Course Announcement: MTH 654
Large Scale Scientific Computing Methods

http://www.math.oregonstate.edu/~mpesz/teaching/65

Course content: theory
- theory and implementation details for solving large linear and nonlinear systems of equations
- Newton-Krylov methods, multigrid and domain decomposition

Course content: weekly lab
- introduction to parallel computing
- how to function in a high performance computing environment
- a module on multicore architectures and on programming GPUs for using NVIDIA CUDA programming environment

Students: the class is designed for motivated graduate students and well prepared undergraduates.

Contact me with questions – also on scheduling - mpesz@math.oregonstate.edu

INSTRUCTOR: MAŁGORZATA PESZYŃSKA, MATHEMATICS DEPARTMENT
Class MTH 655/659 information

• Attendance in labs required:
  – Fridays **(8:30-)9:00-10:00-** in MLC Kidder 108 computer lab
  – (start 8:30-can leave at 10:00)
  – must complete each lab project

• Individual project: paper and (optional) presentation in March

• Fill out questionnaire
  – must have OSU ID and ONID username

• NO CLASS this Wednesday

• Reading/review:
  • see http://www.math.oregonstate.edu/~mpesz/teaching/654_F09/
Class MTH 655/659 information

- **Algorithms and theory**
  - nonlinear problems: Newton-based for $F(U)=0$
  - linear solvers: Jacobi family, Krylov family (CG, PCG, GMRES)
  - parallel implementation theory
  - domain decomposition
  - multigrid
  - Additional topics as time permits:
    - primer on optimization (nonlinear, continuous, unconstrained)
    - Non-numerical algorithms and their parallel implementation

- **Implementation**
  - MATLAB prototypes for testing properties of algorithms and applications
  - Fortran (C for geeks) for REAL scientific computing
    - overview of Unix will be given
  - Fortran+MPI for parallel implementation on a cluster
  - Module on programming GPUs
Solution of nonlinear equations $F(U) = 0$

- Ex.: find $x$:
  
  $$e^{-x} = x^2$$

- Solution:
  
  $$x \approx 0.7$$

- Set-up
  
  $$f(x) = e^x x^2 - 1$$

- Solve
  
  $$f(x) = 0$$
Find $x \in (a, b)$: $f(x) = 0$
Use Newton’s method

- given an initial guess $x^{(0)}$, iterate ... $x^{(1)}, x^{(2)}, x^{(3)}, ...$
Newton: step 1
also known as method of tangents

- the next guess (iterate) is found by
  \[ x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} \]
Newton: step2

\[ x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} \]
Newton: step3

\[ x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} \]
Newton: steps 4, 5

- until convergence
  - residual is small \( \left| f(x^{(k)}) \right| \leq \tau \)
  - subsequent iterates do not differ much \( \left| x^{(k)} - x^{(k-1)} \right| \leq \beta \)
Properties of Newton’s method

• Iteration

\[ x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})} \]

written as

\[ \begin{cases} \partial^{(k+1)} = (f'(x^{(k)}))^{-1} f(x^{(k)}) \\ x^{(k+1)} = x^{(k)} - \partial^{(k+1)} \end{cases} \]

• Convergence: what conditions?
  – local convergence: for what initial guess?
    • conditions?
  – global: (for any initial guess) how?
    • line search, trust regions, and other

• Use in optimization

\[ \min_x J(x) \]

\[ f(x) = J'(x) \]
Newton’s method in N-dimensions

• 1D variant for $f(x)=0$

$$f : \mathbb{R} \rightarrow \mathbb{R}, \, x \in \mathbb{R}$$

$$\begin{cases} 
\hat{\partial}^{(k+1)} = (f'(x^{(k)}))^{-1} f(x^{(k)}) \\
N \quad \text{ } x^{(k+1)} = x^{(k)} - \hat{\partial}^{(k+1)}
\end{cases}$$

• N-D Variant for $F(U) = 0$

$$F : \mathbb{R}^{Nx1} \rightarrow \mathbb{R}^{Nx1}, \, U \in \mathbb{R}^{Nx1}$$

$$\begin{cases} 
\partial^{(k+1)} = (DF(U^{(k)}))^{-1} F(U^{(k)}) \\
N \quad \text{ } U^{(k+1)} = U^{(k)} - \partial^{(k+1)}
\end{cases}$$

$$DF \in \mathbb{R}^{NxN}$$
Newton’s method efficiency and scaling

- Solve an N-dimensional problem \( F(U) = 0 \)

\[
F : R^{Nx1} \mapsto R^{Nx1}, U \in R^{Nx1}
\]

- using Newton’s method:

\[
\begin{align*}
\text{find} & \quad F(U^{(k)}), DF(U^{(k)}) \\
\text{solve} & \quad \delta^{(k+1)} = (DF(U^{(k)}))^{-1} F(U^{(k)}) \\
\text{update} & \quad U^{(k+1)} = U^{(k)} - \delta^{(k+1)}
\end{align*}
\]
How to solve most accurately/efficiently

