Practical Multigrid

Travis Whyte

MAP55672 Case Studies

Outline for this Module

1. A general overview of geometric multigrid and its ingredients to introduce notation and jargon
2. Some discussion of smoothers
3. The specifics of multigrid
4. A case study of multigrid for the 2D Discrete Laplacian
5. If we have time, I will introduce smooth aggregation multigrid for cases where geometric multigrid fails and show how we can use this as a preconditioner for Krylov subspace method

This is not meant to be a difficult module so we will spend some time during class working on exercises to illustrate the concepts and assemble the pieces of the assignment together

Getting Materials Etc

THe assignments, code and lecture slides available on github

git clone https://github.com/twLQCD/MAP55672_MG.git

To use Matlab on Chuck, use ssh with forwarding:

ssh -X user@chuck.tchpc.tcd.ie

and load the module with "module load matlab" and launch matlab by typing "matlab" in the command line

A General Overview of Multigrid

- Multigrid is a method to solve a system of linear equations

$$\boldsymbol{Ax} = \boldsymbol{b} \tag{1}$$

  much like methods you have seen before (Conjugate Gradient, GMRES, Gauss-Seidel etc)

- These iterative methods can converge slowly due to components of the error that are in the direction of the eigenvectors associated with small eigenvalues are slow to converge

- The basic idea of multigrid is to exploit a hierarchy of discretizations where the slow to converge components of the error become easier to deal with on coarser discretizations

A General Overview of Multigrid: Ingredients

- Say we have a matrix $\boldsymbol{A}$ coming from a fine discretization of a domain Ω (maybe this is through a finite difference of a PDE), and we want to solve Eq. (1)

- What are the ingredients that we need to make a multigrid solver?
  1. Our fine level matrix $\boldsymbol{A}$
  2. A coarser discretization
  3. A way to move between the original problem and the coarse problem $\rightarrow$ *transfer* matrices
  4. A matrix associated with the coarse discretization
  5. A smoother $\rightarrow$ A "solver" which exposes the slow to converge components of the error

Transfer Matrices

DISCLAIMER: Transfer matrices can be very different for each of the problems that you are working with. We will see these for a specific example

Let $\boldsymbol{A} \in \mathcal{R}^{n\times n}$ and $\boldsymbol{v} \in \mathcal{R}^{n\times 1}$, be the matrix associated with the fine discretization (or level) of the domain Ω, and $\hat{\boldsymbol{v}} \in \mathcal{R}^{m\times 1}$ be a vector associated with the coarse discretization of the domain

To get from the fine level to the coarse level, we need to *restrict* the fine level vector. We can do this with the *restriction* matrix $\boldsymbol{R} \in \mathcal{R}^{m\times n}$ such that

$$\boldsymbol{R}\boldsymbol{v} = \boldsymbol{v_c} \tag{2}$$

How do we do the opposite? With the *prolongation* matrix, $\boldsymbol{P} \in \mathcal{R}^{n\times m}$

$$\boldsymbol{P}\boldsymbol{v_c} = \boldsymbol{v} \tag{3}$$

Coarse level Matrix

So (generally speaking) if we have our transfer matrices $\boldsymbol{R}$ and $\boldsymbol{P}$, we can go back and forth between the fine and coarse levels, but we still need a matrix for the coarse level

This can be very easily done. The coarse grid matrix is simply

$$\boldsymbol{A_c} = \boldsymbol{RAP} \tag{4}$$

where we can see that $\boldsymbol{A_c} \in \mathcal{R}^{m \times m}$.

The hope is then that the coarse grid matrix $\boldsymbol{A_c}$ approximates the hard to resolve components of the error, but since it is a smaller matrix, it will be easier to deal with these difficult components with the coarse grid matrix

What is a smoother?

