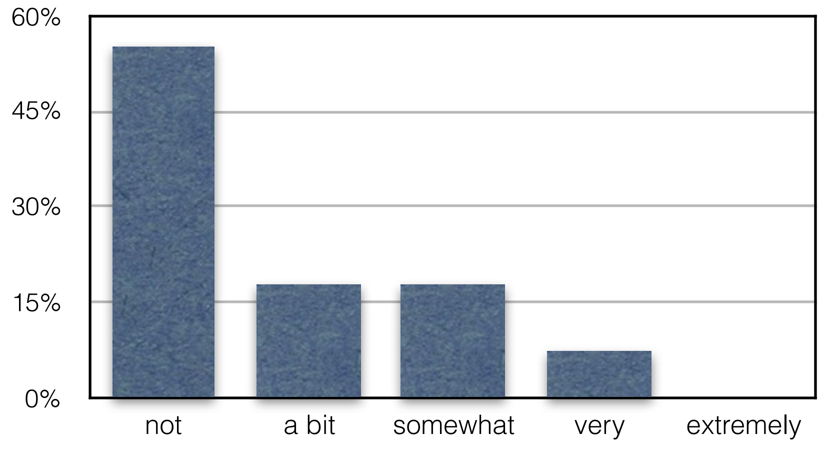
• Muddiest Points from Class 02/01

- *"How much of these derivations do we have to know with all the eigenvalues and iterative methods? Will we have to do the whole process like in the slides or just the end formulas?"*
  - You are expected to know and work with the index formulas
  - The linear algebra derivations are for background and the spectral radius, only

- *"let's assume for simplicity: delta(x)=delta(y)=(delta). What is (delta) on right hand side. How is it taken towards right hand side and put before f(i,j).*
  - The Δ in that formula is simply the mesh spacing and it's brought to the right hand side by multiplying both sides by Δ^2

- *"Do these equations and methods change when time becomes a variable?"*
  - if time is replacing one of the independent variables, the methods stay identical

- *"I got a little lost at the point Jacobi method, and its calculation in coding."*
- *"I guess in general I am a little unsure about the code implementation of the Point-Jacobi method […]"*
- *"Could you be more explicit with how to code some of the solutions for the poisson equation."*
  - since I ran out of time last class, this is next….



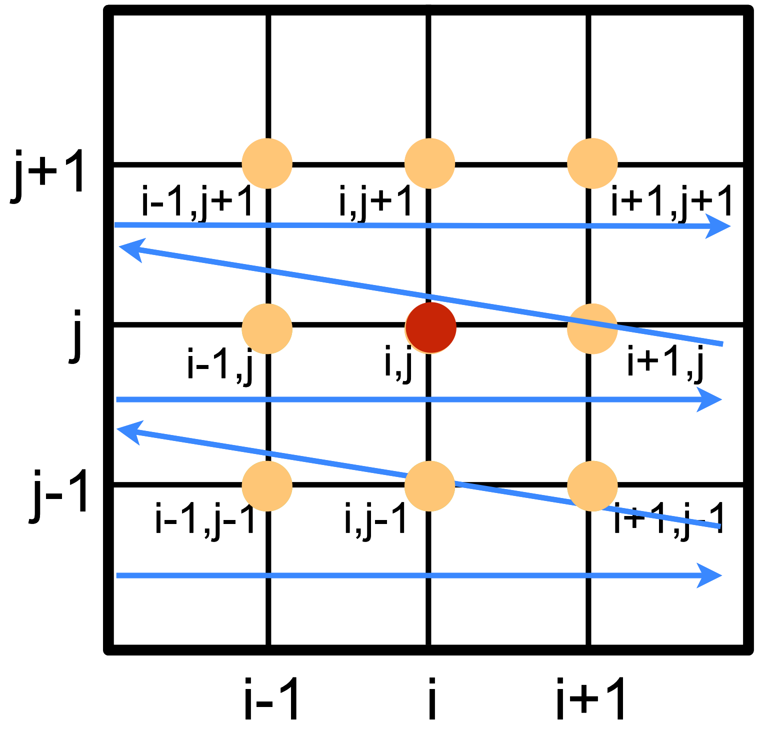
Step 7: Solve $A\vec{\varphi} = \vec{b}$: Iterative Methods

Gauss-Seidel Method

PJ: $\varphi_{i,j}^{(k+1)} = \frac{1}{4}\left(\varphi_{i-1,j}^{(k)} + \varphi_{i+1,j}^{(k)} + \varphi_{i,j-1}^{(k)} + \varphi_{i,j+1}^{(k)}\right) - \frac{1}{4}b_{i,j}$