- a linear problem?
  \[ AU = b \quad O(N^3) \]

- a nonlinear problem?
  \[ F(U) = 0 \quad O(\#\text{iters} \times N^3) \]

The answer depends …
- on the underlying application
- on properties of A, F
Linear solvers: how to solve $AU = b$

- Problem: solve

$$AU = b$$

$$A \in \mathbb{R}^{N \times N}, \quad U \in \mathbb{R}^{N \times 1}, \quad b \in \mathbb{R}^{N \times 1}$$

- How large is $N$?
  - does $A$ fit in computer memory? (8 bytes x NxN = ?)
  - is $A$ full (dense) / sparse?
  - how does the speed of the method (number of FLOPs) scale with $N$? What is the exponent in $O(N^\alpha)$

- Two main classes of methods
  - direct
  - iterative
<table>
<thead>
<tr>
<th>Linear solvers:</th>
<th>direct</th>
<th>versus</th>
<th>iterative</th>
</tr>
</thead>
</table>

- **Ex.: Gauss-Jordan elimination (or QR decomposition)**

\[
\begin{bmatrix}
* & 0 & 0 \\
* & * & * \\
* & * & 0 \\
* & * & *
\end{bmatrix}
\]

- requires storage \(O(N^2)\)
- scales \(O(N^3)\)
- do not preserve sparsity
- special variants ILU, ICCL
  - band direct solvers exist \(O(N^2)\)

- IDEA:

\[
A U = b
\]

iterate \(U^{(k)} \mapsto U^{(k+1)}\)

\(k = 1,2,\ldots\) until convergence

requires only product \(Y = AV\)

- no storage necessary
- stationary methods: \(O(N^2 \log N)\)
  - Jacobi, G-S, SOR
- non-stationary methods \(O(N^{1.17})\)
  - Krylov family:
    - CG, PCG, GMRES
- multigrid \(O(N^1)\)

All scaling information for 3D linear PDE models, optimal parameters [Heath’97]
Motivation: solving large systems of nonlinear PDEs

- PDEs = partial differential equations
- PDEs are mathematical models of
  - continuum mechanics
  - fluid flow in subsurface and surface waters
  - gas dynamics
  - heat conduction
  - transport of contaminants
  - and more ....
- Let us call a generic system of (coupled nonlinear) PDEs

\[ F(U) = 0 \]
Steps of solving large systems of nonlinear PDEs

• Coupled nonlinear PDEs
  • PDEs imposed over a region D in space and time interval (0,T)
  • boundary conditions on boundary of D
  • initial conditions at t=0

• Numerical discretization of DEs/PDEs
  – discretize in space: grid over D
    • finite differences, elements, volumes
  – discretize in time, use time step \( \Delta t \)
    • finite differences
  – ANALYSIS of schemes: MTH 552, 553, 654, 655 (FE)
    • Error \( U - U_h \)
    • for accuracy we must have MANY grid points in D, small \( \Delta t \)

• Solve the system \( F_h(U_h) = 0 \) as fast as possible
  • solving general linear systems: MTH 551
Example: linear PDE on a simple domain

- model

\[ F(U) = 0 \]

is Poisson equation

\[ - \Delta U = b \]

- discretized model

\[ - \Delta_h U_h = b_h \]
Discrete (linear) model

- linear discrete model

\[ F_h(U_h) = 0 \]

or

\[ -\Delta_h U_h = b_h \quad \text{or} \quad A U = b \]

\[ U_h \equiv \begin{bmatrix} u_{1,1} \\ u_{2,1} \\ \vdots \\ u_{9,7} \\ u_{9,8} \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_{71} \\ U_{72} \end{bmatrix} = U \]
Structure of $\Delta_h$

$u_{3,4} \equiv U_{30}$

$U_h = \begin{bmatrix} u_{1,1} & u_{2,1} & \cdots & u_{9,7} & u_{9,8} \\ \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_{71} \\ U_{72} \end{bmatrix}$
Details on the stencil in $\Delta_h$
Matrix of the system $AU = b$
How to best solve $AU = b$

• **Exploit**
  - sparsity of $A$
  - band structure of $A$

• **Exploit positive definiteness of $A$**

• **Exploit its origin**
  - PDE

• **Modern methods:**
  - multigrid
  - domain decomposition
  - parallel algorithms
Divide D and conquer \( AU = b \)

- Iterative solver: must communicate data between yellow and blue zones
- Domain decomposition
  - overlapping or nonoverlapping
- Matrix/vector view: similar to block decomposition
- Implement on single processor or multiprocessor computer
Domain decomposition: overlapping

- blue and yellow values lag between iterations
- similar to block Jacobi solver
Domain decomposition: non-overlapping

- external iteration
- solve for interface degrees of freedom (external iteration)

- form the Schur complement (eliminate interface unknowns)
Solve system $AU = b$ on a multiprocessor computer system

- Need to communicate data between processors
  - distributed memory or shared memory?
Parallel solution: distributed memory multiprocessor systems

