Parallel Algebraic Multigrid Methods for Fusion and Higher Order PDEs

Sophie Boileau, Atmik Das, **Kellen Kanarios**, Lucia Krajčoviechová

Mentor: Dr. Jean-Michel Maldague
Sponsors: Dr. Robert Falgout, Dr. Wayne Mitchell, Dr. Daniel Osei-Kuffuor

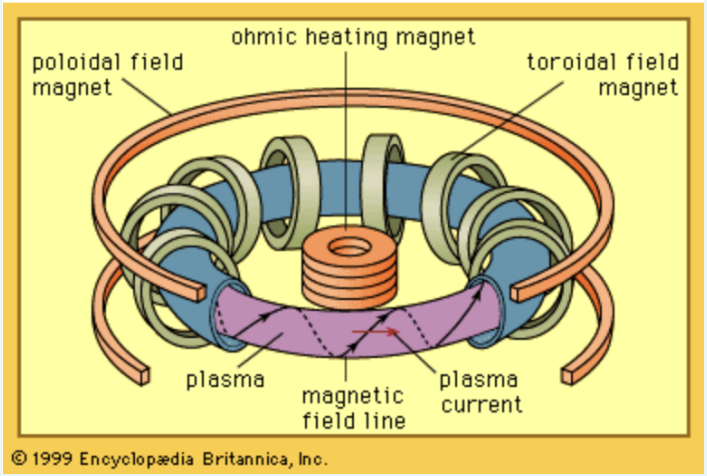
This material is based upon work jointly supported by Lawrence Livermore National Laboratory and the National Science Foundation under Grant No. 2015553.

Motivation

COGENT Proxy Problem

$$-au_{xx} - bu_{yy} + acu_{yyxx} = g, \quad b, c \gg a$$

- Thermonuclear fusion by magnetic confinement.



Preliminaries

- Transferring a PDE into a linear system

$$Au = f.$$

Preliminaries

- Transferring a PDE into a linear system

$$Au = f.$$

- Example: 2D Poisson equation

$$-u_{xx} - u_{yy} = f.$$

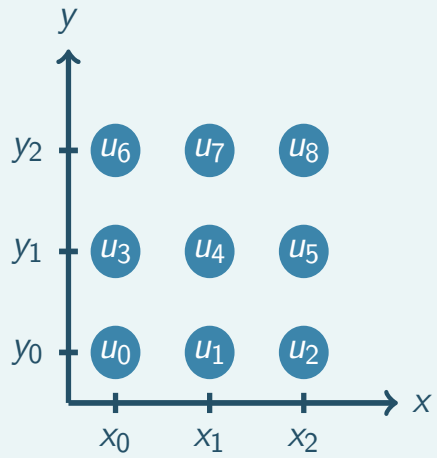
Preliminaries

- Transferring a PDE into a linear system

$$Au = f.$$

- Example: 2D Poisson equation

$$-u_{xx} - u_{yy} = f.$$

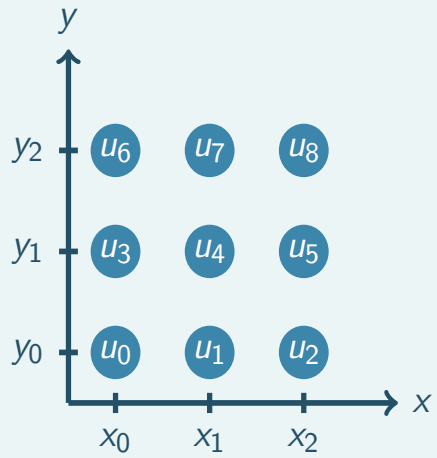


Its discretization (finite differences) on a 3×3 grid has

Preliminaries

- Transferring a PDE into a linear system

$$Au = f.$$



- Example: 2D Poisson equation

$$-u_{xx} - u_{yy} = f.$$

Its discretization (finite differences) on a 3×3 grid has

$$A = \underbrace{\begin{pmatrix} 4 & -1 & & -1 & & \\ -1 & 4 & -1 & & \ddots & \\ & -1 & 4 & \ddots & & \\ -1 & & \ddots & \ddots & \ddots & \end{pmatrix}}_{\text{matrix}} \quad \text{or} \quad A \sim \underbrace{\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}}_{\text{stencil}}.$$

Taxonomy of Multigrid

1. Iterative Methods
2. Multigrid
3. Algebraic Multigrid
4. Our Approach

Iterative Methods

- Consider the linear system

$$Au = b.$$

Iterative Methods

- Consider the linear system

$$Au = b.$$

- Iterative methods are of the form

$$u^{(k+1)} := u^{(k)} + M^{-1} r^{(k)}.$$

Iterative Methods

- Consider the linear system

$$Au = b.$$

- Iterative methods are of the form

$$u^{(k+1)} := u^{(k)} + M^{-1} r^{(k)}.$$

- Error propagation is

$$e^{(k+1)} = (I - M^{-1}A)e^{(k)} = \sum_{i=1}^{N} (I - M^{-1}\lambda_i) c_i^{(k)} v_i.$$

Iterative Methods

- Consider the linear system

$$Au = b.$$

- Iterative methods are of the form

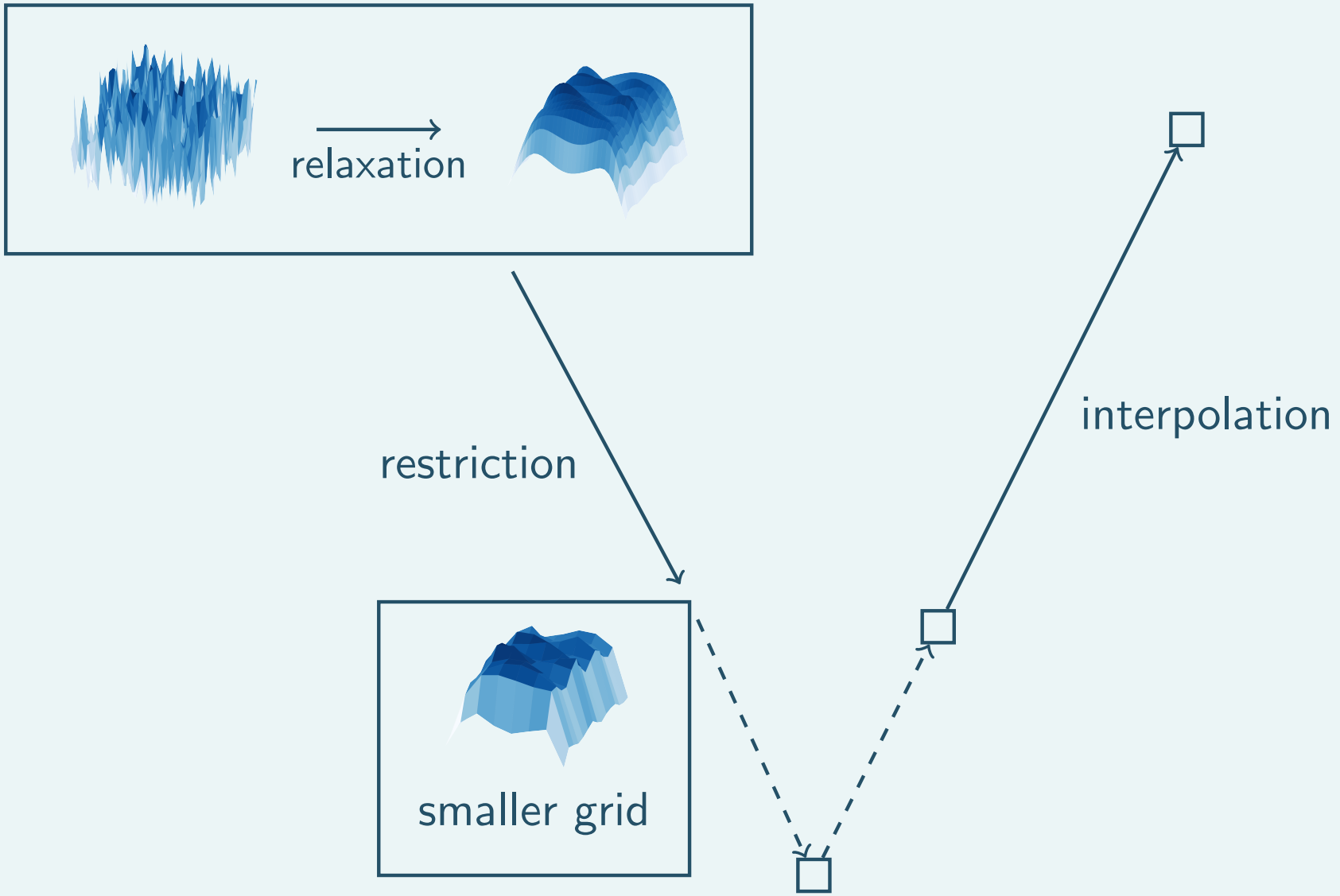
$$u^{(k+1)} := u^{(k)} + M^{-1} r^{(k)}.$$

- Error propagation is

$$e^{(k+1)} = (I - M^{-1}A)e^{(k)} = \sum_{i=1}^{N} (I - M^{-1}\lambda_i) c_i^{(k)} v_i.$$

- Small eigenvalues $\rightarrow$ **slow** convergence.

Multigrid



Constructing Smaller System

To Do

- Define restriction and interpolation operators R and P.
- Decide how to pick points for coarser grid.

Constructing Smaller System

To Do

- **Define restriction and interpolation operators R and P.**
- Decide how to pick points for coarser grid.

- Let P be interpolation (prolongation) and P^T restriction.

Constructing Smaller System

To Do

- **Define restriction and interpolation operators R and P.**
- Decide how to pick points for coarser grid.

- Let P be interpolation (prolongation) and P^T restriction.
- The coarse-grid operator is defined by the Galerkin procedure, $A_c = P^T A P$.

Constructing Smaller System

To Do

- **Define restriction and interpolation operators R and P.**
- Decide how to pick points for coarser grid.

- Let P be interpolation (prolongation) and P^T restriction.
- The coarse-grid operator is defined by the Galerkin procedure, $A_c = P^T A P$.
- This gives the "best" coarse-grid correction in the sense that the solution e_c of the coarse system satisfies

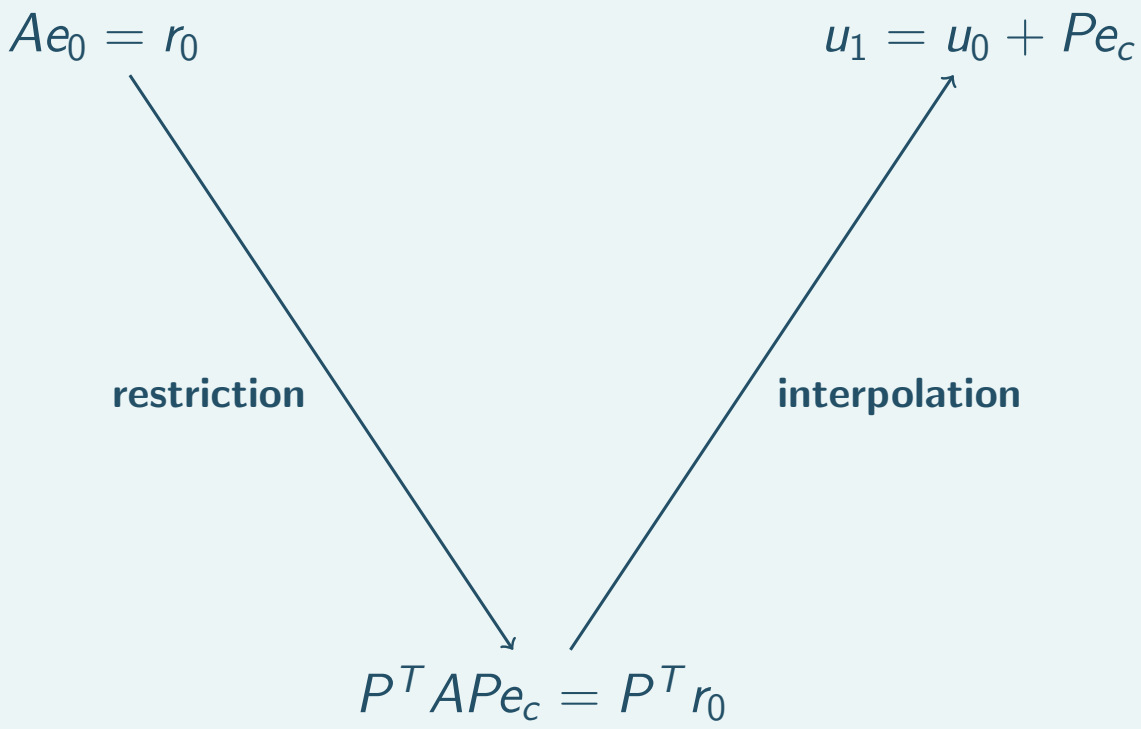
$$A_c e_c = P^T r,$$

$$e_c = \arg\min \|e - P e_c\|_A .$$

Constructing Smaller System

To Do

- Define restriction and interpolation operators R and P.
- Decide how to pick points for coarser grid.



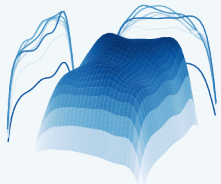
Constructing Smaller System

To Do

- Define restriction and interpolation operators R and P.
- **Decide how to pick points for coarser grid.**

- Error is assumed **smooth**

2D-Poisson



$$-u_{xx} - u_{yy} = f$$

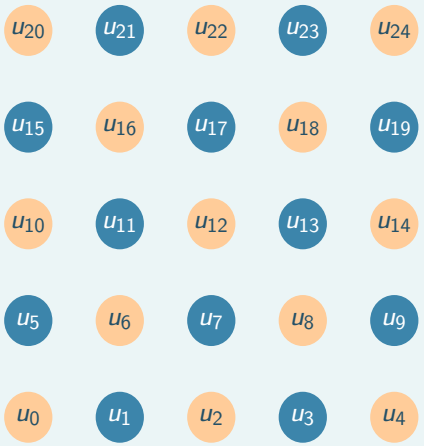
- Isotropic

Constructing Smaller System

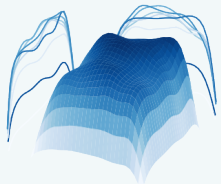
To Do

- Define restriction and interpolation operators R and P.
- **Decide how to pick points for coarser grid.**

- Error is assumed **smooth**



2D-Poisson



$$-u_{xx} - u_{yy} = f$$

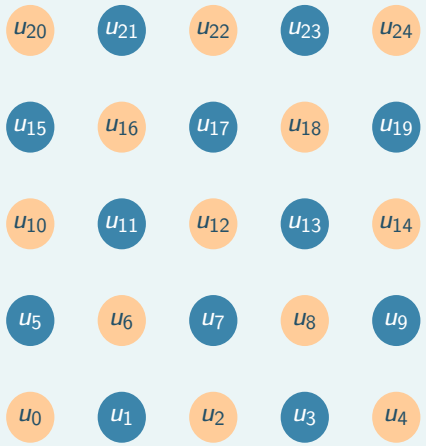
- Isotropic

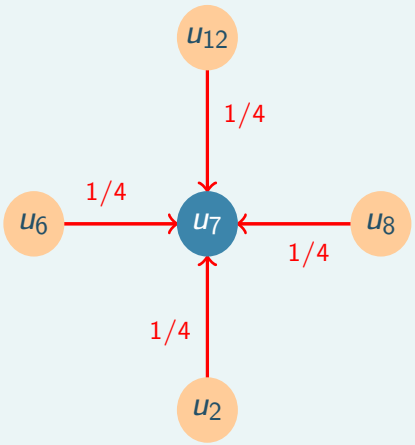
Constructing Smaller System

To Do

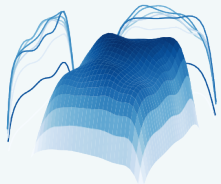
- Define restriction and interpolation operators R and P.
- **Decide how to pick points for coarser grid.**

- Error is assumed **smooth**, so local linear interpolation often works.