- In Point-Jacobi, we use only $\vec{\varphi}^{(k)}$ to calculate $\vec{\varphi}^{(k+1)}$
- Idea: Why don't we use $\vec{\varphi}^{(k+1)}$ instead of $\vec{\varphi}^{(k)}$ for nodes we have already calculated?

  for our sorting, for $\vec{\varphi}_{i,j}^{(k+1)}$ this would be $\vec{\varphi}_{i,j-1}^{(k+1)}$ and $\vec{\varphi}_{i-1,j}^{(k+1)}$



$$\Rightarrow \quad \varphi_{i,j}^{(k+1)} = \frac{1}{4}\left(\varphi_{i-1,j}^{(k+1)} + \varphi_{i+1,j}^{(k)} + \varphi_{i,j-1}^{(k+1)} + \varphi_{i,j+1}^{(k)}\right) - \frac{1}{4}b_{i,j}$$

➡ to code: use just one storage vector!

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

Gauss-Seidel Method

- How to code?
    - use 1 storage vector that gets overwritten

```
while not converged (or loop from k=1 to #requested iterations)
   loop from j = 1 to N-1
      loop from i = 1 to M-1
        phi(i,j) = 0.25*(phi(i-1,j)+phi(i+1,j)+...
      end loop i
   end loop j
end while
```

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

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

Gauss-Seidel Method

L: - lower triangular part of A
U: - upper triangular part of A
Note: $A \neq LU$

- in matrix notation: $A_1 = D - L$ and $A_2 = U$
  - What are the eigenvalues of $\left(A_1^{-1} A_2\right)$?

$$\lambda_{m,n} = \frac{1}{4}\left[\cos\left(\frac{m\pi}{M}\right) + \cos\left(\frac{n\pi}{N}\right)\right]$$

$$m = 1, \ldots, M-1$$
$$n = 1, \ldots, N-1$$

  - factor 2 faster than Point Jacobi (and need only half the storage!)

PJ: $\lambda_{m,n} = \frac{1}{2}\left[\cos\left(\frac{m\pi}{M}\right) + \cos\left(\frac{n\pi}{N}\right)\right]$

AEE471/MAE561 Computational Fluid Dynamics

- Find the error $f(x) = x^2 \sin 2x$

calculate f' using second-order central differences & first-order one-sided differences at the boundaries

M	L_{oo}	L_1	L_2	order L_{oo}	order L_1	order L_2
4	8.6495E+00	2.8059E+00	4.1037E+00			
8	4.3229E+00	8.2479E-01	1.5102E+00	1.001	1.766	1.442
16	2.1145E+00	2.3354E-01	5.4596E-01	1.032	1.820	1.468
32	1.0393E+00	6.7400E-02	2.0848E-01	1.025	1.793	1.389
64	5.3528E-01	2.0900E-02	9.2200E-02	0.957	1.691	1.176
128	5.1374E-01	7.1600E-03	5.0500E-02	0.059	1.543	0.868
256	5.0332E-01	2.7600E-03	3.2400E-02	0.030	1.378	0.642
512	4.9821E-01	1.1700E-03	2.2200E-02	0.015	1.235	0.547
1024	4.9567E-01	5.3400E-04	1.5500E-02	0.007	1.134	0.515

If there's a bug in the code, where is it most likely located?

A: finite difference formula in the interior — show of hands

B: finite difference formula at the boundary — discuss

C: range of loop over the mesh is incorrect — show of hands

D: no idea

AEE471/MAE561 Computational Fluid Dynamics

- Find the error

$f(x) = x^2 \sin 2x$

calculate f’ using second-order central differences & first-order one-sided differences at the boundaries

M	L_{oo}	L_1	L_2	order L_{oo}	order L_1	order L_2
4	8.6495E+00	3.6100E+00	4.5992E+00			
8	1.2105E+01	4.5787E+00	6.0783E+00	-0.485	-0.343	-0.402
16	2.4056E+01	1.0123E+01	1.3435E+01	-0.991	-1.145	-1.144
32	4.8143E+01	2.1921E+01	2.8115E+01	-1.001	-1.115	-1.065
64	9.6303E+01	4.5612E+01	5.7250E+01	-1.000	-1.057	-1.026
128	1.9266E+02	9.2921E+01	1.1539E+02	-1.000	-1.027	-1.011
256	3.8532E+02	1.8748E+02	2.3162E+02	-1.000	-1.013	-1.005
512	7.7065E+02	3.7657E+02	4.6404E+02	-1.000	-1.006	-1.002
1024	1.5413E+03	7.5472E+02	9.2885E+02	-1.000	-1.003	-1.001

If there’s a bug in the code, where is it most likely located?

