Chandler Justice - A02313187

November 27, 2023
Where we are going with the class

- Complete Assn. 7 today
- Assn 8 - Using Jacobi to compute iteration solns to $Ax = b$
- Assn 9 - Speed code up using OpenMP

Iterative methods to computing solutions: Jacobi iteration & more
let: $A = L + D + U$
$Ax = b \rightarrow (L + D + U)x = b \rightarrow Dx = b - (L + U)x$
$x = D^{-1}(b - (L + U)x)$

- L: A lower triangular matrix with zeros on the diagonal
- D: A Matrix with non-zero values only in the diagonal
- U: An upper triangular matrix with zeros on the diagonal

$x^{k+1} = D^{-1}(b - (L + U)x^k)$
$x^{k+1} \rightarrow Ax^* = b$
$x^{k+1} \rightarrow x^*$
Note: A is diagonally dominant

Implementation for Jacobi
$Ax + b \rightarrow x^{k+1} = D^{-1}(b - (L + U)x^k) \rightarrow x^*$
input: $A \in \mathbb{R}^{n \times m}, b \in \mathbb{R}^n, x^0 \in \mathbb{R}^n$, tol maxIter
initialize: error = 10 * tol; iter = 0;
loop:

```
while (error > tol && iter < maxIter){
for(int i = 0; i ¡ n; i++){
double sum = $b_i$;
for(int j = 0; j ¡ n; j++){
sum += $a_{i,j} * x_0$;
for(int k = i + 1; j ¡ n; j++){
sum += $a_{i,j} * x_0$}
$x_i = sum/a_{i,i};$}
double error = 0.0;
for(int i = 0; i ¡ n; i++){
double value = $x1_i - x0_i$;
$error+ = val + val_i$}
error = sqrt(error);
iter++;}
```
