PHY407: Computational Physics
Fall, 2017

Lecture 12: Recap and next steps

FILL OUT THE ONLINE EVALUATIONS

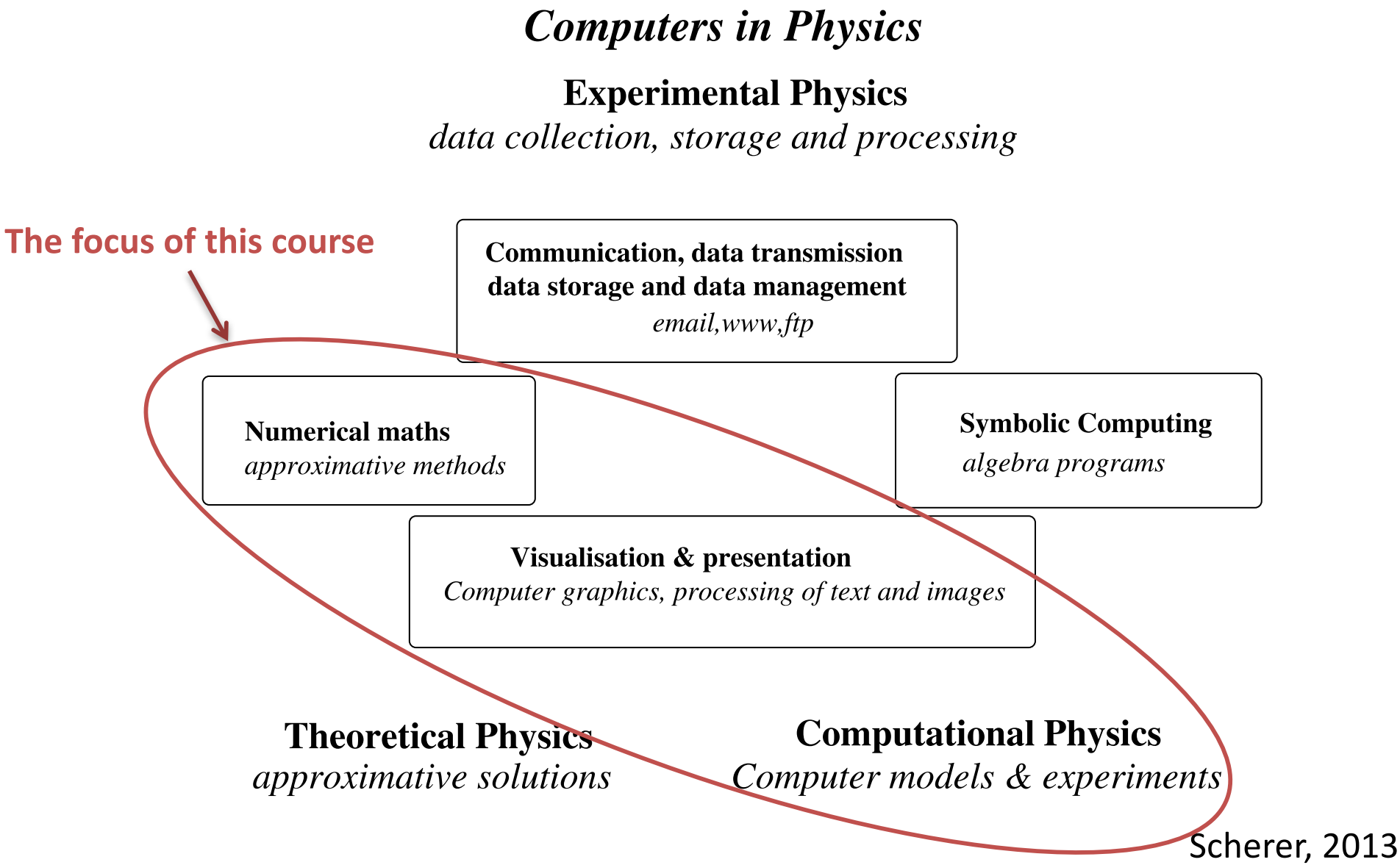
Summary & Status

- ☑ Weeks 1-3: Programming basics, numerical errors, numerical integration and differntiation.
- ☑ Weeks 4-5: Solving linear & nonlinear systems and Fourier transforms.
- ☑ Week 6: ODEs Part 1: RK4, Leapfrog, Verlet, adaptive time stepping; customizing python output
- ☑ Week 7: ODEs Part 2: Bulirsch-Stoer, Boundary Value Problems/shooting,
- ☑ Weeks 8-9: PDEs Part 1: Elliptic equation solvers, leapfrog time stepping, FTCS, Crank-Nicholson, Spectral Methods
- ☑ Week 10: Stochastic methods, Part 1: random numbers, monte carlo integration
- ☑ Week 11: Stochastic methods, Part 2: statistical mechanics ideas, simulated annealing approach to optimization.
- ❑ Week 12: recap, discussion, extensions.

PHY407: Computational Physics
Fall, 2017

Lecture 12: Recap and next steps

Think of all the ways we use computers in Physics …



Scherer, 2013

Computational physics applications are more advanced …

Data from lab and field obs

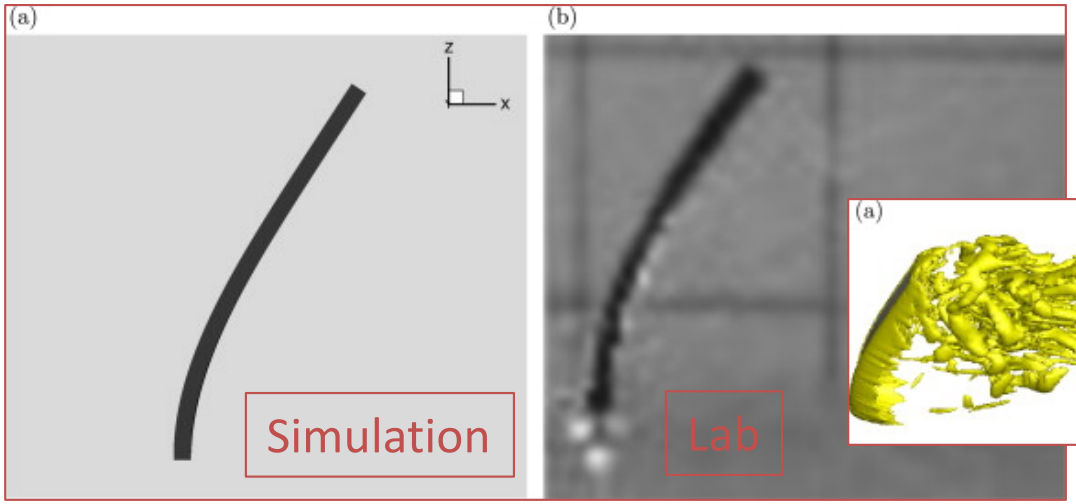
Mathematical model
(Empirical/dynamical)

$$\rho_s \frac{d^2 u_i}{dt^2} + \eta_d \frac{du_i}{dt} = \frac{\partial \sigma_{ij}}{\partial x_j} + \rho_s b_i,$$