2D-Poisson



$$-u_{xx} - u_{yy} = f$$

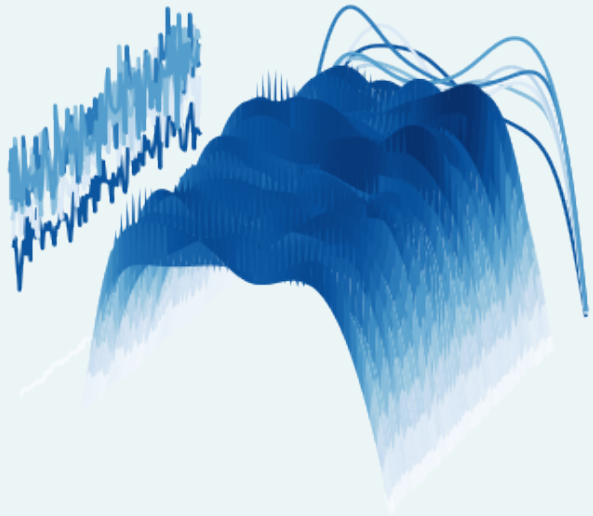
- Isotropic

Anisotropic Example

- Anisotropic, 5×5 grid

  $$-u_{xx} - \epsilon u_{yy} = f, \quad 0 < \epsilon \ll 1.$$

- $\sin y$ is an ϵ-eigenfunction.

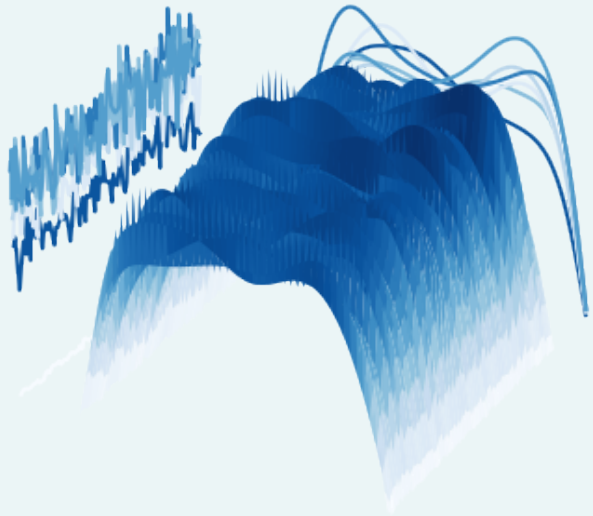


Anisotropic Example

- Anisotropic, 5×5 grid

  $$-u_{xx} - \epsilon u_{yy} = f, \quad 0 < \epsilon \ll 1.$$

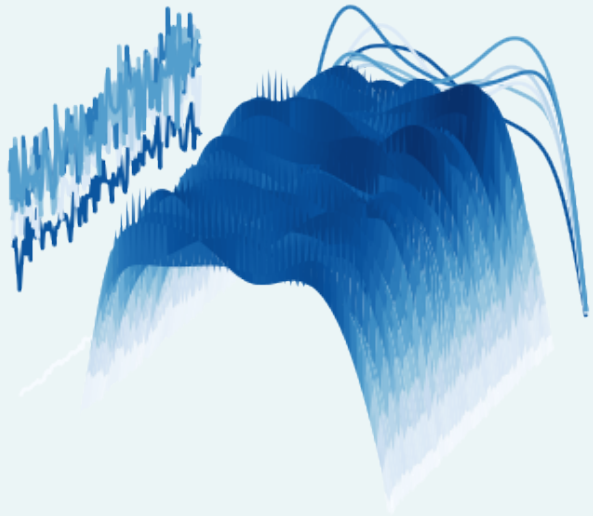
- $\sin y$ is an ϵ-eigenfunction.



WHAT DO WE DO?!?

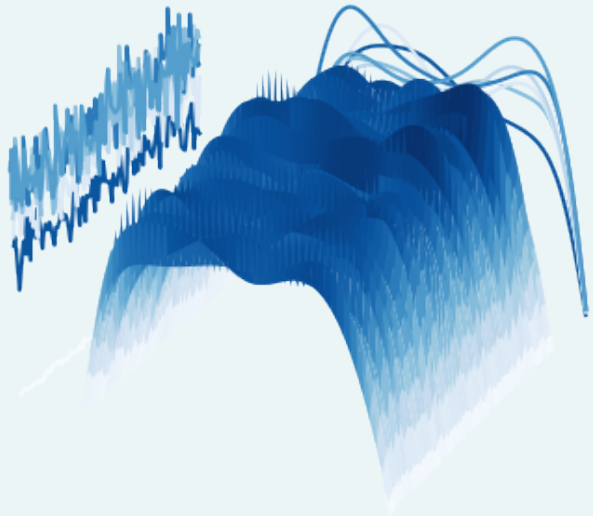
Anisotropic Example

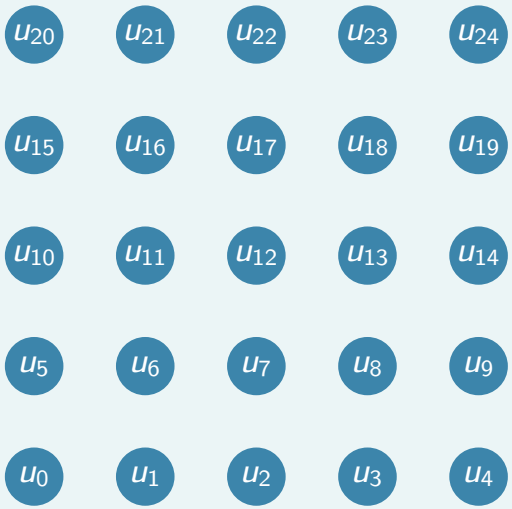
- Use knowledge of underlying PDE and grid.



Anisotropic Example

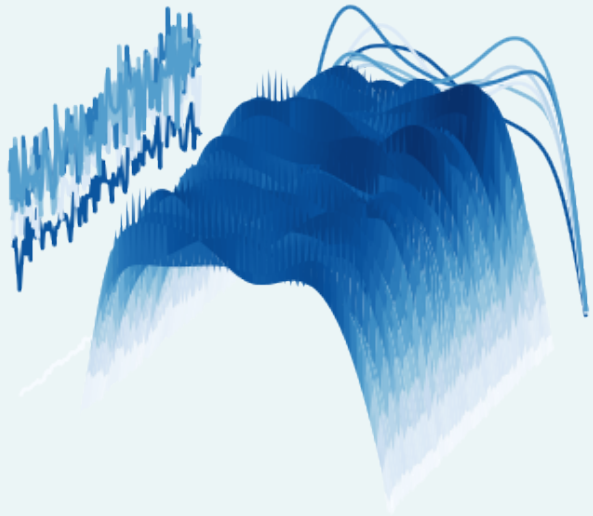
- Use knowledge of underlying PDE and grid.

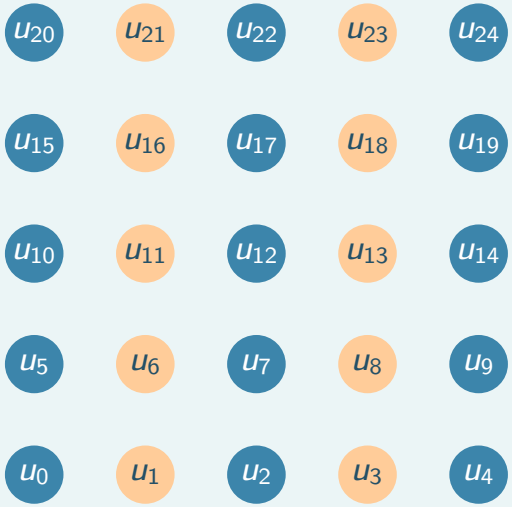




Anisotropic Example

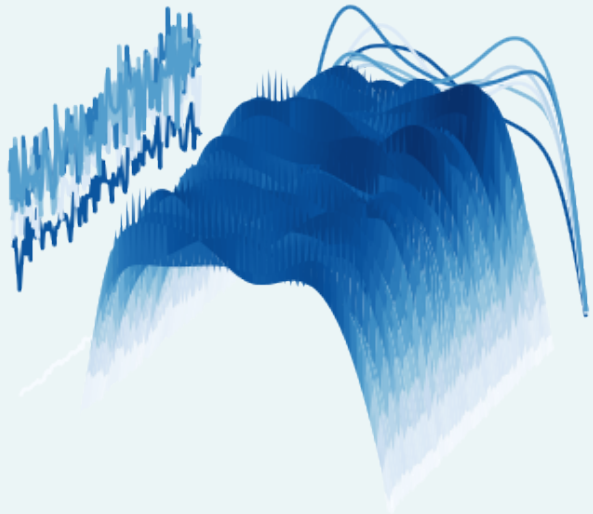
- Use knowledge of underlying PDE and grid.

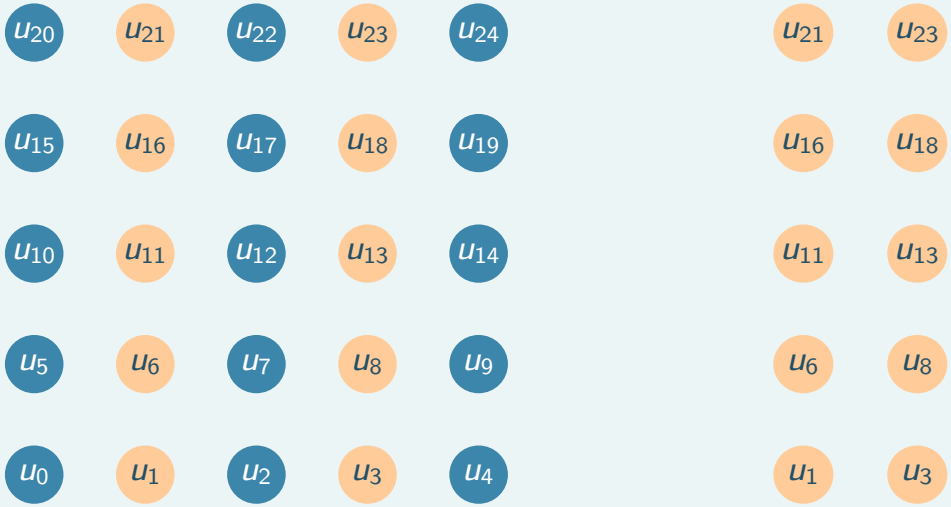




Anisotropic Example

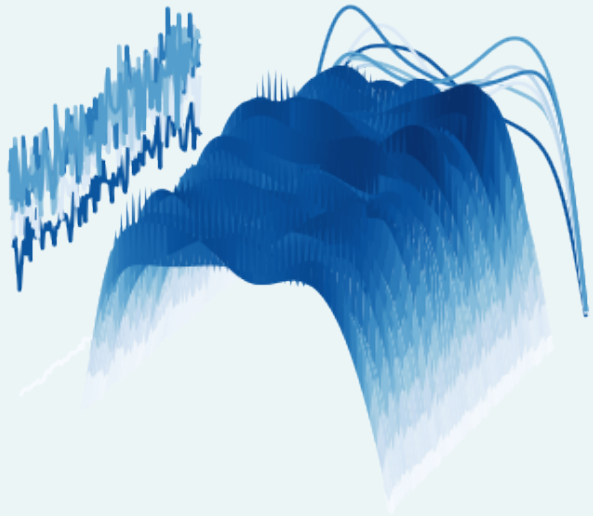
- Use knowledge of underlying PDE and grid.

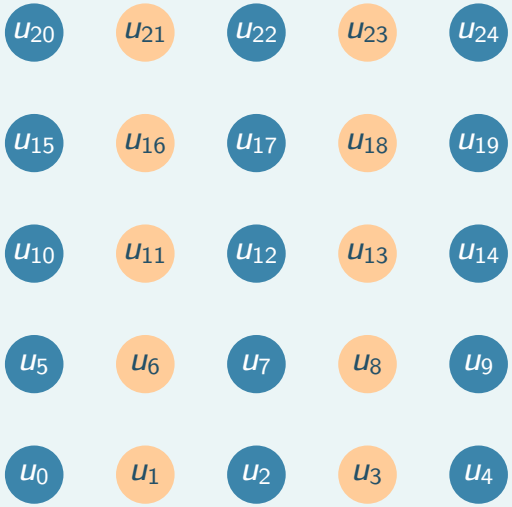




Anisotropic Example

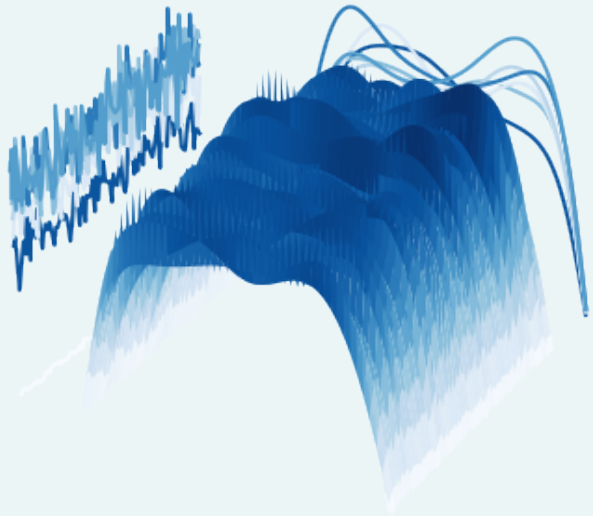
- Use knowledge of underlying PDE and grid.





