Scientific Computing (Phys 2109/Ast 3100H)
II. Numerical Tools for Physical Scientists

SciNet HPC Consortium University of Toronto

Lecture 2: Integration, ODE solvers, Molecular Dynamics

January 2012

Lessons from HW from lecture 1

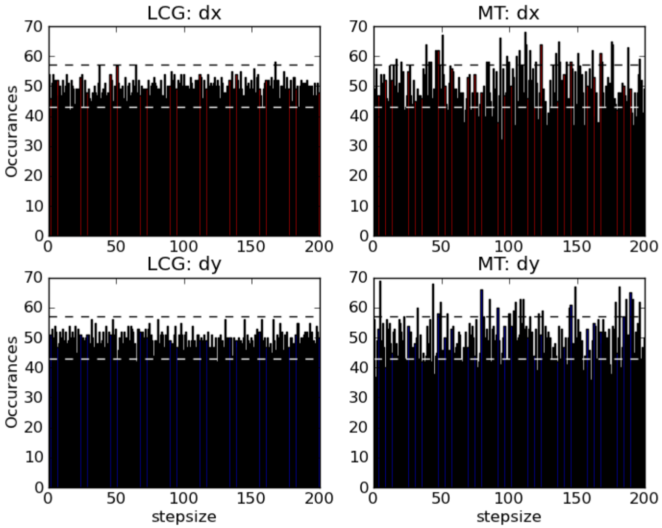
Floating point sums
r

Forward sum	=	1
Backward sum	=	1.25
Pairwise sum	=	2
Pairwise sum (sorted)	=	2
Forward sum error	=	1
Backward sum error	=	0.75
Pairwise sum (unsorted) error	=	0
Pairwise sum (sorted) error	=	0

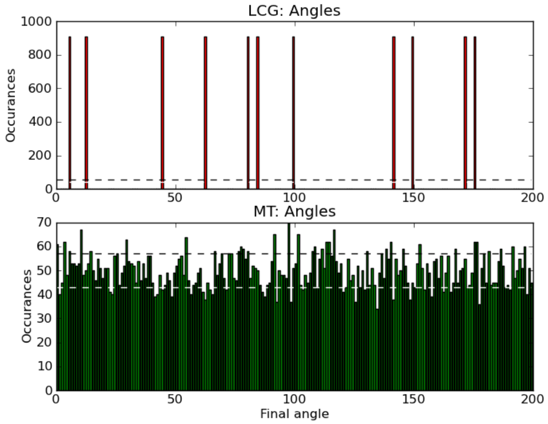
- Doing the summation forward just results in 1; at the first step, 1 + 1.e-8 = 1 + (something less than machine epsilon) = 1, and all 1e8 following steps then play out the same way.
- Even doing the sum backwards doesn't help; as soon as you add up enough 1e-8's to sum up to (1.e-8/machine epsilon), which is about 1/4, the following 1.e-8s don't contribute to sum, then you get the final 1.

Lessons from HW from lecture 1

- dx/dy histograms seems reasonable, although the variance in the LCG case seems somewhat less than expected.



Distribution of angles

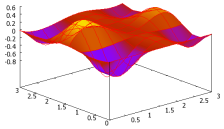


- Consecutive numbers out of the LCG are very strongly correlated, leading to just a handful of final angles picked out
- Moral of story – don't make up your own RNG. Even if simple statistics look reasonable, could get bitten.