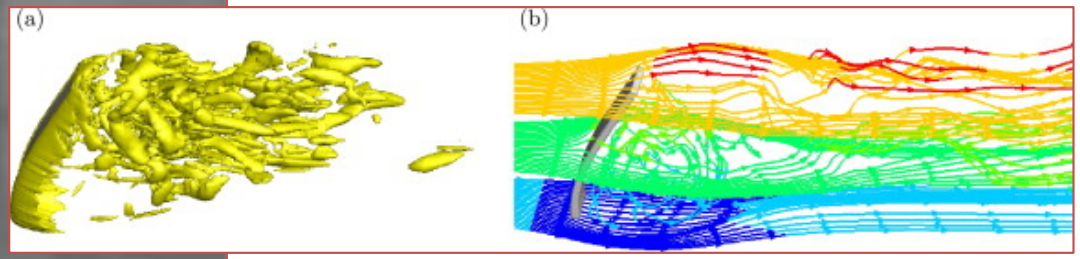
Numerical method
Algorithm
Code/Program

$$\left[K_T + \frac{\gamma}{\beta \Delta t} C + \frac{1}{\beta \Delta t^2} M\right] \{\Delta u^{n+1,s+1}\} = \{P^{n+1}\} - \{F^{n+1,s}\} - [M]\left\{\ddot{u}_n + \frac{1}{\beta \Delta t^2}(u^{n+1,s} - \tilde{u})\right\} - [C]\left\{\tilde{v} + \frac{\gamma}{\beta \Delta t}(u^{n+1,s} - \tilde{u})\right\},$$

Computation
Simulation
Analysis



In-depth analysis of simulation



… but we have learned lessons that stand the test of time.

Take-Home Lessons

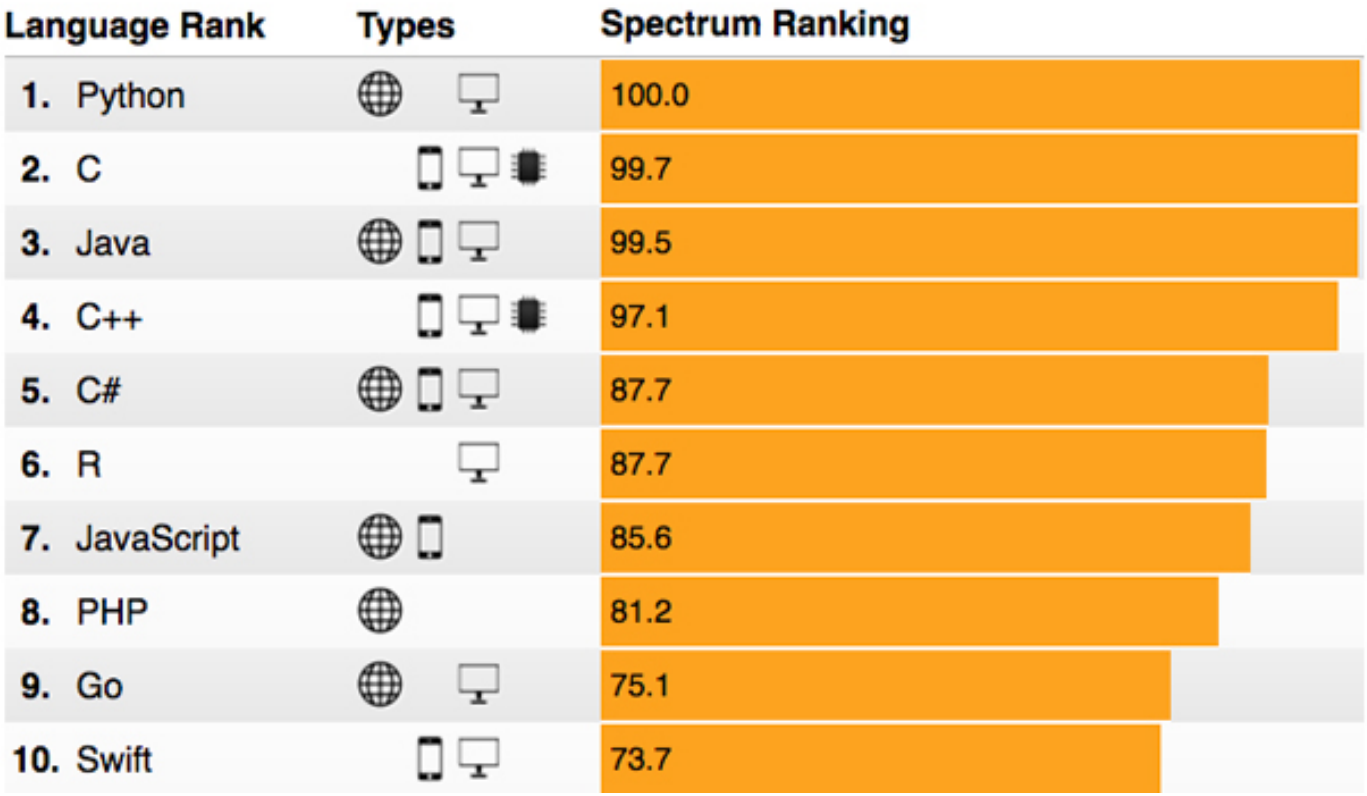
- We have to invest at least some thought into program design. Without pseudocoding we are dead in the water.
- We have to account for many different sources of error and their relationships: algorithmic, numerical roundoff, instability.
- We need to be quantitative: we need to estimate accuracy, rates of convergence, performance.
- We need to take advantage of smart programming approaches:
  - E.g. grouping calculations in smart ways (FFTs, Bulirsch-Stoer)
  - Investing time into extrapolation to get good estimates of the errors taken in the next step (adaptive time stepping)
- We should take advantage of canned routines, but we need to understand how to measure their performance (speed, accuracy, rate of convergence).