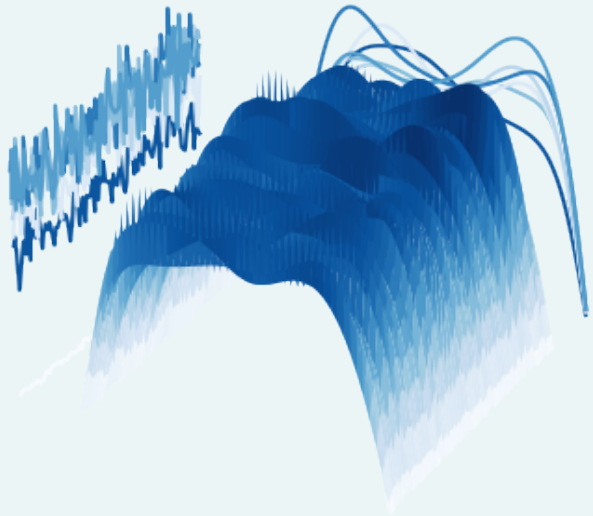
Anisotropic Example

- Use knowledge of underlying PDE and grid.

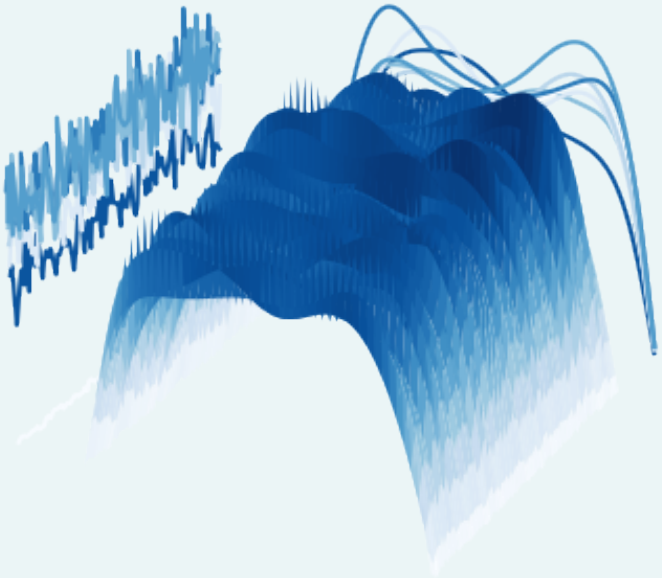


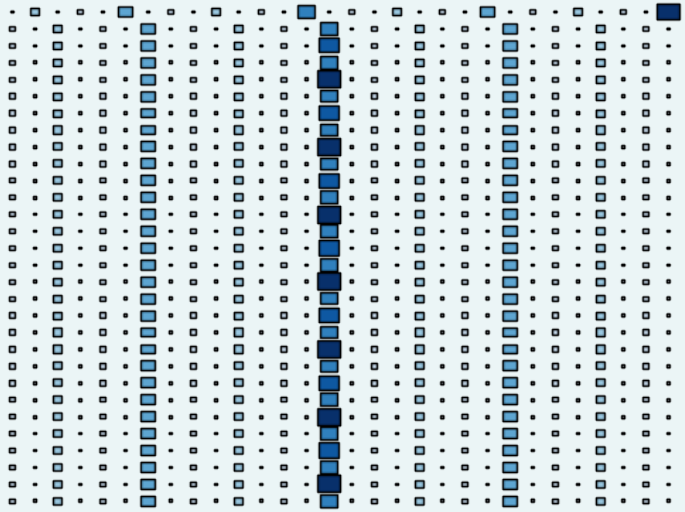
Anisotropic Example

- Use knowledge of underlying PDE and grid.
- Line-relaxation can smooth oscillatory error on coarser grids.



Algebraic Multigrid (AMG)





- Algebraic multigrid follows multigrid principles using only the coefficients of the matrix.
- More robust - can be used in time-dependent simulations.

Strength of Connection

- Smooth error is characterized by small eigenvectors, i.e.

$$e^T A e = \lambda \|e\|^2 \ll 1.$$

Strength of Connection

- Smooth error is characterized by small eigenvectors, i.e.

$$e^T A e = \lambda \|e\|^2 \ll 1.$$

- We can derive

$$e^T A e = \sum_{i<j} (-a_{ij})(e_i - e_j)^2 \ll 1.$$

Strength of Connection

- Smooth error is characterized by small eigenvectors, i.e.

$$e^T A e = \lambda \|e\|^2 \ll 1.$$

- We can derive

$$e^T A e = \sum_{i<j} (-a_{ij})(e_i - e_j)^2 \ll 1.$$

- "Algebraically smooth" in direction of **large negative** matrix coefficients.

Strength of Connection

- Smooth error is characterized by small eigenvectors, i.e.

$$e^T A e = \lambda \|e\|^2 \ll 1.$$

- We can derive

$$e^T A e = \sum_{i<j} (-a_{ij})(e_i - e_j)^2 \ll 1.$$

- "Algebraically smooth" in direction of **large negative** matrix coefficients.

Definition

A component u_i is θ-strongly connected with a component u_j if

$$-a_{ij} \geq \theta \max_{k \neq j} \{-a_{ik}\}.$$

Example

- Five-point stencil discretization on a 5×5 grid

$$-u_{xx} - \epsilon u_{yy} = f, \quad 0 < \epsilon \ll 1, \quad A \sim \begin{bmatrix} & -\epsilon & \\ -1 & 2+2\epsilon & -1 \\ & -\epsilon & \end{bmatrix}.$$

Example

- Five-point stencil discretization on a 5×5 grid

$$-u_{xx} - \epsilon u_{yy} = f, \quad 0 < \epsilon \ll 1, \quad A \sim \begin{bmatrix} & -\epsilon & \\ -1 & 2+2\epsilon & -1 \\ & -\epsilon & \end{bmatrix}.$$

- Construct strength adjacency matrix:

$$S = \begin{pmatrix} 0 & 1 & & & \\ 1 & 0 & 1 & & \\ & 1 & 0 & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & 0 \end{pmatrix}$$

$S_{ij} = 1$ if u_i and u_j are strongly connected.

Example

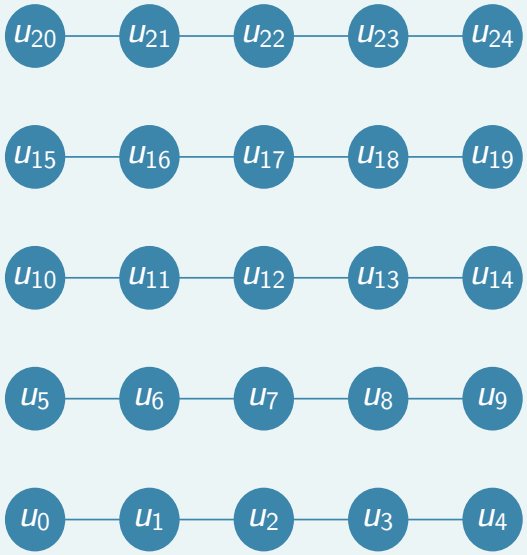
- Five-point stencil discretization on a 5×5 grid

$$-u_{xx} - \epsilon u_{yy} = f, \quad 0 < \epsilon \ll 1, \quad A \sim \begin{bmatrix} & -\epsilon & \\ -1 & 2+2\epsilon & -1 \\ & -\epsilon & \end{bmatrix}.$$

- Construct strength adjacency matrix:

$$S = \begin{pmatrix} 0 & 1 & & & \\ 1 & 0 & 1 & & \\ & 1 & 0 & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & 0 \end{pmatrix}$$

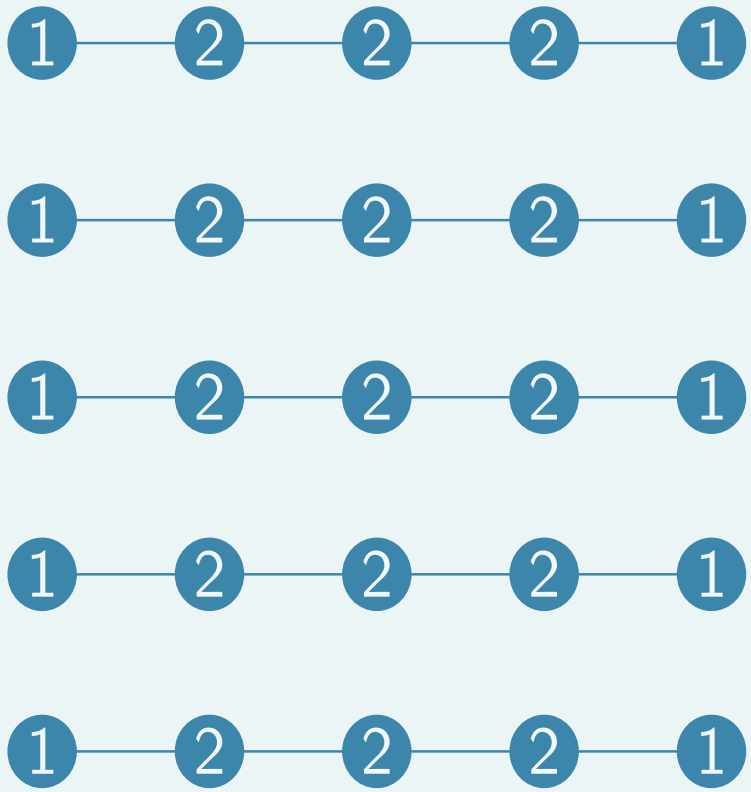
$S_{ij} = 1$ if u_i and u_j are strongly connected.



Graphical representation of S.

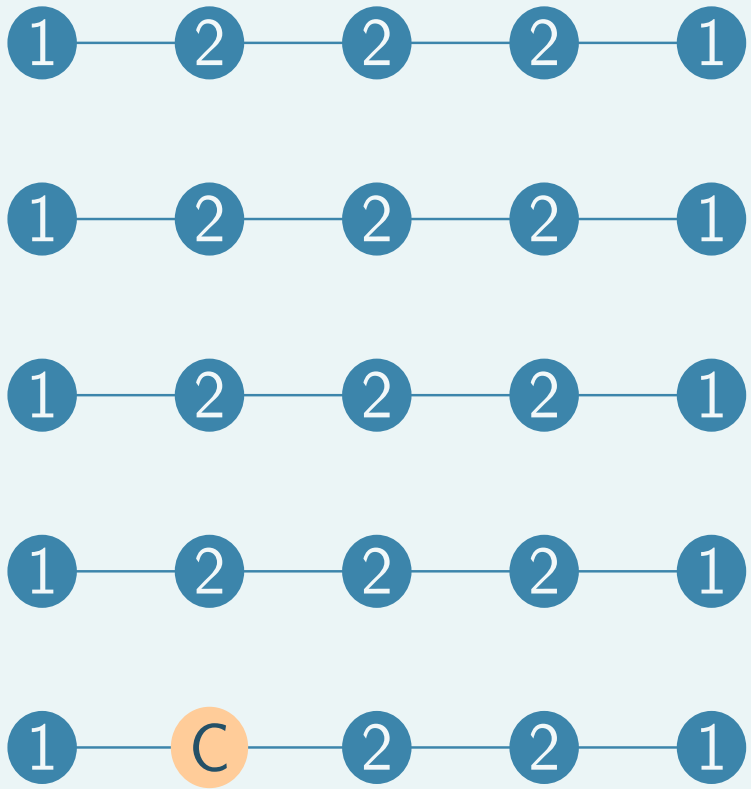
Algebraic Coarsening

- How to coarsen?
  1. **Label each node with # strong connections,**



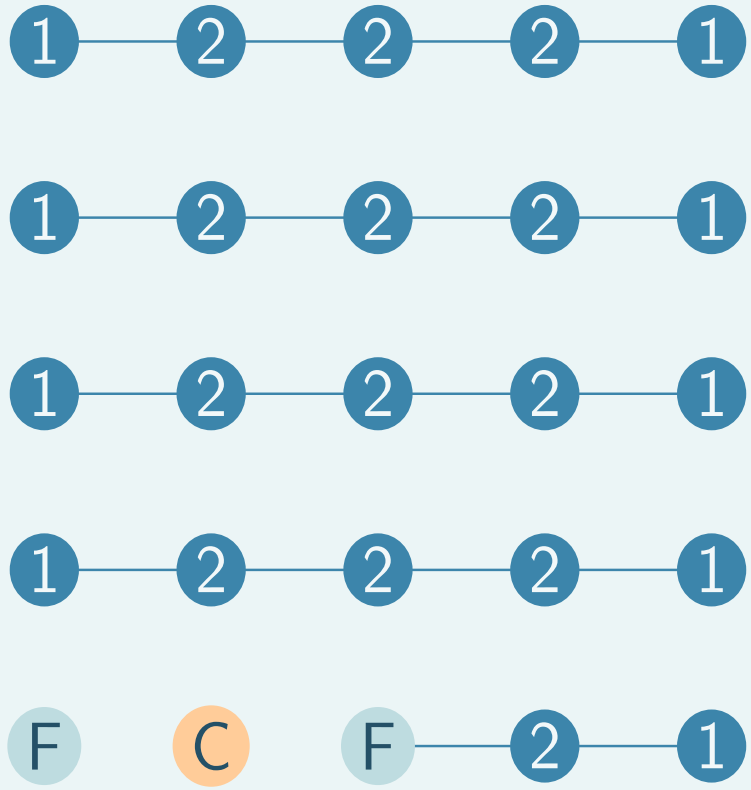
Algebraic Coarsening

- How to coarsen?
  1. Label each node with # strong connections,
  2. **Pick highest node as a C-point,**



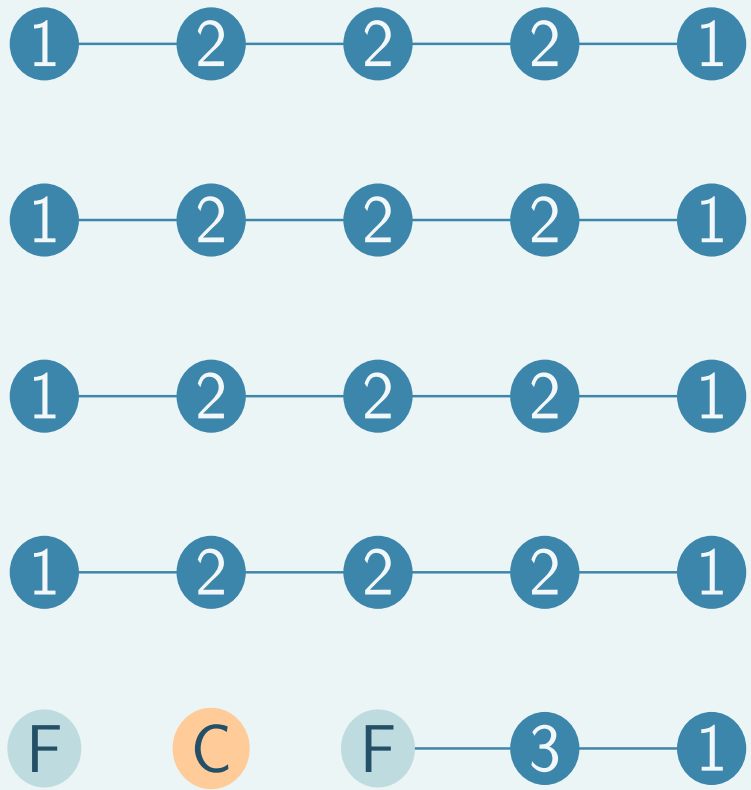
Algebraic Coarsening

- How to coarsen?
  1. Label each node with # strong connections,
  2. Pick highest node as a C-point,
  3. **All connected nodes to the selected are F-points,**



