Homework 4

Please answer the following questions in complete sentences in a typed, clearly prepared manuscript and submit the solution by the due date on Blackboard (early morning on Monday, September 25th, 2017)

Updates

- Update 1 (Problem 1.2, corrected the syntax!)
- Fixed the $f(x_i, x_j)$ comment.
- Added a note about row and column indexing
- Updated the pseudo-code for building the matrix to use row indexing to mirror the problem description

Problem 0: Homework checklist

- Please identify anyone, whether or not they are in the class, with whom you discussed your homework. This problem is worth 1 point, but on a multiplicative scale.
- Make sure you have included your source-code and prepared your solution according to the most recent Piazza note on homework submissions.

Problem 1: Backsolve

*For this problem, you must use Julia.*

1. Implement backsolve and forward solve as functions in Julia. Show and document your code.

2. Construct a random upper-triangular linear system via:

   ```
   A = triu(randn(n,n));
   b = randn(n);
   ```

   Compare the performance of your backsolve to Julia’s backslash method to solve a linear system.

3. Write a routine to solve a linear system of equations using the LU factorization in Julia both with and without pivoting.

   (Use `lufact(A, Val{false})` for no pivoting in LU in Julia.)

4. Use your backsolve code, along with Julia’s lu factorization in order to implement your own linear solver. Present a paragraph or two (and a figure or two) comparing it’s speed and accuracy to Julia’s own solver and focus on the effect of pivoting.
Problem 2: Poisson’s equation

In this problem, we’ll meet one of the most common matrices studied in numerical linear algebra: the 2d-Laplacian. We arrive at this matrix by discretizing a partial differential equation. Poisson’s equation is:

\[ \Delta u = f \]

where \( u(x, y) \) is a continuous function defined over the unit-plane (i.e. \( 0 \leq x \leq 1, 0 \leq y \leq 1 \)), \( f(x, y) \) is a continuous function defined over the same region, and \( \Delta \) is the Laplacian operator:

\[ \Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}. \]

Given a function \( f \), we want to find a function \( u \) that satisfies this equation. There are many approaches to solve this problem theoretically and numerically. We’ll take a numerical approach here.

Suppose we discretize the function \( u \) at regular points \( x_0, \ldots, x_n \), and \( y_0, \ldots, y_n \) where \( x_i = y_i = i/n \) so that we have:

\[ u(x, y) \approx \text{grid of } u(x_0, y_0) \cdots u(x_n, y_0) \ldots u(x_0, y_n) \cdots u(x_n, y_n). \]

For this discretization, note that

\[
\Delta u(x_i, y_j) \approx n^2 (u(x_{i-1}, y_j) - 2u(x_i, y_j) + u(x_{i+1}, y_j)) \\
+ n^2 (u(x_i, y_{j-1}) - 2u(x_i, y_j) + u(x_i, y_{j+1})) \\
= n^2 (u(x_{i-1}, y_j) + u(x_i, y_{j-1}) - 4u(x_i, y_j) + u(x_{i+1}, y_j) + u(x_i, y_{j+1})) \\
= f(x_i, y_j).
\]

What we’ve done here is use the approximation:

\[ \frac{\partial^2 u}{\partial x^2} \approx \frac{1}{h^2} (u(x-h) - 2u(x) + u(x+h)) \]

for both partial terms.

We need this equation to hold at each point \( x_i, y_j \). But note that there are some issues with this equation at the boundary values (where \( x = 0 \) or 1, or where \( y = 0 \) or 1).

For this problem, we’ll make it very simple and set:

\[ u(0, y_j) = u(1, y_j) = u(x_i, 0) = u(x_i, 1) = 0. \]

Now, we’ll do what we always do! Turn this into some type of matrix equation!

Let \( U \) be an \( n+1 \times n+1 \) matrix that we’ll index from zero instead of one:

\[
U = \begin{bmatrix}
U_{0,0} & \cdots & U_{0,n} \\
\vdots & \ddots & \vdots \\
U_{n,0} & \cdots & U_{n,n}
\end{bmatrix},
\]

where \( U_{i,j} = u(x_i, y_j) \). At this point, we are nearly done. What we are going to do is turn Poisson’s equation into a linear system. This will be somewhat like how we turned image resampling into a matrix vector equation in the first homework.

There was a typo here before with row-vs-column indexing. Either is okay for this problem. Just be consistent and mention what you are using.
In order to write $U$ as a vector, we’ll keep the convention from last time:

$$
U = \begin{bmatrix}
u_1 & \cdots & u_{n+1} \\
u_{n+2} & \cdots & u_{2(n+1)} \\
\vdots & \ddots & \vdots \\
u_{n(n+1)+1} & \cdots & u_{(n+1)(n+1)}
\end{bmatrix}.
$$

Let $u$ be the vector of elements here. Note that our approximation to $\Delta u$, just involved a linear combination of the elements of $u$. This means we have a linear system:

$$Au = f$$

where the rows of $A$ and $f$ correspond to equations of the form:

$$\frac{1}{h^2} \left(u(x_{i-1}, y_j) + u(x_i, y_{j-1}) - 4u(x_i, y_j) + u(x_{i+1}, y_j) + u(x_i, y_{j+1}) = f(x_i, y_j) \right).$$

1. Let $n = 3$. Write down the $16 \times 16$ linear equation for $u$ including all the boundary conditions. Note that you can encode the boundary conditions by adding a row of $A$ where: $u_i = 0$.

2. Write a Julia or Matlab/Python code to construct the matrix $A$ and vector $f$ when $n = 10$ and $f(x, y) = 1$. Here’s some pseudo-code to help out:

```julia
function laplacian(n::Integer, f::Function)
    N = (n+1)^2
    A = zeros(N,N)
    fvec = zeros(N)
    # 2017-09-21 added transpose to mirror the row-indexing in the problem
    # but see note above, you can do either, just be consistent.
    G = reshape(1:N, n+1, n+1)'
    # index map, like we saw before;
    h = 1.0/(n)
    for i=0:n
        for j=0:n
            row = G[i+1,j+1]
            if i==0 || j == 0 || i == n || j == n
                # we are on a boundary
                fvec[row] = 0.0
                # fill in A[row,:]
            else
                fvec[row] = f(i*h, j*h)*h^2
                # fill in A[row,:]
            end
        end
    end
    return A, fvec
end
A, f = laplacian(10, (x,y) -> 1.0)
```

3. Solve for $u$ using Julia’s or Matlab’s backslash solver, and show the result using the `mesh` function (Matlab) or `surface` function (Plots.jl in Julia).

**Problem 3: The Schur Complement**

Suppose we wish to solve

$$Mx = b$$

and further suppose that you KNOW some of the values of $x$. 
Let us permute and partition $M$ to be a block system:

$$Mx = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

where $x_1$ is what you know.

1. Show how to solve for $x_2$ given $x_1$. Under what conditions is this possible?

2. Now, suppose that you have a very kind, but very confused dog that happened to eat your flash memory stick holding the piece of $x_1$ that you knew. However, you had saved your computed $x_2$ on your Purdue account, and so you have a backup. (This means you can assume that computing $x_2$ from $x_1$ is possible for this problem if you determined it wasn't always possible above.) Can you get $x_1$ back?

3. Combine these two parts to derive a single linear system to compute $x_1$ without computing $x_2$. The system you'll derive is called the Schur complement.

**Problem 4: Ranking with Linear Systems**

Watch the video on sports linear systems available on Blackboard. (this was recorded in 2014). No questions about it, but this material is now fair-game for the Midterm. Your answer to this homework problem should be a question about the material presented in the lecture.