1) Adjust your path by typing:
   ```
   set path = ($path .)
   ```
2) Setup a local workspace in your own directory with the following commands:
   ```
   mkdir isis_work
   cp -r /nfs/o2-10/scratch/tra1/isis_work/* isis_work
   cd isis_work
   make all
   ```
   (Note: If you're using o2-10, just copy from /scratch/tra1/isis_work/)

The files you've copied fall into several categories:

- `convex_hull.m fv.m mmread.m mmwrite.m rsort.m show_eigs.m`
  These Matlab files will be used to extract eigenvalue information from the matrices we examine.
- `hb2mtxstrm  hbrhs2mtx`
  These are utilities to convert Harwell-Boeing matrices to a Matrix-Market format.
- `genmat.f matpde.f libskit.a`
  These files allow us to generate test matrices that result from the finite-difference discritization of a PDE. libskit is the SPARSKIT2 library, whose conversion routines are used by matgen in the matrix generation process.
- `Makefile make.options`
  Makefiles for everything you'll need to experiment.
- `seqDriver.cc genPrecon.cc timers.c driverUtils.cc`
  Code to use ISIS++ to solve systems of linear equations and collect timing data.
- `run-genPrecon.sh run-seqDriver.sh`
  These files serve as a shell for the ISIS++ driver routines. Edit these files to conduct your experiments.
- `/jpgs`
  Eigenvalue information for select matrices precomputed in matlab and stored as a jpg.
- `/mtx`
  Interesting matrices to examine.

Notes on genmat/matpde

MATPDE was written by H. Ellman, University of Maryland. This generator computes a five-point central finite difference discretization of the two-dimensional variable-coefficient linear elliptic equation

$$-(p\,u_x)_x - (q\,u_y)_y + r\,u_x + (r\,u)_x + s\,u_y + (s\,u)_y + t\,u = f$$

where p, q, r, s and t are the functions of x and y, the domain is the unit square $(0,1)\times(0,1)$, and the boundary condition are Dirichlet. The code handles arbitrary p, q, r, s and t, but is set up initially with $p = \exp(-xy)$, $q = \exp(xy)$, $r = \beta\,(x+y)$, $s = \gamma\,(x+y)$, and $t = 1/(1+x+y)$, $\beta = 20$, $\gamma = 0$. To change these parameters, edit the file `matpde.f` and recompile.

1) Create a matrix using matgen:
```
matgen.x
enter the mesh dimensions nx, ny
15, 15
enter name of file to output matrix
pde225.hb
Matrix generation successful.
```
2) The matrix has been created and stored using a Harwell-Boeing format. To use ISIS++, we need to convert this to a Matrix-Market format. Type the following:
`hb2mtxstrm pde225.hb > pde225.mtx` (creates a MM-format matrix)
`hbrhs2mtx pde225.hb pde225` (creates a MM-format right-hand side vector)
3) Now edit the file `run-seqDriver.sh` to show the following:
```
seqDriver.x \
-A pde225.mtx -b pde225.txt \
-solver gmres -pc spai
```
4) Now execute run `run-seqDriver.sh`.
```
...
PC calculate wall clock time:   2.980232e-06 sec.
Solve wall clock time:          6.687452e+00 sec.
Solve & PC calc time:           6.687455e+00 sec.
```
You'll see timing data similar to this, as well as a count of the number of iterations required for convergence. Copy this data onto blanks on your worksheet. Repeat for the case of a diagonal preconditioner ("diagonal") and no preconditioner ("identity").
5) Now have a look at the images `pde225.jpg`, `pde225_diag.jpg`, and `pde225_spai.jpg` in /jpgs. You'll see (in the complex plane) the eigenvalues, their convex hull, the field of values, and the bounding box for the field of values. Note the change in structure of the eigenvalues between the nonpreconditioned and preconditioned systems.
6) If you want to generate these images yourself, you may do so in matlab. First, let's generate the preconditioner matrices. Edit the file `run-genPrecon.sh` to show the preconditioner, rhs, and matrix you want. then execute `run-genPrecon.sh`. It will produce a Matrix-Market format file `precon.mtx`. This file contains the explicit preconditioner matrix, *not* the preconditioned system! Rename this file to something more meaningful:
`mv precon.mtx pde225_spai.mtx`
Now, load matlab, then type the following:
```
» A = mmread('pde225.mtx');       (load the matrix pde225.mtx)
» show_eigs(A);                   (look at its eigenvalues)
» P = mmread('pde225_spai.mtx'); (load the SPAI preconditioner matrix)
» B = P*A;                        (apply the preconditioner)
» show_eigs(B);                   (now examine the preconditioned system)
```
Repeat this process to investigate other matrices.

pde225: http://math.nist.gov/MatrixMarket/data/NEP/matpde/pde225.html
e05r0300: http://math.nist.gov/MatrixMarket/data/SPARSKIT/drivcav/e05r0300.html
e05r0500: http://math.nist.gov/MatrixMarket/data/SPARSKIT/drivcav/e05r0500.html

Problem: PDE225	**15×15 system from genmat**				
Solver	**Preconditioner**	**Num. Iters.**	**PC Calc. Time**	**Solve Time**	**Total Time**
GMRES(90)	Identity				
	Diagonal				
	SPAI				
Solver	**Preconditioner**	**Num. Iters.**	**PC Calc. Time**	**Solve Time**	**Total Time**
BICGSTAB	Identity				
	Diagonal				
	SPAI				
Problem: PDE1600	**40×40 system from genmat**				
Solver	**Preconditioner**	**Num. Iters.**	**PC Calc. Time**	**Solve Time**	**Total Time**
GMRES(90)	Identity				
	Diagonal				
	SPAI				
Solver	**Preconditioner**	**Num. Iters.**	**PC Calc. Time**	**Solve Time**	**Total Time**
BICGSTAB	Identity				
	Diagonal				
	SPAI				
Problem: e05r0300	**from MatrixMarket**				
Solver	**Preconditioner**	**Num. Iters.**	**PC Calc. Time**	**Solve Time**	**Total Time**
GMRES(90)	Identity				
	Diagonal				
	SPAI				
Solver	**Preconditioner**	**Num. Iters.**	**PC Calc. Time**	**Solve Time**	**Total Time**
BICGSTAB	Identity				
	Diagonal				
	SPAI				
Problem: e05r0500	**from MatrixMarket**				
Solver	**Preconditioner**	**Num. Iters.**	**PC Calc. Time**	**Solve Time**	**Total Time**
GMRES(90)	Identity				
	Diagonal				
	SPAI				
Solver	**Preconditioner**	**Num. Iters.**	**PC Calc. Time**	**Solve Time**	**Total Time**
BICGSTAB	Identity				
	Diagonal				
	SPAI				