At the most basic level, a smoother is an iterative solver. We call it a smoother because it is able to resolve the highly oscillatory components of the error very quickly, leaving only the smooth, low frequency components behind

The highly oscillatory components are those that are in the directions of the eigenvectors corresponding to large eigenvalues

The smooth low frequency components are those that are in the directions of the eigenvectors corresponding to small eigenvalues

Lets see how we can demonstrate this...

Smoothing → Stationary Iterative Methods

Lets consider our system of linear equations we want to solve again:

$$\boldsymbol{Ax} = \boldsymbol{b} \tag{5}$$

and consider a splitting of the matrix $\boldsymbol{A}$:

$$\begin{aligned}(\boldsymbol{A} - \boldsymbol{B} + \boldsymbol{B})\boldsymbol{x} &= \boldsymbol{b} \\ \boldsymbol{Bx} &= (\boldsymbol{B} - \boldsymbol{A})\boldsymbol{x} + \boldsymbol{b} \\ \boldsymbol{x} &= (\boldsymbol{I} - \boldsymbol{B}^{-1}\boldsymbol{A})\boldsymbol{x} + \boldsymbol{B}^{-1}\boldsymbol{b}\end{aligned} \tag{6}$$

This suggests an iterative method of the form

$$\boldsymbol{x_{k+1}} = (\boldsymbol{I} - \boldsymbol{B}^{-1}\boldsymbol{A})\boldsymbol{x_k} + \boldsymbol{B}^{-1}\boldsymbol{b} \tag{7}$$

Subtracting Eq. (6) from Eq.(7) gives

$$\boldsymbol{e_{k+1}} = (\boldsymbol{I} - \boldsymbol{B}^{-1}\boldsymbol{A})\boldsymbol{e_k} \tag{8}$$

Smoothing → Stationary Iterative Methods

We call the matrix $=(1\text{A})\boldsymbol{M} = (\boldsymbol{I} - \boldsymbol{B}^{-1}\boldsymbol{A}) theerroriterationmatrix$

Lets demonstrate smoothing with a specific example: if we let $\boldsymbol{B} = \boldsymbol{D}$ with $\boldsymbol{D}$ the diagonal of $\boldsymbol{A}$, then this is the Jacobi Iteration Method

Lets let $\boldsymbol{A}$ be the 1D Discrete Laplacian with Dirichlet boundary conditions:

$$\boldsymbol{A} = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2 \end{bmatrix} \tag{9}$$

coming from the discretization of the Laplace equation using finite differences

Smoothing $\rightarrow$ Stationary Iterative Methods

In this case, the eigenvectors of $\boldsymbol{M}$ are the same as those of $\boldsymbol{A}$. Let $(\lambda, \boldsymbol{v})$ be an eigenpair of $\boldsymbol{A}$, then

$$\begin{aligned} \boldsymbol{M}\boldsymbol{v} &= (\boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{A})\boldsymbol{v} \\ &= \boldsymbol{v} - \lambda \boldsymbol{D}^{-1}\boldsymbol{v} \\ &= \boldsymbol{v} - \frac{\lambda}{2}\boldsymbol{v} \\ &= (1 - \frac{\lambda}{2})\boldsymbol{v} \\ &= \sigma \boldsymbol{v} \end{aligned} \tag{10}$$

Smoothing → Stationary Iterative Methods

For the discrete 1D Laplacian, the eigenvalues and eigenvectors have an analytical form

$$\begin{aligned} v_{i,j} &= \sqrt{\frac{2}{n+1}}\sin\left(\frac{ij\pi}{n+1}\right) \\ \lambda_i &= 4\sin^2\left(\frac{\pi j}{2(n+1)}\right) \end{aligned} \tag{11}$$

So lets look and see what we mean by “low frequency” and “high frequency” modes...