Lecture 2 of Part II

Numerical Integration

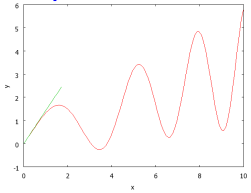
$$\mathcal{I} = \int_{\mathcal{D}} f(x)\, d^d x$$



Ordinary Differential Equation

$$\sum_n a_n(x,y)\frac{d^n y}{dx^n} = f(x,y)$$

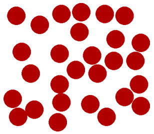
$+$ boundary/initial conditions



Molecular Dynamics Simulations

$$m_i \ddot{r}_i = f_i(\{r\}, \{\dot{r}_j\}, t)$$

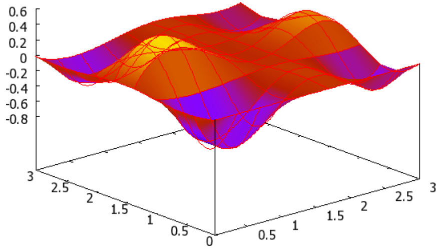
$+$ initial conditions



Numerical Integration

Numerical Integration

$$\mathcal{I} = \int_{\mathcal{D}} \mathbf{f(x)}\, \mathbf{d^d x}$$



Large variety of methods, depending on **d**, $\mathbf{f(x)}$ and x

For $\mathbf{d = 1}$:

$$\mathcal{I} = \int_{\mathbf{a}}^{\mathbf{b}} \mathbf{f(x)}\, \mathbf{dx}$$

1. Regular grid
2. Gaussian Quadrature

Small **d**:

1. Regular grid
2. Recursive Quadrature

Large **d**:

1. Monte Carlo

Numerical Integration in $\mathbf{d = 1}$

Regularly spaced grid method #1

On small interval $[a, a+h]$, interpolate using values at a few points.

- Interpolating polynomial of degree 0 using mid-point:

$$\int_a^{a+h} f(x)\,dx \approx h\,f\left(a + \frac{h}{2}\right)$$

- Linear interpolation based on end-points: Trapezoidal rule

$$\int_a^{a+h} f(x)\,dx \approx \frac{h}{2}\left[f(a) + f(a+h)\right]$$

- Compose trapezoidal rule $n\times$ on sub-intervals $[kh, (k+1)h]$ ($k = 0, \ldots, n-1$; $h = (b-a)/n$): Extended trapezoidal rule

$$\int_a^{b} f(x)\,dx \approx h\left[\frac{f(a)+f(b)}{2} + \sum_{k=1}^{n-1} f(a+kh)\right] + \mathcal{O}\left(\frac{1}{n^2}\right)$$

SciNet compute • calcul CANADA

Numerical Integration in $\mathbf{d = 1}$

Regularly spaced grid method #2

- Interpolating function of degree $\mathbf{2}$ on $\mathbf{[a, a + h]}$ using end-points and mid-point:
  Simpsons' rule

$$\int_a^{a+2h} f(x)\,dx \approx \left[\tfrac{2}{3}f(a) + \tfrac{4}{3}f(a + \tfrac{h}{2}) + \tfrac{2}{3}f(a + h)\right]$$

- Compose $\mathbf{n}$ times on full interval:
  Extended Simpsons' rule

$$\int_a^{b} f(x)\,dx \approx h\Big[\tfrac{1}{3}f(a) + \tfrac{4}{3}f(a + h) + \tfrac{2}{3}f(a + 2h) + \tfrac{4}{3}f(a + 3h) + \tfrac{2}{3}f(a + 4h) + \cdots + \tfrac{1}{3}f(b)\Big] + \mathcal{O}\left(\frac{1}{n^4}\right)$$

Numerical Integration in $\mathbf{d = 1}$

Method using unevenly spaced grid: Gaussian quadrature

- Based on orthogonal polynomials on the interval.
  E.g. Legendre, Chebyshev, Hermite, Jacobi polynomials
- Compute and $\mathbf{f_i = f(x_i)}$ then

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} v_i f_i$$

  with choice of $\mathbf{x_i}$ and $\mathbf{v_i}$ based on zeroes of polynomial of degree $\mathbf{n}$ and of integrals of orthogonal polynomials.
- Well-defined procedure to find $\mathbf{\{x_i\}}$ and $\mathbf{\{v_i\}}$
  (see e.g. *Numerical Recipes*).
- Error roughly the same as Simpsons' rule but as if $\mathbf{n \to 2n}$.

Numerical Integration in $\mathbf{d = 1}$

Specifiying accuracy

We may know the order of the error term, but not the accuracy.

$\Downarrow$

Good numerical integration routines increases $\mathbf{n}$ until some specified accuracy is achieved.

- Easier with fixed grid because old points get reused.
- But in standard Gaussian quadrature, the $\{\mathbf{x_i}\}$ for $\mathbf{n}$ and for $\mathbf{n+1}$ have no points in common.
- Gauss-Kronrod methods allow reuse, but require specific sequences of $\mathbf{n}$ (e.g. 10, 21, 43, 87).

Numerical Integration in $\mathbf{d = 1}$

Adaptive schemes

If a function is not smooth or behaves differently throughout the domains, divide and apply the above techniques to subdomains.

Weight functions

$$\mathcal{I} = \int_a^b w(x) f(x)\, dx$$

There are ways to include weight **w** in the scheme.

- If **w** is standard, this can be done by changing the polynomials
- If **w** has singularities, this may remove numerical difficulties.

Don't code these yourself! Schemes like this, as well as Gaussian quadratures, are implemented in libraries such as the gsl.

Numerical Integration in $\mathbf{d > 1}$ but small.

Why multidimensional integration is hard:

- Requires $\mathcal{O}(\mathbf{n}^\mathbf{d})$ points if its 1d counterpart requires $\mathbf{n}$.
- A function can be peaked, and peak can easily be missed.
- The domain itself can be complicated.

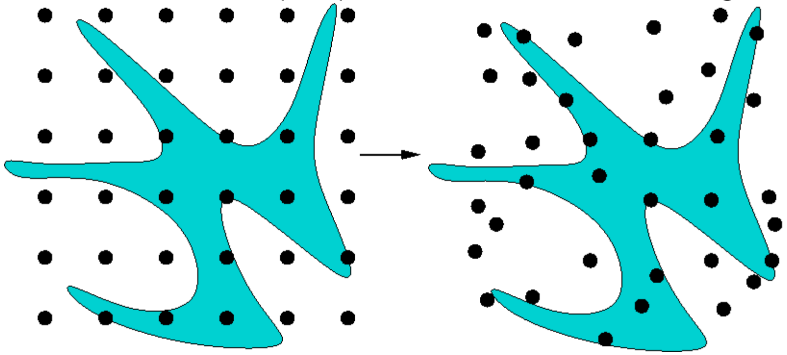
Numerical Integration in $\mathbf{d} > 1$ but small.

So what should you do?

- If you can reduce the **d** by exploiting symmetry or doing part of the integral analytically, do it!
- If you know the function to integrate is smooth and its domain is fairly simple, you could do repeated 1d integrals (fixed-grid or Gaussian quadrature)
- Otherwise, you'll have to consider Monte Carlo.

Monte Carlo Integration

Use random numbers to pick points at which to evaluate integrand.



Similar to the rejection/acceptance scheme of the previous lecture.

- Convergence always as $\mathbf{1/\sqrt{n}}$, regardless of $\mathbf{d}$.
- Simple and flexible.
- Can generalize to focus on important parts.

Importance Sampling

$$\mathcal{I} = \int_V f(x)\,dx$$

Suppose $f(x)$ is non-zero only in specific x regions.

- Want to place more points in region where integrand is large.
- Define function $w(x)$ that tells which regions are significant.
  - Require $w(x) > 0$ for any point x in volume where $f \neq 0$.
  - Re-express integral as:

$$\mathcal{I} = \int_V \frac{f(x)}{w(x)}\,w(x)\,dx$$

- Draw a set of n points $\{x_1, \ldots, x_n\}$ weighted by $w(x)$, then

$$\bar{I} \approx \frac{1}{n}\sum_{i=1}^{n}\frac{f(x_i)}{w(x_i)}$$

- Converges to right answer for $n \to \infty$ as $1/\sqrt{n}$.

