AMath 483/583 — Lecture 23

Outline:

- Linear systems: LU factorization and condition number
- Heat equation and discretization
- Iterative methods

Sample codes:

- $UWHPSC/codes/openmp/jacobi1d_omp1.f90
- $UWHPSC/codes/openmp/jacobi1d_omp2.f90

Notes:

Announcements

Homework 6 is in the notes and due next **Friday**.

Quizzes for this week's lectures due next **Wednesday**.

Office hours today 9:30 – 10:20.

Next week:

Monday: no class

Wednesday: Guest lecture —

Brad Chamberlain, Cray

Chapel: A Next-Generation Partitioned
Global Address Space (PGAS) Language

Notes:

AMath 483/583 — Lecture 23

Outline:

- Linear systems: LU factorization and condition number
- Heat equation and discretization
- Iterative methods

Sample codes:

- $UWHPSC/codes/openmp/jacobi1d_omp1.f90
- $UWHPSC/codes/openmp/jacobi1d_omp2.f90

Notes:

DGESV — Solves a general linear system

```
      SUBROUTINE DGESV( N, NRHS, A, LDA, IPIV,
     &                  B, LDB, INFO )
```

`NRHS` = number of right hand sides

`B` = matrix whose columns are right hand side(s) on input
solution vector(s) on output.

`LDB` = leading dimension of `B`.

`INFO` = integer returning 0 if successful.

`A` = matrix on input, L,U factors on output,

`IPIV` = Returns pivot vector (permutation of rows)
`integer, dimension(N)`
Row `I` was interchanged with row `IPIV(I)`.

Notes:

Gaussian elimination as factorization

If A is nonsingular it can be factored as

$$PA = LU$$

where

P is a permutation matrix (rows of identity permuted),

L is lower triangular with 1's on diagonal,

U is upper triangular.

After returning from `dgesv`:
`A` contains L and U (without the diagonal of L),
`IPIV` gives ordering of rows in P.

Notes:

Gaussian elimination as factorization

Example:

$$A = \begin{bmatrix} 2 & 1 & 3 \\ 4 & 3 & 6 \\ 2 & 3 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & 1 & 3 \\ 4 & 3 & 6 \\ 2 & 3 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1/2 & -1/3 & 1 \end{bmatrix} \begin{bmatrix} 4 & 3 & 6 \\ 0 & 1.5 & 1 \\ 0 & 0 & 1/3 \end{bmatrix}$$

`IPIV` = (2,3,1)

and `A` comes back from `DGESV` as:

$$\begin{bmatrix} 4 & 3 & 6 \\ 1/2 & 1.5 & 1 \\ 1/2 & -1/3 & 1/3 \end{bmatrix}$$

Notes:

`dgesv` examples

See `$UWHPSC/codes/lapack/random`.

Sample codes that solve the linear system $Ax = b$ with a random $n \times n$ matrix A, where the value n is run-time input.

`randomsys1.f90` is with static array allocation.

`randomsys2.f90` is with dynamic array allocation.

`randomsys3.f90` also estimates condition number of A.

$$\kappa(A) = \|A\| \, \|A^{-1}\|$$

Can bound relative error in solution in terms of relative error in data using this:

$$Ax^* = b^* \text{ and } A\tilde{x} = \tilde{b} \implies \frac{\|\tilde{x} - x^*\|}{\|x^*\|} \le \kappa(A) \frac{\|\tilde{b} - b^*\|}{\|b^*\|}$$

Notes:

Heat Equation / Diffusion Equation

Partial differential equation (PDE) for $u(x,t)$
in one space dimension and time.

u represents temperature in a 1-dimensional metal rod.

Or concentration of a chemical diffusing in a tube of water.

The PDE is

$$u_t(x,t) = Du_{xx}(x,t) + f(x,t)$$

where subscripts represent partial derivatives,

$D =$ diffusion coefficient (assumed constant in space & time),

$f(x,t) =$ source term (heat or chemical being added/removed).

Also need initial conditions $u(x,0)$
and boundary conditions $u(x_1,t)$, $u(x_2,t)$.

Notes:

Steady state diffusion

If $f(x,t) = f(x)$ does not depend on time and if the boundary conditions don't depend on time, then $u(x,t)$ will converge towards steady state distribution satisfying

$$0 = Du_{xx}(x) + f(x)$$

(by setting $u_t = 0$.)

This is now an ordinary differential equation (ODE) for $u(x)$.