So what's next?

- Look for more advanced algorithms (more accurate, more efficient): parallel computing, etc.
- Look at other computer languages.

IEEE Spectrum Top 10 Languages 2017 (changes each year)

A lot of applications still use Fortran! (But it's way down on the list.)



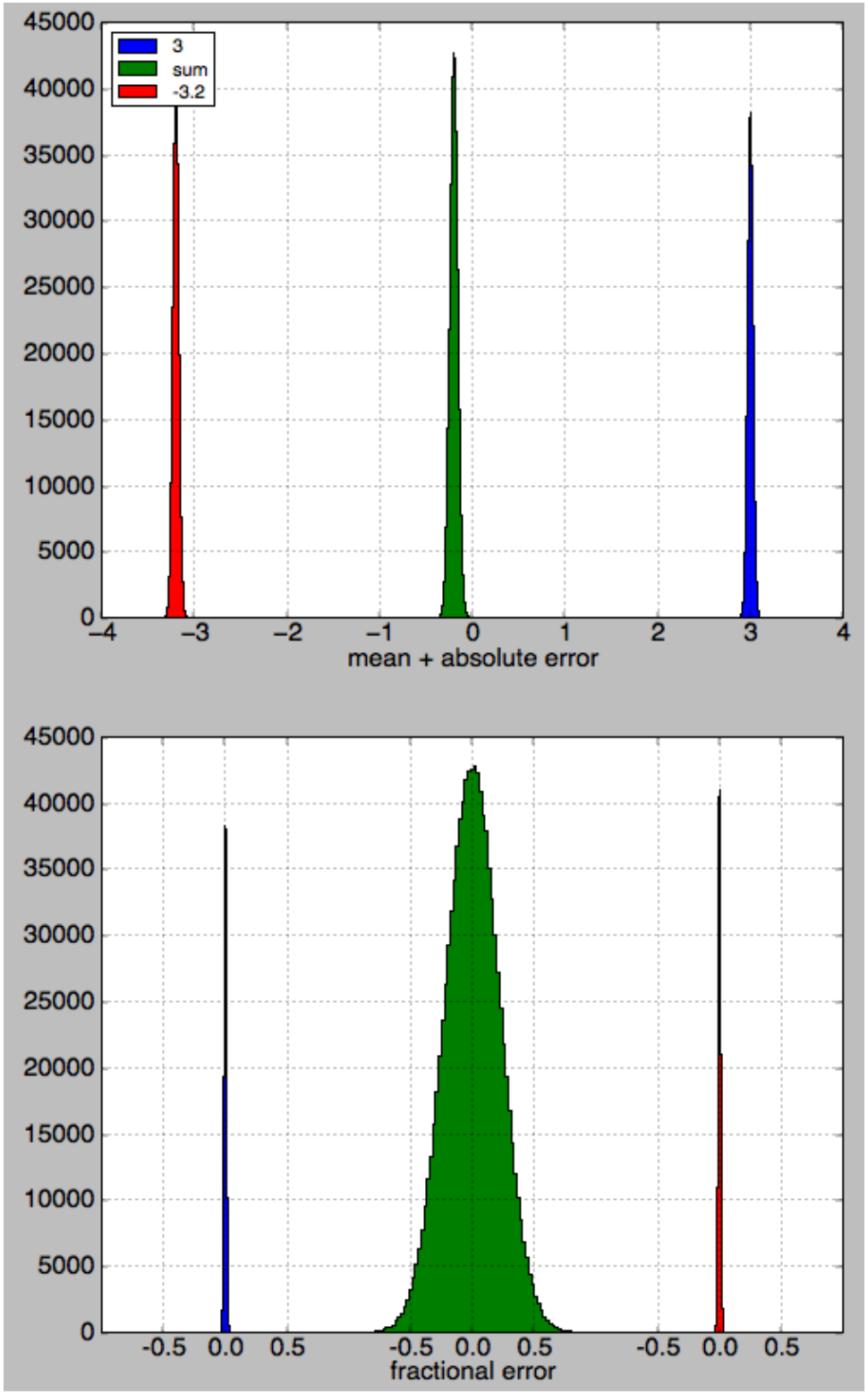
Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C		99.7
3. Java		99.5
4. C++		97.1
5. C#		87.7
6. R		87.7
7. JavaScript		85.6
8. PHP		81.2
9. Go		75.1
10. Swift		73.7

http://spectrum.ieee.org/computing/software/the-2017-top-programming-languages

Roundoff Error

We discussed how small roundoff errors could lead to large fractional errors when subtracting large numbers.

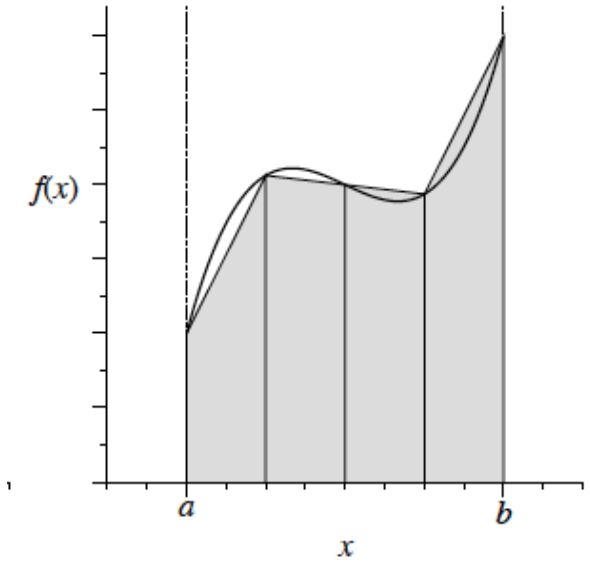
This makes differentiation less accurate and makes large sums hard to compute.