Smoothing → Stationary Iterative Methods

We have a bit of an issue here! This can be observed by looking at the eigenvalues of the error iteration matrix, $\boldsymbol{M}$, which we saw are related to the eigenvalues of $\boldsymbol{A}$

$$\begin{aligned}
\sigma &= 1 - \frac{\lambda}{2} \\
&= 1 - \frac{4\sin^2\left(\frac{\pi j}{2(n+1)}\right)}{2} \\
&= 1 - 2\sin^2\left(\frac{\pi j}{2(n+1)}\right) \\
&= \cos\left(\frac{\pi j}{(n+1)}\right)
\end{aligned} \tag{12}$$

The magnitude of λ_1 is the same as that of λ_n. This means the high modes won't be efficiently taken care of.

Smoothing $\rightarrow$ Stationary Iterative Methods

Instead of scaling by $\boldsymbol{D}^{-1}$, lets introduce a scalar weighting factor ω so that $\boldsymbol{D}^{-1} \rightarrow \widetilde{\boldsymbol{D}^{-1}} = \omega \boldsymbol{D}^{-1}$. This is called the weighted Jacobi method

The eigenvalues of the error iteration matrix then become

$$\begin{aligned} \sigma &= 1 - 2\omega \sin^2\left(\frac{\pi j}{2(n+1)}\right) \\ &= 1 - \omega + \omega \cos\left(\frac{\pi j}{(n+1)}\right) \end{aligned} \tag{13}$$

Now the eigenvalues of the low modes and the high modes are no longer the same!

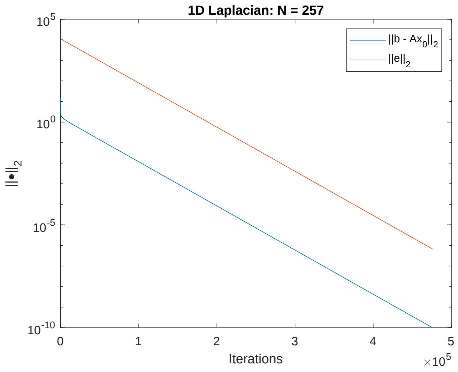
Exercises

1. Copy the contents of the jacobi.m file and create a new file called "weighted_jacobi.m". Modify the code to accept the scalar weighting parameter ω and modify the algorithm to use it
2. Create the 1D discrete laplacian using the **laplacian_1D** function with $p = 5$. This will create a grid of size $n_x = 2^p - 1 = 31$. Create the error iteration matrices for the weighted and unweighted cases with $\omega = 2/3$. Calculate the eigenvalues of both with Matlab's **eig** and make a plot of abs(σ) for both cases and observe how the spectrum of the two differ
3. Create a right hand side with "b = randn(size(A,1),1)". Make a plot of the vector. Using Matlab's backslash operator $x = A\backslash b$, calculate the solution to $\boldsymbol{Ax} = \boldsymbol{b}$. Using your new weighted jacobi function and an initial guess of all zeros, perform 1 iteration of weighted jacobi to get x_0, the solution after 1 iteration. Create the error vector $e = x - x_0$. Make a plot of ee. Repeat this again with 2,3 and 4 iterations.

Smoothers continued...

Why don't we just use weighted Jacobi as a solver?

1. Jacobi scales like $O(n^2) \rightarrow$ it's very slow!
2. It does not converge for all matrices. The spectral radius of the iteration matrix must be less than 1 for convergence: $\rho(M) < 1$.
3. The error does not converge at the same rate as the residual norm



Onward to Multigrid

Now that we know how smoothers work, lets start getting into the details of multigrid

The key idea behind multigrid is that the smooth components of the error that are difficult for the smoother to reduce act like the high frequency components of the error when they are transferred to a coarser grid where this large component of the error can be efficiently removed by solving the coarse grid linear equations

Ingredients:

1. Fine grid matrix $\boldsymbol{A}$ and right hand side vector $\boldsymbol{b}$
2. A smoother
3. Transfer matrices: restriction matrix $\boldsymbol{R}$ and prolongation matrix $\boldsymbol{P}$
4. Coarse grid matrix: $\boldsymbol{A_c} = \boldsymbol{RAP}$