We can solve this on an interval, say $0 \le x \le 1$ with

Boundary conditions:

$$u(0) = \alpha, \qquad u(1) = \beta.$$

Notes:

Steady state diffusion

More generally: Take $D = 1$ or absorb in f,

$$u_{xx}(x) = -f(x) \qquad \text{for } 0 \le x \le 1,$$

Boundary conditions:

$$u(0) = \alpha, \qquad u(1) = \beta.$$

Can be solved exactly if we can integrate f twice and use boundary conditions to choose the two constants of integration.

Example: $\alpha = 20, \quad \beta = 60, \quad f(x) = 0$ (no heat source)

Solution: $u(x) = \alpha + x(\beta - \alpha) \qquad \Longrightarrow \; u''(x) = 0.$

No heat source $\Longrightarrow$ linear variation in steady state ($u_{xx} = 0$).

Notes:

Steady state diffusion

More generally: Take $D = 1$ or absorb in f,

$$u_{xx}(x) = -f(x) \qquad \text{for } 0 \le x \le 1,$$

Boundary conditions:

$$u(0) = \alpha, \qquad u(1) = \beta.$$

Can be solved exactly if we can integrate f twice and use boundary conditions to choose the two constants of integration.

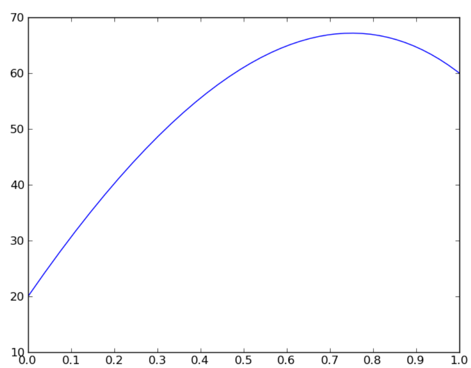
More interesting example:

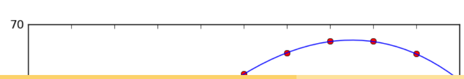
Example: $\alpha = 20, \quad \beta = 60, \quad f(x) = 100e^x,$

Solution: $u(x) = (100e - 60)x + 120 - 100e^x.$

Notes:

Steady state diffusion





Notes:

Finite difference method

Define grid points $x_i = i\Delta x$ in interval $0 \le x \le 1$, where

$$\Delta x = \frac{1}{n+1}$$

So $x_0 = 0,\ x_{n+1} = 1$, and the n grid points $x_1,\ x_2,\ \ldots,\ x_n$ are equally spaced inside the interval.

Let $U_i \approx u(x_i)$ denote approximate solution.

We know $U_0 = \alpha$ and $U_{n+1} = \beta$ from boundary conditions.

Idea: Replace differential equation for $u(x)$ by system of n algebraic equations for U_i values ($i = 1,\ 2,\ \ldots,\ n$).

Notes:

Finite difference method

$U_i \approx u(x_i)$

$u_x(x_{i+1/2}) \approx \frac{U_{i+1}-U_i}{\Delta x}$

$u_x(x_{i-1/2}) \approx \frac{U_i-U_{i-1}}{\Delta x}$

So we can approximate second derivative at x_i by:

$$\begin{aligned} u_{xx}(x_i) &\approx \frac{1}{\Delta x}\left(\frac{U_{i+1}-U_i}{\Delta x} - \frac{U_i - U_{i-1}}{\Delta x}\right) \\ &= \frac{1}{\Delta x^2}\left(U_{i-1} - 2U_i + U_{i+1}\right) \end{aligned}$$

This gives coupled system of n linear equations:

$$\frac{1}{\Delta x^2}\left(U_{i-1} - 2U_i + U_{i+1}\right) = -f(x_i)$$

for $i = 1,\ 2,\ \ldots,\ n.$ With $U_0 = \alpha$ and $U_{n+1} = \beta$.

Notes:

Tridiagonal linear system

$$\begin{aligned} \alpha - 2U_1 + U_2 &= -\Delta x^2 f(x_1) \qquad (i=1) \\ U_1 - 2U_2 + U_3 &= -\Delta x^2 f(x_2) \qquad (i=2) \end{aligned}$$

Etc.

For $n = 5$:

$$\begin{bmatrix} -2 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \end{bmatrix} = -\Delta x^2 \begin{bmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \\ f(x_4) \\ f(x_5) \end{bmatrix} - \begin{bmatrix} \alpha \\ 0 \\ 0 \\ 0 \\ \beta \end{bmatrix}.$$

General $n \times n$ system requires $O(n^3)$ flops to solve.

Tridiagonal $n \times n$ system requires $O(n)$ flops to solve.

Could use LAPACK routine dgtsv.

Notes:

Heat equation in 2 dimensions

One-dimensional equation generalizes to

$$u_t(x, y, t) = D(u_{xx}(x, y, t) + u_{yy}(x, y, t)) + f(x, y, t)$$

on some domain in the x-y plane, with initial and boundary conditions.

We will only consider rectangle $0 \le x \le 1, \ \ 0 \le y \le 1$.

Steady state problem (with $D = 1$):

$$u_{xx}(x, y) + u_{yy}(x, y) = -f(x, y)$$

This is a PDE in two spatial variables. (Poisson Problem)

Laplace's equation if $f(x, y) \equiv 0$.
$\nabla^2 = (\partial_x^2 + \partial_y^2)$ is the Laplacian operator.

Notes:

Finite difference equations for 2D Poisson problem

Let $U_{ij} \approx u(x_i, y_j)$.

Replace differential equation

$$u_{xx}(x, y) + u_{yy}(x, y) = -f(x, y)$$

by algebraic equations

$$\frac{1}{\Delta x^2}(U_{i-1,j} - 2U_{i,j} + U_{i+1,j}) + \frac{1}{\Delta y^2}(U_{i,j-1} - 2U_{i,j} + U_{i,j+1}) = -f(x_i, y_j)$$

If $\Delta x = \Delta y = h$:

$$\frac{1}{h^2}(U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1} - 4U_{i,j}) = -f(x_i, y_j).$$

Notes:

Finite difference equations for 2D Poisson problem

$$\frac{1}{h^2}(U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1} - 4U_{i,j}) = -f(x_i, y_j).$$

On $n \times n$ grid ($\Delta x = \Delta y = 1/(n+1)$) this gives a linear system of n^2 equations in n^2 unknowns.

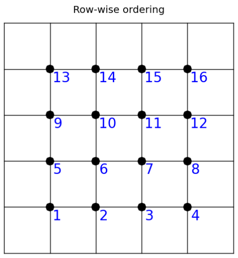
The above equation must be satisfied for $i = 1, \ 2, \ \dots, \ n$ and $j = 1, \ 2, \ \dots, \ n$.

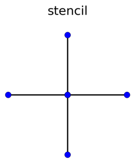
Matrix is $n^2 \times n^2$,
e.g. on 100 by 100 grid, matrix is $10,000 \times 10,000$.
Contains $(10,000)^2 = 100,000,000$ elements.

Matrix is sparse: each row has at most 5 nonzeros out of n^2 elements! But structure is no longer tridiagonal.

Notes:

Finite difference equations for 2D Poisson problem





Matrix has block tridiagonal structure:

$$A = \frac{1}{h^2}\begin{bmatrix} T & I & & \\ I & T & I & \\ & I & T & I \\ & & I & T \end{bmatrix} \qquad T = \begin{bmatrix} -4 & 1 & & \\ 1 & -4 & 1 & \\ & 1 & -4 & 1 \\ & & 1 & -4 \end{bmatrix}$$

Notes:

Iterative methods

Back to one space dimension first...

Coupled system of n linear equations:

$$(U_{i-1} - 2U_i + U_{i+1}) = -\Delta x^2 f(x_i)$$

for $i = 1,\ 2,\ \ldots,\ n.$ With $U_0 = \alpha$ and $U_{n+1} = \beta$.

Iterative method starts with initial guess $U^{[0]}$ to solution and then improves $U^{[k]}$ to get $U^{[k+1]}$ for $k = 0,\ 1,\ \ldots$.

Note: Generally does not involve modifying matrix A.

Do not have to store matrix A at all, only know about stencil.

Notes:

Jacobi iteration

$$(U_{i-1} - 2U_i + U_{i+1}) = -\Delta x^2 f(x_i)$$

Solve for U_i:

$$U_i = \frac{1}{2}\left(U_{i-1} + U_{i+1} + \Delta x^2 f(x_i)\right).$$

Note: With no heat source, $f(x) = 0$,
the temperature at each point is average of neighbors.

Suppose $U^{[k]}$ is a approximation to solution. Set

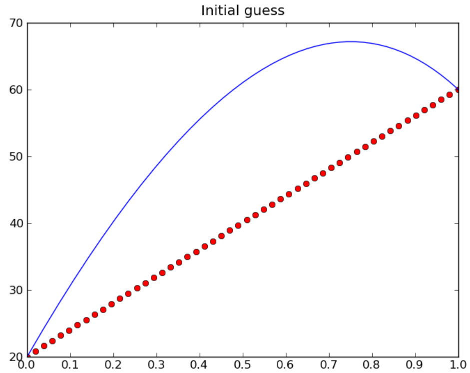
$$U_i^{[k+1]} = \frac{1}{2}\left(U_{i-1}^{[k]} + U_{i+1}^{[k]} + \Delta x^2 f(x_i)\right) \quad \text{for } i = 1,\ 2,\ \ldots,\ n.$$

Repeat for $k = 0,\ 1,\ 2,\ \ldots$ until convergence.