A: finite difference formula in the interior — show of hands

B: finite difference formula at the boundary — discuss

C: range of loop over the mesh is incorrect — show of hands

D: no idea

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

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

$\omega > 1$: over-relaxation
$\omega < 1$: over-relaxation
$\omega = 1$: 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

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

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

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

- What are the eigenvalues of G_{SOR}?

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

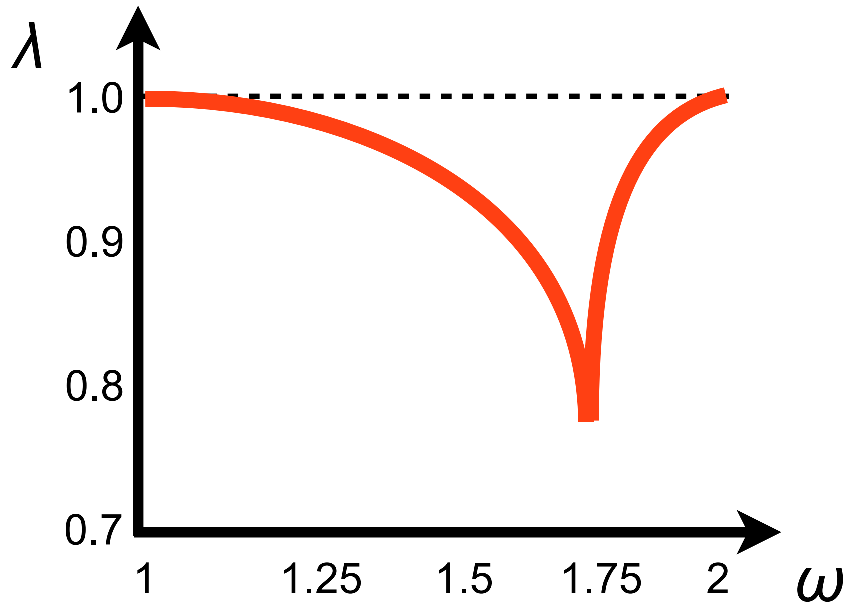
Successive Over-Relaxation (SOR)

- What are the eigenvalues of G_{SOR}?

$$\sqrt{\lambda} = \frac{1}{2}\left(\pm|\mu|\omega \pm \sqrt{\mu^2\omega^2 - 4(\omega-1)}\right)$$

μ: eigenvalues of Point Jacobi

➡ choose ω such that λ is minimum: $\frac{d\lambda}{d\omega} = 0$ unfortunately, this has no solution



$\Rightarrow$ minimum at $\frac{d\lambda}{d\omega} = \infty$

$$\boxed{\omega = \frac{2}{1+\sqrt{1-\rho_{PJ}^2}}}$$

- Recall

1D: $\rho_{PJ} = \cos\left(\frac{\pi}{M}\right)$ 2D: $\rho_{PJ} = \frac{1}{2}\left[\cos\left(\frac{\pi}{M}\right) + \cos\left(\frac{\pi}{N}\right)\right]$

- Consequence: ω depends on M,N because of ρ_{PJ}

- for uniform meshes, one calculates ω a-priori
- for non-uniform meshes, use numerical experiments starting with $\omega \approx 1.7\ldots1.9$

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:

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

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

$\vec{x}_i$: eigenvectors

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

➡ 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?

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

calculate $d_{i,j}$ with Gauss-Seidel (using SOR updated values) $d_{i,j} = \widetilde{\varphi}_{i,j} - \varphi_{i,j}^{(k)}$

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

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

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

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

(2D)

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

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

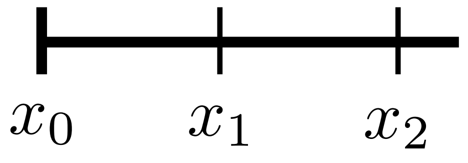
  - using 2nd-order central finite differences

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

  - iterate, for example using Point-Jacobi

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

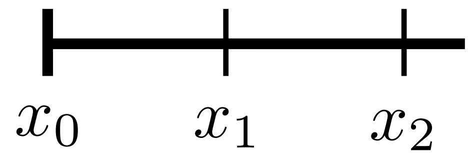
    - Problem: this won't work for points on the boundary (i=0 or i=M)



    - 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 do 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 do 1 to M-1)
    - must update boundary value after each iteration (this is not fully consistent with Gauss Seidel, though)