HW5

All problems due **Friday** February **15**

5.1 Fundamentals: Jacobi and Gauss-Seidel Methods

3 pts Consider the 5 equations for 5 unknowns, written in matrix form at right.

$$\begin{bmatrix} -2 & 0 & 3 & -6 & -1 \\ 5 & 6 & 7 & 8 & 9 \\ 2 & 4 & -4 & 1 & 2 \\ 7 & 4 & -1 & 7 & 2 \\ -1 & 3 & -1 & 0 & 0 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 5 \\ 4 \\ 3 \\ 2 \\ 1 \end{pmatrix}$$

Reorder the equations to form a *new* $Ax = b$ problem, written in matrix form, where the new matrix A is "strictly diagonally dominant" (**or** at least the **"best you can do"** to make as **"strong"** a diagonal as possible).

5.2 Fundamentals: Jacobi and Gauss-Seidel Methods

5 pts Consider the 4x4 problem $Ax = b$ at right, for which I have already ordered equations to create as "strong" a diagonal as I can (at *least* the largest element in each row is on the diagonal).

$$\begin{bmatrix} 2 & -1 & 0 & 1 \\ 1 & -2 & 1 & 1 \\ -1 & 1 & 4 & -1 \\ 1 & 0 & -1 & 2 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ -1 \\ 2 \end{pmatrix}$$

Starting with the initial guess vector $\underline{\boldsymbol{x}}^{\boldsymbol{0}} = [0\ \ 1\ \ 0\ \ -1]^T$, calculate by-hand (and *show your work!!*) ...

+3 a) the first two iterations of the solution ($\underline{\boldsymbol{x}}^{\boldsymbol{1}}$ and $\underline{\boldsymbol{x}}^{\boldsymbol{2}}$) using the *Jacobi* iteration method.

+2 b) the first two iterations of the solution ($\underline{\boldsymbol{x}}^{\boldsymbol{1}}$ and $\underline{\boldsymbol{x}}^{\boldsymbol{2}}$) using the *Gauss-Seidel* iteration method.

5.3 Fundamentals: Errors and Norms.

6 pts Consider the same 4x4 problem from 5.2, with the exact solution $\underline{\boldsymbol{x}}$ and iterations $\underline{\boldsymbol{x}}^{\boldsymbol{5}}$ and $\underline{\boldsymbol{x}}^{\boldsymbol{6}}$ given at right. Do the following *by-hand*:

$$\underline{x} = \begin{pmatrix} 0 \\ 2 \\ 1 \\ -1 \end{pmatrix}$$

$$\underline{x}^5 = \begin{pmatrix} 0.2 \\ 1.7 \\ 0.7 \\ 0.2 \end{pmatrix} \qquad \underline{x}^6 = \begin{pmatrix} 0.5 \\ 2.3 \\ 0.7 \\ -0.5 \end{pmatrix}$$

+3 a) For the k=6th iteration, evaluate the true ***error***, the ***proxy*** for the error, and the ***residual error***, expressed as vectors (<u>don't</u> take norms).

+3 b) Evaluate the L_1, L_2, and L_∞ **norms** of the exact solution vector $\underline{\boldsymbol{x}}$.

Show your work and *explain* everything you do

HW5 All problems due **Friday** February **15**

PLEASE READ THIS! It sets up the "Sagging Cable" problem 5.4

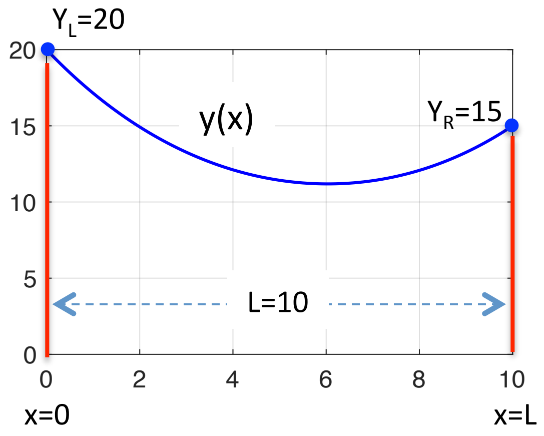
A flexible cable is supported between two poles, height Y_L at left and Y_R at right, separated by distance L. A differential equation representing the cable shape can be *approximated* as:

$$\frac{d^2y}{dx^2} - \frac{1}{2a}\frac{dy}{dx} = \frac{1}{a}$$

where *a* is a constant representing the properties of the cable (light, high-tension cables have a high value of *a*; heavy, low-tension cables have a low value of *a*).