A Two Level V-Cycle

Before jumping right in, it is useful to consider the action of multigrid on a vector $\boldsymbol{v}$. The simplest version of multigrid is a two level V-cycle
For our vector $\boldsymbol{v}$, we restrict it to the coarse grid:

$$\boldsymbol{v_c} = \boldsymbol{Rv} \tag{14}$$

Perform a linear solve on the coarse grid:

$$\begin{aligned} \boldsymbol{x_c} &= (\boldsymbol{A_c})^{-1}\boldsymbol{Rv} \\ \boldsymbol{x_c} &= (\boldsymbol{RAP})^{-1}\boldsymbol{Rv} \end{aligned} \tag{15}$$

And prolong it back to the fine grid:

$$\boldsymbol{v'} = \boldsymbol{P}(\boldsymbol{RAP})^{-1}\boldsymbol{Rv} = \boldsymbol{Sv} \tag{16}$$

This seems to suggest an iterative process...lets go back to our equation we used to get to the Jacobi method...

A Two Level V-Cycle

We had that we could split our system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$ into

$$\boldsymbol{x} = (\boldsymbol{I} - \boldsymbol{B}^{-1}\boldsymbol{A})\boldsymbol{x} + \boldsymbol{B}^{-1}\boldsymbol{b} \tag{17}$$

Which gave us the inspiration to make it iterative:

$$\boldsymbol{x}_{k+1} = (\boldsymbol{I} - \boldsymbol{B}^{-1}\boldsymbol{A})\boldsymbol{x}_k + \boldsymbol{B}^{-1}\boldsymbol{b} \tag{18}$$

And using Eq. (17) and Eq. (18) gave us the error iteration matrix

$$\boldsymbol{e}_{k+1} = (\boldsymbol{I} - \boldsymbol{B}^{-1}\boldsymbol{A})\boldsymbol{e}_k \tag{19}$$

Instead of $\boldsymbol{B}^{-1} = \boldsymbol{D}^{-1}$ or $\omega\boldsymbol{D}^{-1}$, lets set it equal to $\boldsymbol{S}$ of Eq. (16)

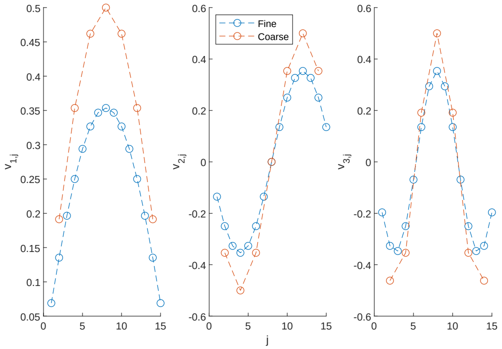
$$\boldsymbol{e}_{k+1} = (\boldsymbol{I} - \boldsymbol{P}(\boldsymbol{RAP})^{-1}\boldsymbol{RA})\boldsymbol{e}_k \tag{20}$$

This is the multigrid error iteration matrix

A Two Level V-cycle

We can see from Eq. (20) that this is solving a projection of the error $\boldsymbol{e_k}$ on the coarse grid

If $\boldsymbol{e_k}$ has had the high frequency components of the error removed from smoothing, then the coarse grid solve removes the low frequency components of the error which become high frequency on the coarse grid



Mode Analysis

Lets look and see what is actually happening to the eigenmodes on the coarse grid

Lets take the 1D Laplacian again for our example. Begin by writing the eigendecomposition of $\boldsymbol{A}$

$$\boldsymbol{A} = \boldsymbol{V}\Lambda\boldsymbol{V}^{\boldsymbol{T}} \tag{21}$$

and create the coarse grid operator

$$\begin{aligned}
\boldsymbol{A_c} &= \boldsymbol{RAP} \\
&= \boldsymbol{RV}\Lambda\boldsymbol{V}^{\boldsymbol{T}}\boldsymbol{P} \\
&= \frac{1}{4}\boldsymbol{P}^{\boldsymbol{T}}\boldsymbol{V}\Lambda\boldsymbol{V}^{\boldsymbol{T}}\boldsymbol{P} \\
&= \boldsymbol{P}^{\boldsymbol{T}}\boldsymbol{V}\frac{1}{2}\sqrt{\boldsymbol{\Lambda}}\frac{1}{2}\sqrt{\boldsymbol{\Lambda}}\boldsymbol{V}^{\boldsymbol{T}}\boldsymbol{P}
\end{aligned} \tag{22}$$

Modal Analysis

By assigning $\boldsymbol{W} = \frac{1}{2}\sqrt{\boldsymbol{\Lambda}}\boldsymbol{V}^{\boldsymbol{T}}\boldsymbol{P}$, we have

$$\begin{aligned} \boldsymbol{A_c} &= \boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{W} \\ \boldsymbol{V_c}\Lambda_{\boldsymbol{c}}\boldsymbol{V_c^T} &= \boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{W} \end{aligned} \tag{23}$$

Let's look at the first column of $\boldsymbol{W}^{\boldsymbol{T}}$ for a 1D grid of size $N_x = 7$

$$(\boldsymbol{W}^{\boldsymbol{T}})_1 = \sqrt{\frac{\lambda_1}{2(N_x+1)}} \begin{bmatrix} \sin(\frac{\pi}{n+1}) + 2\sin(\frac{2\pi}{n+1}) + \sin(\frac{3\pi}{n+1}) \\ \sin(\frac{3\pi}{n+1}) + 2\sin(\frac{4\pi}{n+1}) + \sin(\frac{5\pi}{n+1}) \\ \sin(\frac{5\pi}{n+1}) + 2\sin(\frac{6\pi}{n+1}) + \sin(\frac{7\pi}{n+1}) \end{bmatrix} \tag{24}$$

We see that the restriction has mixed the frequencies of the eigenmodes of the fine matrix on the coarse grid!

A Two Level V-Cycle

All of this analysis is very nice and now we know how multigrid reduces the error. But how do you actually get the solution from $\boldsymbol{Ax} = \boldsymbol{b}$?

Multigrid will transfer the residual vector to the coarse grid. How does this relate to the error?

Suppose we take a few iterations of our smoother to reach a very bad approximation to the solution $\boldsymbol{x_0}$. This is called pre-smoothing. The residual vector is then

$$\begin{aligned} \boldsymbol{r} &= \boldsymbol{b} - \boldsymbol{Ax_0} \\ \boldsymbol{r} &= \boldsymbol{Ax} - \boldsymbol{Ax_0} \\ \boldsymbol{r} &= \boldsymbol{A}(\boldsymbol{x} - \boldsymbol{x_0}) \\ \boldsymbol{r} &= \boldsymbol{Ae} \end{aligned} \tag{25}$$

We don't actually ever have access to the solution $\boldsymbol{x}$ in practical applications so we stick with the first definition of the residual.

A Two Level V-Cycle

We can now transfer the residual to the coarse level to form the right hand side for the coarse solve:

$$\boldsymbol{r_c} = \boldsymbol{RAe} \tag{26}$$

On the coarse grid, we can resolve a large part of the error by performing a solve with the coarse grid matrix:

$$\begin{aligned} \boldsymbol{A_c x_c} &= \boldsymbol{r_c} \\ &= \boldsymbol{RAe} \\ \boldsymbol{x_c} &= \boldsymbol{A_c}^{-1}\boldsymbol{RAe} \\ &= (\boldsymbol{RAP})^{-1}\boldsymbol{RAe} \end{aligned} \tag{27}$$

$\boldsymbol{x_c}$ is what we call the coarse grid correction

A Two Level V-Cycle

Of course, we need to transfer this up to the fine grid and we add it to the solution on the fine grid, $\boldsymbol{x_0}$:

$$\boldsymbol{x_0} = \boldsymbol{x_0} + \boldsymbol{P}(\boldsymbol{RAP})^{-1}\boldsymbol{RAe} \tag{28}$$

We then smooth using $\boldsymbol{x_0}$ as an initial guess to our original system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$. This is called post-smoothing.

We continue this until we have reached convergence of the original system of linear equations.

The Two Level Algorithm

Until convergence, do:

1. Smooth on $\boldsymbol{Ax} = \boldsymbol{b}$ using ν iterations of weighted Jacobi (or other smoother) to get $\boldsymbol{x_0}$
2. Compute the residual $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{Ax}$
3. Restrict the residual to the coarse grid so that $\boldsymbol{r_c} = \boldsymbol{Rr}$
4. Directly solve $\boldsymbol{A_c x_c} = \boldsymbol{r_c}$
5. Prolong and add to the current solution: $\boldsymbol{x_0} = \boldsymbol{x_0} + \boldsymbol{Px_c}$
6. Smooth on $\boldsymbol{Ax} = \boldsymbol{b}$ using ν iterations of weighted Jacobi (or other smoother) using $\boldsymbol{x_0}$ as an initial guess
7. If converged, break

The 2D Discrete Laplacian

The interior points of the 2D Discrete Laplacian is given by the 5-point stencil

$$u_{i,j} = \frac{1}{h^2}\left(-u_{i-1,j} - u_{i+1,j} + 4u_{i,j} - u_{i,j-1} - u_{i,j+1}\right) \tag{29}$$

The discrete 2D Laplacian can be built from the 1D case:

$$\boldsymbol{A_{2D}} = \boldsymbol{A_{1D}} \otimes \boldsymbol{I} + \boldsymbol{I} \otimes \boldsymbol{A_{1D}} \tag{30}$$

How do we create the transfer matrices? We will use *full weighting* to create our restriction matrix $\boldsymbol{R}$ which is related to the prolongation matrix $\boldsymbol{P}$ by

$$\boldsymbol{R} = \frac{1}{4}\boldsymbol{P}^{\boldsymbol{T}} \tag{31}$$

Exercise

Create the restriction and interpolation matrices for the 2D case. A helpful transformation between the fine grid points and the coarse grid points is $i_c = 2i_f$ and $j_c = 2j_f$, where i_c, j_c are coarse grid coordinates and i_f, j_f are fine grid coordinates.

Another helpful conversion is the conversion from cartesian coordinates to linear indexing, which is

$$\mathrm{idx} = i + (j - 1) \times N_x \tag{32}$$

if indexing starts at 1 and

$$\mathrm{idx} = i + j \times N_x \tag{33}$$

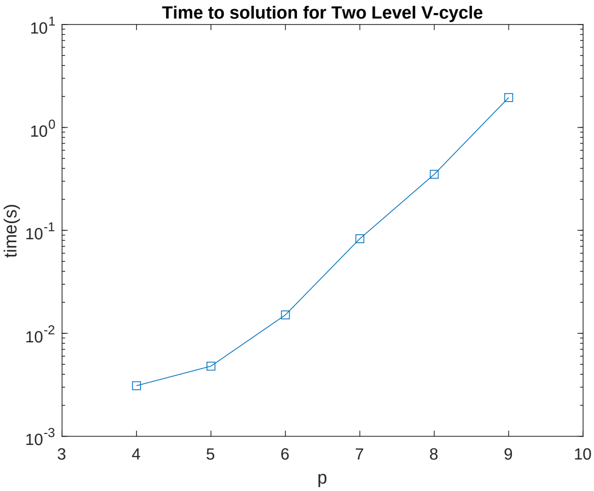
if indexing starts at 0, where i, j are the coordinates in x, y and N_x is the extent of the grid in the x direction

Exercise

1. Create your coarse grid matrix $\boldsymbol{A_c}$. Now we have all of the ingredients for a two grid v cycle.
2. Write a code snippet that calculates the solution to $\boldsymbol{Ax} = \boldsymbol{b}$ with $\boldsymbol{b}$ a random vector using a two grid V-cycle
3. Experiment with different grid sizes. Make a plot of the time to solution vs the grid size (The time can be measured with Matlab's tic() toc() functions). What happens to the convergence as the grid size is increased for our two grid V-cycle?

Beyond Two Levels

From the last exercise, you may have noticed that as the grid size becomes larger, the two grid V-cycle becomes less efficient!



Beyond Two Levels

As we increase the grid size, the coarse grid size is also increasing, which drastically increases the cost of the direct solve

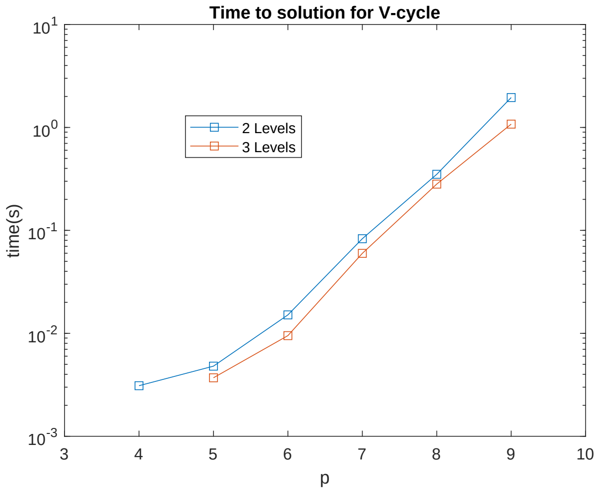
We can rectify this by realizing that multigrid can be done recursively!

For our 2D discrete Laplacian, we can repeat the same process of creating a coarse grid matrix using $\boldsymbol{A_c}$ rather than $\boldsymbol{A}$. Lucky for us, the transfer matrices are the same form for the smaller grids.

This means that we can create a hierarchy of grids recursively, where the direct solve can be done efficiently on the coarsest grid

Beyond Two Levels

Using an extra level brings the time to solution down, but its still not where we would like it to be, so we need more grids!



The Recursive V-Cycle Algorithm

```
$\boldsymbol{x}$ =V-cycle($\boldsymbol{A}_\ell, \boldsymbol{x}_\ell, \boldsymbol{b}_\ell, \omega, \nu, \ell_{max}$)
  1  $\boldsymbol{x}_\ell = \text{smooth}(\boldsymbol{A}_\ell, \boldsymbol{x}_\ell, \boldsymbol{b}_\ell, \omega, \nu)$
  2  $\boldsymbol{r}_\ell = \boldsymbol{b}_\ell - \boldsymbol{A}_\ell \boldsymbol{x}_\ell$
  3  $\boldsymbol{b}_{\ell+1} = \boldsymbol{R}^{\ell}_{\ell+1} \boldsymbol{r}_\ell$
  4  If $\ell == \ell_{max}$
         Solve $\boldsymbol{A}_{\ell_{max}} \boldsymbol{x}_{\ell_{max}} = \boldsymbol{b}_{\ell_{max}}$
     Else
         $\boldsymbol{x}_{\ell+1}$ =V-cycle($\boldsymbol{A}_\ell, \boldsymbol{x}_\ell, \boldsymbol{b}_\ell, \omega, \nu, \ell_{max}$)
  5  $\boldsymbol{x}_\ell = \boldsymbol{x}_\ell + \boldsymbol{P}^{\ell+1}_{\ell} \boldsymbol{x}_{\ell+1}$
  6  $\boldsymbol{x}_\ell = \text{smooth}(\boldsymbol{A}_\ell, \boldsymbol{x}_\ell, \boldsymbol{b}_\ell, \omega, \nu)$
```

Further Reading

Yousef Saad. **Iterative Methods for Sparse Linear Systems**. Second Edition. https://www-users.cse.umn.edu/~saad/IterMethBook_2ndEd.pdf