Lecture 19: Finite Element Methods (2)

Today:

- Finite element methods for boundary value problems (BVPs)
- Implementation, implementation, implementation, implementation
Reminder: the FEM equations

For the 1D Poisson problem with zero BCs:

\[
\frac{1}{\Delta x} \begin{bmatrix}
-2 & 1 & \cdots & 0 & 0 \\
1 & -2 & \cdots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & -2 & 1 \\
0 & 0 & \cdots & 1 & -2
\end{bmatrix} \begin{bmatrix}
u_2 \\
u_3 \\
\vdots \\
u_{n-1} \\
u_n
\end{bmatrix} = \begin{bmatrix}
\int_a^{a+2\Delta x} f(x)b_2(x)\,dx \\
\int_a^{a+3\Delta x} f(x)b_3(x)\,dx \\
\vdots \\
\int_a^{a+(n)\Delta x} f(x)b_{n-1}(x)\,dx \\
\int_a^{a+(n+1)\Delta x} f(x)b_n(x)\,dx
\end{bmatrix}
\]

**Punchline:** lots of zeros in that matrix; easier to solve than the generic form of (1)

**Punchline:** Solve matrix system \(\implies\) get the \(u_1, u_2, \ldots, u_n\) that approximate \(u(x_1), u(x_2), \ldots, u(x_n)\)

**This time:** some coding examples and concept discussions!
Implementation: python

```python
# preamble stuff

# Interval properties
a = 2
b = 4
L = (b-a)

# Initialize
alpha = uex(a, a, b)
beta = uex(b, a, b)

# Consider different n values to see how solution changes as we change n
nv = np.array([40, 80, 120, 240])

# Solve for coefficients
u = LA.solve(G, bv)[0]
u = np.concatenate([[alpha], u, [beta]])

# Get error
err[] = LA.norm(u - uex(xj, a, b))/ LA.norm(uex(xj, a, b))

# Then do some plotting...
```
And the results...
Some concept questions

The \( n = 40 \) plot was jagged because there weren’t enough \( x \) points to represent the solution with sufficient resolution. Could we have made smoother plots keeping \( n \) fixed?

Yes! We have a representation for our approximation function for \( \text{any } x \in [a, b] \)

\[
 u_a(x) = \sum_{j=2}^{n} u_j b_j(x)
\]

These are the coefficients we solved for from the matrix system

These are the “hat” basis functions we defined Monday

We simply plotted \( u_a \) evaluated at the nodes \( x_j, j = 1, \ldots, n + 1 \)

Would this approach make the approximate solution more accurate?

No! The error in the solution scales as \( O(\Delta x^2) \), where \( \Delta x \) is the spacing used for the coefficients. We can make the plot smoother by introducing more plotting points, but the error between the true and approximate solution will not change.

How would you change the FEM solution for a different BVP?

Use the right energy inner product and build the matrix system using that energy IP!