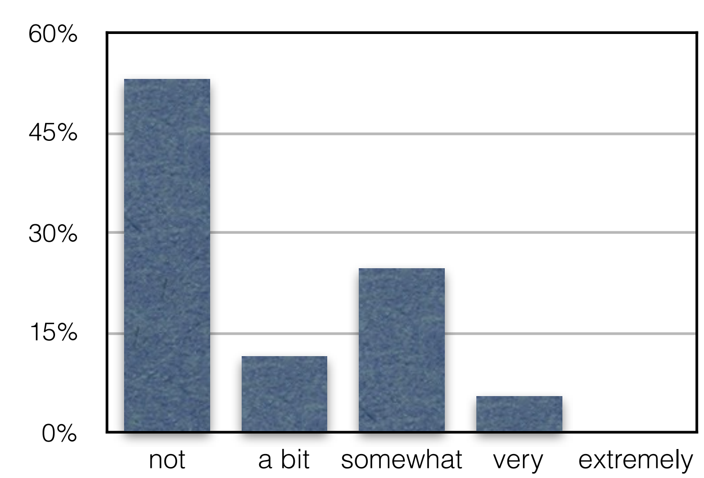
- Muddiest Points from Class 02/08
  - *"I'm a bit confused how the multi-grid is any good once you get to a very few elements, like the lowest possible, two elements. I understand that would be very fast, but at that level is it even producing any useful results?"*
  - *"Are we forsaking accuracy when we getting higher speed with the coarser mesh? If so why is accuracy not as important?"*
  - *"To clarify: we're using the finer mesh to help minimize the error while using the coarser mesh to help increase the speed at which convergence occurs?"*
    – Multi-grid is exactly that: use coarser meshes to reduce the large wave number residuals quickly and use finer meshes to reduce the truncation errors to obtain an accurate solution quickly
  - *"What is the exact meaning of "Don't code with matrices" ? When we are solving Aφ=b, we set φ as an array to calculate. Is this what your mean?"*
    – Do not think of arrays as matrices.
    – Don't code with matrices just means you should not solve systems of linear equations (A**φ**=**b**) setting up a matrix A and using matrix operations
  - *"I recall you had mentioned that we would only be doing the multi grid method in 1D. How do the double indices come into play in a 1D application?"*
    – I should have been more precise: We will only do multi-grid in 1D using the mesh type we covered so far.
    – We will do 2D multi-grid for a new mesh type we will cover today.
  - *"How does this all lead int fluid analysis?"*
    – Poisson equations will enable us to calculate the pressure and stream function for fluid flows (later in the semester)



Multigrid

- How to code this?
  - write iteration, prolongation, restriction as subroutines/functions
  - could use recursive calls with dynamic memory allocations

  OR

  - pre-compute and store grid level support data for all grid levels in vectors
    `M(1:p), N(1:p), h(1:p)`
  - do not make new arrays/variables for each grid level, instead use

    `rhs (0:M(1),0:N(1),1:p):` $\vec{f}$, $\vec{r}^{h\to 2h}$, $\vec{r}^{2h\to 4h}$, ...

    `r   (0:M(1),0:N(1),1:p):` $\vec{r}^{h}$, $\vec{r}^{2h}$, ...