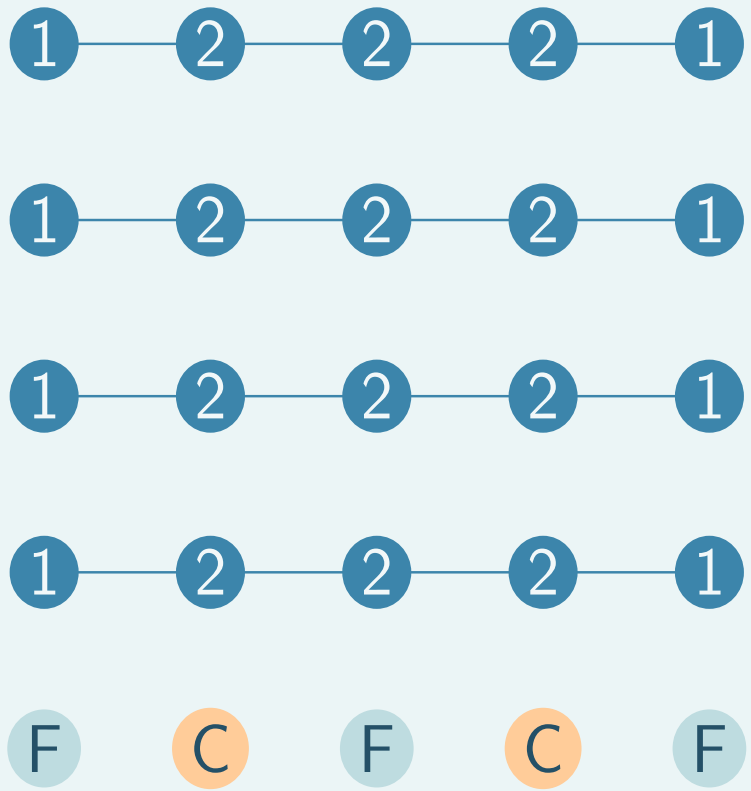
Algebraic Coarsening

- How to coarsen?
  1. Label each node with # strong connections,
  2. Pick highest node as a C-point,
  3. All connected nodes to the selected are F-points,
  4. **Increment the nodes connected to F-points,**



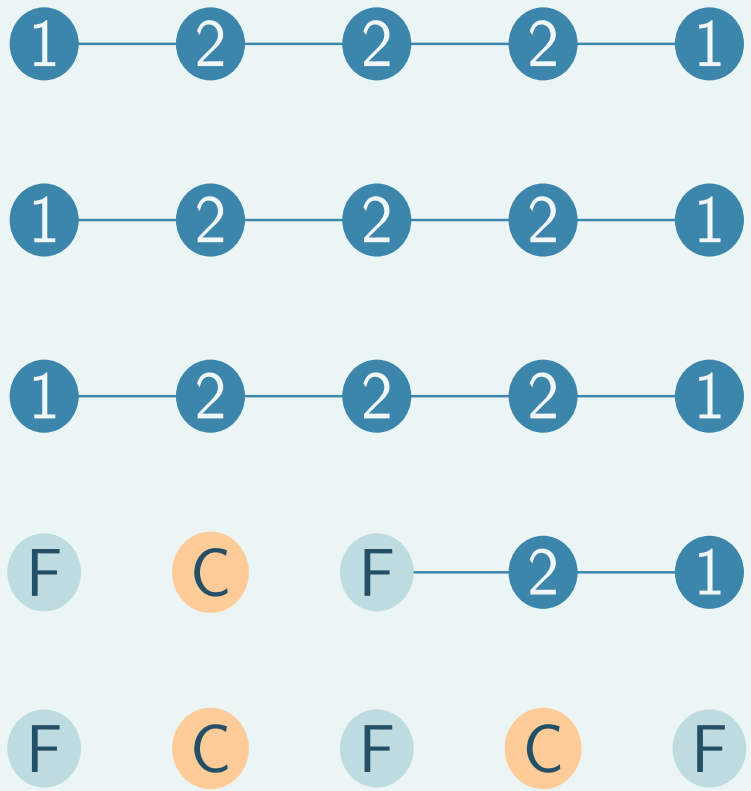
Algebraic Coarsening

- How to coarsen?
  1. Label each node with # strong connections,
  2. Pick highest node as a C-point,
  3. All connected nodes to the selected are F-points,
  4. Increment the nodes connected to F-points,
  5. **Stop when all are F-points or C-points.**



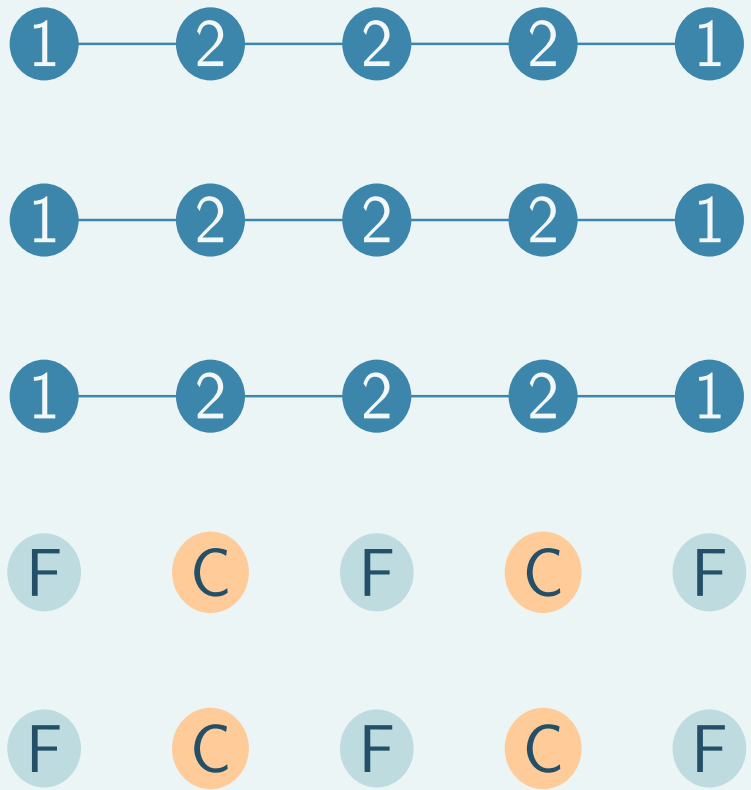
Algebraic Coarsening

- How to coarsen?
  1. Label each node with # strong connections,
  2. Pick highest node as a C-point,
  3. All connected nodes to the selected are F-points,
  4. Increment the nodes connected to F-points,
  5. **Stop when all are F-points or C-points.**



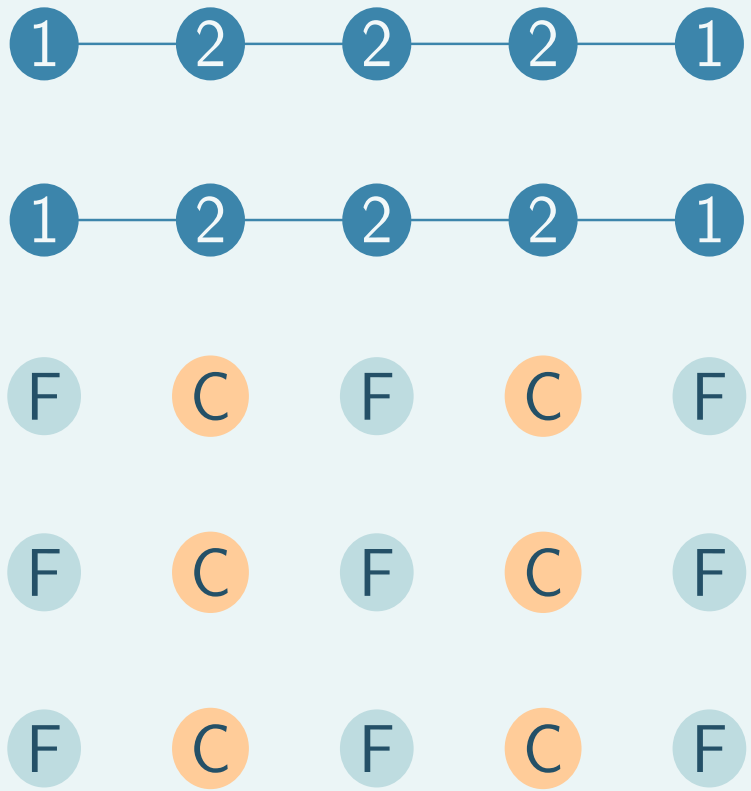
Algebraic Coarsening

- How to coarsen?
  1. Label each node with # strong connections,
  2. Pick highest node as a C-point,
  3. All connected nodes to the selected are F-points,
  4. Increment the nodes connected to F-points,
  5. **Stop when all are F-points or C-points.**



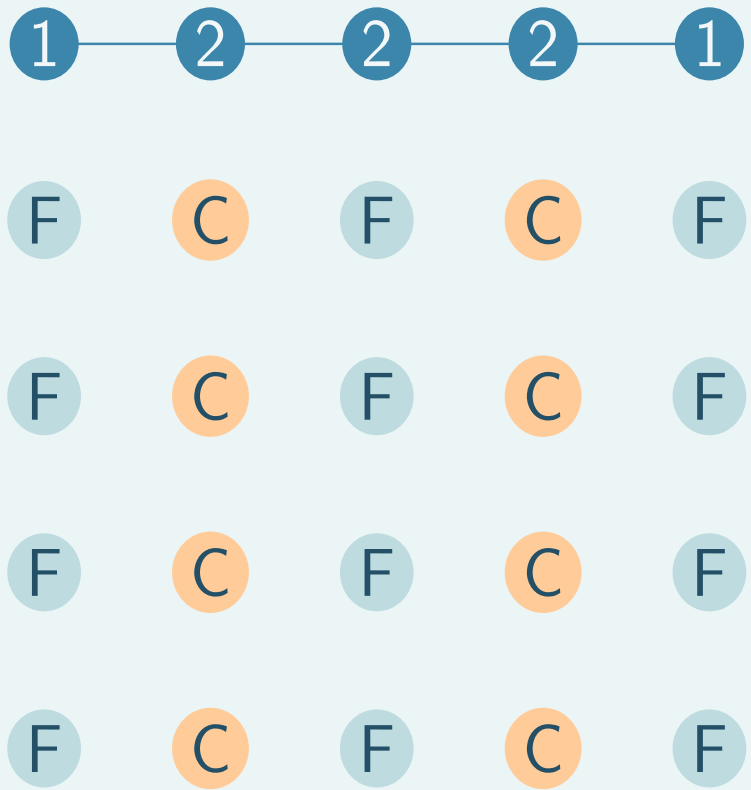
Algebraic Coarsening

- How to coarsen?
  1. Label each node with # strong connections,
  2. Pick highest node as a C-point,
  3. All connected nodes to the selected are F-points,
  4. Increment the nodes connected to F-points,
  5. **Stop when all are F-points or C-points.**



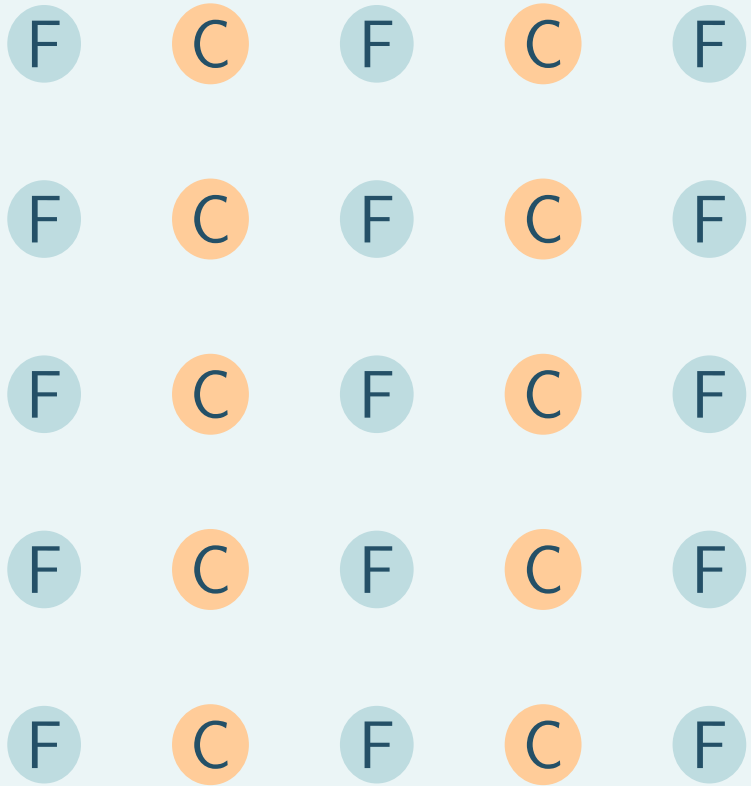
Algebraic Coarsening

- How to coarsen?
  1. Label each node with # strong connections,
  2. Pick highest node as a C-point,
  3. All connected nodes to the selected are F-points,
  4. Increment the nodes connected to F-points,
  5. **Stop when all are F-points or C-points.**

