Structured grids and finite differences

the Poisson equation, continued

Ed Bueler
Dept. of Mathematics and Statistics, UAF

16 February 2016

outline for today

we are in the midst of Chapter 3 of the book:

- FD method generates linear system $A\mathbf{u} = \mathbf{b}$
  - with care, A is symmetric and positive definite
- build a 2D structured grid with PETSc's `DMDA`
  - distributed across processors, with "ghosts"
- look at the code `poisson.c`
  - `Vec` and `Mat` assembled with grid indices i, j
- experiment with:
  - visualization

FD scheme gives linear system

- Poisson equation:

$$-\nabla^2 u = f$$

- FD equations for our Poisson boundary-value problem:

$$-\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_x^2} - \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_y^2} = f_{i,j} \quad (*)$$

$$u_{0,j} = 0, \quad u_{m_x-1,j} = 0, \quad u_{i,0} = 0, \quad u_{i,m_y-1} = 0$$

  - grid indices $i = 0, \ldots, m_x - 1$ and $j = 0, \ldots, m_y - 1$
  - $(*)$ applies at each interior point
  - boundary conditions are trivial equations: "$1\, u = 0$"
- linear system of $L = m_x m_y$ equations in L unknowns

$$A\mathbf{u} = \mathbf{b}$$

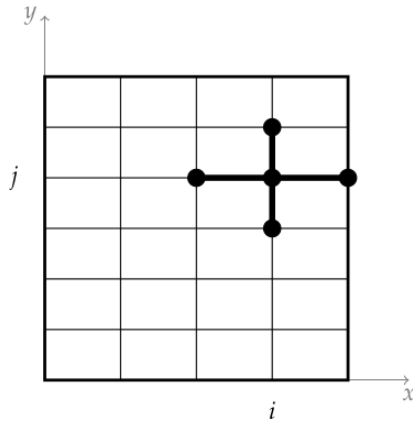
where A is $L \times L$ matrix and $\mathbf{u}, \mathbf{b}$ are $L \times 1$ column vectors

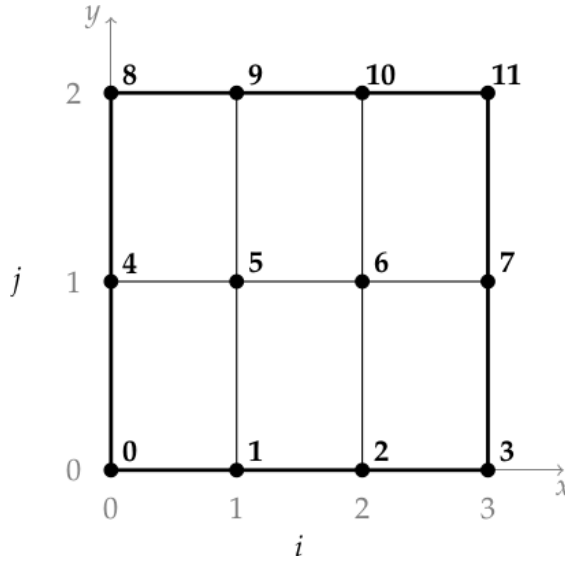
ordering of unknowns

- actually building linear system requires global ordering of unknowns: $k = 0, 1, \ldots, L$
- $m_x = 4$ and $m_y = 3$ case has $L = 12$:

$$\begin{bmatrix} 1 & & & & & & & & & & & \\ & 1 & & & & & & & & & & \\ & & 1 & & & & & & & & & \\ & & & 1 & & & & & & & & \\ & & & & 1 & & & & & & & \\ c & & & b & a & b & & & c & & & \\ & c & & & b & a & b & & & c & & \\ & & & & & & 1 & & & & & \\ & & & & & & & 1 & & & & \\ & & & & & & & & 1 & & & \\ & & & & & & & & & 1 & & \\ & & & & & & & & & & & 1 \end{bmatrix} \begin{bmatrix} u_{0,0} \\ u_{1,0} \\ u_{2,0} \\ u_{3,0} \\ u_{0,1} \\ u_{1,1} \\ u_{2,1} \\ u_{3,1} \\ u_{0,2} \\ u_{1,2} \\ u_{2,2} \\ u_{3,2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ f_{1,1} \\ f_{2,1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

  - only $k = 5, 6$ eqns are *not* b.c.s
- (weak) diagonal dominance: $a = |2b + 2c|$
- matrix is not symmetric
- surprisingly-large condition number for small example: $\kappa(A) = 43.16$





symmetrize Dirichlet conditions & scale equations

- non-symmetry comes from having boundary values appearing on the left, so: move these over
- also: multiply by cell area $\Delta A = h_x h_y$ to make coefficients $O(1)$
- get better system $A\mathbf{u} = \mathbf{b}$:

$$\begin{bmatrix} 1 & & & & & & & & & & & \\ & 1 & & & & & & & & & & \\ & & 1 & & & & & & & & & \\ & & & 1 & & & & & & & & \\ & & & & 1 & & & & & & & \\ & & & & & \alpha & \beta & & & & & \\ & & & & & \beta & \alpha & & & & & \\ & & & & & & & 1 & & & & \\ & & & & & & & & 1 & & & \\ & & & & & & & & & 1 & & \\ & & & & & & & & & & 1 & \\ & & & & & & & & & & & 1 \end{bmatrix} \begin{bmatrix} u_{0,0} \\ u_{1,0} \\ u_{2,0} \\ u_{3,0} \\ u_{0,1} \\ u_{1,1} \\ u_{2,1} \\ u_{3,1} \\ u_{0,2} \\ u_{1,2} \\ u_{2,2} \\ u_{3,2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ h_x h_y f_{1,1} \\ h_x h_y f_{2,1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- matrix A is now symmetric, positive definite, and better-scaled than before: $\kappa(A) = 5.83$

PETSc can manage the grid

- PETSc `DMDA` object holds layout of your structured grid
  - it can convert grid indices i, j to unknown order k
  - you tell it: type/size of stencil
  - you tell it: which grid points are b.c.s
  - it decides on parallel distribution of grid (unless you tell it)
- create it this way:

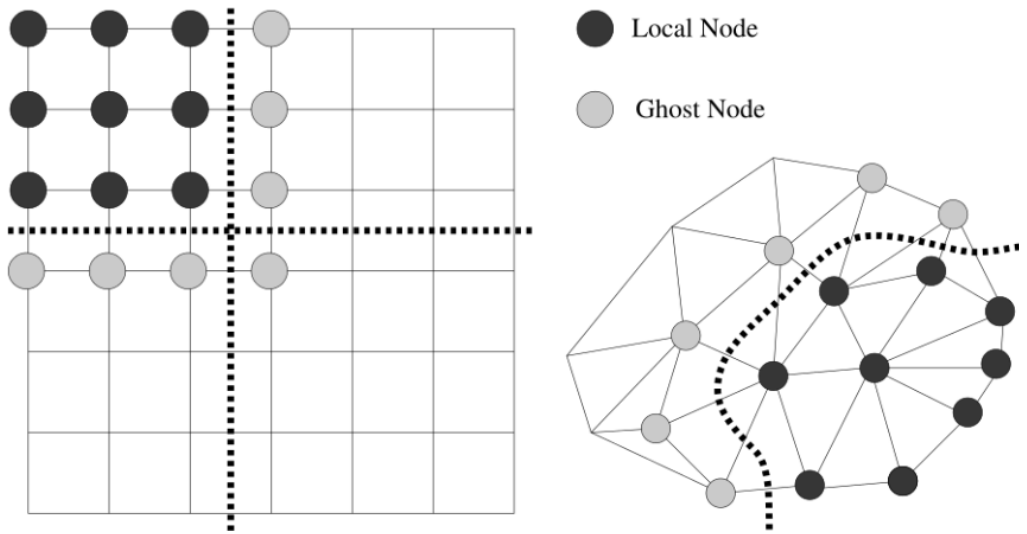
```
DMDACreate2d(MPI_Comm comm,
  DMBoundaryType bx, DMBoundaryType by, DMDAStencilType stype,
  PetscInt M, PetscInt N, PetscInt m, PetscInt n,
  PetscInt dof, PetscInt s,
  const PetscInt lx[], const PetscInt ly[], DM *da)
```

where

`comm`	MPI communicator
`bx,by`	use one of `DM_BOUNDARY_NONE, ..._GHOSTED, ..._PERIODIC`
`stype`	either `DMDA_STENCIL_BOX` or `DMDA_STENCIL_STAR`
`M,N`	global dimension in each direction
`m,n`	number of processes in each dimension or `PETSC_DECIDE`
`dof`	number of degrees of freedom per node (`dof` $>$ 1 for systems of PDEs)
`s`	stencil width (our star stencil has `s` $=$ 1)
`lx,ly`	number of nodes for each processor or `NULL`
`da`	output: the `DMDA` object

to compute $A\mathbf{u}$ you need "ghosted" values

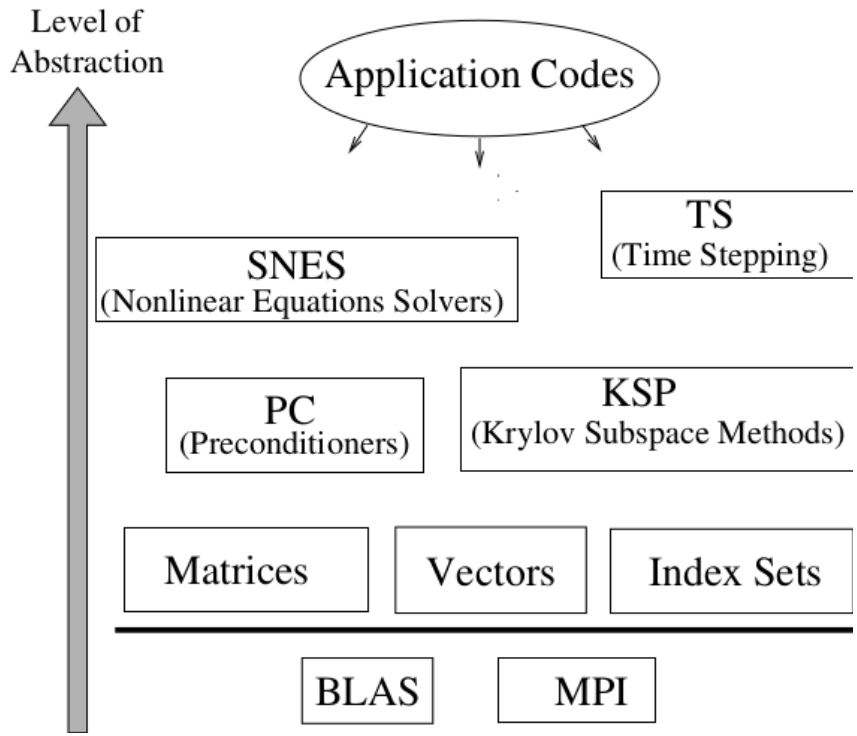
- suppose $A\mathbf{u} = \mathbf{b}$ is solved by a Krylov method for A generated from FD method (or FV or FEM . . .)
- need mat-vec products $\mathbf{w} = A\mathbf{v}$ where ownership of $\mathbf{v}$ is spread across grid
- each processor-owned "patch" of $\mathbf{w} = A\mathbf{v}$ needs to access grid neighbor values of $\mathbf{v}$ owned by neighbor process
- thus PETSc "local"/"ghost" node distinction:



  - left fig. shows `STENCIL_BOX` but our FD is `STENCIL_STAR`

a message from our sponsor: MPI

- PETSc `Vec` and `Mat` are distributed across processes
  - by rows
  - this idea needs no grid
- always under the hood: message passing to do $A\mathbf{v}$
- `DMDA` lets us think of a `Vec` as distributed across process-owned patchs



time to look at the code: `c/ch3/poisson.c`

- `main()`:
  - calls `DMDACreate2d()`
  - assembles linear system by calling our code (below)
  - `KSPCreate()`,`KSPSetOperators()`,`KSPSetFromOptions()`
  - `KSPSolve()`
  - numerical error using `VecAXPY()` and `VecNorm()`

user-written assembly code:

- `formMatrix()` builds A by rows using grid indices i, j
  - `MatSetValuesStencil()` sets matrix entries using i, j
  - note `DMDALocalInfo`; will show in a moment
- `formExact()` computes exact soln $u(x, y)$ for error calc
  - use `DMDAVecGetArray()` to set `Vec` entries using i, j
- `formRHS()` computes $f(x, y)$

run it

- compile and run:

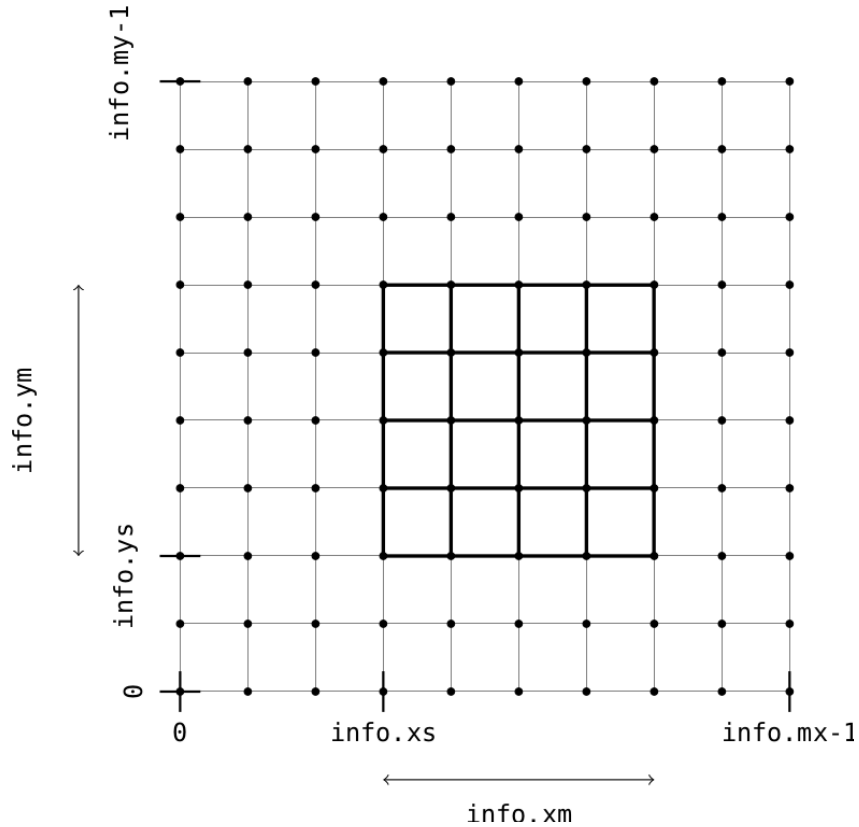
```
$ cd c/ch3/
$ make poisson
$ ./poisson
```

- examine and control behavior:

```
$ ./poisson -ksp_monitor
$ ./poisson -ksp_view
$ ./poisson -da_refine 1
$ ./poisson -da_refine 4
$ ./poisson -da_grid_x 5 -da_grid_y 7
$ ./poisson -ksp_monitor -da_refine 4 \
    -ksp_monitor_solution draw
$ ./poisson -ksp_monitor -ksp_rtol 1.0e-3
```

one process' portion of the grid: `DMDALocalInfo`

- call `DMDAGetLocalInfo(da,&info)` to get a `struct` of type `DMDALocalInfo`; gives 3 integers in each dimension:



- 2D loop over process' portion of the grid:

```
for (j = info.ys; j < info.ys+info.ym; j++) {
  for (i = info.xs; i < info.xs+info.xm; i++) {
    ...
```

ask PETSc to show how grid is distributed

- use option `-dm_view`

```
$ ./poisson -dm_view
$ mpiexec -n 2 ./poisson -dm_view
$ mpiexec -n 2 ./poisson -dm_view draw -draw_pause 2
$ mpiexec -n 4 ./poisson -da_grid_x 5 -da_grid_y 7 \
    -dm_view draw -draw_pause 2
```

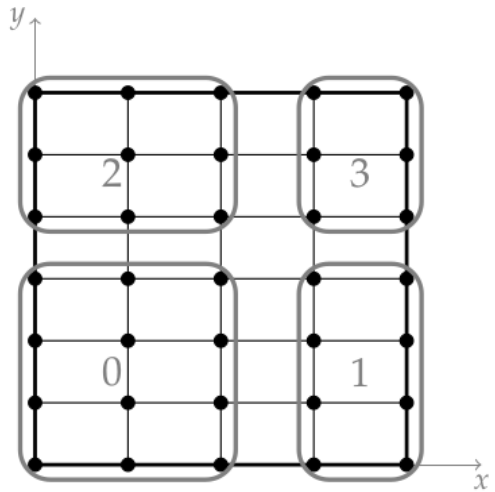


Figure 3.3: The same grid as in Figure 3.2, distributed across four MPI processes, with `rank` $\in \{0,1,2,3\}$ in gray, by `DMDACreate2d()`.