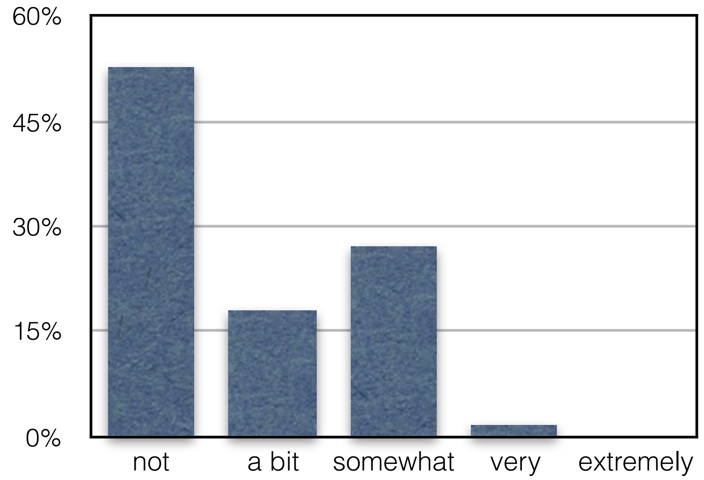
Muddiest Points from Class 02/06

- *"Just to clarify, we need to calculate the Point Jacobi eigenvalues in order to calculate the eigenvalues for GSOR?"*
  – Yes. But in practice all you need to do is calculate ω.

- *"Is fi,j or bi,j something that is always given or is it determined via gaussian elimination?"*
  – The right-hand-side of the PDEs are always given or can be calculated from known values.

- *"When breaking the matrix into upper, lower and diagonal components, shouldn't the decomposition be of the form A = D + L + U? The choice of A1 and A2 taken in class results in A = D - L - U. Or is it that the definition of L and U reverse from what I am assuming?"*
  – I very likely misspoke in class. L and U are the minus lower, respective minus upper triangular parts of A, excluding the diagonals (the slides themselves are correct).
  – So $A = D - L - U$ is actually correct



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

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = f$$

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

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

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = f$$

Multigrid Acceleration

$$A\vec{\varphi}^{\,(n)} = \vec{f} - \vec{r}^{\,(n)} \qquad \textit{(ii)}$$

- Residual in matrix form (never code this!)

$$\vec{r} = \vec{f} - A\vec{\varphi}^{(n)}$$

(A includes $1/\Delta^2$)

- Residual in index form (code this!)

$$r_{i,j} = f_{i,j} - \left[\left(\frac{\partial^2 \varphi}{\partial x^2} \right)_{i,j} + \left(\frac{\partial^2 \varphi}{\partial y^2} \right)_{i,j} \right]$$

use finite differences for derivatives

- What to do at Dirichlet boundaries?

error is zero, thus residual is zero!

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: $\varphi(x) = -\dfrac{1}{k^2\pi^2}\sin(k\pi x)$

- define mesh: $M+1$ points

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

- use 2nd-order central differences

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

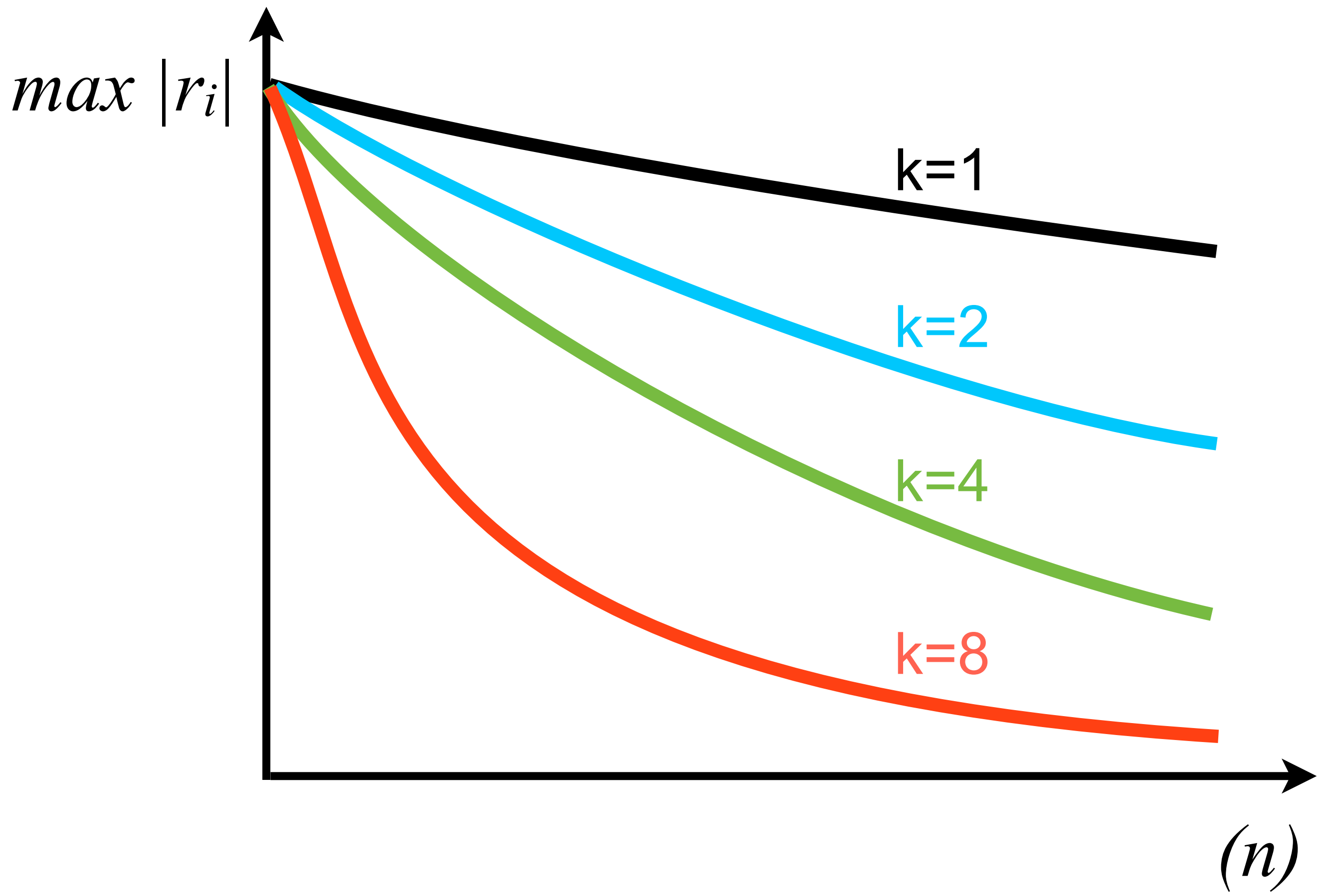
- use $\vec{\varphi} = \vec{0}$ as initial guess $\Rightarrow \quad \vec{r}^{(0)} = \vec{f} - A\vec{\varphi}^{(0)} = \vec{f}$