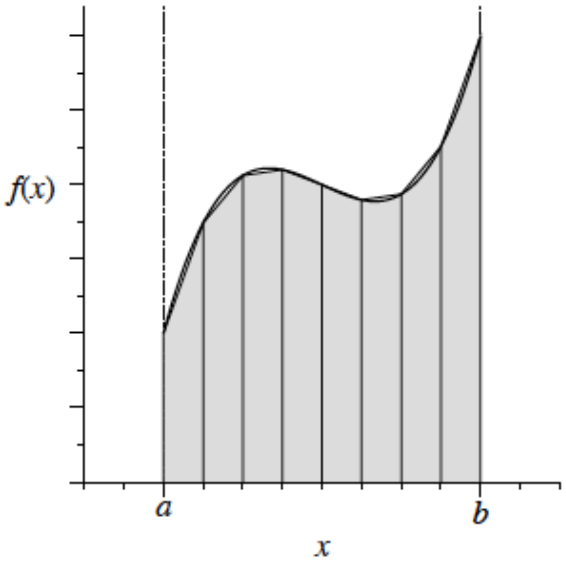


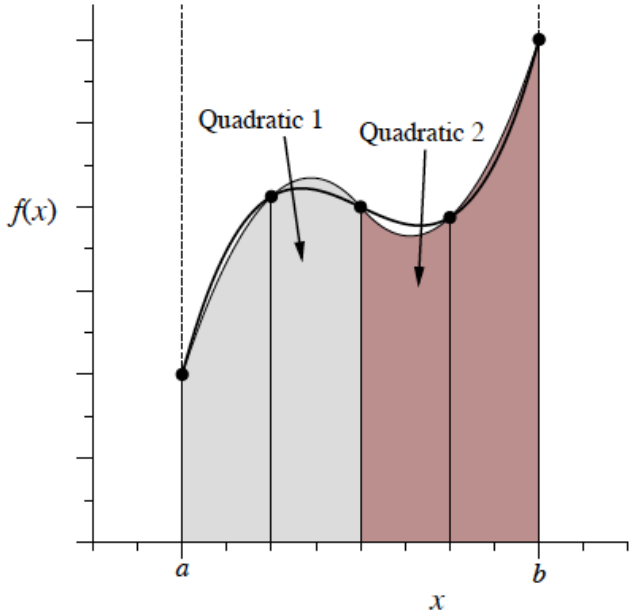
Solving Integrals with Newton-Cotes formulas

These are suited for equally spaced sampling.

Degree	Polynomial	Coefficients
1 (trapezoidal rule)	Straight line	$\frac{1}{2}, 1, 1, \ldots, 1, \frac{1}{2}$
2 (Simpson's rule)	Quadratic	$\frac{1}{3}, \frac{4}{3}, \frac{2}{3}, \frac{4}{3}, \ldots, \frac{4}{3}, \frac{1}{3}$
3	Cubic	$\frac{3}{8}, \frac{9}{8}, \frac{9}{8}, \frac{3}{4}, \frac{9}{8}, \frac{9}{8}, \frac{3}{4}, \ldots, \frac{9}{8}, \frac{3}{8}$
4	Quartic	$\frac{14}{45}, \frac{64}{45}, \frac{8}{15}, \frac{64}{45}, \frac{28}{45}, \frac{64}{45}, \frac{8}{15}, \frac{64}{45}, \ldots, \frac{64}{45}, \frac{14}{45}$







Solving Integrals with Gaussian Quadrature

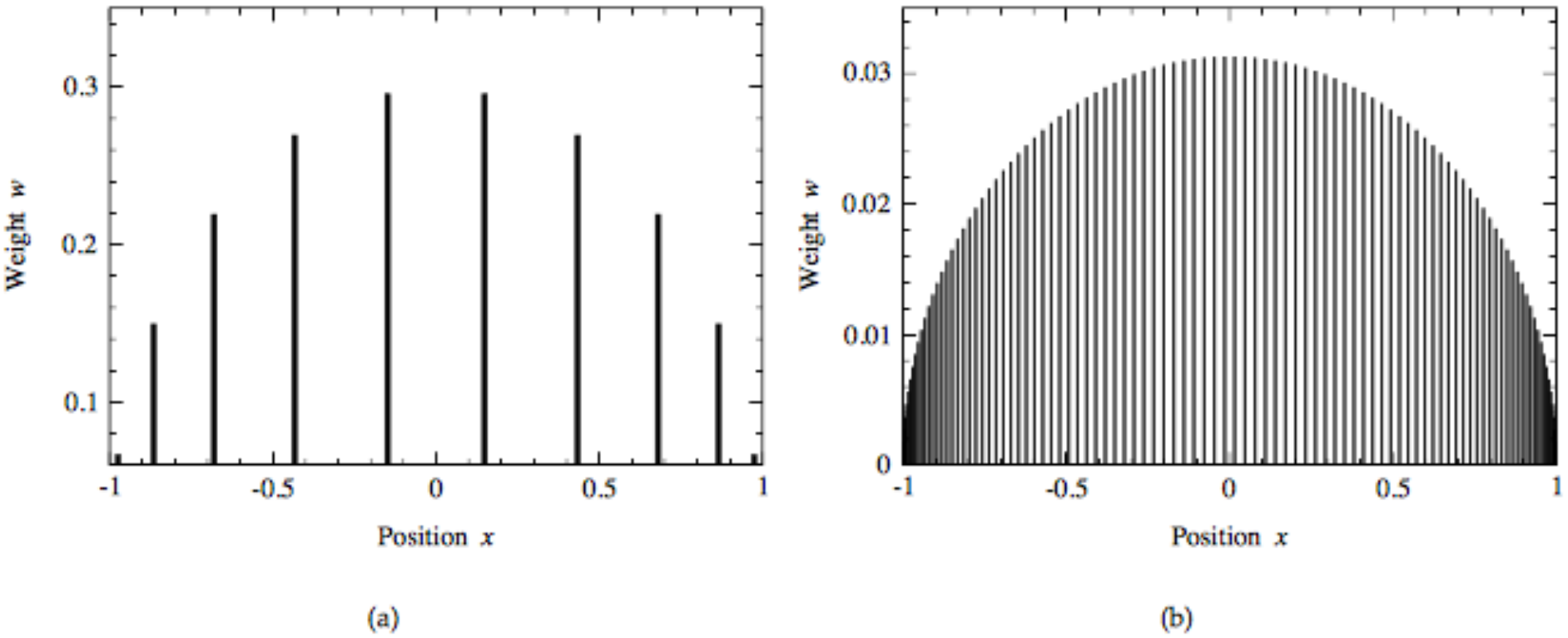


Figure 5.4: Sample points and weights for Gaussian quadrature. The positions and heights of the bars represent the sample points and their associated weights for Gaussian quadrature with (a) $N = 10$ and (b) $N = 100$.

$$\int_a^b f(x)dx \approx \sum_{k=1}^{N} \mathrm{w}_k f(x_k)$$

Solving in a way to get exact results for higher order polynomials (adjusting weights and samples, like in importance sampling).

