Parallel PDE Solvers in Python

Bill Spotz
Sandia National Laboratories

Scientific Python 2006
August 18, 2006
Computational Sciences at Sandia

- Chemically reacting flows
- Climate modeling
- Combustion
- Compressible flows
- Computational biology
- Electrical modeling
- Heat transfer
- Load balancing

- Materials modeling
- MEMS modeling
- Mesh generation
- Optimization and uncertainty quantification
- Seismic imaging
- Shock and multiphysics
- Structural dynamics
The Trilinos Project

- Provide a central repository for Sandia’s solver technology
- Increase code-reuse
- Organized on concept of “packages”
- Minimize package interdependence
- Maximize package interoperability
- Provide a framework for SQE and SQA
  - Compliance with requirements
  - Nightly test harness
- High degree of developer autonomy
- Open source: GNU Lesser License
- Web site: http://software.sandia.gov/trilinos
- Next release: Version 7.0, September, 2006
- Trilinos Users Group Meeting, November 7-9, 2006
The Interoperability Problem

- ~20 “core” packages → ~400 interface package boundaries
- The purpose of the Thyra package is to provide tools and definitions for a common interface
  - Packages that code to the interface should be able to interact with each other
  - Important, but relatively new effort within Trilinos (replacing TSF):
    - Thyra
    - RTOp
    - Stratimikos
    - Rythmos
    - ...
  - SciDAC TOPS proposal: universal operability
  - Actively pursuing funding for python implementation of Thyra (some prototyping done)
PyTrilinos

• **Linear Algebra Services**
  – **Epetra** (with extensive NumPy compatibility and integration)
  – **EpetraExt** (coloring algorithms and some I/O)

• **Linear Solvers**
  – **Amesos** (LAPACK, KLU, UMFPACK, ScaLAPACK, SuperLU, SuperLUDist, DSCPACK, MUMPS)
  – **AztecOO**

• **Preconditioners**
  – **IFPACK**
  – **ML**

• **Nonlinear Solvers**
  – **NOX** (python wrappers not yet caught up to recent redesigns)

• **Meta-Solvers**
  – **LOCA** (python wrappers not yet caught up to recent redesigns)
  – **Anasazi** (early development stage)

• **Tools and Utilities**
  – **Teuchos** (ParameterList class only)
  – **TriUtils**
  – **Galeri**
  – **Thyra** (early development stage)
  – **New_Package**
Trilinos documentation is handled by doxygen: special comments within code
- Web pages updated twice daily

Python wrappers are generated using swig ... doxygen does not work with swig interface files
- %feature("autodoc", "1");

>>> help(Epetra.Vector.Dot)
    Dot(*args) unbound PyTrilinos.Epetra.Vector method
    Dot(self, Epetra_Vector A) -> double

Currently working to provide much more extensive documentation highlighting differences between C++ and python interfaces
- Release 7.0 in September
PyTrilinos.Epetra

• Communicators
  – Comm
  – SerialComm
  – MpiComm
  – PyComm

• Maps
  – BlockMap
  – Map
  – LocalMap

• Vectors
  – MultiVector
  – Vector
  – IntVector

• SerialDense objects
  – SerialDenseOperator
  – SerialDenseMatrix
  – SerialDenseVector
  – SerialDenseSolver
  – IntSerialDenseMatrix
  – IntSerialDenseVector

• Graphs
  – CrsGraph

• Operators
  – Operator
  – RowMatrix
  – CrsMatrix
PyTrilinos.Epetra and NumPy

- Array-like classes inherit from `numpy.UserArray`
  - MultiVector
  - Vector
  - IntVector
  - SerialDenseMatrix
  - SerialDenseVector
  - IntSerialDenseMatrix
  - IntSerialDenseVector

- Methods throughout Epetra have arguments that accept or produce pointers to C arrays
  - Python input arguments accept python sequences
  - Python output arguments produce ndarrays
PyTrilinos.Teuchos

- Teuchos::ParameterList
  - Used by several Trilinos packages to set problem parameters
  - Maps string names to arbitrary-type values
  - Python implementation allows dictionary substitutions
  - Hybrid PyDictParameterList objects are returned
  - The following conversions are supported:

<table>
<thead>
<tr>
<th>Python</th>
<th>Dir</th>
<th>C / C++</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>⇔</td>
<td>bool</td>
</tr>
<tr>
<td>int</td>
<td>⇔</td>
<td>int</td>
</tr>
<tr>
<td>float</td>
<td>⇔</td>
<td>double</td>
</tr>
<tr>
<td>string</td>
<td>⇔</td>
<td>std::::string</td>
</tr>
<tr>
<td>string</td>
<td>⇔</td>
<td>char *</td>
</tr>
<tr>
<td>dict</td>
<td>⇒</td>
<td>ParameterList</td>
</tr>
<tr>
<td>wrapped ParameterList</td>
<td>⇔</td>
<td>ParameterList</td>
</tr>
<tr>
<td>wrapped PyDictParameterList</td>
<td>⇒</td>
<td>ParameterList</td>
</tr>
</tbody>
</table>
PyTrilinos Demonstration

- **Governing equation:** \(- \frac{d^2 u}{dx^2} + c \frac{du}{dx} = 0, \ x \in [0,1]\)

- **Boundary conditions:** \(u(0) = 0, \ u(1) = 1\)

- **Exact solution:** \(u(x) = \frac{e^{cx} - 1}{e^c - 1}\)

- **CDS:** \(- \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + c \frac{u_{i+1} - u_{i-1}}{2h} = 0\)

- **Oscillations:** \(ch = \frac{c}{n - 1} > 2\)