- solve with Gauss-Seidel

code example MG1

Example #1:

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



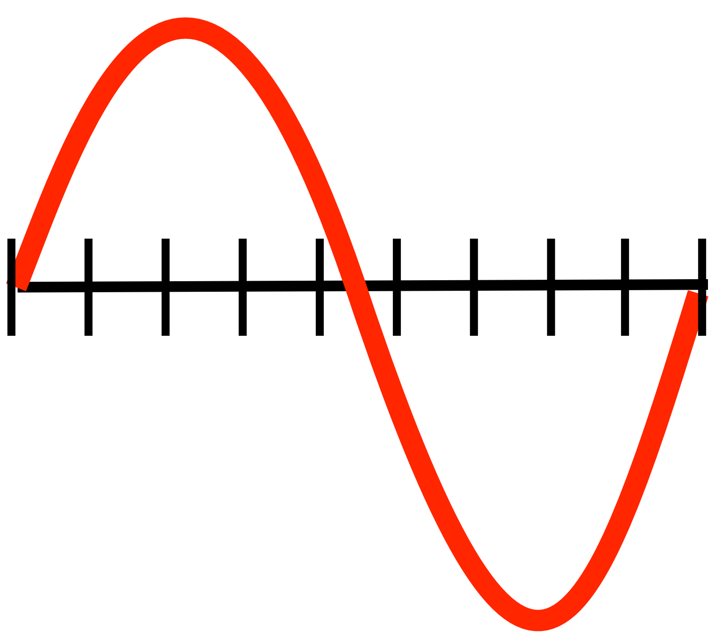
$\Rightarrow$ larger k converge faster

Example #2:

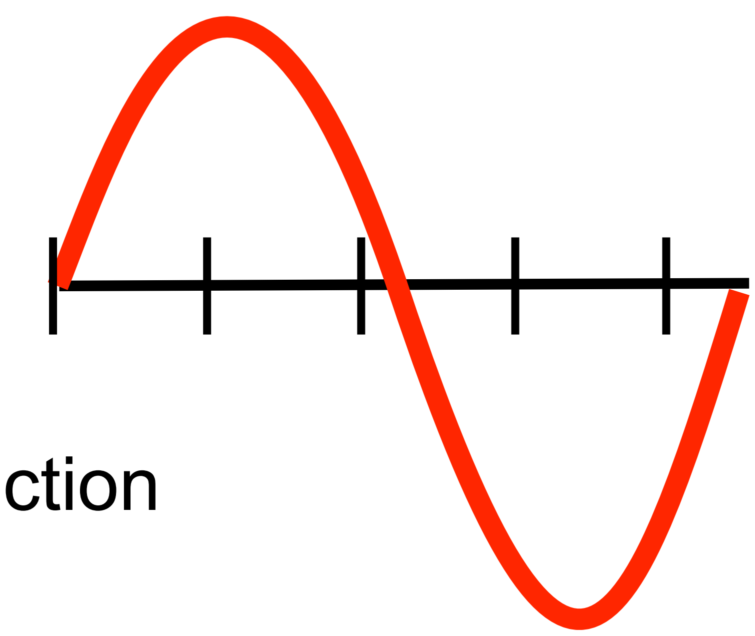
code example MG2 & MG3

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

- 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!



same function



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

Multigrid Methods

- Let $k = k_m$ with $1 \le \frac{k_m}{2\pi} \le \frac{M}{4}$

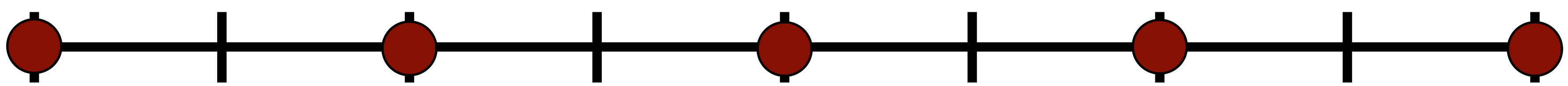
  - need 2 points/wavelength minimum

    $\Rightarrow \frac{k_{\max}}{2\pi} = \frac{M}{2}$

  - lower half of allowed wavenumbers

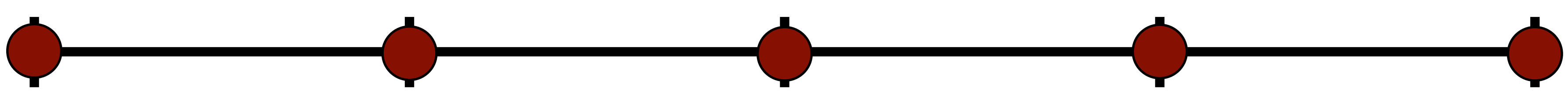
    $\Rightarrow$ **slowly varying**

- Let's evaluate $\sin(k_m x_i)$ at even index points only: $x_i = 2i\frac{L}{M} = \frac{2i}{M}$



$$\Rightarrow \quad \sin(k_m x_i) = \sin\left(\frac{k_m 2i}{M}\right) = \sin\left(\frac{k_m i}{\frac{M}{2}}\right)$$

  - same as function evaluated at **every** point of mesh with spacing $\frac{L}{\frac{M}{2}}$



  - on mesh with $M/2+1$ mesh points, function goes up to the maximum allowed wavenumber $\Rightarrow$ **rapidly varying**

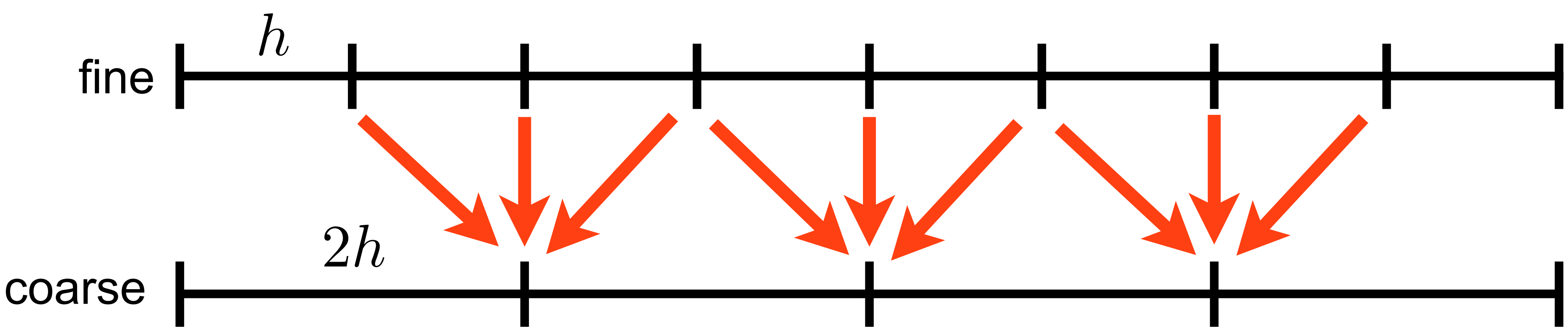
- Can turn slowly varying function into rapidly varying function by coarsening mesh ('skipping' mesh points)

Multigrid Methods

- Ideas behind Multigrid Methods
  - reduce slowly varying parts of residual on coarser grids, since they are rapidly varying there
  - transfer improved solution from coarse grid back to fine grid

- This will require transfer operations between grids
  - fine ➙ coarse: **restriction**
  - coarse ➙ fine: **prolongation**

Restriction



$M^h + 1$ points

$M^{2h} + 1$ points

$$M^{2h} = \frac{M^h}{2}$$

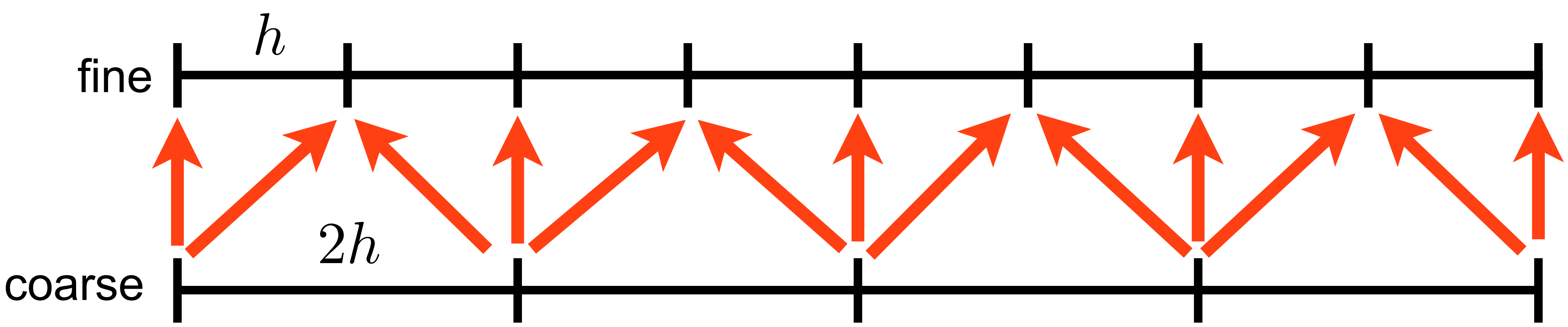
- Option #1:
  - take every 2nd point

$$r_i^{h\rightarrow 2h} = r_{2i}^h \qquad i = 1, 2, \ldots, M^{2h} - 1 \qquad \text{(all interior points)}$$