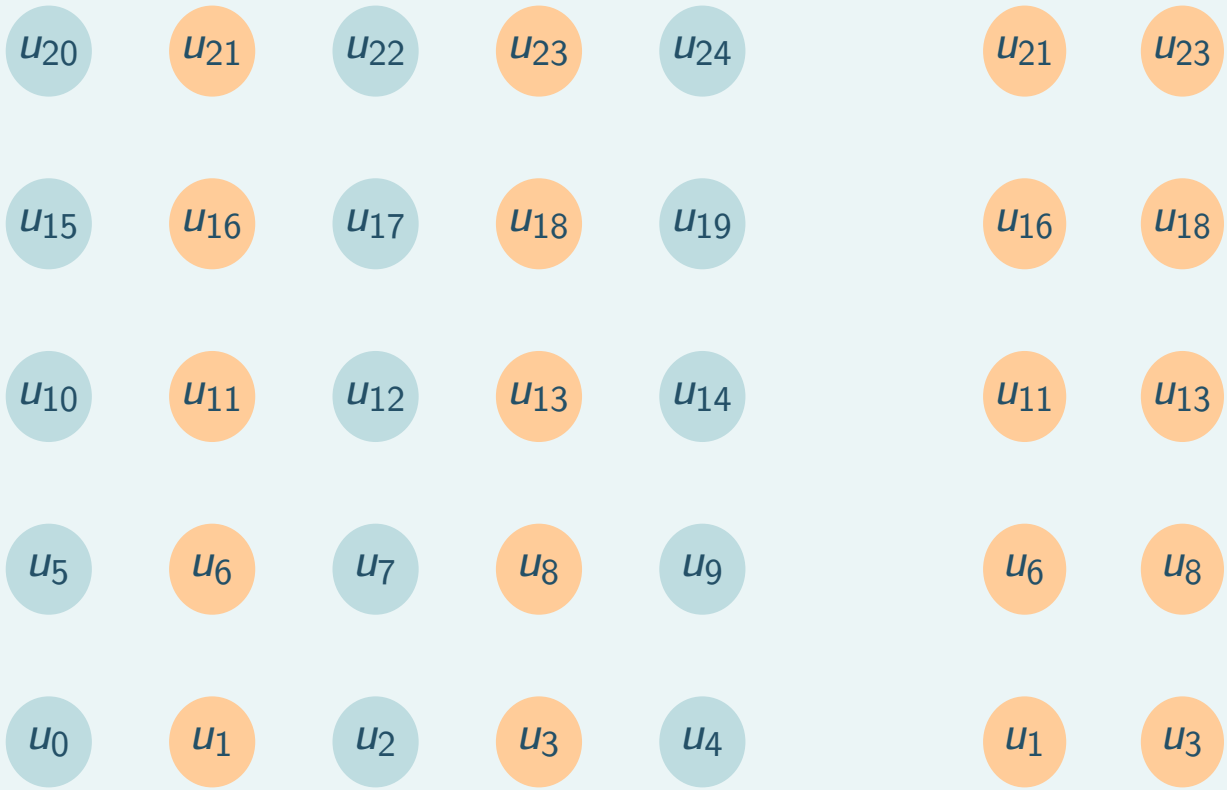


Algebraic Coarsening

- How to coarsen?
  1. Label each node with # strong connections,
  2. Pick highest node as a C-point,
  3. All connected nodes to the selected are F-points,
  4. Increment the nodes connected to F-points,
  5. **Stop when all are F-points or C-points.**



Algebraic Coarsening



Algebraic Interpolation

- Weighted average of neighbouring coarse points.

Algebraic Interpolation

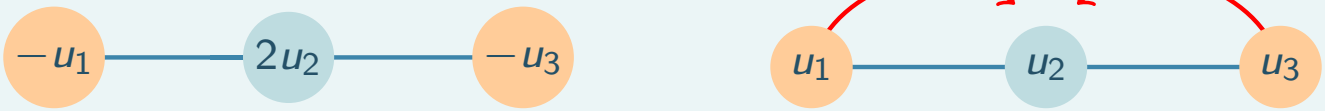
- Weighted average of neighbouring coarse points.
- For the previous problem,

$$u_2 = \left(\frac{a_{21}}{a_{21}+a_{23}}\right) u_1 + \left(\frac{a_{23}}{a_{21}+a_{23}}\right) u_3 = \frac{1}{2}u_1 + \frac{1}{2}u_3.$$

Algebraic Interpolation

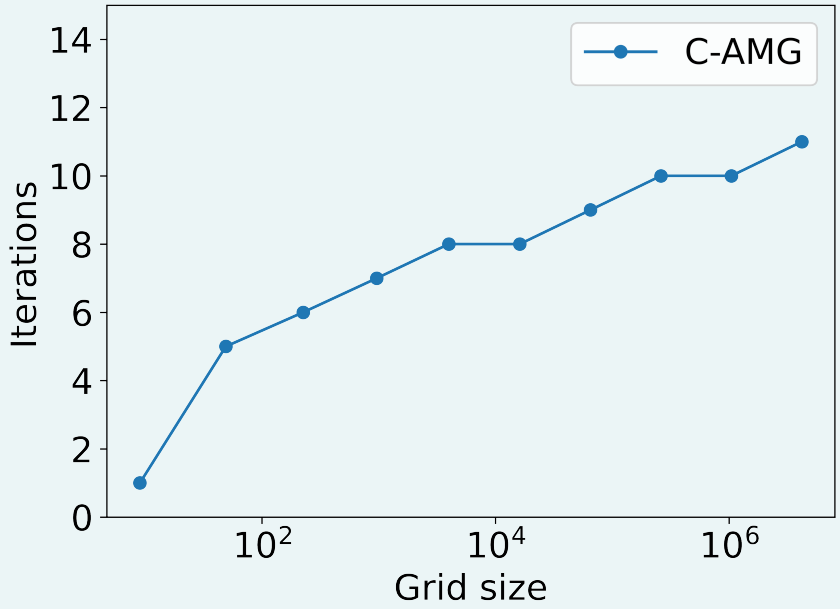
- Weighted average of neighbouring coarse points.
- For the previous problem,

$$u_2 = \left(\frac{a_{21}}{a_{21}+a_{23}}\right) u_1 + \left(\frac{a_{23}}{a_{21}+a_{23}}\right) u_3 = \frac{1}{2}u_1 + \frac{1}{2}u_3.$$



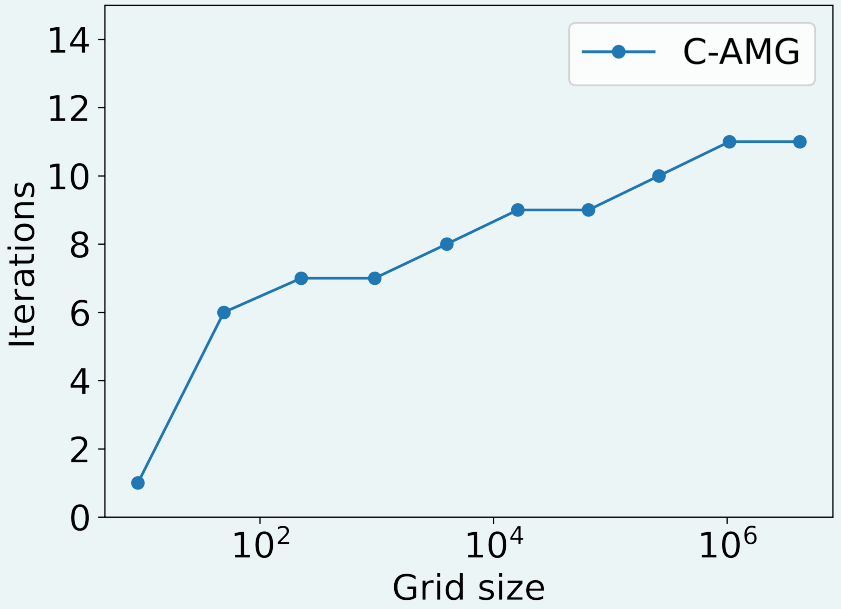
RECOVERED LOCAL LINEAR INTERPOLATION

C-AMG Performance on Diffusion



Isotropic diffusion:

$A = -u_{xx} - u_{yy}$



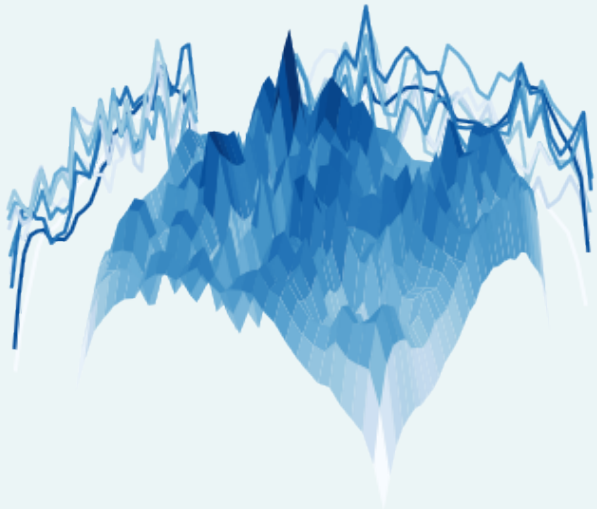
Anisotropic in y diffusion:

$A = -0.00001u_{xx} - u_{yy}$

Problem

- Nine-point stencil discretization on a 5×5 grid:

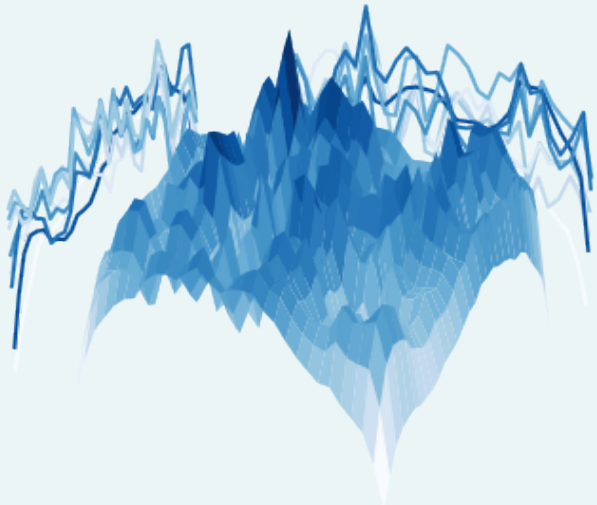
$$-u_{xxyy} = f, \quad A \sim \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

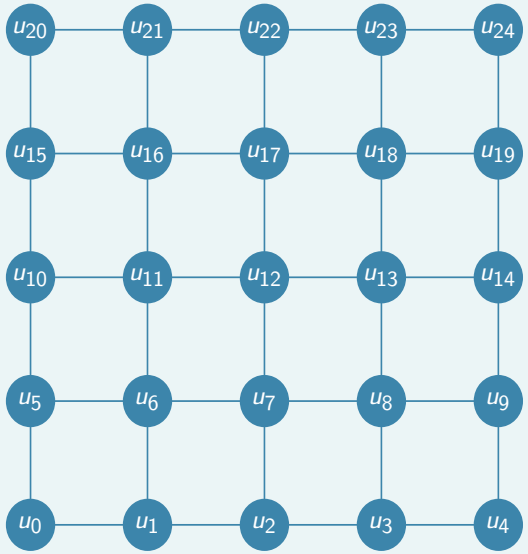


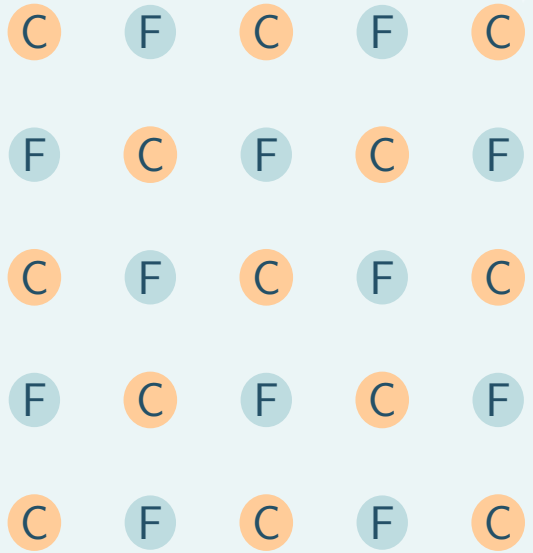
Problem

- Nine-point stencil discretization on a 5×5 grid:

$$-u_{xxyy} = f, \quad A \sim \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$





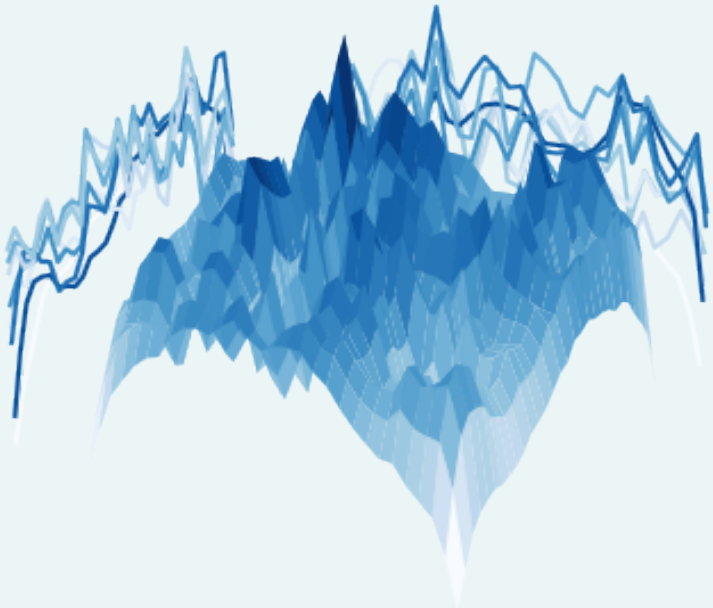


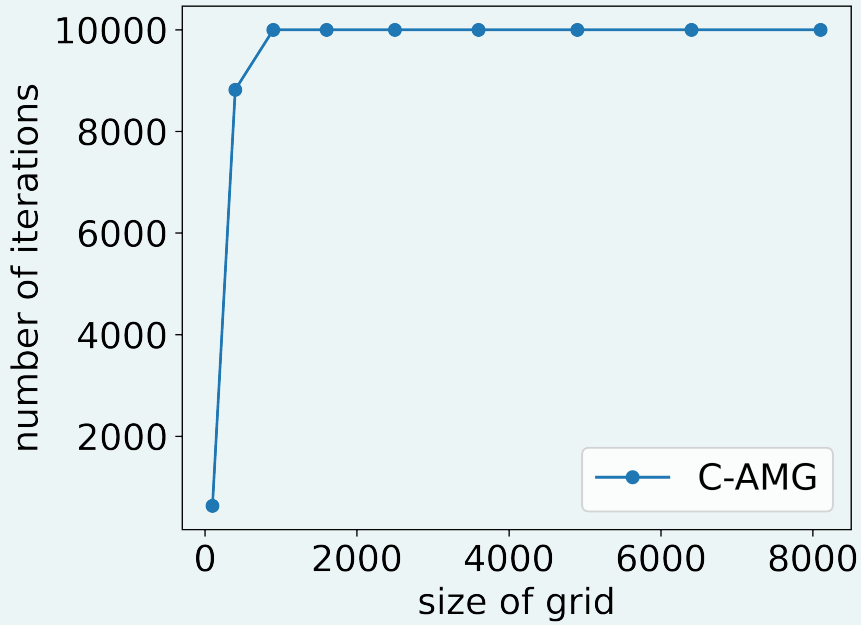
Problem

Standard solving techniques break down for this problem.

COGENT Proxy Problem

$$-au_{xx} - bu_{yy} + acu_{yyxx} = g, \quad b, c \gg a$$





Preliminaries: Total Reduction

- Can write $Au = f$ as equivalent system:

$$\begin{pmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{pmatrix} \begin{pmatrix} u|_f \\ u|_c \end{pmatrix} = \begin{pmatrix} f|_f \\ f|_c \end{pmatrix}$$

Preliminaries: Total Reduction

- Can write $Au = f$ as equivalent system:

$$\begin{pmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{pmatrix} \begin{pmatrix} u|_f \\ u|_c \end{pmatrix} = \begin{pmatrix} f|_f \\ f|_c \end{pmatrix}$$

- Take "ideal" P_* such that $P_* e|_c = e$.
  - $e_c = \arg\min \|e - Pe_c\|_A$.

Preliminaries: Total Reduction

- Can write $Au = f$ as equivalent system:

$$\begin{pmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{pmatrix} \begin{pmatrix} u|_f \\ u|_c \end{pmatrix} = \begin{pmatrix} f|_f \\ f|_c \end{pmatrix}$$

- Take "ideal" P_* such that $P_* e|_c = e$.
- For $S = \begin{pmatrix} I_F \\ 0 \end{pmatrix}$,

$$I - A^{-1}A = \underbrace{\left(I - P_* \left(P_*^T A P_*\right)^{-1} P_*^T A\right)}_{\text{coarse-grid correction}} \underbrace{\left(I - S \left(S^T A S\right)^{-1} S^T A\right)}_{\text{F-relaxation}}.$$

Preliminaries: Total Reduction

- Can write $Au = f$ as equivalent system:

$$\begin{pmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{pmatrix} \begin{pmatrix} u|_f \\ u|_c \end{pmatrix} = \begin{pmatrix} f|_f \\ f|_c \end{pmatrix}$$

- Take "ideal" P_* such that $P_* e|_c = e$.
- For $S = \begin{pmatrix} I_F \\ 0 \end{pmatrix}$,

$$I - A^{-1}A = \underbrace{\left(I - P_* \left(P_*^T A P_*\right)^{-1} P_*^T A\right)}_{\text{coarse-grid correction}} \underbrace{\left(I - S \left(S^T A S\right)^{-1} S^T A\right)}_{\text{F-relaxation}}.$$

- Want multigrid method with this error propagation.

F-Relaxation

- Use

$$\widetilde{M^{-1}} := S(S^T A S)^{-1} S^T = \begin{pmatrix} A_{FF}^{-1} & 0 \\ 0 & 0 \end{pmatrix}.$$

F-Relaxation

- Use

$$\widetilde{M^{-1}} := S(S^T A S)^{-1} S^T = \begin{pmatrix} A_{FF}^{-1} & 0 \\ 0 & 0 \end{pmatrix}.$$

- Update is

$$u^{(k+1)} = u^{(k)} + \widetilde{M^{-1}} r^{(k)}$$

F-Relaxation

- Use

$$\widetilde{M^{-1}} := S(S^T A S)^{-1} S^T = \begin{pmatrix} A_{FF}^{-1} & 0 \\ 0 & 0 \end{pmatrix}.$$

- Update is

$$u^{(k+1)} = u^{(k)} + \widetilde{M^{-1}} r^{(k)}$$

- Error propagation is

$$E_F := I - \widetilde{M^{-1}} A = I - S \left(S^T A S \right)^{-1} S^T A.$$

Coarse Grid Correction

- From multigrid,

$$P_*^T A P_* e_c = P_*^T r \implies e_c = (P_*^T A P_*)^{-1} A e_k.$$

Coarse Grid Correction

- From multigrid,

$$P_*^T A P_* e_c = P_*^T r \implies e_c = (P_*^T A P_*)^{-1} A e_k.$$

- Then

$$u_{k+1} = u_k + P_* e_c.$$

Coarse Grid Correction

- From multigrid,

$$P_*^T A P_* e_c = P_*^T r \implies e_c = (P_*^T A P_*)^{-1} A e_k.$$

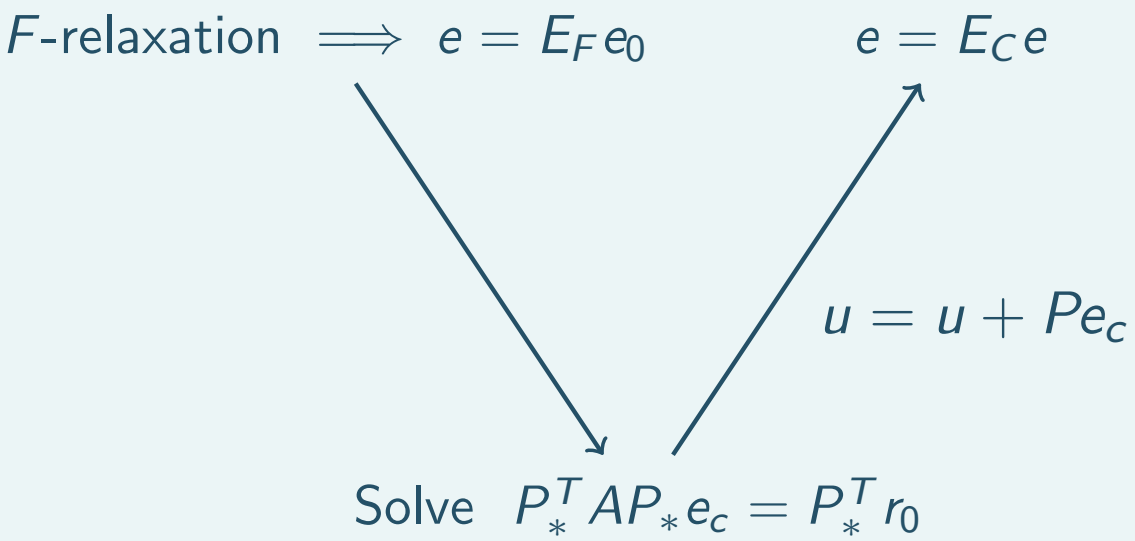
- Then

$$u_{k+1} = u_k + P_* e_c.$$

- Can derive

$$E_C := I - P_* (P_*^T A P_*)^{-1} P_*^T A.$$

Multigrid Reduction



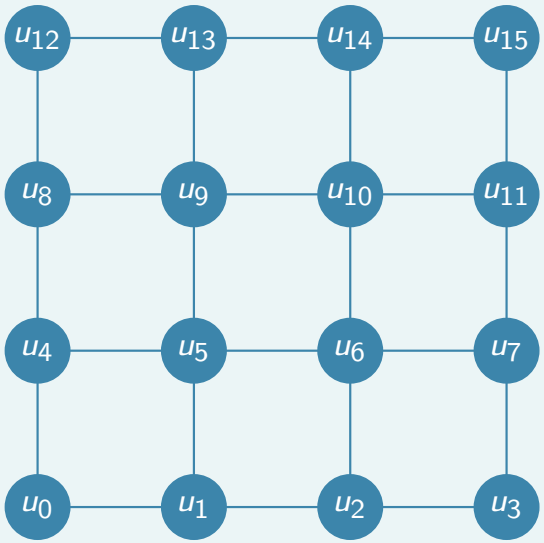
- Exact two-grid method.
  - $e = E_C E_F e_0 = 0$

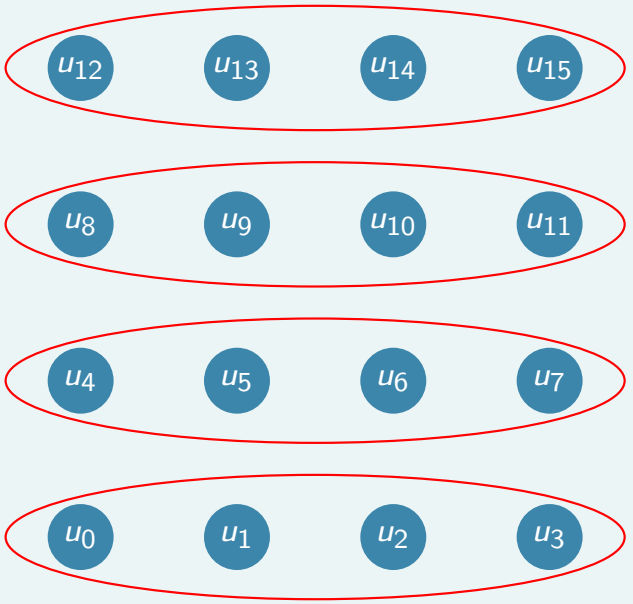
Solving Fourth Order Term

- First focus on fourth order term.
- We will take multigrid reduction approach.
  1. Approximate coarse grid correction
  2. Approximate F-relaxation.

Line Relaxation

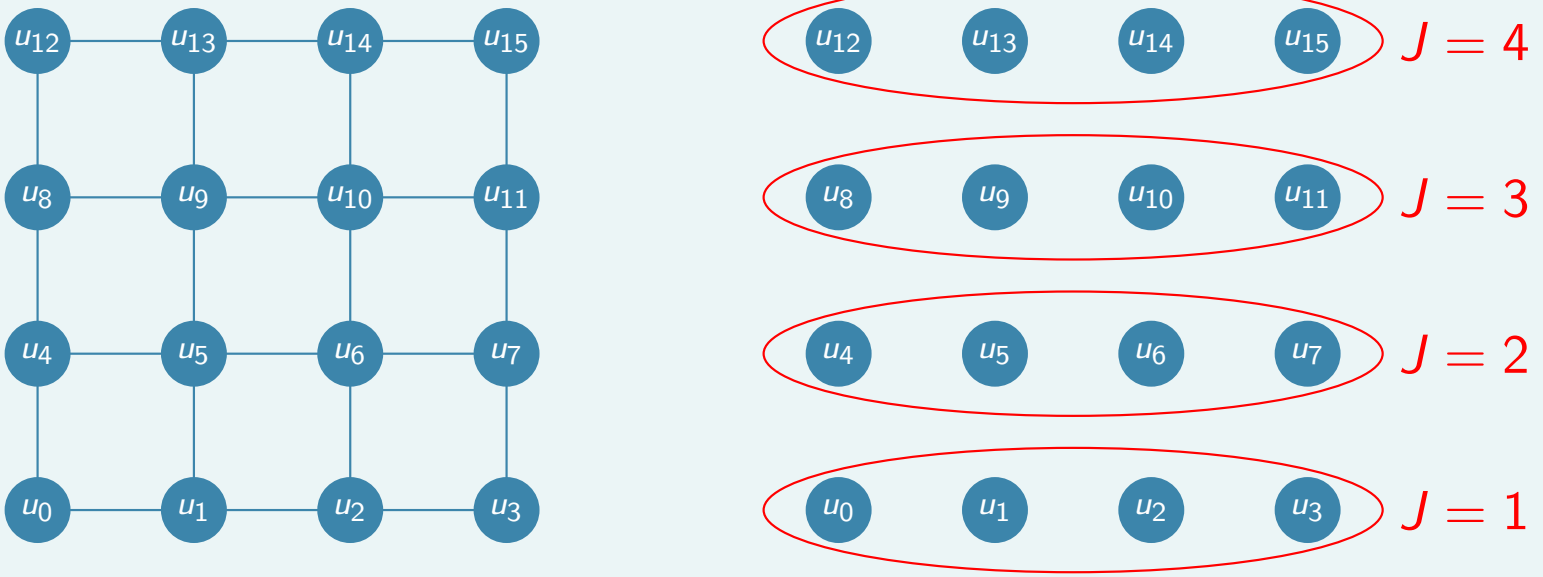
- Partition grid into lines.





Line Relaxation

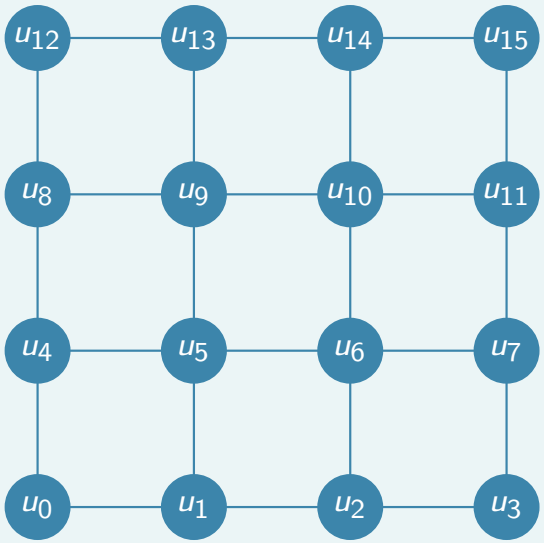
- Partition grid into lines.

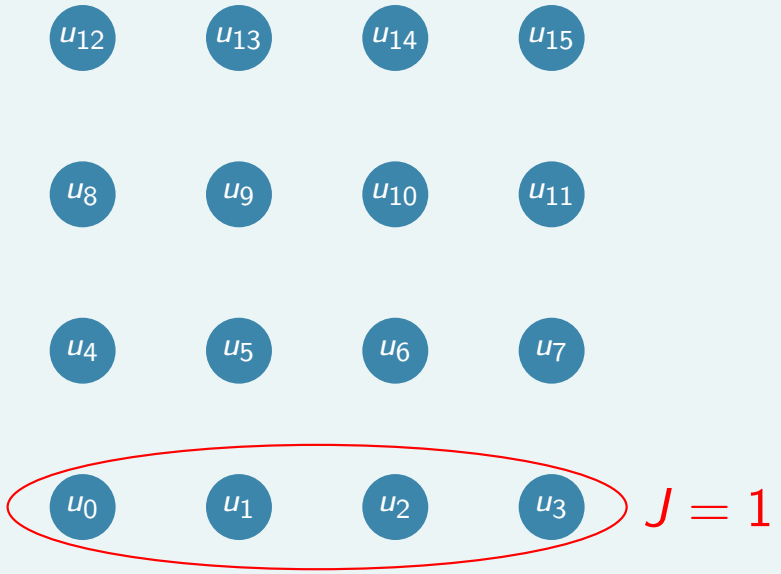


- Let J be the Jth line.

Line Relaxation

- Partition grid into lines.





- Let J be the Jth line.
- Solve each line exactly.

Line Relaxation

- For $J = 1$.

$$A_{J,J} u_J = \begin{bmatrix} 4 & -2 & & \\ -2 & 4 & -2 & \\ & -2 & 4 & -2 \\ & & -2 & 4 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{bmatrix} = f_J - A_{J,J+1} u_{J+1}$$

- Systems are tridiagonal - can be solved in $O(n)$.

Line Relaxation

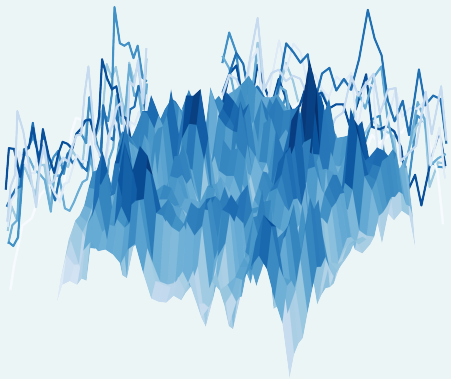
- Why do we do this?

