PECOS

Predictive Engineering and Computational Sciences

Software Verification Workshop

Nicholas Malaya, Chris Simmons

Center for Predictive Engineering and Computational Sciences (PECOS)
Institute for Computational Engineering and Sciences (ICES)
The University of Texas at Austin

July 9th, 2012

Workshop

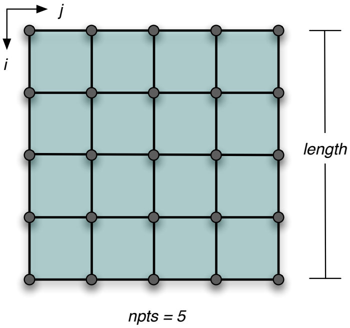
This talk is online:

`users.ices.utexas.edu/~nick`

Goals

- Walk you through the process of code verification
- Build/install MASA
- Do a grid-refinement study for solution verification
- Write some code to have a little fun - do something simple and use MASA

Problem: Solve 2D Laplacian using Finite-Differencing



Recall:

- Laplace's Equation in 2D:

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$$

- For the verification exercise, we will replace the RHS above with a forcing function $f(x, y)$ that we get from MASA

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = f(x, y)$$

Problem: Solve 2D Laplacian using Finite-Differencing

Outline

- *Goal:* Write a program in C/C++, F90, or Matlab/Octave which solves the two-dimensional Laplacian on a square domain
- *Inputs*:
  - # of points in one direction (*npts*)
  - the physical dimension of one side (*length*)
- *Output*: l_2 error between your numerical solution and an exact solution derived from a manufactured solution in MASA

$$l_2 = \sqrt{\frac{\sum_{i=1}^{\mathrm{N}} (\phi_i - \phi_i^{\text{exact}})^2}{N}}$$

- *Runs*: Run your snazzy code for npts $= 5, 9, 17, and 33$ and plot l_2 norm as a function of $1/h$ where $h = length/(npts - 1)$

Finite-difference Scheme

Method

- Let us use a simple FD approximation for the Laplacian
- Assume a constant spacing mesh for convenience
- Central-differencing

$$\nabla^2 \phi_{i,j} \approx \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{h^2} + \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{h^2} + O(h^2)$$

- Use this formula to build the coefficient entries into a linear system $Ax = b$.
- The size of the linear system is the number of solution points. Since we are on a square domain, $N = npts * npts$
- You may find it convenient to use a mapping from a 2D index $\phi_{i,j}$ to a 1D index for the solution vector of your linear system, ϕ_{index}

$$index = j + (i * npts);$$

Finite-difference Scheme

Boundary Conditions

- The 5-point FD stencil is incomplete on the boundaries of our square domain
- We need to apply constraints to matrix A to enforce the Dirchlet conditions on the boundaries
- Simplest method to enforce BCs:
  - zero out all matrix entries on the row associated with boundary point, ϕ_i
  - set the diagonal $A(i,i)$ = 1.0
  - Set the RHS function to the desired solution $f_i = \phi_{\text{exact}}$

Finite-difference Scheme

Boundary Conditions

- Let's look at form of system matrix A^* after BCs have been applied for $npts = 3$ (note $A = \frac{1}{h^2} A^*$):

	j = 0	1	2	3	4	5	6	7	8
i = 0:	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
i = 1:	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
i = 2:	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
i = 3:	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
i = 4:	0.00	1.00	0.00	1.00	-4.00	1.00	0.00	1.00	0.00
i = 5:	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
i = 6:	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
i = 7:	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
i = 8:	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

- In this case, we only have *one* active interior solution point

Fortran 90 Reminder: What you need from MASA

```
program main
  use masa
  implicit none

  dx = real(lx)/real(nx)
  dy = real(ly)/real(ny);

  ! initialize the problem
  call masa_init("laplace example","laplace_2d")

  ! evaluate source terms (2D)
  do i=0, nx
     do j=0, ny

        y = j*dy
        x = i*dx

        ! evalulate source term
        field = masa_eval_2d_source_f   (x,y)

        ! evaluate analytical term
        exact_phi = masa_eval_2d_exact_phi (x,y)

     enddo
  enddo

end program main
```

C Reminder: What you need from MASA

```
#include <masa.h>

int main()
{
  err += masa_init("laplace example","laplace_2d");

  // grab / set parameter values
  Lx = masa_get_param("Lx");
  masa_set_param("Ly",42.0);

  for(int i=0;i<nx;i++)
    for(int j=0;j<nx;j++)
      {
        x=i*dx;
        y=j*dy;

        // source term
        ffield    = masa_eval_2d_source_f (x,y);

        // manufactured solution
        phi_field = masa_eval_2d_exact_phi(x,y);

       } // finished iterating over space
} //end program
```

Installing MASA locally

Steps for Building MASA:

- Grab latest tarball (https://red.ices.utexas.edu/attachments/download/1560/masa-0.40.2.tar.gz)
  - https://red.ices.utexas.edu/projects/software/files
- Untar: tar xvfz masa-0.40.2.tar.gz
- Configure: ./configure –prefix=$HOME/masa
- Compile: make -j 2
- Test: make check
- Install locally: make install
- To generate documenations: make docs
  - Can then point a broswer to docs/html/index.html

Linking to your installed MASA

Linking against your local build

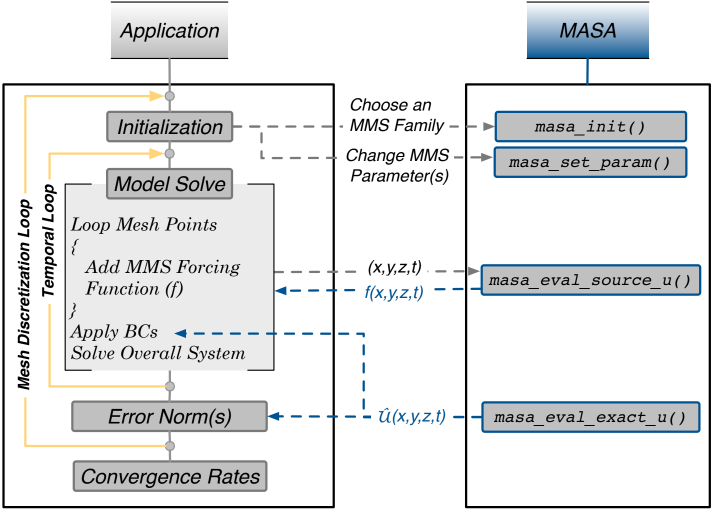
- **C**: Assuming your code is named laplacian.c and you installed masa into $HOME/masa:

```
gcc -I$HOME/masa/include laplacian.c -L$HOME/masa/lib -lmasa
```

- **F90**: Assuming your code is named laplacian.f90

```
gfortran -I$HOME/masa/lib laplacian.f90 -L$HOME/masa/lib -lmasa -lfmasa
```

General Verification Approach Using MMS and MASA



General Program Flow (a C example)

```
int main(int argc, char *argv[])
{
  int n;
  double length;
  pstruct model;                /* primary model data structure */

  /* Parse command-line */

  if(argc < 2)
    {
      printf("\nUsage: laplacian [num_pts] [length]\n\n");
      printf("where \"num_pts\" is the desired number of mesh points and \n");
      printf("\"length\" is the physical length-scale dimension in one direction\n\n");

      exit(1);
    }
  else
    {
      n      = atoi(argv[1]);
      length = (double) atof(argv[2]);
    }

  /* Problem Initialization */

  problem_initialize (n,length,&model);
  assemble_matrix    (1,&model);
  init_masa          (&model);
  apply_bcs          (&model);

  .....
```

General Program Flow (a C example, continued)

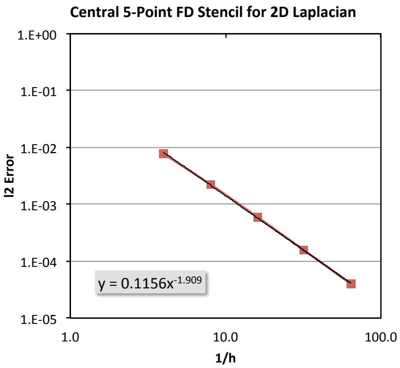
```
  /* Solve linear system */

  solve_gauss       (&model);

  /* Compute Error */

  printf("\n** Error Analysis\n");
  printf("   --> npts     = %i\n",model.npts);
  printf("   --> h        = %12.5e\n",model.h);
  printf("   --> l2 error = %12.5e\n",compute_l2_error(&model));

  return 0;

}
```

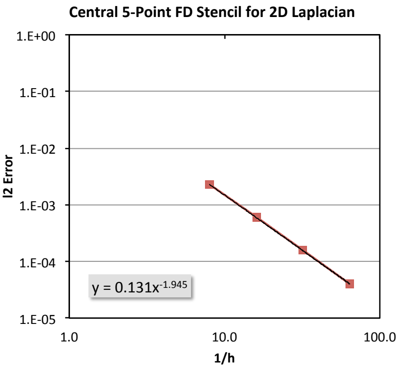
Example Model Data Structure

```
typedef struct pstruct {
  double  *phi;                  /*!< solution variable                     */
  double  *rhs;                  /*!< right-hand side forcing function      */
  double **A;                    /*!< linear system matrix                  */
  double   h;                    /*!< mesh sizing                           */
  int      n;                    /*!< problem size                          */
  int      npts;                 /*!< number of points in single direction */
  int      pad;                  /*!< pad dimension for ghost points        */
} pstruct;
```

Example Results: What we're hoping for

2nd Order Central Finite-difference Scheme





- Example results for $npts = 5, 9, 17, 33, 65$, $length = 1.0$

MASA PDE Examples

Source Terms: Euler

```
// Gas state
ADScalar T  = P / RHO / R;
ADScalar E  = 1. / (Gamma-1.) * P / RHO;
ADScalar ET = E + .5 * U.dot(U);

// Mass, momentum and energy
Scalar   Q_rho   = raw_value(divergence(RHO*U));
RawArray Q_rho_u = raw_value(divergence(RHO*U.outerproduct(U)) +
                   P.derivatives());
 Scalar  Q_rho_e = raw_value(divergence((RHO*ET+P)*U));
```

check out tests/ad_euler.cpp

Thank you for your attention.

Let’s start coding!!!