How does this improve the rate of convergence?

- The statistical uncertainty is related to the variance $\sigma_{\bar{I}}^2$ of $\bar{I}$:

$$\sigma_{\bar{I}}^2 = \frac{1}{n}\sum_{i}^{n}\langle \Delta I_i \Delta I_i \rangle \qquad \text{where} \qquad \Delta I_i = \frac{f(x_i)}{w(x_i)} - \bar{I}$$

  (assuming ΔI_i are statistically independent).
- Vastly different values of $f(x_i)/w(x_i)$ lead to large uncertainty.
- If $\alpha\, w(x_i) = f(x_i)$, then $f(x_i)/w(x_i) = \alpha$ and

$$\left\langle \frac{f(x_i)}{w(x_i)} \right\rangle = I = \alpha \qquad \left\langle \left(\frac{f(x_i)}{w(x_i)}\right)^2 \right\rangle = \alpha^2,$$

  and $\sigma_{\bar{I}}^2 = 0$.
- Generally desire all $f(x_i)/w(x_i)$ to be roughly the same for all sampled points x_i to mimimize $\sigma_{\bar{I}}^2$.

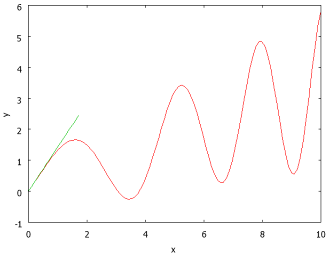
ODE solvers

Ordinary Differential Equations (ODEs)

$$\sum_n a_n(x,y)\frac{d^n y}{dx^n} = f(x,y)$$

Example

$$\frac{d^2y}{dx^2} = -y$$



- Ordinary $\rightarrow$ x is one dimensional (often time).
- Boundary conditions: much like PDEs: next lecture
- Initial conditions: y, $\frac{dy}{dx}$, ..., at $x = x_0$
- Define $y_0 = y$; $y_1 = \frac{dy}{dx}$, ..., $\rightarrow$ set of first order ODEs
  Example:

$$\frac{dy_0}{dx} = y_1$$

$$\frac{dy_1}{dx} = -y_0$$

SciNet compute • calcul CANADA

Numerical approaches

Start from the general form:

$$\frac{dy_i}{dx} = f(x, \{y_j\})$$

- All approaches will evaluate f at discrete points $x_0, x_1, \ldots$.
- Initial conditions: specify $y_i(x_0)$ and $\frac{dy_i}{dx}(x_0)$.
- Consecutive points may have a fixed step size $h = x_{k+1} - x_k$ or may be adaptive.
- $\{y_j(x_{i+1})\}$ may be implicitly dependent on f at that value.

Stiff ODEs

- A stiff ODE is one that is hard to solve, i.e. requiring a very small stepsize **h** or leading to instabilities in some algoritms.
- Usually due to wide variation of time scales in the ODEs.
- Not all methods equally suited for stiff ODEs

ODE solver algorithms: Euler

To solve:

$$\frac{dy}{dx} = f(x, y)$$

Simple approximation:

$$y_{n+1} \approx y_n + hf(x_n, y_n) \qquad \text{“forward Euler”}$$

Rational:

$$y(x_n + h) = y(x_n) + h\frac{dy}{dx}(x_n) + \mathcal{O}(h^2)$$

So:

$$y(x_n + h) = y(x_n) + hf(x_n, y_n) + \mathcal{O}(h^2)$$

- $\mathcal{O}(h^2)$ is the local error.
- For given interval $[x_1, x_2]$, there are $n = (x_2 - x_1)/h$ steps
- Global error: $n \times \mathcal{O}(h^2) = \mathcal{O}(h)$
- Not very accurate, nor very stable (next): don't use.

Stability

Example: solve harmonic oscillator numerically:

$$\frac{dy^{(1)}}{dx} = y^{(2)}$$

$$\frac{dy^{(2)}}{dx} = -y^{(1)}$$

Use Euler ($y_{n+1} \approx y_n + hf(x_n, y_n)$) gives

$$\begin{pmatrix} y^{(1)}_{n+1} \\ y^{(2)}_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & h \\ -h & 1 \end{pmatrix} \begin{pmatrix} y^{(1)}_{n} \\ y^{(2)}_{n} \end{pmatrix}$$

Stability governed by eigenvalues $\lambda_\pm = 1 \pm ih$ of that matrix.
$|\lambda_\pm| = \sqrt{1 + h^2} > 1 \quad \Rightarrow$ Unstable for any h!

ODE solver algorithms: implicit mid-point Euler

To solve:

$$\frac{dy}{dx} = f(x, y)$$

Symmetric simple approximation:

$$y_{n+1} \approx y_n + hf(x_n, (y_n + y_{n+1})/2) \qquad \text{“mid-point Euler”}$$

This is an implicit formula, i.e., has to be solved for y_{n+1}.

Example (Harmonic oscillator)

$$\begin{bmatrix} 1 & -\frac{h}{2} \\ \frac{h}{2} & 1 \end{bmatrix} \begin{bmatrix} y^{[1]}_{n+1} \\ y^{[2]}_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & \frac{h}{2} \\ -\frac{h}{2} & 1 \end{bmatrix} \begin{bmatrix} y^{[1]}_{n} \\ y^{[2]}_{n} \end{bmatrix} \Rightarrow \begin{bmatrix} y^{[1]}_{n+1} \\ y^{[2]}_{n+1} \end{bmatrix} = M \begin{bmatrix} y^{[1]}_{n} \\ y^{[2]}_{n} \end{bmatrix}$$

Eigenvalues M are $\lambda_\pm = \frac{(1 \pm ih/2)^2}{1+h^2/4}$ so $|\lambda_\pm| = 1 \Rightarrow$ Stable for all h

$\boxed{\text{Implicit methods often more stable and allow larger step size } h.}$

ODE solver algorithms: Predictor-Corrector

1. Computation of new point
2. Correction using that new point

- Gear P.C.: keep previous values of **y** to do higher order Taylor series (predictor), then use **f** in last point to correct.

  Can suffer from catestrophic cancellation at very low **h**.

- Runge-Kutta: Refines by using mid-points.

  Workhorse even behind fancier solvers.

4th order version:

$$\begin{aligned}
\mathbf{k_1} &= \mathbf{hf(x, y)} \\
\mathbf{k_2} &= \mathbf{hf(x + h/2, y + k_1/2)} \\
\mathbf{k_3} &= \mathbf{hf(x + h/2, y + k_2/2)} \\
\mathbf{k_4} &= \mathbf{hf(x + h, y + k_3)} \\
\mathbf{y'} &= \mathbf{y} + \frac{\mathbf{k_1}}{\mathbf{6}} + \frac{\mathbf{k_2}}{\mathbf{3}} + \frac{\mathbf{k_3}}{\mathbf{3}} + \frac{\mathbf{k_4}}{\mathbf{6}}
\end{aligned}$$

Further ODE solver techniques

Adaptive methods

As with the integration, rather than taking a fixed **h**, vary **h** such that the solution has a certain accuracy.

Don't code this yourself! Adaptive schemes are implemented in libraries such as the gsl.

Geometric, symplectic and variants

Respects hamiltonian form, better energy conservation.
Will discuss in the context of MD.

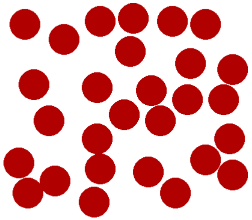
Molecular Dynamics

Molecular Dynamics Simulations

N interacting particles

$$m_i\ddot{r}_i = f_i(\{r\}, \{\dot{r}_j\}, t)$$

$+$ initial conditions



What makes this different from other ODEs?

- Hamiltonian dynamics
- Very expensive evaluation of **f** if **N** is large

For both, we will only touch upon some aspects.

Note that **N**-body simulation fall within this class as well; the numerics does not case whether the particles are molecules or stars.

Hamiltonian dynamics

- Molecular Dynamics aims to compute *equilibrium*, *dynamical* and *transport* properties of *classical many body systems*.
- Many classical systems have Newtonian equations of motion:

$$\dot{\mathbf{r}} = \frac{1}{\mathbf{m}}\mathbf{p} \qquad\qquad \dot{\mathbf{p}} = \mathbf{F} = -\frac{\mathbf{dU}}{\mathbf{dr}},$$

  or $\dot{\mathbf{x}} = \mathbf{L}\mathbf{x}$, with $\mathbf{L}\mathbf{A} = \{\mathbf{A}, \mathbf{H}\}$, where $\mathbf{x} = (\mathbf{r}, \mathbf{p})$.
- Energy $\mathbf{H} = \frac{|\mathbf{p}|^2}{2\mathbf{m}} + \mathbf{U}(\mathbf{r})$ is conserved under the dynamics.
- Potential energy is typically a sum of pair potentials:

$$\mathbf{U}(\mathbf{r}) = \sum_{(\mathbf{i},\mathbf{j})} \varphi(\mathbf{r}_{\mathbf{ij}}) = \sum_{\mathbf{i}=1}^{\mathbf{N}} \sum_{\mathbf{j}=1}^{\mathbf{i}-1} \varphi(\mathbf{r}_{\mathbf{ij}}),$$

  which entails the following expression for the forces **F**:

$$\mathbf{F}_{\mathbf{i}} = -\sum_{\mathbf{j}\neq\mathbf{i}} \frac{\mathbf{d}}{\mathbf{dr}_{\mathbf{i}}} \varphi(\mathbf{r}_{\mathbf{ij}}) = \sum_{\mathbf{j}\neq\mathbf{i}} \underbrace{\varphi'(\mathbf{r}_{\mathbf{ij}}) \frac{\mathbf{r}_{\mathbf{j}} - \mathbf{r}_{\mathbf{i}}}{\mathbf{r}_{\mathbf{ij}}}}_{\mathbf{F}_{\mathbf{ij}}}$$

Hamiltonian dynamics as disguised importance sampling

- If the system is ergodic then time average equals the microcanonical average:

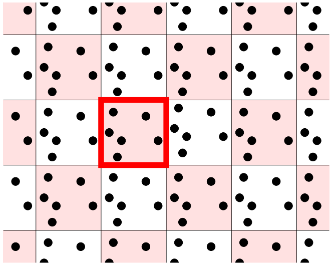
$$\lim_{\mathbf{t}_{\text{final}}\to\infty} \frac{1}{\mathbf{t}_{\text{final}}} \int_0^{\mathbf{t}_{\text{final}}} \mathbf{dt\ A(x(t))} = \frac{\int \mathbf{dx\ A(x)}\ \delta(\mathbf{E}-\mathbf{H(x)})}{\int \mathbf{dx}\ \delta(\mathbf{E}-\mathbf{H(x)})}.$$

- For large **N**, microcanonical and canonical averages are equal for many quantities **A**.

- Need long times $\mathbf{t}_{\text{final}}$!

Boundary conditions

- When simulating finite systems, a wall potential would give finite size effects and destroy translation invariance.
- More benign: *Periodic Boundary Conditions*
- All particles in box have coordinates between $-\mathbf{L}/\mathbf{2}$ and $\mathbf{L}/\mathbf{2}$.
- A particle exiting simulation box is put back at the other end.



- The box with thick red boundaries is our simulation box.
- Other boxes are copies, or “periodic images”
- The other squares contain particles with shifted positions
- “Flat torus”

Force calculations

- A common pair potential is the Lennard-Jones potential

$$\varphi(\mathbf{r}) = 4\varepsilon\left[\left(\frac{\sigma}{\mathbf{r}}\right)^{12} - \left(\frac{\sigma}{\mathbf{r}}\right)^{6}\right],$$

  - σ is a measure of the range of the potential.
  - ε is its strength.
  - The potential is positive for small $\mathbf{r}$: repulsion.
  - The potential is negative for large $\mathbf{r}$: attraction.
  - The potential goes to zero for large $\mathbf{r}$: short-range.
  - The potential has a minimum of $-\varepsilon$ at $2^{1/6}\sigma$.
- Computing all forces in an N-body system requires the computation of $\mathbf{N(N-1)/2}$ forces $\mathbf{F_{ij}}$
- Force Computation often the most demanding part of MD.
- Avoid infinite sums: modify the potential such that it becomes zero beyond a certain *cut-off* distance $\mathbf{r_c}$:

$$\varphi'(\mathbf{r}) = \begin{cases} \varphi(\mathbf{r}) - \varphi(\mathbf{r_c}) & \text{if } \mathbf{r} < \mathbf{r_c} \\ 0 & \text{if } \mathbf{r} \geq \mathbf{r_c} \end{cases}$$

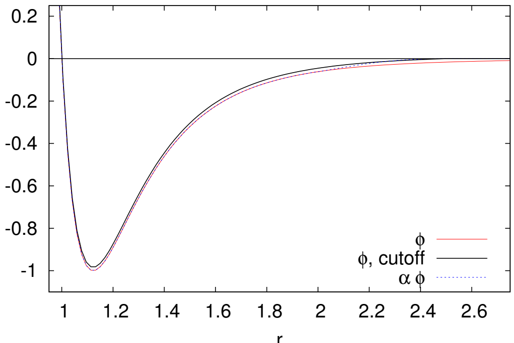
- To also avoid discontinuities in derivatives, one can use a schemes such as

$$\varphi''(r) = \alpha(r)\varphi(r) \tag{1}$$

where

$$\alpha(r) = \begin{cases} 1 & r < r_c' \\ \frac{(r_c - r)^2(r_c - 3r_c' + 2r)}{(r_c - r_c')^3} & r_c' \leq r \leq r_c \\ 0 & r > r_c \end{cases} . \tag{2}$$

Cutoff Lennard-Jones potentials, $\varepsilon=\sigma=1$, $r_c = 2.5$, $r'_c = 2$



Streamlining the force evaluation

Cell divisions

- Divide the simulation box into cells larger than the cutoff $\mathbf{r_c}$.
- Make a list of all particles in each cell.
- In the sum over pairs in the force computation, only sum pairs of particles in the same cell or in adjacent cells.

Neighbour lists (also called Verlet lists)

- Make a list of pairs of particles that are closer than $\mathbf{r_c} + \boldsymbol{\delta}\mathbf{r}$.
- Sum over the list of pairs to compute the forces.
- The neighbour list are to be used in subsequent force calculations as long as the list is still valid.
- Invalidation criterion: a particle has moved more than $\boldsymbol{\delta}\mathbf{r}/\mathbf{2}$.
- Therefore, before a new force computation, check if any particle has moved more than $\boldsymbol{\delta}\mathbf{r}/\mathbf{2}$ since the last list-building. If so, rebuild the Verlet list, otherwise use the old one.

For large systems, turns $\mathbf{N^2}$ into $\mathbf{N}$.

Desirable qualities for a molecular dynamics integrator

- *Accuracy*
- *Efficiency*
- *Stability*
- *Respect physical laws:*
  - Time reversal symmetry
  - Conservation of energy
  - Conservation of linear momentum
  - Conservation of angular momentum
  - Conservation of phase space volume

The most efficient algorithm is then the one that allows the largest possible time step for a given level of accuracy, *while maintaining stability and preserving conservation laws*.

Symplectic integrators

Momentum Verlet Scheme (first version)

$$\begin{aligned} \mathbf{r_{n+1}} &= \mathbf{r_n} + \frac{\mathbf{p_n}}{\mathbf{m}}\mathbf{h} + \frac{\mathbf{F_n}}{\mathbf{2m}}\mathbf{h}^2 \\ \mathbf{p_{n+1}} &= \mathbf{p_n} + \frac{\mathbf{F_{n+1}} + \mathbf{F_n}}{\mathbf{2}}\mathbf{h} \end{aligned}$$

The momentum rule appears to pose a problem since $\mathbf{F_{n+1}}$ is required. But to compute $\mathbf{F_{n+1}}$, we need only $\mathbf{r_{n+1}}$, which is computed in the integration step as well.
Equivalent to position Verlet scheme.

Symplectic integrators

Momentum Verlet Scheme (second version)

The extra storage step can be avoided by introducing the half step momenta as intermediates:

$$\mathbf{p_{n+1/2} = p_n + \frac{1}{2} F_n h}$$
$$\mathbf{r_{n+1} = r_n + \frac{p_{n+1/2}}{m} h}$$
$$\mathbf{p_{n+1} = p_{n+1/2} + \frac{1}{2} F_{n+1} h}$$

Also nice and symmetric:

1. Half momentum step
2. Full position step
3. Half momentum step

First step the same as the last (with updated F).

Symplectic integrators from Hamiltonian splitting methods

- For sampling, one wants a long trajectory (formally $\mathbf{t_f} \rightarrow \infty$).
- It is therefore important that an integration algorithm be stable.
- The momentum Verlet scheme, on the other hand, is much more stable than, say, the Euler scheme.
- To see why, one should re-derive the momentum Verlet scheme from a completely different starting point, using a so-called *Hamiltonian splitting method* (also known as *Geometric integration*).

Symplectic integrators from Hamiltonian splitting methods

Very, very briefly:

- Any Hamiltonian $\mathbf{H}$ has an associated Liouvillean $\mathbf{L} = \{\mathbf{H}, .\}$.
- The Liouvillean generates a flow on phase space: $\mathbf{U(t) = \exp Lt}$.
- Split up Hamiltonian in $\mathbf{K}$ parts, $\mathbf{H_1 \dots H_K}$.
- Gives $\mathbf{K}$ flows: $\mathbf{U_1 \dots U_K}$.
- Baker-Campbell-Hausdorff formula gives approximate factorization, e.g. $\mathbf{H = H_1 + H_2 \Rightarrow U(h) \approx U_1(h/2)U_2(h)U_1(h/2)}$
- Symmetric form reduced order and preserves time reversibility.
- This is momentum Verlet!
- Further using BCH, one can derive a shadow Hamiltonian.
- $\Rightarrow$ simulated system retains all hamiltonian properties.

Homework

Homework 1

Compute numerically:

$$\int_0^3 \mathbf{f(x)\,dx}$$

with

$$\mathbf{f(x) = \ln(x)\sin(x)e^{-x}},$$

using three different methods:

1. Extended Simpsons' rule
2. Gauss-Legendre quadrature
3. Monte Carlo sampling

Compare the convergence of these methods by increasing number of function evaluations.
Hint: what is f(0)?

Homework 2

Using an adaptive 4th order Runge-Kutta approach, with a relative accuracy of 1e-4, compute the solution for $\mathbf{t = [0, 100]}$ of the following set of coupled ODE(Lorenz oscillator)

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dt} &= (\rho - z)x - y \\ \frac{dz}{dt} &= xy - \beta z \end{aligned}$$

with $\sigma = 10, \beta = 8/3, \rho = 28$, and with initial conditions

$$\begin{aligned} x(0) &= 10 \\ y(0) &= 20 \\ z(0) &= 30 \end{aligned}$$

Plot the result.

Hint: study the GSL documentation.