Line Relaxation

- Why do we do this?
- Can smooth oscillatory near-kernel components.

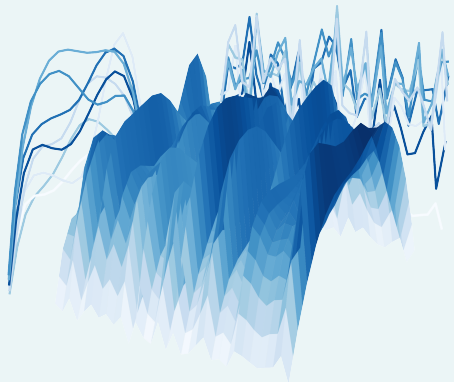
Line Relaxation

- Why do we do this?
- Can smooth oscillatory near-kernel components.



pointwise

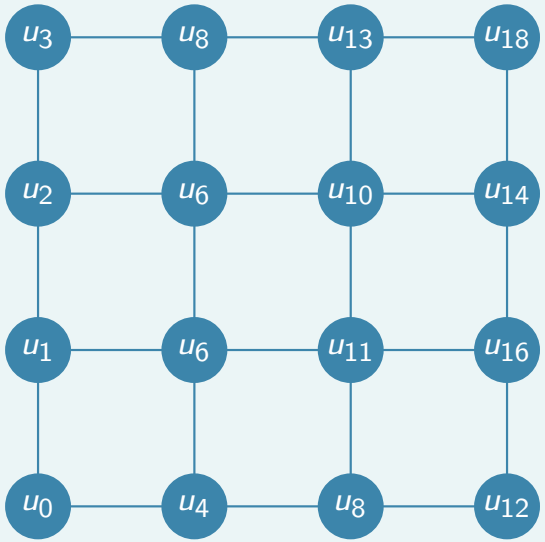
x-line



y-line

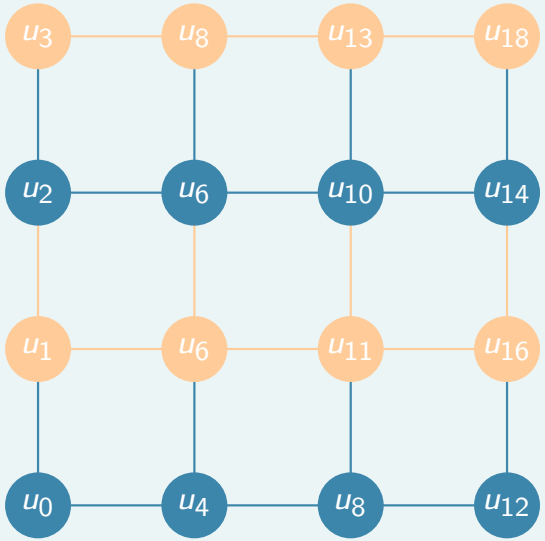
Picking Coarse Points

- Still need to partition points into C/F points



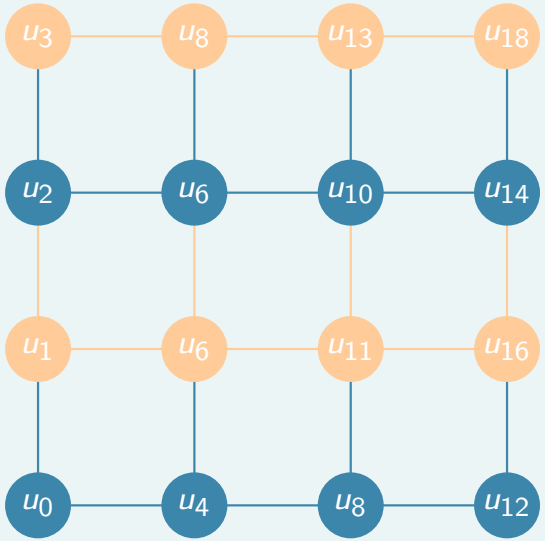
Picking Coarse Points

- Still need to partition points into C/F points
- Choose C-points to be lines in direction opposite relaxation.



Picking Coarse Points

- Still need to partition points into C/F points
- Choose C-points to be lines in direction opposite relaxation.
- Pick F-lines to be independent

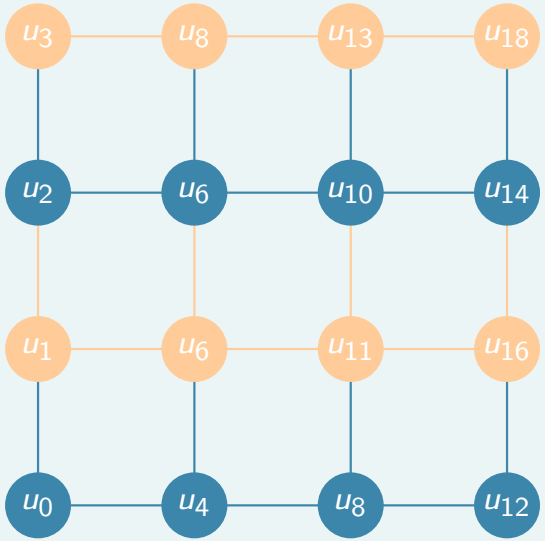


Picking Coarse Points

- Still need to partition points into C/F points
- Choose C-points to be lines in direction opposite relaxation.
- Pick F-lines to be independent
  - F-line relaxation becomes exact.

$$A_{FF}^{-1} = \begin{bmatrix} A_{22}^{-1} & 0 \\ 0 & A_{44}^{-1} \end{bmatrix}$$

  - Can solve in parallel.

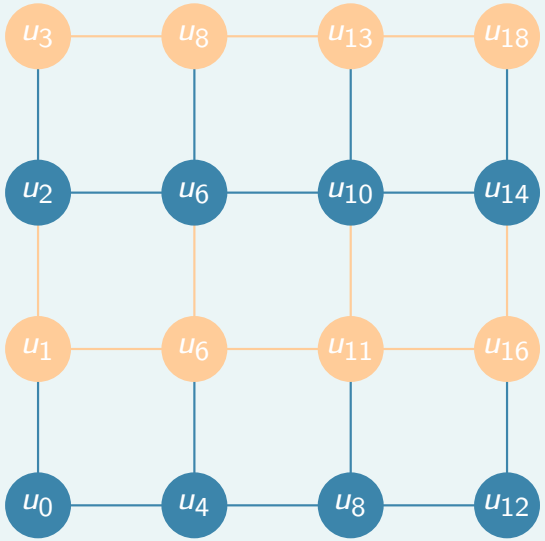


Picking Coarse Points

- Still need to partition points into C/F points
- Choose C-points to be lines in direction opposite relaxation.
- Pick F-lines to be independent
  - F-line relaxation becomes exact.

$$A_{FF}^{-1} = \begin{bmatrix} A_{22}^{-1} & 0 \\ 0 & A_{44}^{-1} \end{bmatrix}$$

  - Can solve in parallel.
- Solve C-lines then F-lines



Important Equation

$$e_J = -A_{J,J}^{-1} A_{J,J-1} e_{J-1} - A_{J,J}^{-1} A_{J,J+1} e_{J+1}$$

Ideal Interpolation

Important Equation

$$e_J = -A^{-1}_{J,J} A_{J,J-1} e_{J-1} - A^{-1}_{J,J} A_{J,J+1} e_{J+1}$$

Ideal Interpolation

Important Equation

$$e_J = -A^{-1}_{J,J} A_{J,J-1} e_{J-1} - A^{-1}_{J,J} A_{J,J+1} e_{J+1}$$

- Fourth-order term has the stencil

$$u_{xxyy} \sim \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

Ideal Interpolation

Important Equation

$$e_J = -A_{J,J}^{-1} A_{J,J-1} e_{J-1} - A_{J,J}^{-1} A_{J,J+1} e_{J+1}$$

- Fourth-order term has the stencil

$$u_{xxyy} \sim \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

- Note that

$$A_{J,J-1} = -\frac{1}{2} A_{J,J} \qquad .$$

Ideal Interpolation

Important Equation

$$e_J = -A_{J,J}^{-1} A_{J,J-1} e_{J-1} - A_{J,J}^{-1} A_{J,J+1} e_{J+1}$$

- Fourth-order term has the stencil

$$u_{xxyy} \sim \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

- Note that

$$A_{J,J-1} = -\frac{1}{2} A_{J,J} = A_{J,J+1}.$$

Ideal Interpolation

Important Equation

$$e_J = -A^{-1}_{J,J} A_{J,J-1} e_{J-1} - A^{-1}_{J,J} A_{J,J+1} e_{J+1}$$

- Fourth-order term has the stencil

$$u_{xxyy} \sim \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

- Note that

$$A_{J,J-1} = -\frac{1}{2} A_{J,J} = A_{J,J+1}.$$

- Therefore,

$$e_J = \frac{1}{2} e_{J-1} + \frac{1}{2} e_{J+1}.$$

Ideal Interpolation

Important Equation

$$e_J = -A_{J,J}^{-1} A_{J-1,J} e_{J-1} - A_{J,J}^{-1} A_{J+1,J} e_{J+1}$$

- Similarly,

$$u_{xxyy} \sim \begin{bmatrix} \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} & -2 & \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} \\ 4 & -2 & \end{bmatrix}$$

- Once again

$$A_{J-1,J} = -\frac{1}{2} A_{J,J} = A_{J+1,J},$$

and

$$e_J = \frac{1}{2} e_{J-1} + \frac{1}{2} e_{J+1}.$$

Ideal Interpolation Cont.

- As a matrix,

$$P = \begin{bmatrix} 1/2 & 1 & 1/2 \end{bmatrix}_c \iff e_J = \frac{1}{2} e_{J-1} + \frac{1}{2} e_{J+1}.$$

Ideal Interpolation Cont.

- As a matrix,

$$P = \begin{bmatrix} 1/2 & 1 & 1/2 \end{bmatrix}_c \iff e_J = \frac{1}{2} e_{J-1} + \frac{1}{2} e_{J+1}.$$

- P is exact, ideal coarse grid correction.
  - Converge in one iteration for two-grid.

Ideal Interpolation Cont.

- As a matrix,

$$P = \begin{bmatrix} 1/2 & 1 & 1/2 \end{bmatrix}_c \iff e_J = \frac{1}{2} e_{J-1} + \frac{1}{2} e_{J+1}.$$

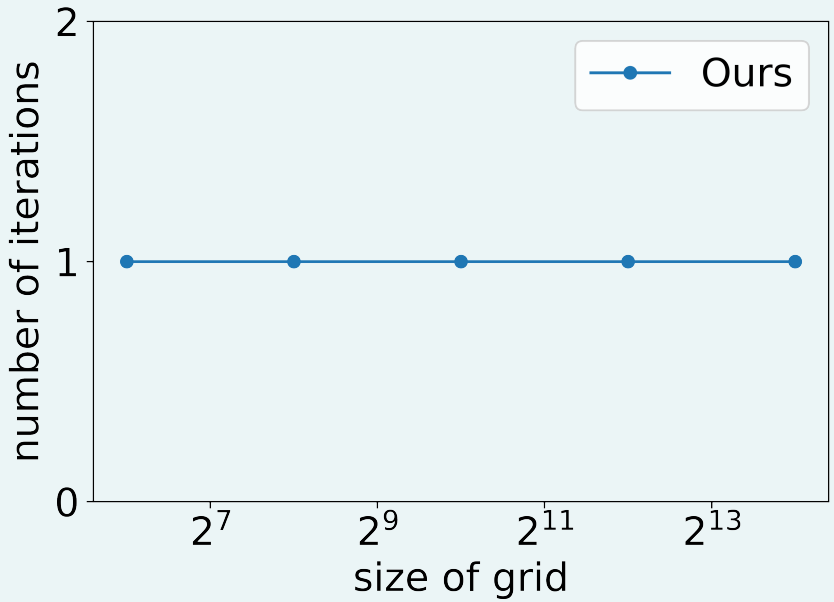
- P is exact, ideal coarse grid correction.
  - Converge in one iteration for two-grid.
- Note $P^T A P \sim \frac{1}{2} A$.
  - Can use same P on each level.
  - Exact coarse grid correction on every level.
  - Should take 1 iteration for n-level solve.

Ideal Interpolation Cont.

- As a matrix,

$$P = \begin{bmatrix} 1/2 & 1 & 1/2 \end{bmatrix}_c \iff e_J = \frac{1}{2} e_{J-1} + \frac{1}{2} e_{J+1}.$$

- P is exact, ideal coarse grid correction.
  - Converge in one iteration for two-grid.
- Note $P^T A P \sim \frac{1}{2} A$.
  - Can use same P on each level.
  - Exact coarse grid correction on every level.
  - Should take 1 iteration for n-level solve.
- Have exact method for fourth order term!



Solving Fourth Order Term

- First focus on fourth order term.
- We will take multigrid reduction approach.
  1. Approximate coarse grid correction
     - Semi-coarsen
     - 2pt ideal interpolation
  2. Approximate F-relaxation.
     - C/F Line relaxation.

Solving Fourth Order Term

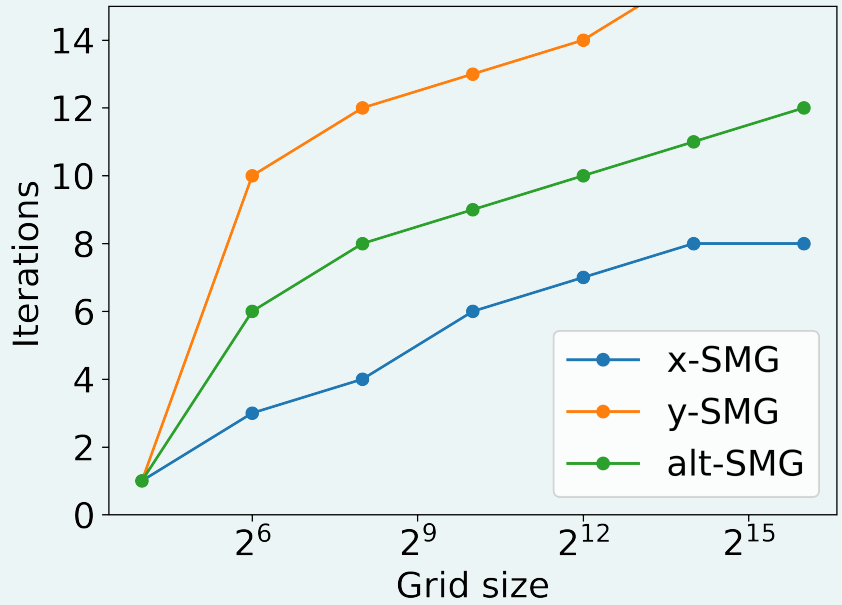
- First focus on fourth order term.
- We will take multigrid reduction approach.
  1. Approximate coarse grid correction
     - Semi-coarsen
     - 2pt ideal interpolation
  2. Approximate F-relaxation.
     - C/F Line relaxation.
- Now we must deal with diffusion term.

Dealing with Diffusion

COGENT Problem

$$-au_{xx} - bu_{yy} + acu_{yyxx} = g, \quad b, c \gg a$$

- Have both diffusion and fourth order term.
  - Showed method is exact for fourth order term.
- Do not have exact coarse-grid correction with diffusion.
- Differences will become more extreme in 3D.



$$-u_{xx} - \epsilon u_{yy} = 0$$

Dealing with Diffusion

- Diffusion term $A = -\epsilon u_{xx} - u_{yy}$ has stencil

$$A \sim \begin{bmatrix} 0 & -1 & 0 \\ -\epsilon & 2+2\epsilon & -\epsilon \\ 0 & -1 & 0 \end{bmatrix}.$$

Dealing with Diffusion

- Diffusion term $A = -\epsilon u_{xx} - u_{yy}$ has stencil

$$A \sim \begin{bmatrix} 0 & -1 & 0 \\ -\epsilon & 2+2\epsilon & -\epsilon \\ 0 & -1 & 0 \end{bmatrix}.$$

- If we semi-coarsen in y,

$$P^T A P \sim \begin{bmatrix} 0 & -1/2 & 0 \\ -\epsilon & 1+2\epsilon & -\epsilon \\ 0 & -1/2 & 0 \end{bmatrix}.$$

Dealing with Diffusion

- Diffusion term $A = -\epsilon u_{xx} - u_{yy}$ has stencil

$$A \sim \begin{bmatrix} 0 & -1 & 0 \\ -\epsilon & 2+2\epsilon & -\epsilon \\ 0 & -1 & 0 \end{bmatrix}.$$

- If we semi-coarsen in y,

$$P^T A P \sim \begin{bmatrix} 0 & -1/2 & 0 \\ -\epsilon & 1+2\epsilon & -\epsilon \\ 0 & -1/2 & 0 \end{bmatrix}.$$

- In general, if $A = -au_{xx} - bu_{yy}$
  - Semi coarsen in $x \implies a = (1/2)a$.
  - Semi coarsen in $y \implies b = (1/2)b$.

Dealing with Diffusion

- Diffusion term $A = -\epsilon u_{xx} - u_{yy}$ has stencil

$$A \sim \begin{bmatrix} 0 & -1 & 0 \\ -\epsilon & 2+2\epsilon & -\epsilon \\ 0 & -1 & 0 \end{bmatrix}.$$

- If we semi-coarsen in y,

$$P^T A P \sim \begin{bmatrix} 0 & -1/2 & 0 \\ -\epsilon & 1+2\epsilon & -\epsilon \\ 0 & -1/2 & 0 \end{bmatrix}.$$

- In general, if $A = -au_{xx} - bu_{yy}$
  - Semi coarsen in $x \implies a = (1/2)a$.
  - Semi coarsen in $y \implies b = (1/2)b$.
- Alternate semi-coarsening $\implies A \to$ isotropic.

Motivation

- Consider $A = -\epsilon u_{xx} - u_{yy}$ for $\epsilon = 1 \times 10^{-3}$.

Motivation

- Consider $A = -\epsilon u_{xx} - u_{yy}$ for $\epsilon = 1 \times 10^{-3}$.
- Semi-coarsen in y 20 times then

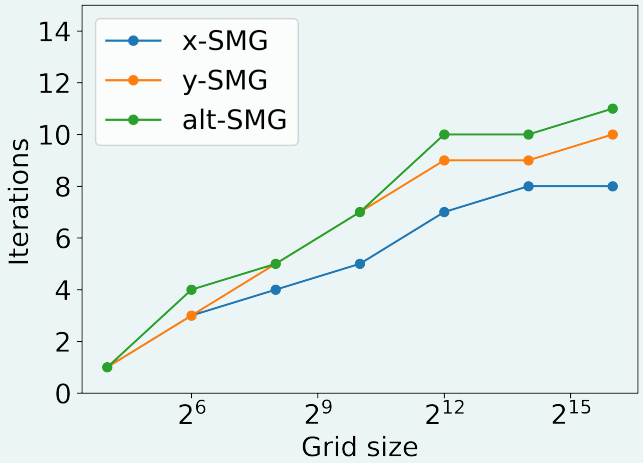
$$A^{(20)} \approx -u_{xx} - \epsilon u_{yy}$$

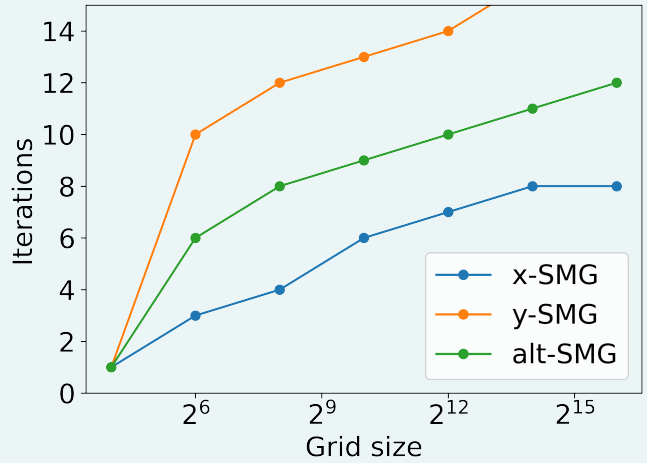
Motivation

- Consider $A = -\epsilon u_{xx} - u_{yy}$ for $\epsilon = 1 \times 10^{-3}$.
- Semi-coarsen in y 20 times then

$$A^{(20)} \approx -u_{xx} - \epsilon u_{yy}$$

- Do not want to semi-coarsen in y for this system





- Differences is amplified for approximate line solve.

Dealing with Diffusion

What we need

- Need to semi-coarsen in right direction when strongly anisotropic.
- Need to alternate to converge to isotropy.

Dealing with Diffusion

What we need

- Need to semi-coarsen in right direction when strongly anisotropic.
- Need to alternate to converge to isotropy.

- To do this, we use "smart" coarsening
  1. Reconstruct a and b.
  2. Coarsen according to the following:

$$\begin{cases} \text{Coarsen in } x, & \text{if a} \geq b \\ \text{Coarsen in } y, & \text{otherwise} \end{cases}$$

Dealing with Diffusion

What we need

- Need to semi-coarsen in right direction when strongly anisotropic.
- Need to alternate to converge to isotropy.

- To do this, we use "smart" coarsening
  1. Reconstruct a and b.
  2. Coarsen according to the following:

$$\begin{cases} \text{Coarsen in } x, & \text{if a} \geq b \\ \text{Coarsen in } y, & \text{otherwise} \end{cases}$$

- Ensures diffusion term is isotropic.

Dealing with Diffusion

What we need

- Need to semi-coarsen in right direction when strongly anisotropic.
- Need to alternate to converge to isotropy.

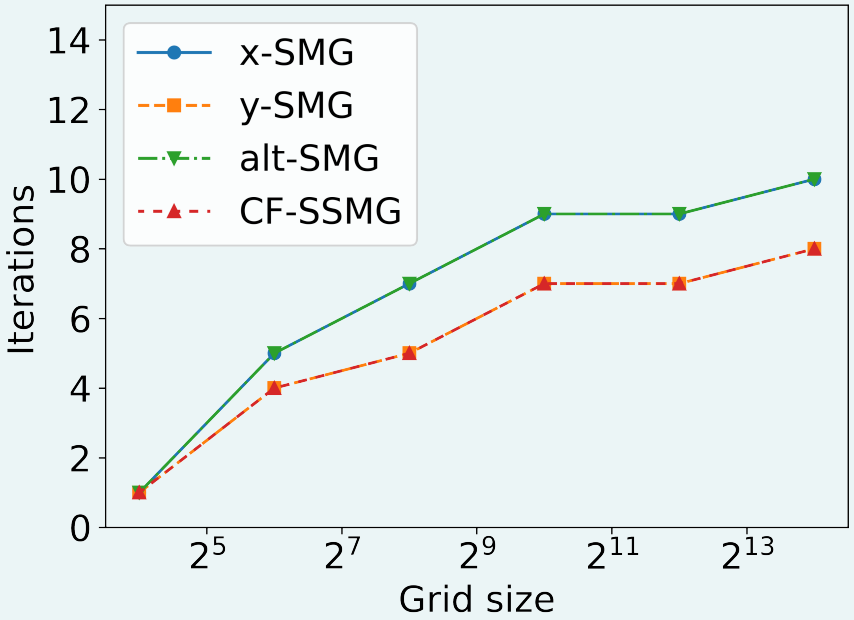
- To do this, we use "smart" coarsening
  1. Reconstruct a and b.
  2. Coarsen according to the following:

$$\begin{cases} \text{Coarsen in } x, & \text{if a} \geq b \\ \text{Coarsen in } y, & \text{otherwise} \end{cases}$$

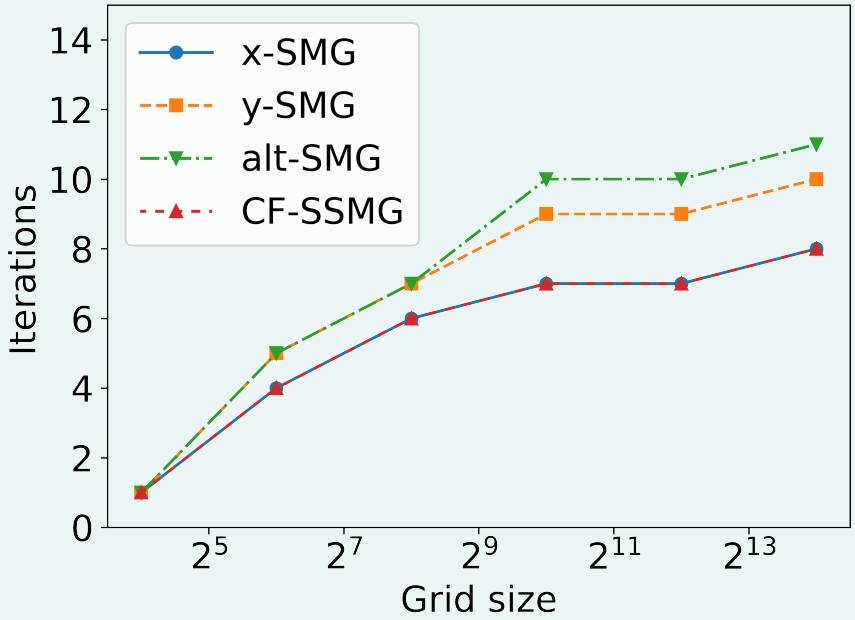
- Ensures diffusion term is isotropic.
- Correctly coarsens in the case of strong anisotropy.

Semi-Coarsening Comparison

- Anisotropic in y diffusion:
  $A = -\epsilon u_{xx} - u_{yy}$



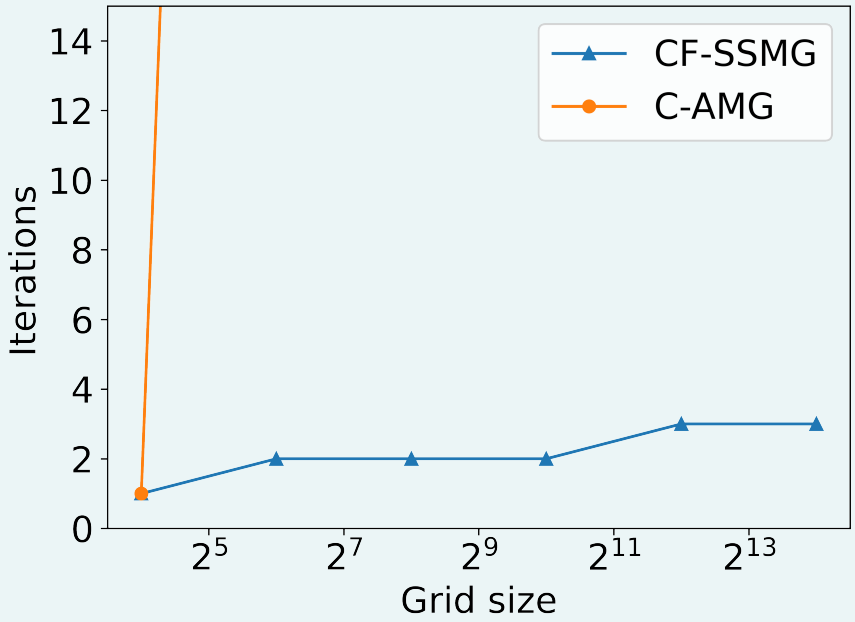
- Anisotropic in x diffusion:
  $A = -u_{xx} - \epsilon u_{yy}$



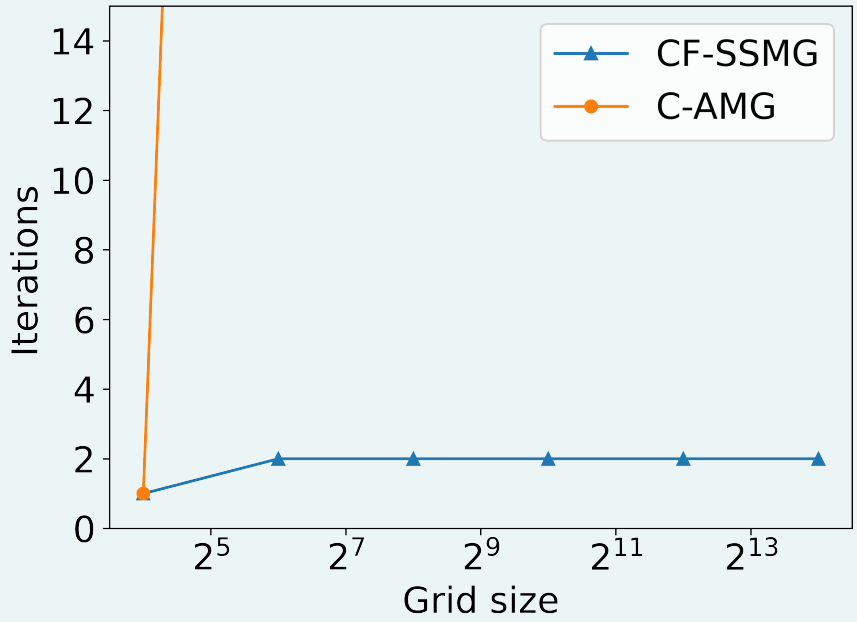
Results

Comparing our algorithm (CF-SSMG) against the prior state of the art algorithm (C-AMG).

- Fourth Order Term:
  $-\epsilon u_{xx} - \epsilon u_{yy} + u_{yyxx} = 0$



- COGENT Problem:
  $-0.0003u_{xx} - 2u_{yy} + 2u_{yyxx} = 0$



QUESTIONS?