Can be shown to converge (eventually... very slow!)

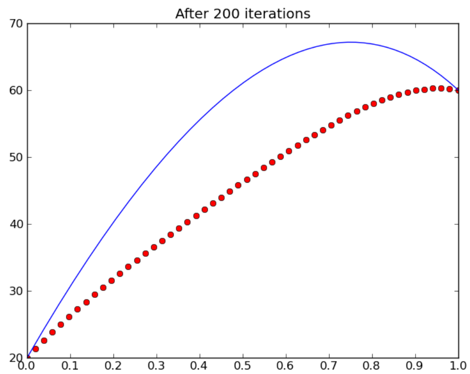
Notes:

Slow convergence of Jacobi



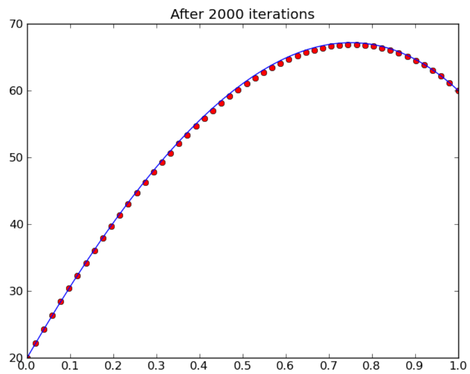
Notes:

Slow convergence of Jacobi



Notes:

Slow convergence of Jacobi



Notes:

Iterative methods

Jacobi iteration is about the worst possible iterative method.

But it's very simple, and useful as a test for parallelization.

Better iterative methods:

- Gauss-Seidel
- Successive Over-Relaxation (SOR)
- Conjugate gradients
- Preconditioned conjugate gradients
- Multigrid

Notes:

Iterative methods – initialization

```
! allocate storage for boundary points too:
allocate(x(0:n+1), u(0:n+1), f(0:n+1))

dx = 1.d0 / (n+1.d0)

!$omp parallel do
do i=0,n+1
    ! grid points:
    x(i) = i*dx
    ! source term:
    f(i) = 100.*exp(x(i))
    ! initial guess (linear function):
    u(i) = alpha + x(i)*(beta-alpha)
    enddo
```


Notes:

Jacobi iteration in Fortran

```
uold = u  ! starting values before updating
do iter=1,maxiter
    dumax = 0.d0
    do i=1,n
      u(i) = 0.5d0*(uold(i-1) + uold(i+1) + dx**2*f(i))
      dumax = max(dumax, abs(u(i)-uold(i)))
      enddo
    ! check for convergence:
    if (dumax .lt. tol) exit
    uold = u  ! for next iteration
    enddo
```

Note: we must use old value at $i-1$ for Jacobi.

Otherwise we get the Gauss-Seidel method.

```
u(i) = 0.5d0*(u(i-1) + u(i+1) + dx**2*f(i))
```

This actually converges faster!

Notes:

Jacobi with OpenMP `parallel do` (fine grain)

See: $UWHPSC/codes/openmp/jacobi1d_omp1.f90

```
uold = u  ! starting values before updating

do iter=1,maxiter

    dumax = 0.d0

    !$omp parallel do reduction(max : dumax)
    do i=1,n
      u(i) = 0.5d0*(uold(i-1) + uold(i+1) + dx**2*f(i))
      dumax = max(dumax, abs(u(i)-uold(i)))
      enddo

    ! check for convergence:
    if (dumax .lt. tol) exit

    !$omp parallel do
    do i=1,n
        uold(i) = u(i) ! for next iteration
        enddo
    enddo
```

Note: Forking threads twice each iteration.

Notes:

Jacobi with OpenMP – coarse grain

General Approach:

- Fork threads only once at start of program.
- Each thread is responsible for some portion of the arrays, from `i=istart` to `i=iend`.
- Each iteration, must copy `u` to `uold`, update `u`, check for convergence.
- Convergence check requires coordination between threads to get global `dumax`.
- Print out final result after leaving parallel block

See code in the repository or the notes:
$UWHPSC/codes/openmp/jacobi1d_omp2.f90

Notes: