

AMath 483/583 — Lecture 19 — May 9, 2011

Today:

- MPI send and receive
- Heat equation and discretization

Wednesday:

- Iterative methods

Read: Class notes and references

Notes:

MPI — Simple example

```
program test1
  use mpi
  implicit none
  integer :: ierr, numprocs, proc_num,

  call mpi_init(ierr)
  call mpi_comm_size(MPI_COMM_WORLD, numprocs, ierr)
  call mpi_comm_rank(MPI_COMM_WORLD, proc_num, ierr)

  print *, 'Hello from Process ', proc_num, &
    ' of ', numprocs, ' processes'

  call mpi_finalize(ierr)
end program test1
```

Always need to: use mpi,
Start with mpi_init,
End with mpi_finalize.

Notes:

mpi module

The `mpi` module includes:

Subroutines such as `mpi_init`, `mpi_comm_size`,
`mpi_comm_rank`, ...

Global variables such as

`MPI_COMM_WORLD`: a communicator,
`MPI_INTEGER`: used to specify the type of data being sent
`MPI_SUM`: used to specify a type of reduction

Remember: Fortran is **case insensitive**:
`mpi_init` is the same as `MPI_INIT`.

Notes:

MPI functions

There are 125 MPI functions.

Can write many program with these 8:

- `MPI_INIT(ierr)` Initialize
- `MPI_FINALIZE(ierr)` Finalize
- `MPI_COMM_SIZE(...)` Number of processors
- `MPI_COMM_RANK(...)` Rank of this processor
- `MPI_SEND(...)` Send a message
- `MPI_RCV(...)` Receive a message
- `MPI_BCAST(...)` Broadcast to other processors
- `MPI_REDUCE(...)` Reduction operation

Notes:

MPI Reduce for vectors

Compute: $\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^m |a_{ij}|$ for an $m \times n$ matrix A .

Suppose there are m processes and the i th process has a vector `arow(1:n)` containing the i th row of A .

Sum the row vectors (using Reduce) and then take maximum of resulting vector...

```
real(kind=8) :: arow(n), arow_abs(n), colsum(n)
...
arow_abs = abs(arow)
call MPI_REDUCE(arow_abs(1), colsum, n, &
               MPI_DOUBLE_PRECISION, MPI_SUM, 0, &
               MPI_COMM_WORLD, ierr)

if (proc_num == 0) then
  anorm = 0.d0
  do j=1,n
    anorm = max(anorm, colsum(j))
  enddo
  print "1-norm of A = ", anorm
endif
```

Notes:

MPI AllReduce

To make a reduction available to *all* processes:

```
call MPI_REDUCE(xnorm_proc, xnorm, 1, &
               MPI_DOUBLE_PRECISION, MPI_SUM, 0, &
               MPI_COMM_WORLD, ierr)
! only Process 0 has the value of xnorm

call MPI_BCAST(xnorm, 1, &
               MPI_DOUBLE_PRECISION, 0, &
               MPI_COMM_WORLD, ierr)
```

Or: simpler and perhaps more efficient...

```
call MPI_ALLREDUCE(xnorm_proc, xnorm, 1, &
                  MPI_DOUBLE_PRECISION, MPI_SUM, &
                  MPI_COMM_WORLD, ierr)
```

Notes:

MPI Send and Receive

`MPI_BCAST` sends from one process to all processes.

Often want to send selectively from Process i to Process j .

Use `MPI_SEND` and `MPI_RECV`.

Need a way to **tag** messages so they can be identified.

The parameter `tag` is an integer that can be matched to identify a message.

Tag can also be used to provide information about what is being sent, for example if a Master process sends rows of a matrix to other processes, the `tag` might be the row number.

Notes:

MPI Send

Send value(s) from this Process to Process `dest`.

General form:

```
call MPI_SEND(start, count, &
              datatype, dest, &
              tag, comm, ierr)
```

where:

- `start`: starting address (variable, array element)
- `count`: number of elements to send
- `datatype`: type of each element
- `dest`: destination process
- `tag`: identifier tag (integer between 0 and 32767)
- `comm`: communicator

Notes:

MPI Receive

Receive value(s) from Process `source` with label `tag`.

General form:

```
call MPI_RECV(start, count, &
              datatype, source, &
              tag, comm, status, ierr)
```

where:

- `source`: source process
- `tag`: identifier tag (integer between 0 and 32767)
- `comm`: communicator
- `status`: integer array of length `MPI_STATUS_SIZE`.

`source` could be `MPI_ANY_SOURCE` to match any source.

`tag` could be `MPI_ANY_TAG` to match any tag.

Notes:

MPI Send and Receive — simple example

```
if (proc_num == 4) then
  i = 55
  call MPI_SEND(i, 1, MPI_INTEGER, 3, 21, &
               MPI_COMM_WORLD, ierr)
endif
if (proc_num == 3) then
  call MPI_RECV(j, 1, MPI_INTEGER, 4, 21, &
               MPI_COMM_WORLD, status, ierr)
  print *, "j = ", j
endif
```

Processor 3 will print `j = 55`

The **tag** is 21. (Arbitrary integer between 0 and 32767)

Blocking Receive: Processor 3 won't return from `MPI_RECV` until message is received.

Run-time error if `num_procs <= 4` (Procs are 0,1,2,3)

R.J. LeVeque, University of Washington AMath 483/583, Lecture 19, May 9, 2011

Notes:

R.J. LeVeque, University of Washington AMath 483/583, Lecture 19, May 9, 2011

Send/Receive example

Pass value of `i` from Processor 0 to 1 to 2 ... to `num_procs-1`

```
if (proc_num == 0) then
  i = 55
  call MPI_SEND(i, 1, MPI_INTEGER, 1, 21, &
               MPI_COMM_WORLD, ierr)
endif

else if (proc_num < num_procs - 1) then
  call MPI_RECV(i, 1, MPI_INTEGER, proc_num-1, 21, &
               MPI_COMM_WORLD, status, ierr)

  call MPI_SEND(i, 1, MPI_INTEGER, proc_num+1, 21, &
               MPI_COMM_WORLD, ierr)

else if (proc_num == num_procs - 1) then
  call MPI_RECV(i, 1, MPI_INTEGER, proc_num-1, 21, &
               MPI_COMM_WORLD, status, ierr)
  print *, "i = ", i
endif
```

R.J. LeVeque, University of Washington AMath 483/583, Lecture 19, May 9, 2011

Notes:

R.J. LeVeque, University of Washington AMath 483/583, Lecture 19, May 9, 2011

MPI Receive

Receive value(s) from Process `source` with label `tag`.

General form:

```
call MPI_RECV(start, count, &
              datatype, source, &
              tag, comm, status, ierr)
```

where:

- `source`: source process
- `tag`: identifier tag (integer between 0 and 32767)
- `comm`: communicator
- `status`: integer array of length `MPI_STATUS_SIZE`.

`source` could be `MPI_ANY_SOURCE` to match any source.

`tag` could be `MPI_ANY_TAG` to match any tag.

R.J. LeVeque, University of Washington AMath 483/583, Lecture 19, May 9, 2011

Notes:

R.J. LeVeque, University of Washington AMath 483/583, Lecture 19, May 9, 2011

MPI Receive — status argument

```
call MPI_RECV(start, count, &
              datatype, source, &
              tag, comm, status, ierr)
```

Elements of the `status` array give additional useful information about the message received.

In particular,

`status(MPI_SOURCE)` is the **source** of the message,
May be needed if `source = MPI_ANY_SOURCE`.

`status(MPI_TAG)` is the **tag** of the message received,
May be needed if `tag = MPI_ANY_TAG`.

Notes:

Another Send/Receive example

Master (Processor 0) sends j th column to Worker Processor j ,
gets back 1-norm to store in `anorm(j)`, $j = 1, \dots, n_{\text{cols}}$

```
! code for Master (Processor 0):
if (proc_num == 0) then
  do j=1,ncols
    call MPI_SEND(a(1,j), nrows, MPI_DOUBLE_PRECISION, &
                 j, j, MPI_COMM_WORLD, ierr)
  enddo
  do j=1,ncols
    call MPI_RECV(colnorm, 1, MPI_DOUBLE_PRECISION, &
                 MPI_ANY_SOURCE, MPI_ANY_TAG, &
                 MPI_COMM_WORLD, status, ierr)
    jj = status(MPI_TAG)
    anorm(jj) = colnorm
  enddo
endif
```

Note: Master may receive back in any order!
`MPI_ANY_SOURCE` will match first to arrive.

The tag is used to tell which column's norm has arrived (`jj`).

Notes:

Send and Receive example — worker code

Master (Processor 0) sends j th column to Worker Processor j ,
gets back 1-norm to store in `anorm(j)`, $j = 1, \dots, n_{\text{cols}}$

```
! code for Workers (Processors 1, 2, ...):
if (proc_num /= 0) then
  call MPI_RECV(colvect, nrows, MPI_DOUBLE_PRECISION, &
               0, MPI_ANY_TAG, &
               MPI_COMM_WORLD, status, ierr)
  j = status(MPI_TAG) ! this is the column number
  colnorm = 0.d0
  do i=1,nrows
    colnorm = colnorm + abs(colvect(i))
  enddo
  call MPI_SEND(colnorm, 1, MPI_DOUBLE_PRECISION, &
               0, j, MPI_COMM_WORLD, ierr)
endif
```

Note: Sends back with tag j .

Notes:

Send may be blocking

```
if (proc_num == 4) then
  i = 55
  call MPI_SEND(i, 1, MPI_INTEGER, 3, 21, &
               MPI_COMM_WORLD, ierr)
  call MPI_RECV(j, 1, MPI_INTEGER, 3, 22, &
               MPI_COMM_WORLD, status, ierr)
endif

if (proc_num == 3) then
  j = 66
  call MPI_SEND(j, 1, MPI_INTEGER, 4, 22, &
               MPI_COMM_WORLD, ierr)
  call MPI_RECV(i, 1, MPI_INTEGER, 4, 21, &
               MPI_COMM_WORLD, status, ierr)
endif
```

Both processors might get stuck in `MPI_SEND`!

Implementation-dependent: waits for send buffer to be free.

Blocking send: `MPI_SSEND`. See [documentation](#)

There are also non-blocking sends and receives:

`MPI_ISEND`, `MPI_IRecv`

Notes:

Heat Equation / Diffusion Equation

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

u represents temperature in a 1-dimensional metal rod, for example.

Or concentration (density) 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,

$f(x, t)$ = source term.

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 $0 \leq x \leq 1$ with

[Boundary conditions](#):

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

Notes:

Steady state diffusion

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

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

Boundary conditions:

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

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

Example: $f(x) = 0$, $\alpha = 20$, $\beta = 60$:

Solution: $u(x) = \alpha + x(\beta - \alpha)$.

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

Notes:

Steady state diffusion

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

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

Boundary conditions:

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

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

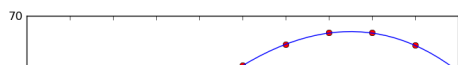
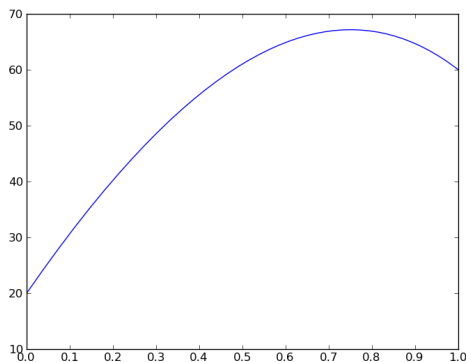
More interesting example:

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

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

Notes:

Steady state diffusion



Notes:

Finite difference method

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

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

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

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

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

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

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:

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

This gives coupled system of n linear equations:

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

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

Notes:

Tridiagonal linear system

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: