# PHAML: A Parallel Adaptive Multilevel Program for Elliptic PDEs

William F. Mitchell

Mathematical and Computational Sciences Division
National Institute of Standards and Technology
Gaithersburg, MD

March 31, 2003
11th Copper Mountain Conference on Multigrid Methods

NIST National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

ITL INFORMATION TECHNOLOGY LABORATORY

# PHAML

- Parallel Hierarchical Adaptive Multi-Level
- Parallel sequel to MGGHAT
- 2D Elliptic PDE solver
- Fortran 90 module/library
- Adaptive finite elements with multigrid
- Message passing parallelism: MPI or PVM
- Tested on most UNIX systems

# Elliptic Boundary Value Problems

$$
\begin{aligned}
-\frac{\partial}{\partial x}p\frac{\partial u}{\partial x} - \frac{\partial}{\partial y}q\frac{\partial u}{\partial y} + ru &= f \quad \text{in } \Omega \subset \Re^2 \\
u &= g_1 \text{ on } \partial\Omega_1 \\
\frac{\partial u}{\partial n} + cu &= g_2 \text{ on } \partial\Omega_2
\end{aligned}
$$

$p, q, r, f, c, g_1$ and $g_2$ functions of $(x, y)$

$\partial\Omega_1 \cup \partial\Omega_2 = \partial\Omega$

$\Omega$ polygonal

Also: time dependent, nonlinear, systems

# Eigenvalue Problems

$$
\begin{aligned}
-\frac{\partial}{\partial x}p\frac{\partial u}{\partial x} - \frac{\partial}{\partial y}q\frac{\partial u}{\partial y} + ru &= \lambda u \ \text{ in } \ \Omega \subset \Re^2 \\
u &= 0 \quad \text{on } \partial\Omega_1 \\
\frac{\partial u}{\partial n} &= 0 \quad \text{on } \partial\Omega_2
\end{aligned}
$$

$p, q, r$ functions of $(x, y)$

$\partial\Omega_1 \cup \partial\Omega_2 = \partial\Omega$

$\Omega$ polygonal, generally a truncation of an infinite domain

# Numerical Methods

- Parallelization of methods in MGGHAT
- Standard finite elements, linear, triangles
- Newest node bisection adaptive refinement
- Hierarchical basis multigrid
- Refinement-tree based grid partitioning
- Full domain partition

# Optional Other Software

- BLAS and LAPACK
- MPI or PVM
- OpenGL, GLUT and F90GL
- PETSc
- MUMPS
- Zoltan, optionally including ParMETIS
- ARPACK
- More are likely to be added
  - Can provide a test bed for comparing programs/methods

# Software Architecture

- Fortran 90 module
  - symbolic constants, data type and operators on it
- Compiles to a library
- User writes main program and subroutines that define the equation
- Two parallel paradigms
  - master/slave
  - SPMD

# Primary Data Structure

- `type phaml_solution_type`
- Create, perform operations on it, and destroy
- Can have multiple objects
  - useful for time dependent, nonlinear, systems, etc.
- Contains all information about an object
  - grid and solution
  - processes information
    - consider the processes to be part of the object
    - spawned when created, terminated when destroyed
  - other state information and options

# User Written External Routines

- Routines to define the problem to be solved
  - pdecoef
  - bcond
  - icond
  - init_grid (defines the domain)
  - true
  - a few others that are seldom used

# Example `pdecoef`

```
subroutine pdecoef(x,y,p,q,r,f)

! pde is
! -( p(x,y)*u ) -( q(x,y)*u ) +r(x,y)*u = f(x,y)
!             x x           y y

real, intent(in) :: x(:), y(:)
real, intent(out), optional :: p(:),q(:),r(:),f(:)

if (present(p)) p = 1.0
if (present(q)) q = 1.0
if (present(r)) r = 0.0
if (present(f)) f = x**2 + y**2

end subroutine pdecoef
```

# Callable Routines

- Operations that PHAML can perform on an object
  - phaml_create, phaml_destroy
  - phaml_solve_pde
  - phaml_evaluate
  - phaml_query
  - phaml_scale
  - phaml_integrate
  - phaml_connect
  - phaml_store, phaml_restore
  - phaml_popen, phaml_pclose

# Optional Arguments

- Does not use a mysterious array to supply parameters
- Does not use a large workspace array that gets chopped up
- All options are specified in individual arguments
  - Nearly all arguments are optional
  - Missing arguments are given reasonable defaults
- Simplifies calling sequence
- Makes code more readable (keyword arguments)
- Improves upward compatibility

# Arguments for `paml_solve_pde`

```
subroutine solve_pde(paml_solution, iterm,                          &
          max_elem, max_node, max_lev, max_refsolveloop,            &
          init_form, comm_freq, partition_size, eq_type,            &
          print_grid_when, print_grid_who, print_error_when,        &
          print_error_who, print_time_when, print_time_who,         &
          print_eval_when, print_eval_who,                          &
          print_header_who, print_trailer_who, clocks,              &
          draw_grid_when, draw_reftree_when, pause_after_draw,      &
          pause_after_phases, pause_at_start, pause_at_end,         &
          uniform, overlap, sequential_node, inc_factor,            &
          error_estimator, refterm, derefine,                       &
          partition_method, predictive,                             &
          solver, preconditioner, mg_cycles, mg_prerelax,           &
          mg_postrelax, iterations, ignore_quad_err,                &
          final_solves, final_mg_cycles,                            &
          num_eval, lambda0)
```

# Example `main` Program

```
program user_main_example
use phaml
type(phaml_solution_type) :: pde
call create(pde, draw_grid_who = MASTER)
call solve_pde(pde,                                    &
               max_node = 20000,                       &
               draw_grid_when = PHASES,                &
               partition_method = ZOLTAN_RCB,          &
               mg_cycles = 2)
call destroy(pde)
end program
```

# Conclusion

- PHAML is a new elliptic PDE solver
- Parallel sequel to MGGHAT
- Adaptive refinement, multigrid, message passing
- Hooks into several other software packages
- Tested on several unixes, compilers
- Now available
  - http://math.nist.gov/phaml
  - william.mitchell@nist.gov