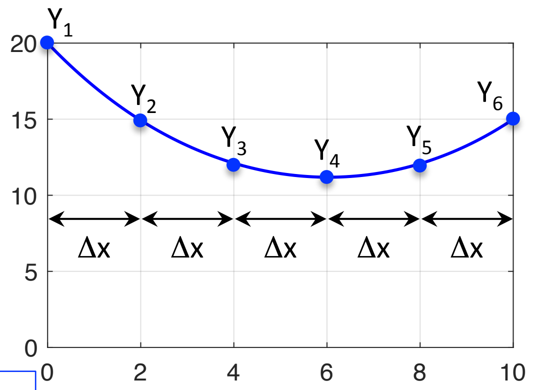
Given the boundary conditions in the figure shown, across poles separated by **L = 10** m, and for a cable with property ***a* = 2**, you *could* solve this equation using calculus from your other math classes, but here we'll solve it NUMERICALLY.

- $y(x=0) = \mathbf{Y_L = 20}$
- $y(x=L) = \mathbf{Y_R = 15}$



Our way to solve the problem is to define variables Y_1 to Y_M to represent the height every **Δ*x*** across the space *x* = 0 to L. Then it turns out you can approximate the derivatives above as:

$$\frac{d^2y(x_i)}{dx^2} \approx \frac{Y_{i-1} - 2Y_i + Y_{i+1}}{(\Delta x)^2} \qquad \frac{dy(x_i)}{dx} \approx \frac{-Y_{i-1} + Y_{i+1}}{2\Delta x}$$



(You'll learn more about approximating derivatives in a month or so; for now, just trust me that this approximation works!) Then substituting all this into the differential equation gives:

$$\boxed{\left(\frac{1}{(\Delta x)^2} + \frac{1}{4a\,\Delta x}\right)Y_{i-1} + \left(\frac{-2}{(\Delta x)^2}\right)Y_i + \left(\frac{1}{(\Delta x)^2} - \frac{1}{4a\,\Delta x}\right)Y_{i+1} = \frac{1}{a}}$$

This linear equation applies for all the "interior" points *i* (*i.e.* $2 \le i \le M-1$). Coupled with the two boundary conditions setting the left and right heights ($Y_1 = Y_L = 20$, and $Y_M = Y_R = 15$) I now can write *M* equations for all *M* unknown Y_i !!

For Example: Take *M* = 6 unknown points (Y_1 to Y_6) across the span, spaced by Δx = 2 meters, as shown by the blue dots above. Then, given cable *a* = 2, I can write *six* equations for the unknowns as:

$$Y_1 = 20 \quad \leftarrow \textit{Left boundary cond}^n$$

$$\frac{5}{16}Y_{i-1} - \frac{1}{2}Y_i + \frac{3}{16}Y_{i+1} = \frac{1}{2} \quad (i = 2{:}5)$$

$$Y_6 = 15 \quad \leftarrow \textit{Right boundary cond}^n$$

Finally, I can collect all *M* = 6 equations in matrix form as:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 5/16 & -1/2 & 3/16 & 0 & 0 & 0 \\ 0 & 5/16 & -1/2 & 3/16 & 0 & 0 \\ 0 & 0 & 5/16 & -1/2 & 3/16 & 0 \\ 0 & 0 & 0 & 5/16 & -1/2 & 3/16 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \end{pmatrix} = \begin{pmatrix} 20 \\ 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \\ 15 \end{pmatrix} \begin{array}{l} \leftarrow \textit{Left boundary cond}^n \\ \left.\begin{array}{l} \\ \\ \\ \\ \end{array}\right\} \textit{Interior equations for } 2 \le i \le 5 \\ \leftarrow \textit{Right boundary cond}^n \end{array}$$

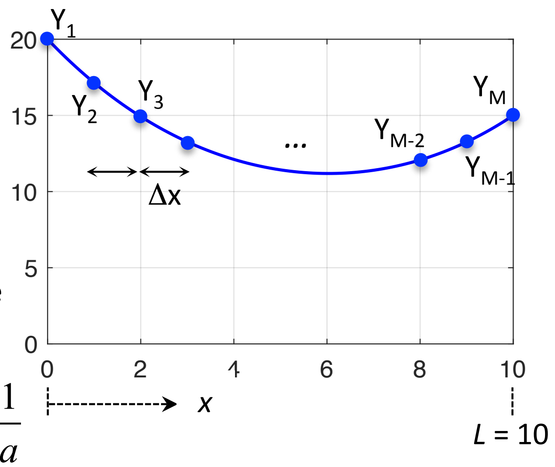
Notice that I ordered the equations the way I did to make the diagonal as strong as possible!

Your job in problem 5.4 is to extend this same example to *many* more points, by creating the matrices for ***M* = 201** equations for unknown variables Y_i to Y_{201}, and then solving for all the Y_i using Jacobi iteration.

HW5 All problems due **Friday** February **15**

5.4 Jacobi Iteration MATLAB code

12 pts Extend the concept from the last page to use the Jacobi Iteration method to solve for the height Y_1 through Y_M at ***M* = 201** equally-spaced (Δx) points along a cable.



Assume characteristics of the cable are the same as before: ***L* = 10** meters, ***a* = 2**, and boundary conditions **Y_L = 20** and **Y_R = 15**.

Like last page, I can approximate the differential equation with the linear relationship below:

$$\left(\frac{1}{(\Delta x)^2}+\frac{1}{4a\,\Delta x}\right)Y_{i-1}+\left(\frac{-2}{(\Delta x)^2}\right)Y_i+\left(\frac{1}{(\Delta x)^2}-\frac{1}{4a\,\Delta x}\right)Y_{i+1}=\frac{1}{a}$$

If you don't really understand exactly where this comes from, don't worry! You're going to be deriving this yourself by Chapter 10. For now, just accept the boxed equation applies for *any* three adjacent Y_{i-1}, Y_i, Y_{i+1} ($2 \le i \le M$-1) separated by a constant Δx. The only difference to the example on the last page is that M is now 201 instead of 6, so the Δx is much smaller (10/200 = 0.05, instead of 2 meters). So, including the two boundary conditions ($Y_1 = Y_L$ and $Y_M = Y_R$), you now can write 201 equations for all 201 unknown Y_i !!

To solve the problem I have posted a script **HW5_4.m** that *already* does all the following for you:

- Initializes variables for the rod (L, a, Y_L, Y_R), and M = 201,
- sets values for convergence tolerances for the L_2 norm of "proxy" error between iterations and the residual error ($tol_X = tol_R$ = sqrt(M) x 10^{-8}, using M = 201),
- calls a function **Jacobi.m** to iteratively solve for all the Y_i until convergence,
- calculates the exact analytic solution $\hat{Y}(x)$ (to compare with your iterative one),
- makes two plots: (i) the $Y_i(x_i)$ distribution for the 1st, 100th, 1000th, 10 000th and final (converged) iterations compared to the exact solution, and (ii) the $\log_{10}$(error) convergence history.

Your job is just to make the function **Jacobi.m** that does the following:

a) EXACTLY interfaces with my main script, using all the same inputs and outputs. To help, I've actually provided the first line of the function, to ensure compatibility. DON'T CHANGE THIS!!

```
function [Yhistory, ErrorX, ErrorR, Iter]=Jacobi(L, a, YL, YR, M, tolX, tolR)
```

b) Create the A and b matrices that represent the M equations for Y_i in the form $A\,Y = b$ where $\underline{Y}$ = $[Y_1, \ldots, Y_M]^T$. Make sure you understand my example for M = 6 on the last page, before tackling this general MxM system..

c) For your initial guess, $\underline{Y}^0$, just use the average value between the supports ½(Y_L+Y_R) for <u>all</u> Y_i.

d) Use a "while" loop to keep iterating until the L_2-norms of *both* the "proxy" error between iterations *and* the residual error are within their tolerances.

e) Calculate the column vector $\underline{Y}^k$ for each iteration k using the **Jacobi** iterative method.

f) Store each iteration as a column vector in the variable `Yhistory`. So, for example, if you're done after 1000 iterations, then Yhistory would be a 201 x 1000 matrix, with each column vector $\underline{Y}^k$ representing one iteration. This will allow us to make plots of what the iterations looked like later.

g) Calculate (and store in vectors `ErrorX` and `ErrorR`) the L_2-norms of <u>both</u> the "proxy" error in $\underline{Y}^k$ and the *residual* error. This will allow us to make a *convergence history* plot later.

5.4 continued …

When it works, HW5_4 will automatically make two plots in one window. Save a .pdf of the plot window called **Y.pdf**. Then study the plots and your calculated variables and answer the following questions:

i. How many iterations did it take to converge? *Hint*: It took me *many* 10,000 s of iterations.

ii. For your final iteration, what's the lowest height of the cable (*i.e* minimum $y(x)$)?

iii. Your convergence history plot of "proxy error" should show an rapid initial improvement in error, and then a (relatively slow) linear convergence rate until completion. Use the slope in the *linear* region to approximate how many iterations it takes for the error to improve by each factor of 10 (*i.e.* gives you one more decimal of accuracy).

Submit online in the HW5 assignment:

- Your working, documented **Jacobi.m** function, and plot **Y.pdf**.
- **Answers** to questions **(i)**, **(ii)** and **(iii)** above in the online "comment" box.

HINTS for doing 5.4

1. Make sure you read and understand how I made my matrices for the $M = 6$ example. Then write out on a piece of paper, before you even think of starting to use MATLAB, what *your* set of 201 equations for 201 unknowns should generally look like. Keep the ordering of the equations similar to my example so your diagonal of A stays as "strong as possible".

2. Sketch out on paper exactly what you need your Jacobi.m function to do! Break it up into high-level jobs, then break that down into smaller parts, THEN start coding.

3. You should check each part of your code as you go along. For example, one thing you need to do is create your A and b matrices. Don't just jump into the M=201 system right away. Instead, try M=6 the first time and make sure you get *exactly* the same A and b matrices from my earlier example using $\Delta x = 2$. If that's correct, then it's much more likely that your 201x201 system will work.

4. Is your actual Jacobi iteration part working? (*i.e.* getting $\underline{Y}^k$ from the previous $\underline{Y}^{k-1}$) You should double-check that part of your code alone against what you *do* know works. Take the 3x3 Ax=b problem we did in class using Jacobi iteration by-hand. We did the next 2 iterations on the board. Your code should give exactly those results starting from those same A and b matrices.

5. What about your *while* loop? We already went over this a lot in HW3.7. I don't fundamentally see anything different in how you would set it up here, except we're using *norms* of *vectors* to represent the errors, rather than just scalar errors before.

6. Any confusion about norms? Well, try them out separately in MATLAB, or by hand. Give it a smaller, known vector of errors as a test case and make sure you're using the *norm* command for L_2-norms in MATLAB properly.

7. Does your code just keep running without stopping? Then get more information WHY. Look at the one of the later iterations – is it getting closer to the final solution, or further away? Look at the "proxy" and residual errors – are they getting smaller, bigger, or kind of staying the same? Most problems here are caused by (a) you made incorrect A and b matrices and/or you did not carefully order the equations to keep a strong diagonal (see hint 1 above), (b) your iteration process is incorrect (see hint 4 above), (c) your while statement is not letting you end (see hint 5 above).

HINTS for doing 5.4 (continued …)

The bottom line is – these are the steps you need to take to develop your code properly the first time, and then to follow if you suspect something is not working. This is helping you be self-reliant, and not just reliant on the TAs. If you come to me or the TAs with a question about the code not working, we're happy to help, but **come prepared!** We're going to ask you to show us that you already checked all the things above, and you'll get turned away until you do so if you don't bring us (for example) something on paper showing how you're planning your code, or your output for trying to make the 6x6 A and b matrices from my earlier example, etc.

We do **not** want you to hand in all your preliminary papers or debugging work with your HW submission. That's just for you to help you develop the code, and to communicate efficiently with us if there are any problems. Only hand in exactly what I asked for in problem 5.4.