• Issues: computation time versus communication time

• Implementation: MPI (Message Passing Interface)
Parallel solution: shared memory

- Processors communicate with one global (shared) memory: bus contention and latency
- Expensive not always scalable solutions
Example: nonlinear PDE on a complicated domain

- given a PDE model
  \[ F(U) = 0, \; x \in D \]
- discretize \( D \)
- define discrete model
  \[ F_h(U_h) = 0 \]
Example of \( F(U) = 0 \) : multi-phase / multi-component flow

### Phase: m, component: M

- **Mass conservation**
  \[
  \frac{\partial (\phi N_M)}{\partial t} + \nabla \cdot U_M = q_M
  \]

- **Volume constraints**
  \[
  \sum_m S_m = 1 \quad \sum_M n_{mM} = 1
  \]

- **Def.: Mass concentration**
  \[
  N_M = \frac{1}{\rho_m^*} \sum_m S_m \rho_m n_{mM}
  \]

- **Def.: Mass flux**
  \[
  U_M = \frac{1}{\rho_m^*} \sum_m \rho_m n_{mM} V_m
  \]

- **Def.: Phase velocity**
  \[
  V_m = -K \frac{k_m}{\mu_m} (\nabla P_m - \rho_m G \nabla D)
  \]

- **Def.: Capillary pressure relation**
  \[
  P_{m_1} - P_{m_2} = P_{m_1,m_2}^c (S_{m_1}, \ldots)
  \]

- **Constitutive eqs.**
  \[
  \rho_m = \rho_m(P_m, n_{mM})
  \]

**Specific model**
Example: simulate oil and gas recovery

- oil and gas displaced by water contained in D: region in which there is oil and gas and water (brine)

\[ F(U) = 0 \]

- must discretize D (decide which scheme to use)

\[ F_h(U_h) = 0 \]

- must solve the system for \( U_h \)

- must post-process the results
  - assess accuracy
  - visualize what is going on
My example: oil and gas recovery

• original domain

• decomposition into 20 processors

• results

• how accurate is this solution?
  – that is another story
  – take MTH 55& and/or 65*?
Summary of choices: algorithm/implementation

- computing platform
  - serial / workstation
  - parallel / distributed memory: MPI
  - parallel / shared memory OpenMP
  - parallel supercomputer (PetaFLOPS): a hybrid?

- discretization method
  - finite differences
  - finite elements

- nonlinear solver
  - Newton-based:
    » local convergence
    » global convergence

- linear solver
  - full or sparse matrix?
    - direct
    - iterative

\[
\begin{align*}
F(U) &= 0 \\
AU &= b
\end{align*}
\]
### Choices: programming environment

<table>
<thead>
<tr>
<th>Interpretative environment</th>
<th>Compiled environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>• ex: MATLAB</td>
<td>• ex.: FORTRAN, C, C++</td>
</tr>
<tr>
<td>• quick development</td>
<td>• requires post-processing (graphics output to files)</td>
</tr>
<tr>
<td>• workstation</td>
<td>• can reuse “dusty shelves” (legacy code)</td>
</tr>
<tr>
<td>• graphics available</td>
<td>• can use highly optimized libraries</td>
</tr>
<tr>
<td>• may not scale</td>
<td>• computational kernels as efficient as computer-ly possible</td>
</tr>
<tr>
<td>• not efficient</td>
<td>• parallel (MPI, OpenMP)</td>
</tr>
<tr>
<td>• not parallel</td>
<td>• can be made portable between supercomputing platforms</td>
</tr>
<tr>
<td>• portable</td>
<td></td>
</tr>
<tr>
<td>• Windows, Unix, MAC</td>
<td></td>
</tr>
<tr>
<td>• but not to supercomputing platforms</td>
<td></td>
</tr>
</tbody>
</table>

In this class we will do both types of implementation
Class MTH 655/659 information

• Algorithms and theory
  • nonlinear problems: Newton-based for F(U)=0
  • linear solvers: Jacobi family, Krylov family (CG,PCG,GMRES)
  • parallel implementation theory
  • domain decomposition
  • multigrid
  • primer on optimization (nonlinear, continuous, unconstrained)

• Implementation
  • MATLAB prototypes for testing properties of algorithms and applications
  • Fortran (C for geeks) for REAL scientific computing
    – overview of Unix will be given
  • Fortran+MPI for parallel implementation on a cluster
  • Module on use of GPUs for scientific computing

• Current information
  • http://www.math.oregonstate.edu/~mpesz/teaching/654_F09
Class MTH 655/659 information

• Attendance in labs required:
  – Fridays (8:30-)9:00-10:00-(10:30) in MLC
  – (start 8:30-can leave at 10:30)
  – must complete each lab project

• Individual project: paper and (optional) presentation in Nov./Dec.

• Fill out questionnaire
  – must have OSU ID and ONID username

• NO CLASS this Wednesday
  – some other no-class dates TBA

• Reading/review:
  • http://www.math.oregonstate.edu/~mpesz/teaching/654_F09/