So what's next?

- Calculus:
  - Extension of Gaussian Quad: Gauss-Kronrod
  - Other special orthogonal bases for easy integration/differentiation (DFT, Chebychev polynomials, spherical harmonics)

FILL OUT THE ONLINE EVALUATIONS

Solving Linear Systems Ax=v

- Gaussian elimination, partial pivoting, LU decomposition

$$L_N L_{N-1} \ldots L_0 Ax = Ux = \begin{pmatrix} + & + & + & + & + \\ 0 & + & + & + & + \\ 0 & 0 & + & + & + \\ 0 & 0 & 0 & + & + \\ 0 & 0 & 0 & 0 & + \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ \vdots \\ x_N \end{pmatrix} = L_N L_{N-1} \ldots L_0 v$$

$$L = \left(L_N L_{N-1} \ldots L_0\right)^{-1} = L_0^{-1} \ldots L_N^{-1} = \begin{pmatrix} + & 0 & 0 & 0 & 0 \\ + & + & 0 & 0 & 0 \\ + & + & + & 0 & 0 \\ + & + & + & + & 0 \\ + & + & + & + & + \end{pmatrix}$$

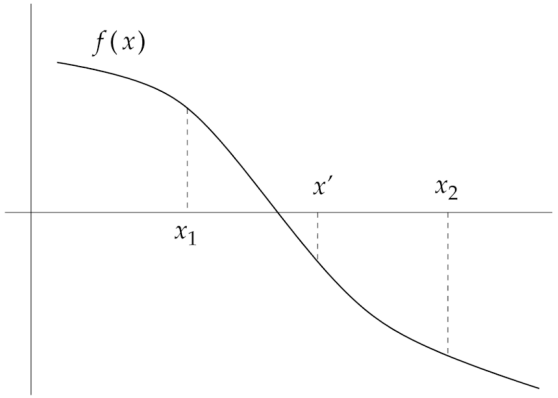
$$LU = A, \text{ so } LUx = v, \text{ or } Ux = L^{-1}v$$

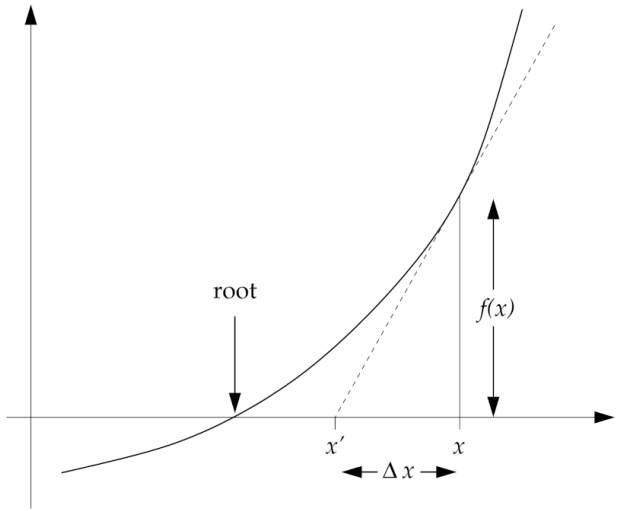
Solving Eigenvalue and Nonlinear Systems

- Using the QR method, we solved

$$Av = \lambda v \text{ or } \mathrm{AV}=\mathrm{VD}$$

- Using relaxation, Newton, bisection, golden ratio search (etc.) we found roots of equations and solved local max/min problems.



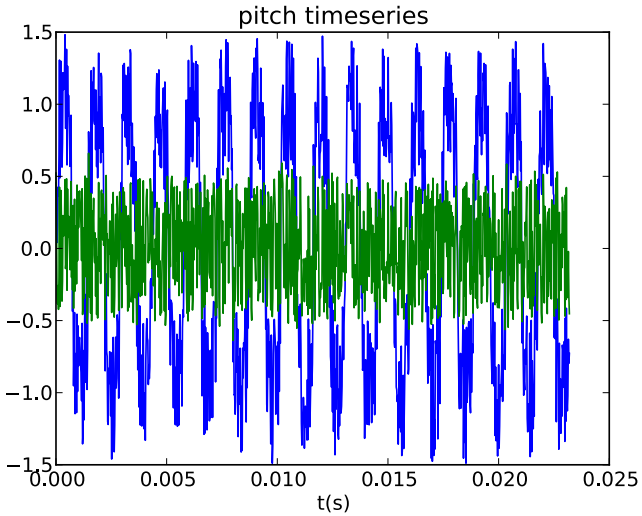


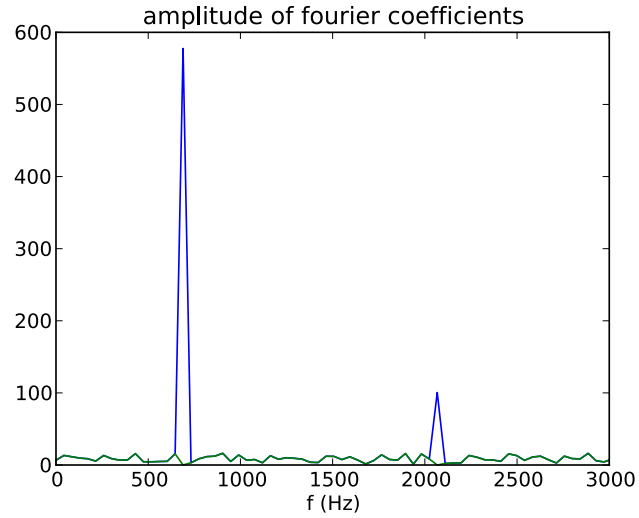
Fourier Transforms, DFTs, FFTs

$$y_k = f(x_k), \quad c_k = N\gamma_k$$

$$\text{DFT: } c_k = \sum_{n=0}^{N-1} y_n \exp\left(-i\frac{2\pi kn}{N}\right) \Rightarrow c_k = c^*_{N-k} \text{ for real } y(x)$$

$$\text{Inverse DFT: } y_n = \frac{1}{N}\sum_{k=0}^{N-1} c_k \exp\left(i\frac{2\pi kn}{N}\right)$$





- Do you remember what the connection of the DFT to the Trapezoidal Rule of integration is?
- FFT: grouping your calculations in a smart way: $O(N^s)$ -> $O(N \log N)$

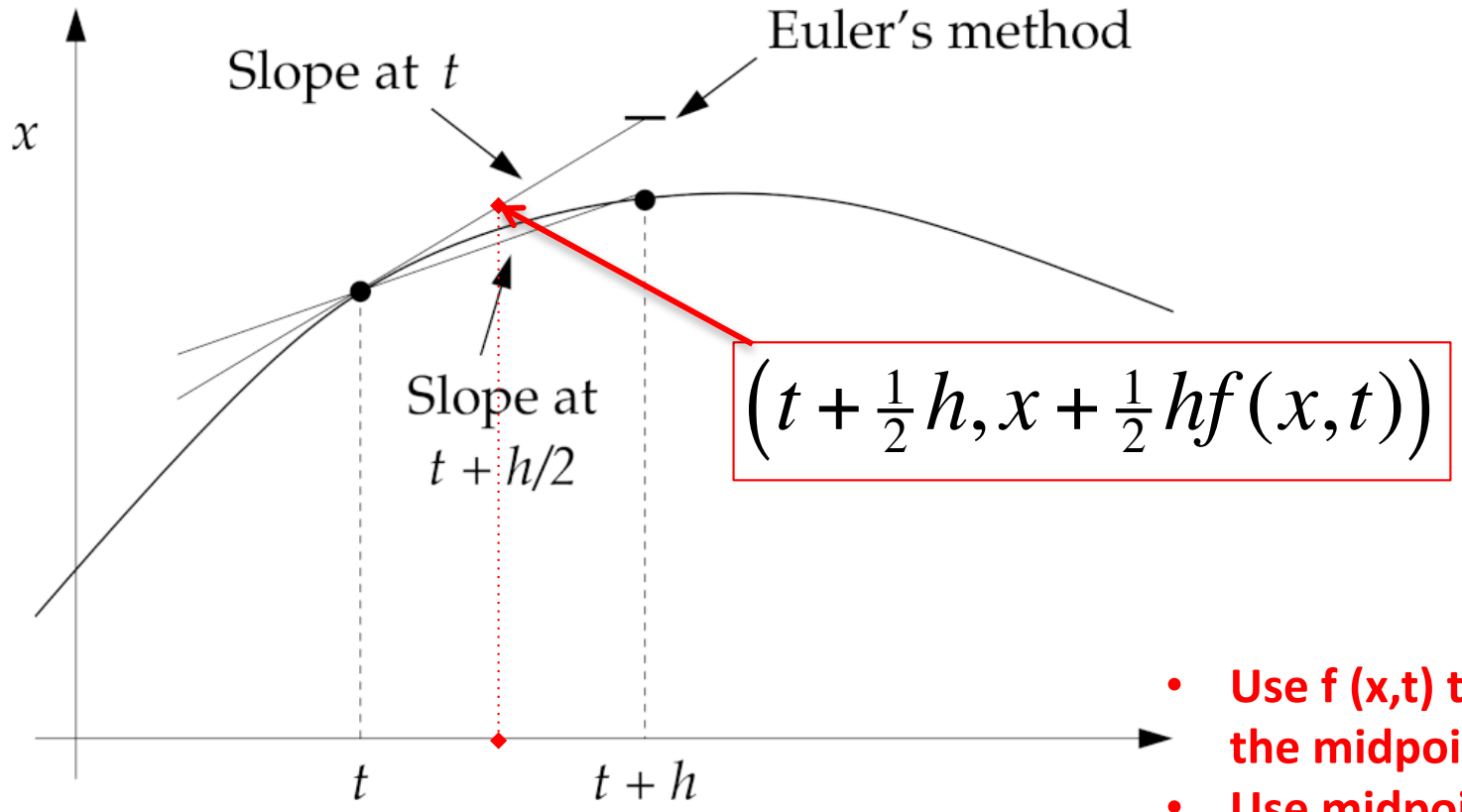
So what's next?

- Linear/nonlinear systems:
  - Working efficiently with sparse systems (as in banded.py): conjugate gradient, Lanczos
  - Singular Value Decomposition for non-square or near singular matrices
  - Cholesky decomposition: LU decomposition for symmetric positive definite matrices.
  - Optimization: gradient descent, simplex …
- Fourier Transforms:
  - Convolution/deconvolution
  - Correlation/autocorrelation
  - Filtering, data windowing, dealing with unevenly sampled data
  - Wavelet transforms (localized functions)

Solving ODEs

- RK methods: using current information to estimate next slope to higher and higher accuracy.
- Leapfrog/Verlet: midpoint methods that are reversible.
- Adaptive time stepping: extrapolating to the next step to see if it is acceptable.

Runge-Kutte Methods



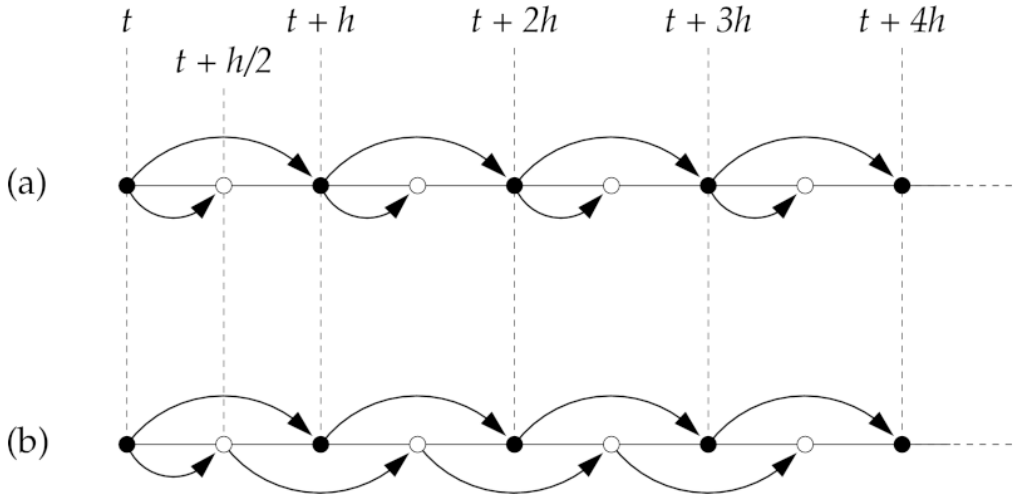
- **Use f (x,t) to estimate the midpoint.**
- **Use midpoint estimate to get improved slope estimate.**
- **Can go to higher and higher accuracy (RK4)**

Second order Runge-Kutte:

$$x(t+h) = x(t) + hf\left(x + \tfrac{1}{2}hf(x,t), t + \tfrac{1}{2}h\right)$$

Leapfrog Methods

$$\text{RK2: } x(t+h) = x(t) + hf\left(x + \tfrac{1}{2}hf(x,t), t + \tfrac{1}{2}h\right)$$



RK2: Next step requires f info from midpoint ahead.

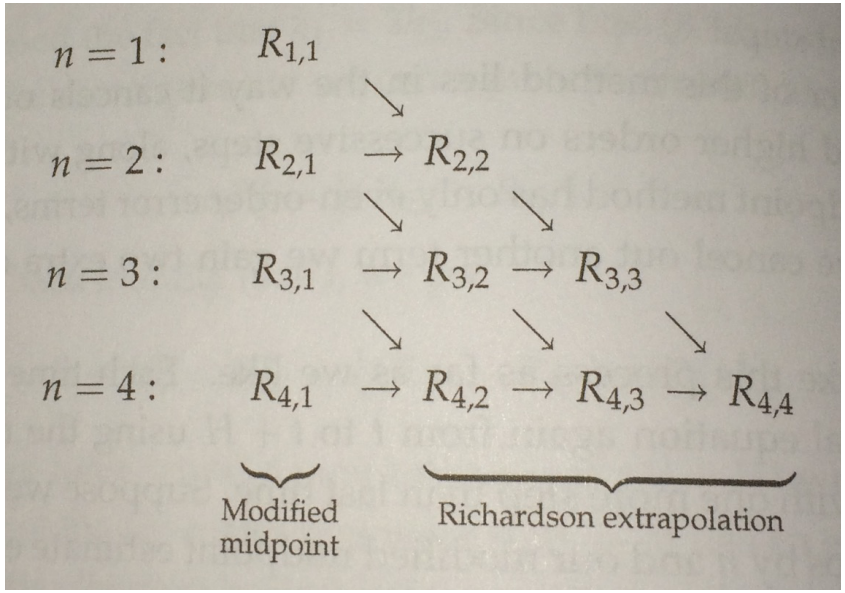
LF: Reversible, especially applicable to

- Leapfrog: next midpoint from last one.

$$x(t+h) = x(t) + hf\left(x\left(t + \tfrac{1}{2}h\right), t + \tfrac{1}{2}h\right)$$

$$x\left(t + \tfrac{3}{2}h\right) = x\left(t + \tfrac{1}{2}h\right) + hf\left(x(t+h), t+h\right)$$

Bulirsch-Stoer



$$x_0 = x(t)$$

$$y_1 = x_0 + \tfrac{1}{2} Hf(x_0, t)$$

$$x_1 = x_0 + Hf(y_1, t + \tfrac{1}{2} H)$$

$$x_1' = y_1 + \tfrac{1}{2} Hf(x_1, t + H)$$

$$x(t+H)_{\text{final}} = \tfrac{1}{2}[x_1 + x_1'] = R_{1,1}$$

$$x(t+H) = R_{1,1} + c_1 H^2 + O(H^4)$$

$$= R_{2,1} + c_1 (H/2)^2 + O\left((H/2)^4\right)$$

$$\text{So } x(t+H) = R_{2,1} + \frac{1}{3}(R_{2,1} - R_{1,1}) + O\left((H/2)^4\right)$$

$$R_{3,2} = R_{3,1} + \tfrac{4}{5}(R_{3,1} - R_{2,1})$$

$$R_{3,3} = R_{3,2} + \tfrac{16}{65}(R_{3,2} - R_{2,2})$$

$$x(t+H) = R_{3,3} + O\left((H/3)^6\right)$$

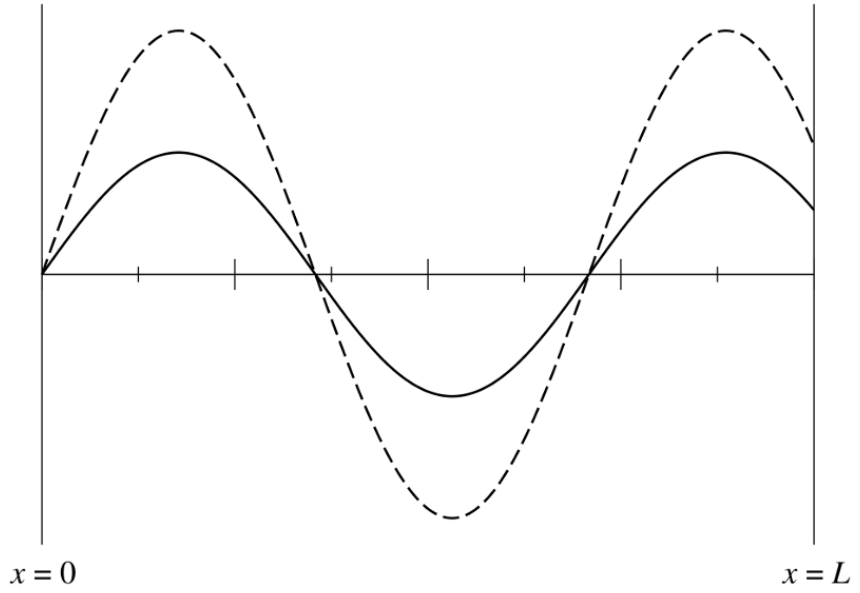
$$\cdots$$

$$x(t+H) = R_{n,m+1} + O\left((H/n)^{2m+2}\right)$$

Shooting Method for Solving BVPs

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2}+V(x)\psi=E\psi,$$

$$\psi(x=0)=\psi(x=L)=0$$



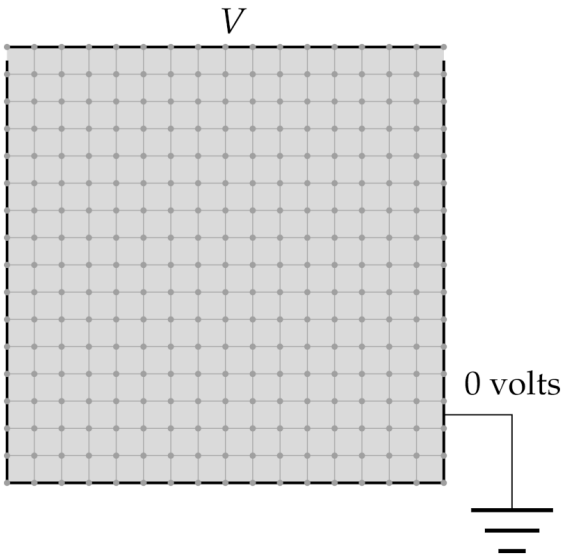
- Integrating ODEs using accurate time stepping to satisfy boundary conditions.

Method Stability

- We touched on this very lightly.
- It's important for you to understand why Euler Forward is unstable, implicit methods are more stable, and how having a stable method is not always ideal.
- We tried algebraic and Von Neumann approaches to stability analysis.

Solving PDEs

- Elliptic equations: Jacobi/Gauss-Seidel relaxation.
- FTCS: easy to code but unstable.
- Crank Nicholson: stable, matrix based (solving a linear system)
- Spectral (using FT concepts)

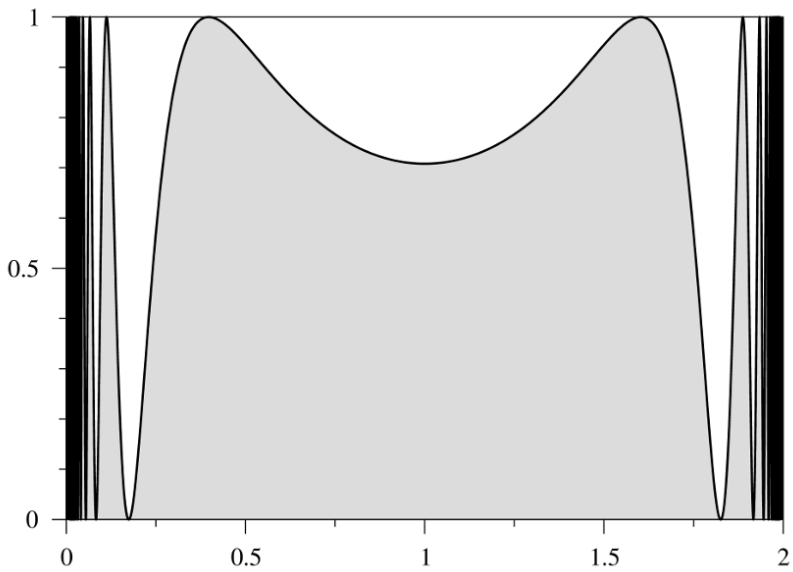


So what's next?

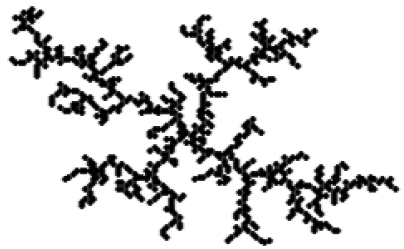
- ODEs:
  - Solving stiff equations: Rosenbrock, semi-implicit extrapolation
  - Predictor-corrector methods
- PDEs
  - Lax-Wendroff (higher order than CN)
  - Finite element methods, spectral element methods, finite volume methods
  - Different gridding systems (triangular, icosohedral, nested, adaptive)

Random and Monte Carlo Methods

- Random number generators
- Transformation of distributions
- Monte Carlo integration: hit or miss, importance sampling.
- Simulations of random processes.



$$I = \int_a^b f(x)dx = \left\langle \frac{f(x)}{w(x)} \right\rangle_w \int_a^b w(x)dx$$



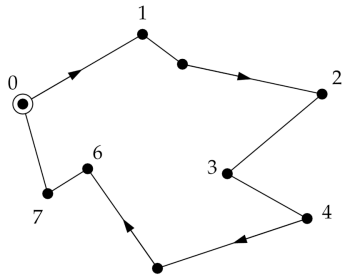
Stat Mech and Simulated Annealing

- Using importance sampling to choose the most likely values of occurrence from Boltzmann stats

$$< X >= \frac{1}{N} \sum_{k=1}^{N} X_k$$

$$p_k = \frac{w_k}{\sum_{i=1}^{ALL} w_i} = \frac{P(E_k)}{\sum_{i=1}^{ALL} P(E_i)} = P(E_k)$$

- Markov Chain method: reducing global problem to a series of probabilistic decisions that converges to a reasonable solution.
- Simulated annealing: extending Markov chain idea to solve global max/min problems.



So what's next?

- Monte Carlo:
  - Sampling strategies (sparse sampling of multidimensional parameter space)
  - Stochastic differential equations

So what's next?

- More fun with python
  - Good graphics capabilities (using basemap, mayavi, etc.)
  - Computer algebra with sympy and other programs.
  - Exploring other features of anaconda (ipython notebooks, other ides)
- More work on design for complex projects
  - Better pseudocoding/debugging.
  - Version control and open source (github)
  - Best practices: software-carpentry.org

FILL OUT THE ONLINE EVALUATIONS

So what's next?

- We'll finish off marking your labs soon.
- Keep working on your projects and keep in touch about your progress in computational physics.
- Good luck in exams, post-graduate plans!
- Have a great holiday!