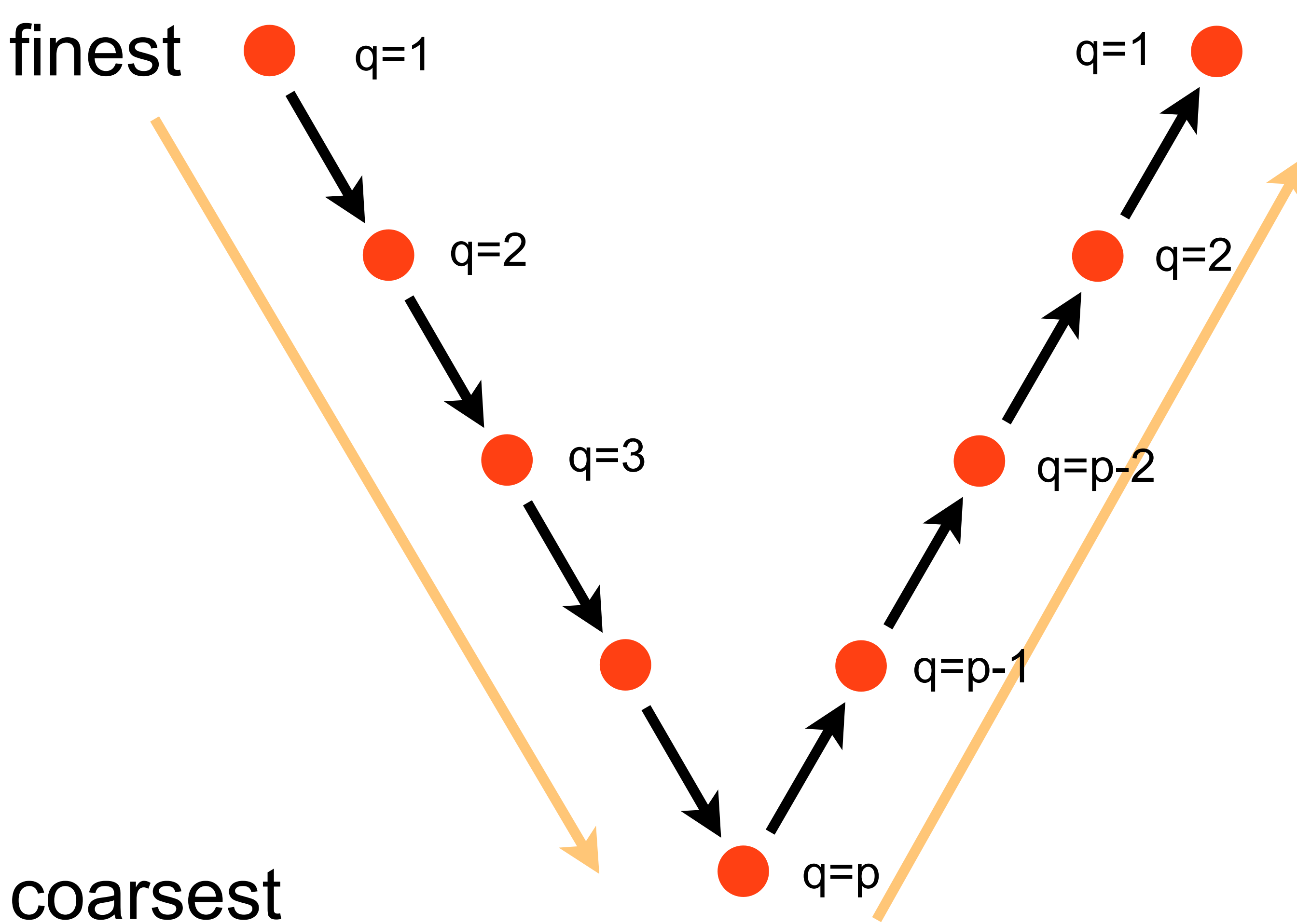
    `eps (0:M(1),0:N(1),1:p):` — , $\vec{\epsilon}^{2h}$, $\vec{\epsilon}^{4h}$, ...

    `epsc(0:M(1),0:N(1),1:p):` $\vec{\epsilon}^{2h\to h}$, $\vec{\epsilon}^{4h\to 2h}$, ...

  - this wastes some memory but makes coding easier

Multigrid: V-cycle

- How to code single V-cycle iteration?



V-cycle
1 iteration

```
eps     = 0.0
phi     = GaussSeidel (phi,rhs(:,1),h(1),M(1))
r(:,1) = calcResidual(phi,rhs(:,1),h(1),M(1))
```

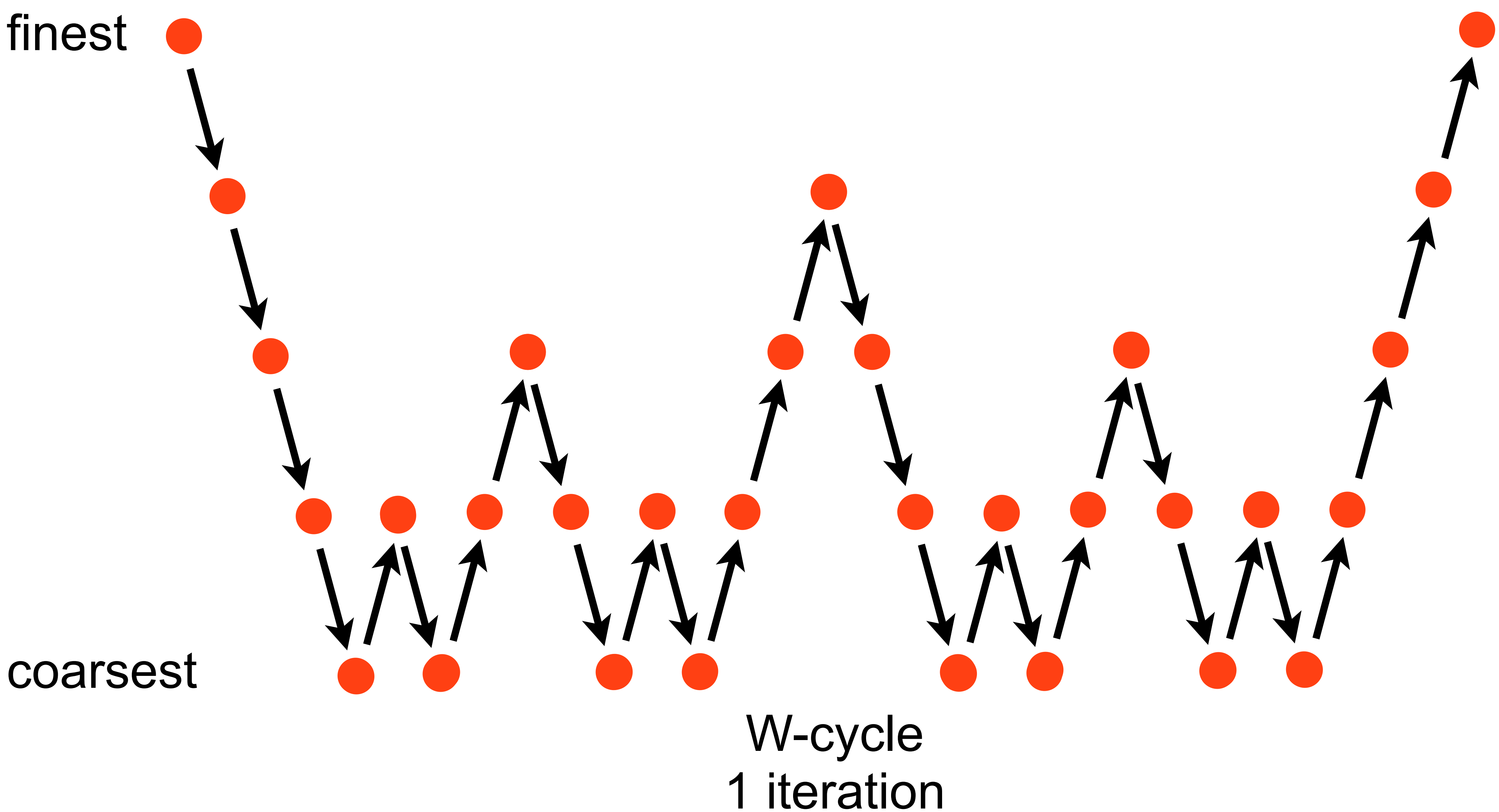
```
loop q from 2 to p
   rhs(:,q) = restrict     (r(:,q-1),M(q))
   eps(:,q) = GaussSeidel (eps(:,q),rhs(:,q),
                           h(q),M(q))
   r  (:,q) = calcResidual(eps(:,q),rhs(:,q),
                           h(q),M(q))
end loop q
```

```
loop q from p-1 to 2
   epsc(:,q) = prolong(eps(:,q+1),M(q))
   eps (:,q) = correct(eps(:,q),epsc(:,q),M(q))
   eps (:,q) = GaussSeidel(eps(:,q),rhs(:,q),
                           h(q),M(q))
end loop q
```

```
epsc(:,1) = prolong(eps(:,2),M(1))
phi       = correct(phi,epsc(:,1),M(1))
```

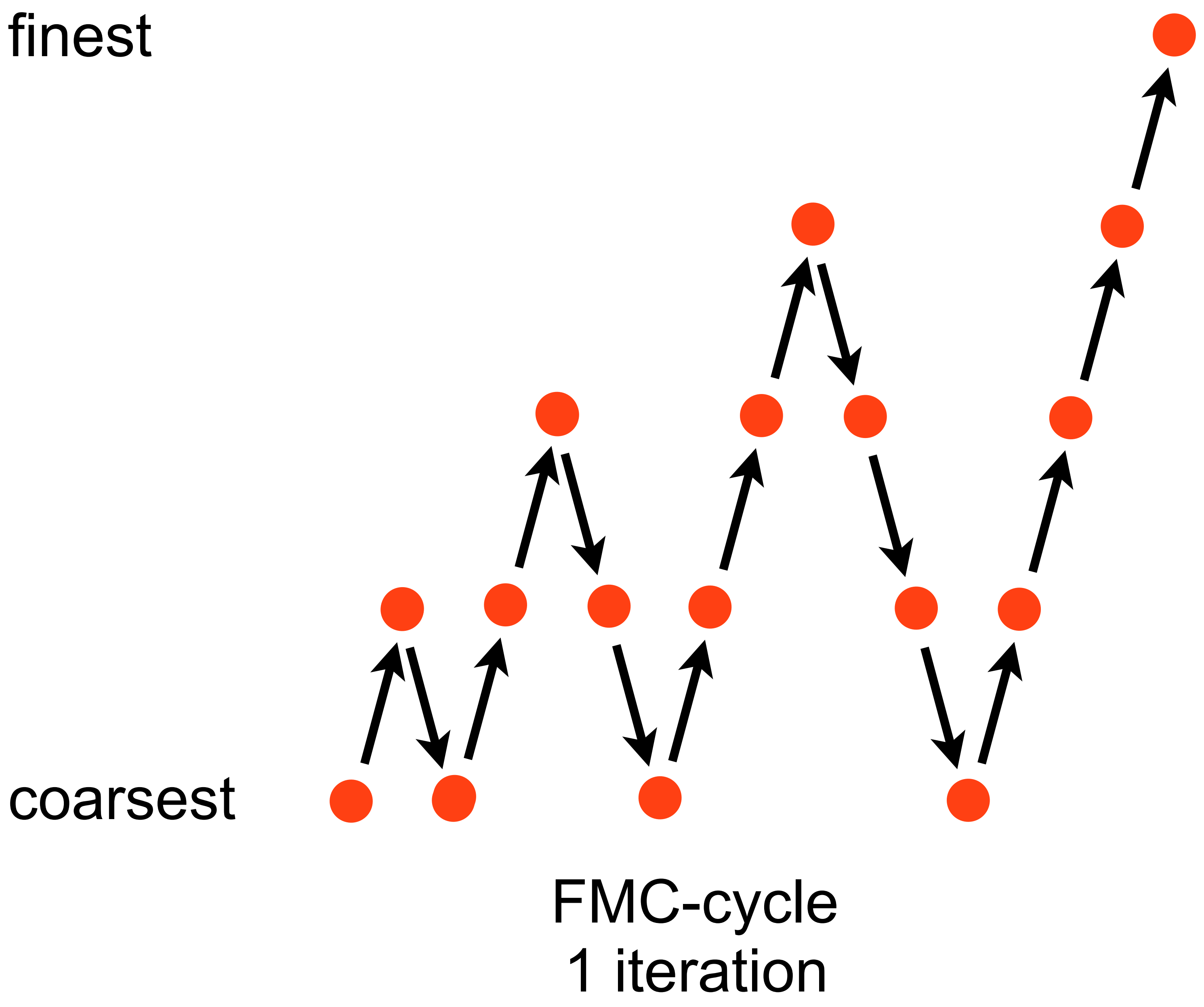
comment: `rhs(:,1)` must contain PDE right hand side

Multigrid: W-Cycle



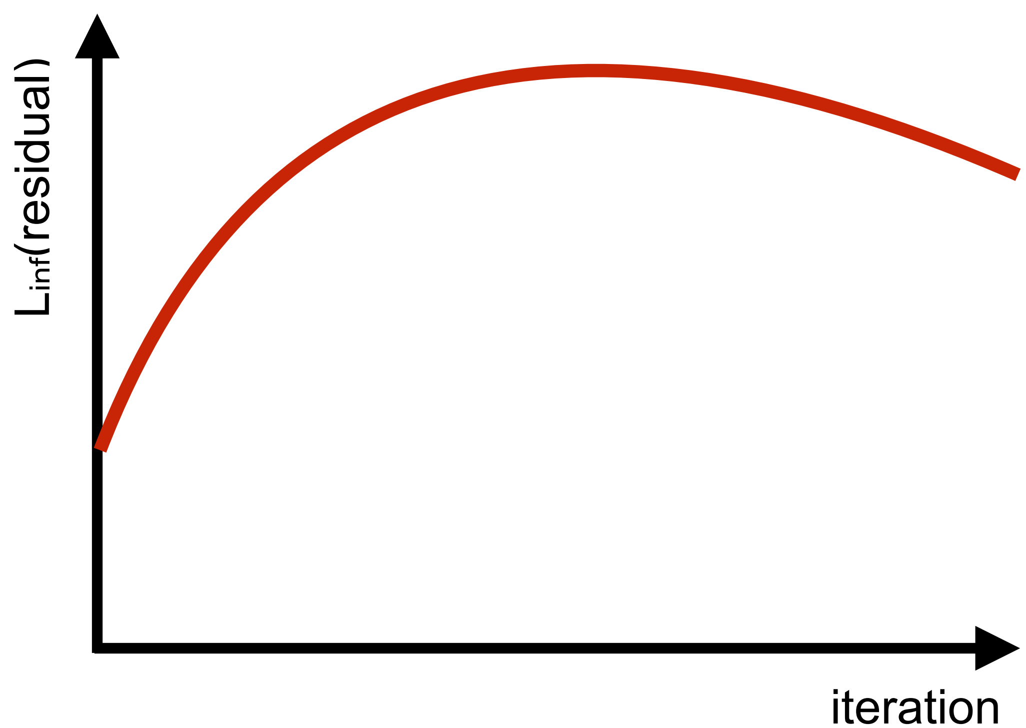
Multigrid: Full Multigrid Cycle

- start at coarsest grid level



- Challenge Question:

For an iterative method, e.g., Gauss Seidel, the following L_{inf}(residual) vs iteration plot is obtained



Does the increase in residual norm indicate there is an error in the code?

A: Yes

B: No

Show of hands

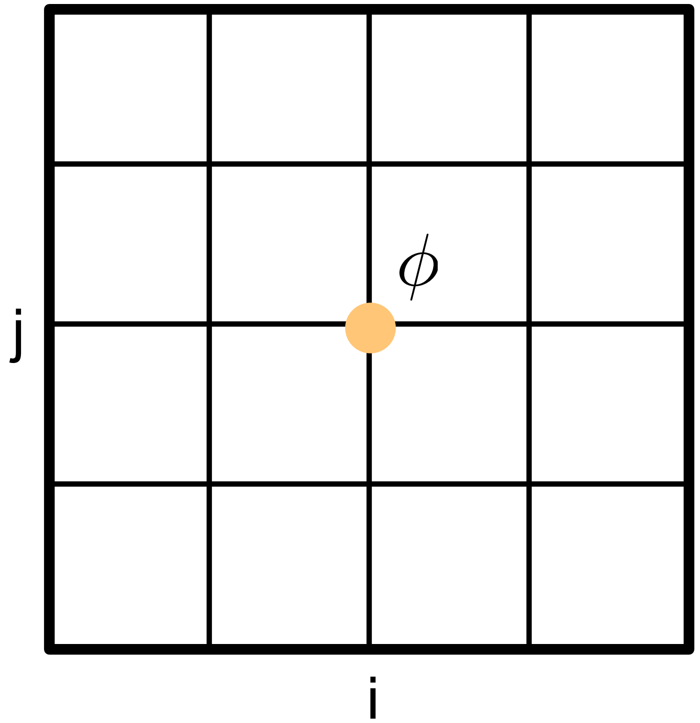
Discuss (1-2 mins)

Show of hands

Next: need to revisit meshing

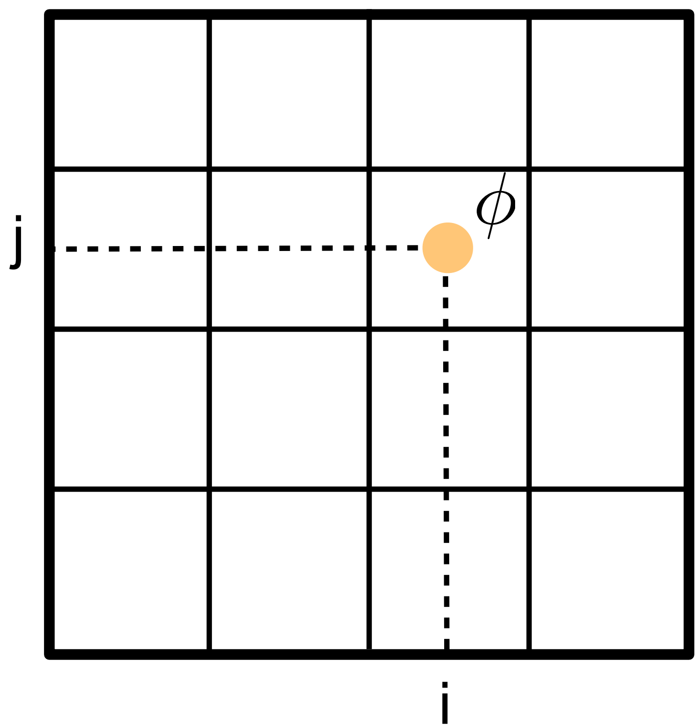
- until now, we have used the following meshes
  - variables are located at the intersection of grid lines

node based mesh



  - but, we could also locate variables @ cell centers!

cell centered mesh

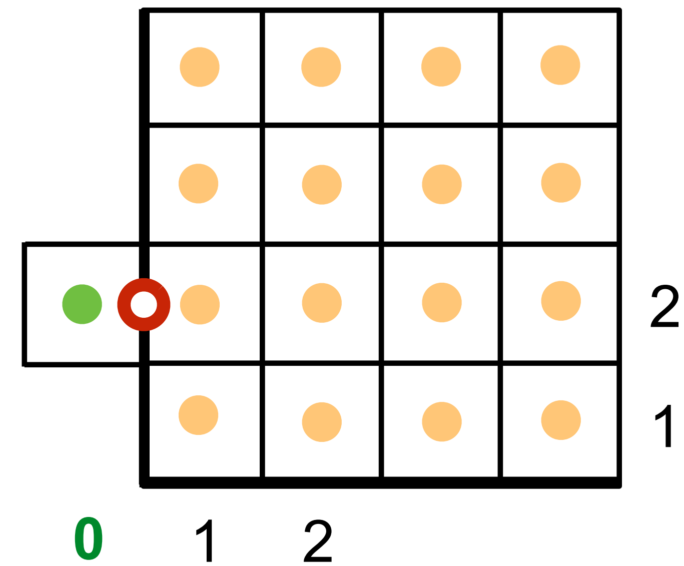


  - index i,j refers to cell (element) center

How do cell centered meshes impact boundary conditions?

- **Dirichlet boundary:**
  - there's no longer a variable located on the boundary to set to the given Dirichlet value
  - Trick: add a "virtual" **ghost cell** outside the boundary
  - choose the ghost cells' value such that an interpolation to the boundary location with appropriate order is equal to the Dirichlet value
    - Example: 2nd-order

$$\phi_{bc,j} = \frac{\phi_{0,j} + \phi_{1,j}}{2} \quad \Rightarrow \quad \phi_{0,j} = 2\phi_{bc,j} - \phi_{1,j}$$

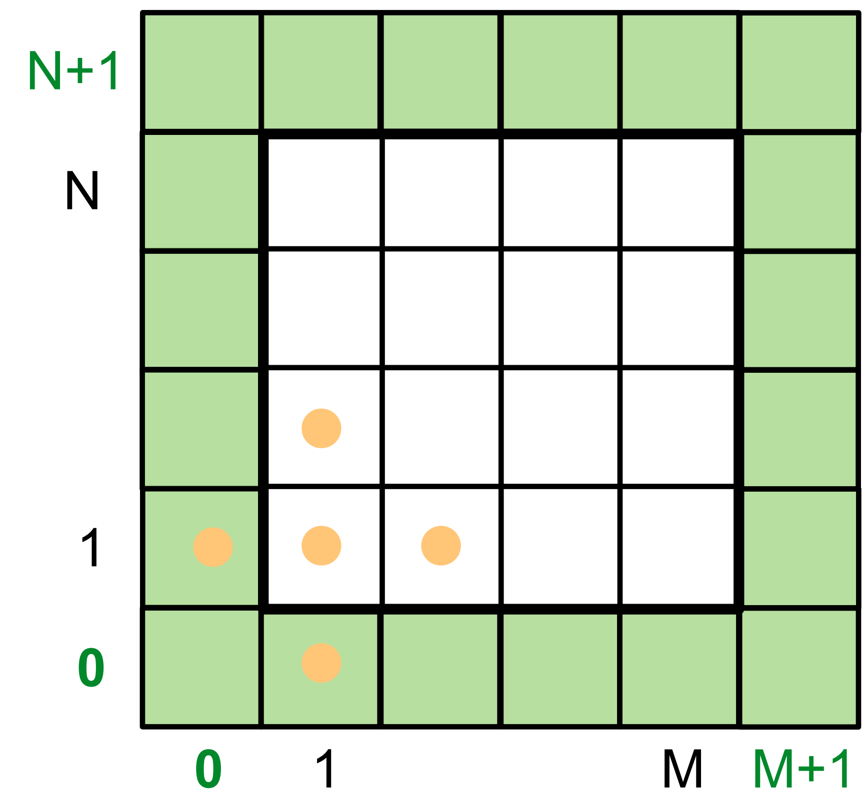


  - extends the mesh by a layer of ghost cells all around

```
phi(0:M+1,0:N+1)
```

  - Benefit: can use regular stencil even adjacent to boundaries with ghost cell values

```
for j=1:N
   for i=1:M
```

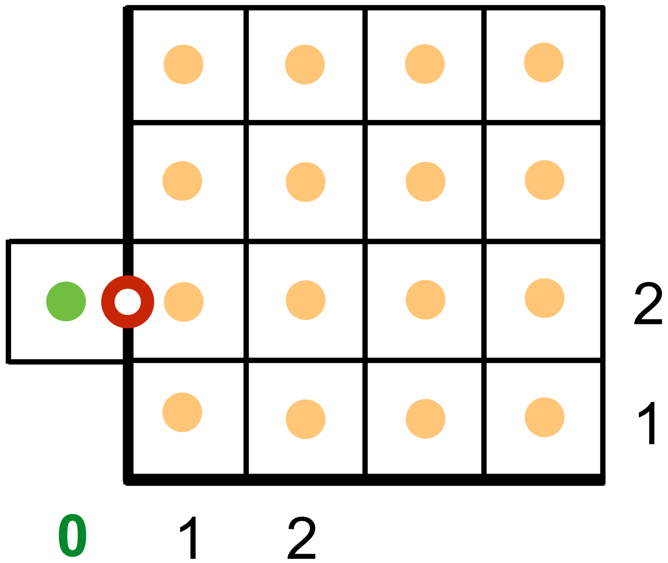


How do cell centered meshes impact boundary conditions?