- Option #2:
  - average (usually better)

$$r_i^{h\rightarrow 2h} = \frac{1}{4}\left(r_{2i-1}^h + 2r_{2i}^h + r_{2i+1}^h\right) \qquad i = 1, 2, \ldots, M^{2h} - 1$$

Prolongation



$M^h + 1$ points

$M^{2h} + 1$ points

$$M^{2h} = \frac{M^h}{2}$$

- simple copy of aligned mesh points

$$\epsilon_{2i}^{2h \to h} = \epsilon_i^{2h} \qquad i = 0, 1, \ldots, M^{2h}$$

- average non-aligned mesh points

$$\epsilon_{2i+1}^{2h \to h} = \frac{1}{2}\left(\epsilon_i^{2h} + \epsilon_{i+1}^{2h}\right) \qquad i = 0, 1, \ldots, M^{2h} - 1$$

Multigrid Methods

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = f$$

- Ideas behind Multigrid Methods
  - reduce slowly varying parts of residual on coarser grids, since they are rapidly varying there
  - transfer improved solution from coarse grid back to fine grid

- How can we do this in practice?

  Let $\varphi_{i,j}^{(k)}$ be the estimate to the solution $\varphi(x_i, y_j)$ after k iterations

  How can we improve this estimate?

  - do another iteration step to k+1, or
  - what about if we add the error $e_{i,j}^{(k)}$ to $\varphi_{i,j}^{(k)}$?

$$e_{i,j}^{(k)} = \varphi(x_i, y_j) - \varphi_{i,j}^{(k)} \quad \Rightarrow \quad \varphi_{i,j}^{(k)} + e_{i,j}^{(k)} = \varphi(x_i, y_j)$$

if we knew the error we could calculate the exact solution

the entire point of multigrid is to estimate the error on the finest mesh as quickly as possible

Multigrid Methods

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = f$$

the entire point of multigrid is to estimate the error on the finest mesh as quickly as possible

$$e_{i,j}^{(k)} = \varphi(x_i, y_j) - \varphi_{i,j}^{(k)}$$

how can we calculate the error? Let's take the Laplacian

$$\nabla^2 e_{i,j}^{(k)} = \nabla^2 \varphi(x_i, y_j) - \nabla^2 \varphi_{i,j}^{(k)}$$

$$\left(\frac{\partial^2 e^{(k)}}{\partial x^2}\right)_{i,j} + \left(\frac{\partial^2 e^{(k)}}{\partial y^2}\right)_{i,j} = f(x_i, y_j) - \left[\left(\frac{\partial^2 \varphi^{(k)}}{\partial x^2}\right)_{i,j} + \left(\frac{\partial^2 \varphi^{(k)}}{\partial y^2}\right)_{i,j}\right]$$

$$\left(\frac{\partial^2 e^{(k)}}{\partial x^2}\right)_{i,j} + \left(\frac{\partial^2 e^{(k)}}{\partial y^2}\right)_{i,j} = r_{i,j}^{(k)}$$

$$\frac{e_{i+1,j}^{(k)} - 2e_{i,j}^{(k)} + e_{i-1,j}^{(k)}}{h^2} + \frac{e_{i,j+1}^{(k)} - 2e_{i,j}^{(k)} + e_{i,j-1}^{(k)}}{h^2} = r_{i,j}^{(k)}$$

solve with Gauss-Seidel:

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

Multigrid Methods

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = f$$

the entire point of multigrid is to estimate the error on the finest mesh as quickly as possible

$$e_{i,j}^{(k)} = \varphi(x_i, y_j) - \varphi_{i,j}^{(k)}$$

how can we calculate the error?

$$\frac{e_{i+1,j}^{(k)} - 2e_{i,j}^{(k)} + e_{i-1,j}^{(k)}}{h^2} + \frac{e_{i,j+1}^{(k)} - 2e_{i,j}^{(k)} + e_{i,j-1}^{(k)}}{h^2} = r_{i,j}^{(k)}$$

solve with Gauss-Seidel:

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

But, instead of solving this on the fine mesh h, solve it on a factor 2 coarser mesh 2h to speed up convergence

$$e_{i,j}^{2h^{(k+1)}} = \frac{1}{4}\left(e_{i+1,j}^{2h^{(k)}} + e_{i-1,j}^{2h^{(k+1)}} + e_{i,j+1}^{2h^{(k)}} + e_{i,j-1}^{2h^{(k+1)}}\right) - \frac{1}{4}(2h)^2 r_{i,j}^{h\to 2h^{(k)}}$$

need to restrict $r_{i,j}^{(k)}$ to $r_{i,j}^{h\to 2h^{(k)}}$

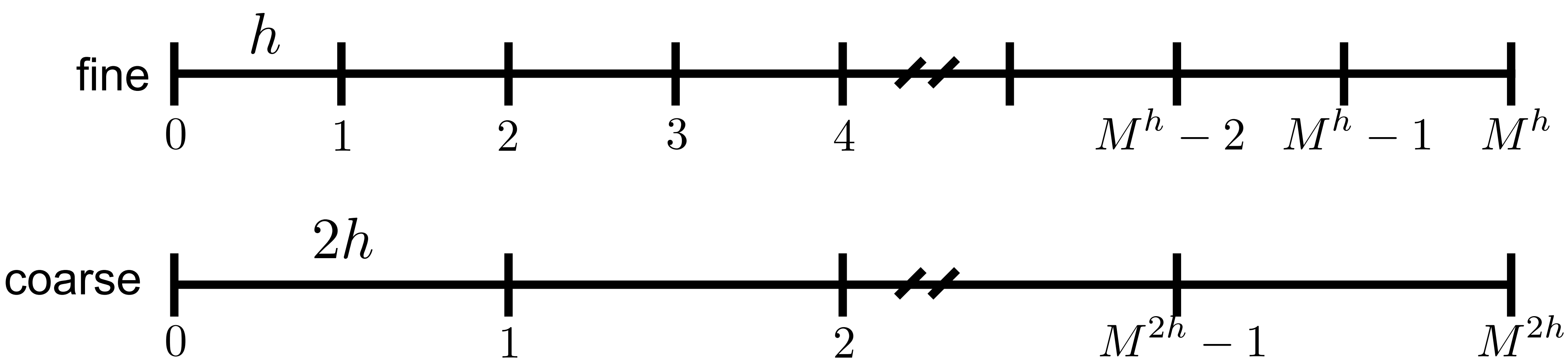
need initial guess: $e_{i,j}^{2h^{(0)}} = 0$

now iterate using Gauss-Seidel

after iterating, need to prolong solution $e_{i,j}^{2h^{(k)}}$ to $e_{i,j}^{2h\to h^{(k)}}$

finally improve/correct solution on fine mesh $\varphi_{i,j}^{(0)} = \varphi_{i,j}^{(k)} + e_{i,j}^{2h\to h^{(k)}}$

Dual Grid Multigrid in Matrix form (never code this)



$$M^{2h} = \frac{M^h}{2}$$

a) on fine gird, perform a few iterations for $A\vec{\varphi} = \vec{f}$ with $\vec{\varphi}^{h^{(0)}}$ as initial guess

b) calculate residual: $\vec{r}^h = \vec{f} - A\vec{\varphi}^{h^{(k)}}$

c) restrict residual to coarse grid: $\vec{r}^{h \to 2h}$

d) on coarse grid, perform a few iterations for $A\vec{\epsilon}^{2h} = \vec{r}^{h \to 2h}$ with $\vec{\epsilon}^{2h^{(0)}} = 0$

e) prolong error to fine grid: $\vec{\epsilon}^{2h \to h}$

f) apply correction to fine grid solution: $\vec{\varphi}^{h^{(0)}} = \vec{\varphi}^{h^{(k)}} + \vec{\epsilon}^{2h \to h}$

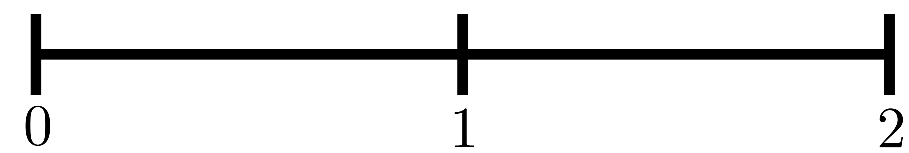
g) goto step a) until norm of residual drops below acceptable threshold

reminder: never code matrices, use loops with index form instead

Multigrid

- Why stop at 2 grids?
  - go all the way to coarsest possible mesh
  - possible if $M^h = 2^p$

$$M^{2^p h} = 2$$



Multigrid

a) on fine gird, perform a few iterations for $A\vec{\varphi} = \vec{f}$ with $\vec{\varphi}^{h^{(0)}}$ as initial guess

b) calculate residual: $\vec{r}^{h} = \vec{f} - A\vec{\varphi}^{h^{(k)}}$

c) restrict residual to coarse grid: $\vec{r}^{h\to 2h}$

d) on coarse grid, perform a few iterations for $A\vec{\epsilon}^{2h} = \vec{r}^{h\to 2h}$ with $\vec{\epsilon}^{2h^{(0)}} = 0$

- d.b) calculate residual to error equation: $\vec{r}^{2h} = \vec{r}^{h\to 2h} - A\vec{\epsilon}^{2h^{(k)}}$
- d.c) restrict residual to next coarser grid: $\vec{r}^{2h\to 4h}$
- d.d) on next coarser grid, perform a few iterations for $A\vec{\epsilon}^{4h} = \vec{r}^{2h\to 4h}$ with $\vec{\epsilon}^{4h^{(0)}} = 0$
  - d.d.b-d) continue with ...b) - ...d) on next coarser mesh $4h$ all the way to ph mesh
  - d.d.e-g) do steps ...e) - ...g) from coarsest mesh ph all the way to $4h$ mesh
- d.e) prolong error to next finer grid: $\vec{\epsilon}^{4h\to 2h}$
- d.f) apply correction to next finer grid solution (error): $\vec{\epsilon}^{2h^{(0)}} = \vec{\epsilon}^{2h^{(k)}} + \vec{\epsilon}^{4h\to 2h}$
- d.g) on next finer grid, perform a few iterations for $A\vec{\epsilon}^{2h} = \vec{r}^{h\to 2h}$

e) prolong error to fine grid: $\vec{\epsilon}^{2h\to h}$

f) apply correction to coarse grid solution: $\vec{\varphi}^{h^{(0)}} = \vec{\varphi}^{h^{(k)}} + \vec{\epsilon}^{2h\to h}$

g) goto step a) until norm of residual drops below acceptable threshold

Recursive algorithm (but we know the number of levels beforehand)

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):` $\vec{f}$, $\vec{r}^{h\rightarrow 2h}$, $\vec{r}^{2h\rightarrow 4h}$, ...

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

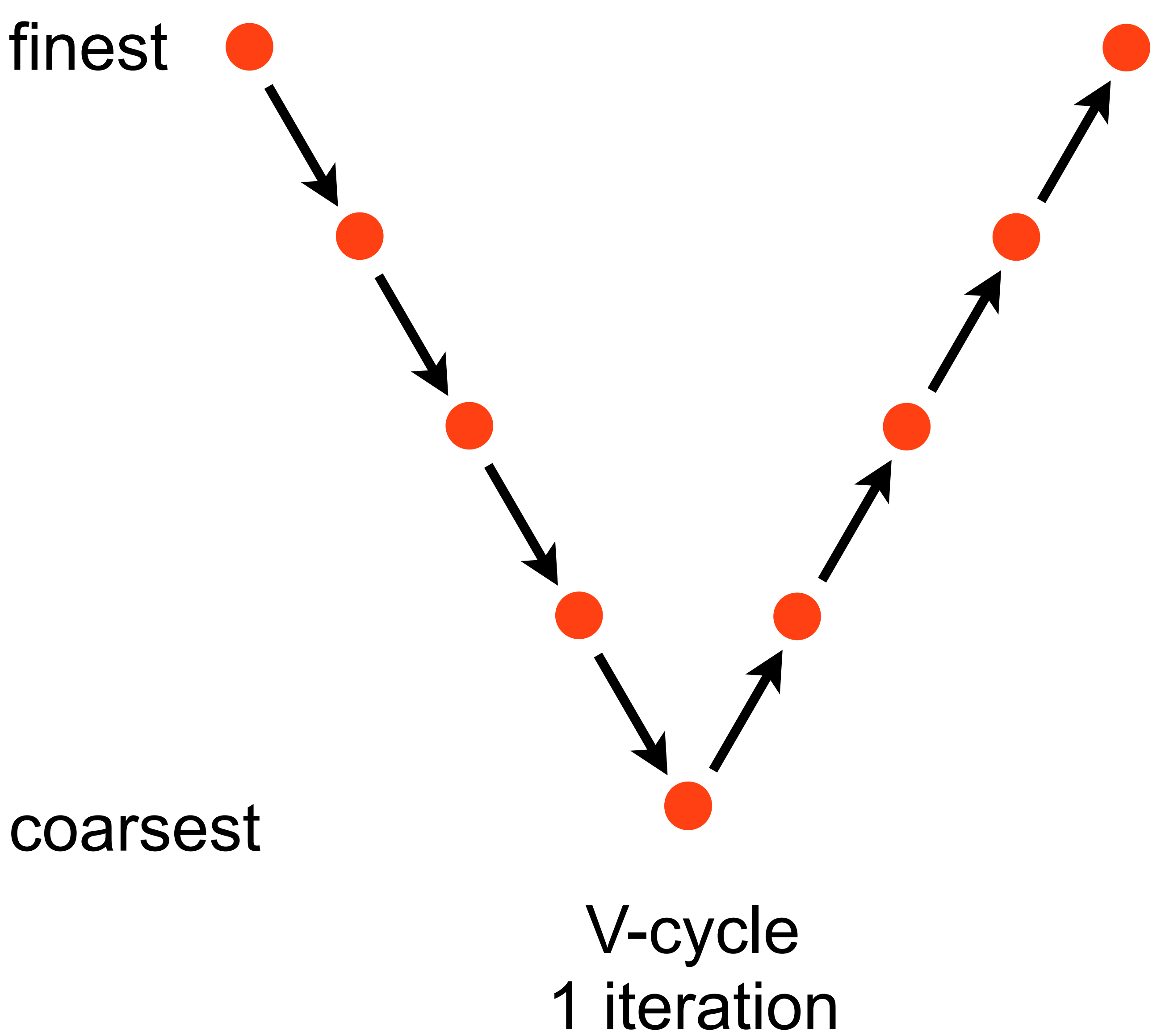
    `eps (0:M(1),0:N(1),1:p):` — , $\vec{\epsilon}^{2h}$, $\vec{\epsilon}^{4h}$, ...

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

  - this wastes some memory but makes coding easier

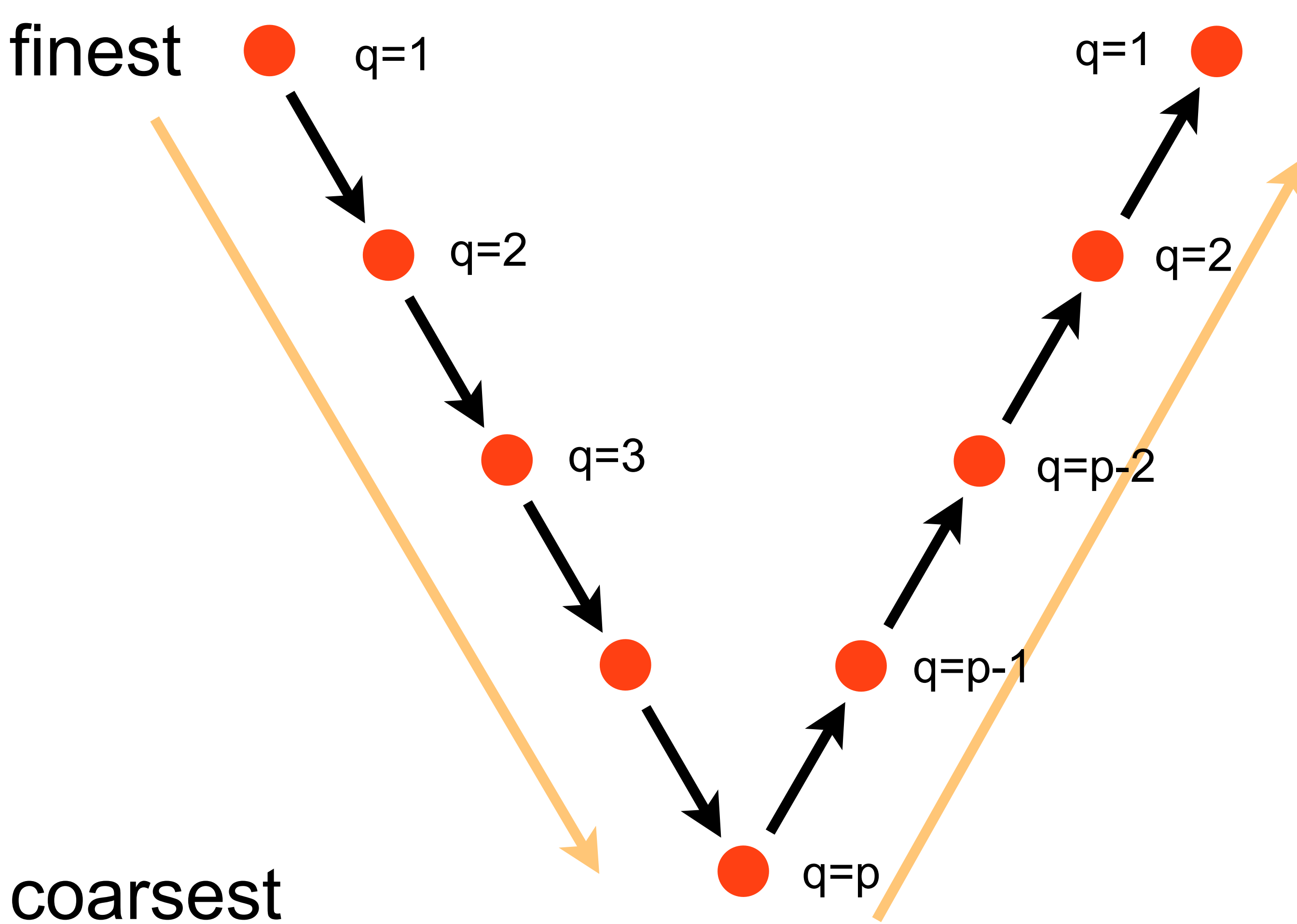
Multigrid: V-cycle

- How to traverse the different grid levels?
  - Many options!



Multigrid

- 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
```

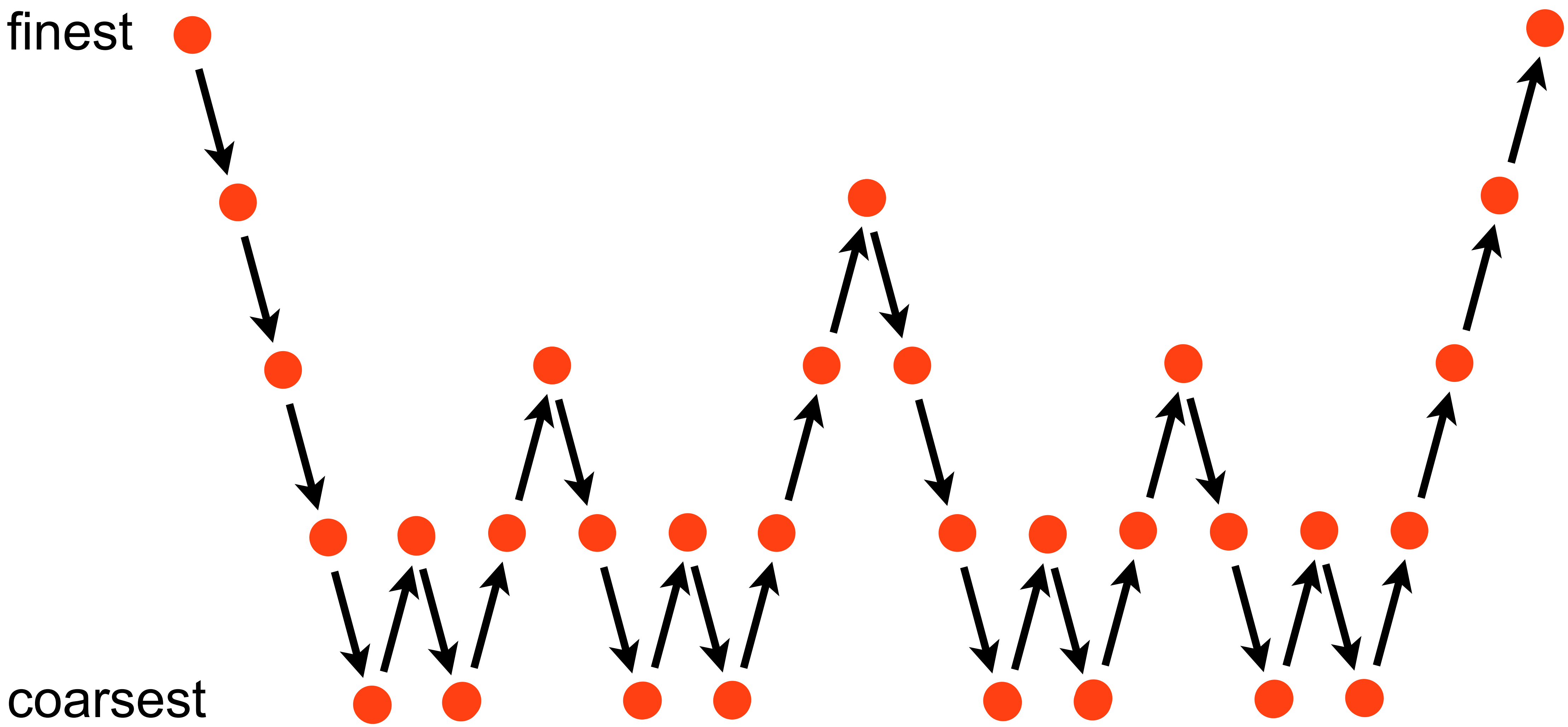
```
loop q from p-1 to 2
   epsc(:,q) = prolong(eps(:,q+1),M(q))
   eps (:,q) = correct(eps(:,q),epsc(:,q),M(q))
   eps (:,q) = GaussSeidel(eps(:,q),rhs(:,q),
                            h(q),M(q))
end loop q
```

