Numerical Methods for (Astro)Physics
Course Objectives

- To understand the considerations that go into designing a numerical algorithm
- To understand the limitations of numerical methods
- To gain exposure into some of the common methods used in astrophysics
- To share our own personal experiences with computation
- To understand when we should write our own code vs. use an existing library.
Class Business

- Does everyone have access to a Linux or Unix machine to work from?

- What programming languages do each of you typically use?
  - Past students have used:
    - Fortran
    - C
    - C++
    - Matlab (but please make sure it works in octave)
    - python
“Don't worry head, the computer will do our thinking now”
—Homer Simpson
Question...

- When you see a numerical result presented in the literature, we want to be able to answer the following:
  - What assumptions were made about the physics?
  - What approximations were made to the system?
  - How well tested in the simulation code?
Course Texts

- *Computational Physics*, by Mark Newman
  - Generally good coverage on most of the topics we’ll discuss
  - Lots of physics examples
  - Inexpensive
  - Main recommended book
Course Texts

- **An Introduction to Computational Physics**, by Tao Pang
  - Also good coverage of the topics (up to PDEs)
  - Lots of physics examples
  - Inexpensive
Course Texts

- *Numerical Methods for Physics* by Alejandro Garcia
  - Broad coverage
  - More PDE stuff than Pang, but we'll still do things differently

- Others
  - There are LOTS of other texts on computation for physics. Pick a reasonably priced one and it will be a good basis.
Course Texts

  - Focuses on methods used for simulation, not the core, underlying basics
  - Includes code, but not very well documented
  - Best serves as a review to the literature
Course Texts

- **Numerical Recipes** by Press et al.
  - Classic introduction to a host of methods
  - Good source of references
  - Code is not open source
    - If you use these routines in your projects, you are not allowed to share your project with anyone else!
  - Best thought of as a way to learn about the core ideas in the methods, and then follow the references to get more.
  - Many critics argue that the methods are old and have not kept up with new developments in numerical analysis (see for example: http://www.uwyo.edu/buerkle/misc/wnotnr.html)
  - Many open source libraries exist that offer similar routines
  - Well tested code from libraries is often better.
Course Texts

- *Effective Computation in Physics* by Scopatz & Huff
  - Introduces UNIX shell and python
  - Covers good programming practices
    - testing, version control, debugging, documentation
  - Introduces some data structures
  - Introduces parallel programming
Course Texts

• For the discussion on PDEs, we’ll focus on finite-volume methods and use my notes:
  
  - https://github.com/Open-Astrophysics-Bookshelf/numerical_exercises
Other Resources

- Links to papers, web resources/tutorials, and notes are provided on the course webpage:
  - http://bender.astro.sunysb.edu/classes/numerical_methods/
- Wikipedia also provides some good links and basic info.
- You!
  - You bring your own experiences with you.
  - Share your experiences in class.
General Caveats

• There are a wide range of numerical methods
  – For each class of problem, there are many different algorithms to choose from, each with their own strengths and weaknesses
  – Some algorithms are preferred in some fields and relatively unknown in others

• Our goal here is to focus on the basic ideas, not cover every possible method for each problem.
  – We want to learn some of the general “got-yas” of the various methods.

• The best way to learn how these things work is to code up simple versions yourself.
  – After that, it is often best to use a well-tested library routine if it fits your needs.
Class Work

- There will be several assignments (usually involving writing some short code)
  - 1 – 2 problems per major topic
  - You can write in any language you wish.
  - If it's not obvious, provide a short README on how to use your code.
  - Come to me for help—the point of all of this is to learn the basics of numerical methods

- The last homework will be more of a project—you will choose some interesting algorithm and code it up / use it for an interesting problem, and share it with the class

- Grades: 100% homework
<table>
<thead>
<tr>
<th>class meeting</th>
<th>topic</th>
<th>Newman Ch.</th>
<th>Pang Ch.</th>
<th>Garcia Ch.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>overview / basics of computation</td>
<td>–</td>
<td>1</td>
<td>§1.5</td>
</tr>
<tr>
<td>2</td>
<td>good programming practices</td>
<td>2</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>3–4</td>
<td>differentiation / integration / order-of-accuracy</td>
<td>5</td>
<td>3</td>
<td>§10.2</td>
</tr>
<tr>
<td>5</td>
<td>interpolation / root finding</td>
<td>–</td>
<td>2, 3</td>
<td>–</td>
</tr>
<tr>
<td>6–8</td>
<td>ODEs</td>
<td>8</td>
<td>4</td>
<td>2, 3</td>
</tr>
<tr>
<td>9</td>
<td>linear algebra</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>FFTs</td>
<td>7</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>fitting</td>
<td>–</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>12–14</td>
<td>advection / hyperbolic PDEs</td>
<td>9</td>
<td>–</td>
<td>7, 9</td>
</tr>
<tr>
<td>15–16</td>
<td>Poisson equation / elliptic PDEs</td>
<td>9</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>17–18</td>
<td>diffusion / parabolic PDEs</td>
<td>–</td>
<td>7</td>
<td>6, 9</td>
</tr>
<tr>
<td>19–20</td>
<td>Monte Carlo</td>
<td>10</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>21</td>
<td>genetic algorithms</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>22</td>
<td>optimization</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>23–24</td>
<td>parallel computing</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>25–26</td>
<td>machine learning</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>27–28</td>
<td>computational fluid dynamics</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
Much like the telescope is an observer's tool, supercomputers allow us to perform experiments on a model system.
Virtual experimentation allows us to

- Test new ideas / perform sensitivity studies to refine our understanding
- Understand processes otherwise hidden from view
The complexity for minimum component costs has increased at a rate of roughly a factor of two per year... Certainly over the short term this rate can be expected to continue, if not increase.

—Gordon Moore (1965)

We are still in a period of exponential growth.

Today's supercomputers are capable of tens of 1000s of trillions of floating point operations per second (PFLOP/S).
The fastest computers keep getting faster.
Computation on the Cosmological Scale

- Small inhomogeneities in the early Universe seed structure
- More than 10 billion particles
- Self-gravity dominates the evolution

Simulating the growth of structure and the formation of galaxies. (Springel et al. 2005)
The merger of the Milky Way and Andromeda

What happens to the Sun?
Computation on the Galactic Scale

An exceptionally detailed simulation of the formation of the first stars in the universe.

Visualization: Kähler (ZIB), Cox, Patterson, Levy (NCSA), Simulations (Tom Abel, Greg Bryan, Mike Norman)

Formation of Micro-Galaxy

The First Star within it

© Discovery Channel

The First Supernova
Is it a single white dwarf or merging white dwarfs?—model both and see which looks more like nature.

(Roepke and Hillebrandt 2005)
Giant Impact hypothesis for the formation of the Moon
(Alastair Cameron)