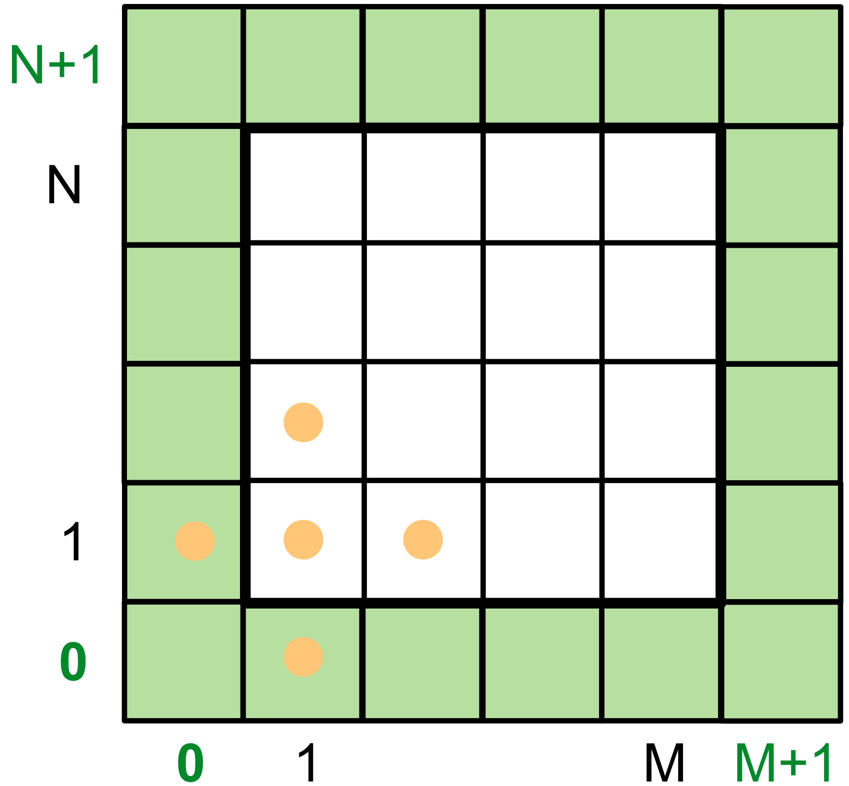
- **Neumann boundary:**
  - Trick: use ghost cell value to calculate derivative on the boundary
    - Example: 2nd-order

$$g = \left.\frac{\partial \phi}{\partial x}\right|_{bc,j} = \frac{\phi_{1,j} - \phi_{0,j}}{2\,\frac{h}{2}} + O(h^2)$$

$$\Rightarrow \quad \phi_{0,j} = \phi_{1,j} - hg$$



    - this sets the ghost cell value!
  - Again: can use regular stencil even adjacent to boundaries with ghost cell values
  - for higher order, add additional ghost cells

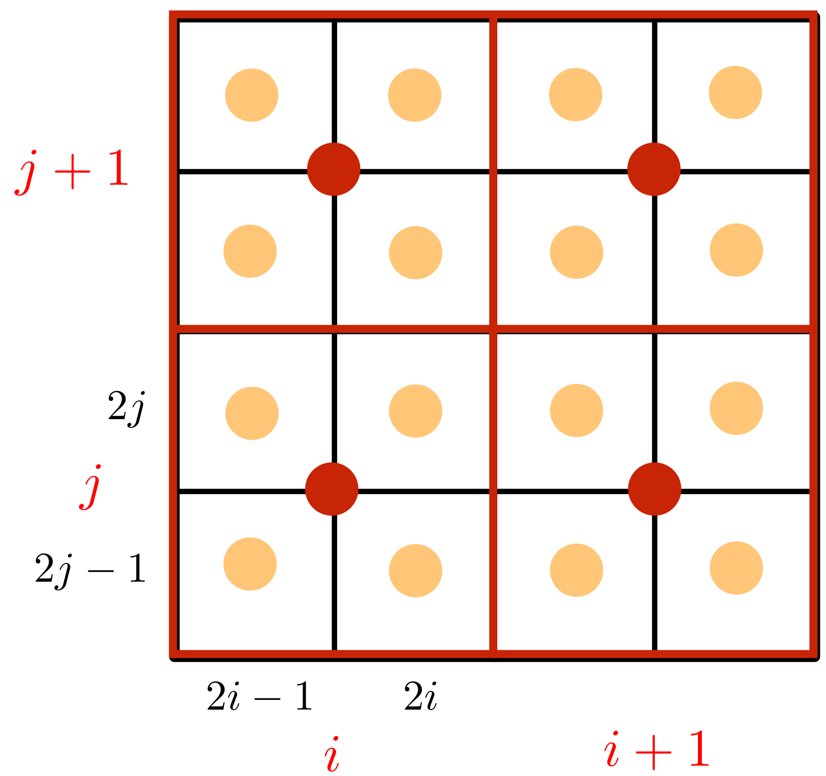


- **Solution procedure for cell centered meshes**
  - update interior cells (j=1:N, i=1:M)
  - after all interior cells are updated, directly calculate ghost cell values with updated interior values

- Small drawback: ghost cells in Gauss-Seidel are not updated and thus may lag one iteration

How do cell centered meshes impact Multigrid methods?

- **Prolongation**



here: i,j are coarse grid indices

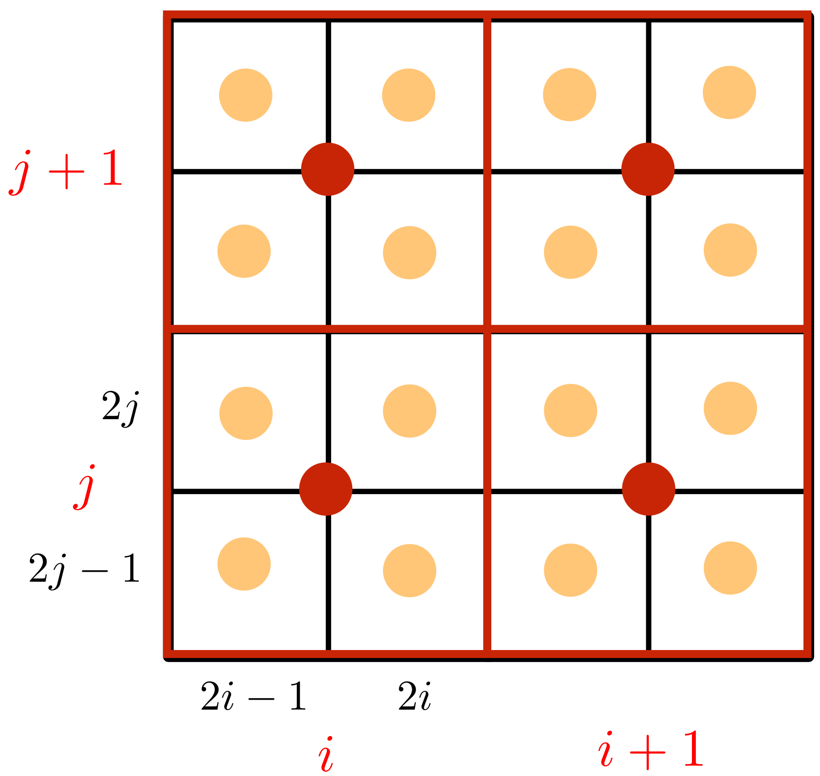
- Option #1: Constant “interpolation”

$$\epsilon^{2h\rightarrow h}_{2i-1:2i,2j-1:2j} = \epsilon^{2h}_{i} \qquad i = 1, 2, \ldots, M^{2h}\;,\; j = 1, 2, \ldots, M^{2h}$$

- Option #2: Bilinear interpolation

How do cell centered meshes impact Multigrid methods?

- **Restriction** (needs to be adjoint of Prolongation)



here: i,j are coarse grid indices

- Option #1: Adjoint to constant “interpolation”

$$r_{i,j}^{h\rightarrow 2h} = \frac{1}{4} \sum_{j'=2j-1}^{2j} \sum_{i'=2i-1}^{2i} r_{i',j'}^{h} \qquad i = 1,2,\ldots,M^{2h}\,,\; j = 1,2,\ldots,M^{2h}$$

- Option #2: Adjoint to bilinear interpolation

- Finally, a comment on Poisson equation with all Neumann boundaries

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = f(x,y) \qquad\qquad \left.\frac{\partial \phi}{\partial n}\right|_{bc} = g(x,y)$$

  - if $\phi(x,y)$ is a solution, so is $\phi(x,y) + const$
  - iterative solution may "drift"
  - this is usually not a problem for convergence checks, since these use the residual

$$r(x,y) = f(x,y) - \left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}\right)$$

  - but, excessive "drift" may cause finite precision problems, since it can lead to differences of large numbers
  - Fix: subtract the mean of ϕ from ϕ after convergence or after some number of iterations

$$\phi_{i,j} \rightarrow \phi_{i,j} - \frac{1}{MN}\sum_{j=1}^{N}\sum_{i=1}^{M}\phi_{i,j}$$

- Challenge Question:

Solve $\frac{\partial^2 \varphi}{\partial x^2} = \sin(x)$ on domain $0 \leq x \leq 2\pi$ with bc $\varphi(0) = \varphi(2\pi) = 0$

with second order central differences using Gauss-Seidel and initial guess $\varphi^{(0)} = 0$

Question: Is the exact solution to the PDE $\varphi(x) = -\sin(x)$ the solution to the Gauss-Seidel method after infinitely many iterations?

A: Yes B: No C: No Idea

Show of Hands

Discuss (1-2mins). (also discuss why)

Show of Hands