```
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

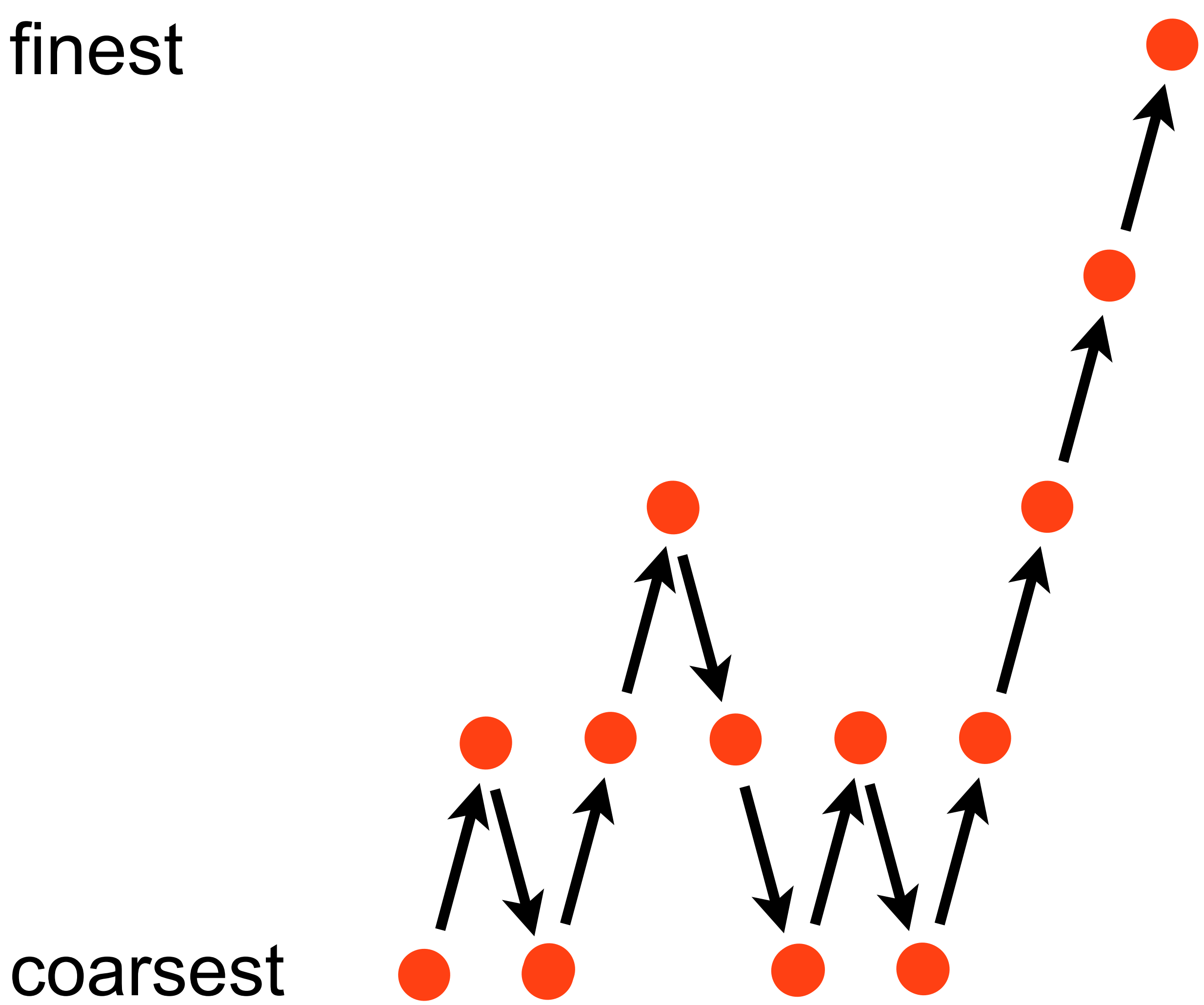
- How to traverse the different grid levels?



W-cycle
1 iteration

Multigrid: FMC

- Full Multi Grid cycle:
  - start at coarsest grid level



FMC-cycle
1 iteration