PHYS 410: Computational Physics
Finite Difference Solution of the Gravitational N-Body Problem

1. THE GRAVITATIONAL N-BODY PROBLEM

1.1 Physical & Mathematical Formulation

- Consider N point particles, labelled by an index i, with masses m_i

$$m_i, \;\; i = 1, 2, \ldots N$$

and position vectors, $\mathbf{r}_i(t)$

$$\mathbf{r}_i(t) \equiv [x_i(t),\, y_i(t),\, z_i(t)]\,, \;\; i = 1, 2, \ldots N$$

where we have established a standard set of Cartesian coordinates (x, y, z) with some arbitrarily chosen origin. (In practice, however, it may be most convenient to choose the origin at the center of mass of the system.)

- We wish to study the dynamics of the system due to the (attractive) Newtonian gravitational force exerted by each particle on every other particle.

- Combining Newton's second law, as well as the law of gravitation, we have the basic equations of motion in vector form

$$m_i\, \mathbf{a}_i = G \sum_{j=1,\, j\neq i}^{N} \frac{m_i\, m_j}{r_{ij}^2}\, \hat{\mathbf{r}}_{ij}\ , \qquad i = 1, 2, \ldots N\ , \qquad 0 \le t \le t_{\max} \tag{1}$$

where

- $\mathbf{a}_i = \mathbf{a}_i(t)$ is the acceleration of the i-th particle
- G is Newton's gravitational constant
- r_{ij} is the magnitude of the separation vector $\mathbf{r}_{ij}$ between particles i and j:

$$\mathbf{r}_{ij} \equiv \mathbf{r}_j - \mathbf{r}_i$$

$$r_{ij} \equiv |\mathbf{r}_j - \mathbf{r}_i|$$

and we recall that the magnitude of any vector, $\mathbf{w} = [w_x,\, w_y,\, w_z]$, is given by:

$$w \equiv |\mathbf{w}| = \sqrt{w_x^2 + w_y^2 + w_z^2}$$

- $\hat{\mathbf{r}}_{ij}$ is the unit vector in the direction from particle i to particle j (i.e. in the direction of the separation vector:)

$$\hat{\mathbf{r}}_{ij} \equiv \frac{\mathbf{r}_j - \mathbf{r}_i}{r_{ij}} \tag{2}$$

- **Important:** From now on, for brevity of notation we will use

$$\sum_{j=1,\, j\neq i}^{N} \to \sum_{j}$$

and $i = 1, 2, \ldots, N$ and $0 \le t \le t_{\max}$ will be implied.

- For the purposes of computation, it turns out to be more convenient to use (2) in (1) to get

$$m_i\,\mathbf{a}_i = G\sum_j \frac{m_i\,m_j}{r_{ij}^3}\,\mathbf{r}_{ij} \tag{3}$$

where we note that

$$r_{ij}^3 = \left[(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2\right]^{3/2}$$

- It is also convenient to non-dimensionalize the system of equations, which in this case means choosing units in which $G = 1$, which we will hereafter do

- We have

$$\mathbf{a}_i(t) = \frac{d^2\mathbf{r}(\mathbf{t})}{dt^2}$$

so (3) becomes (with $G = 1$)

$$m_i\,\mathbf{a}_i = m_i\,\frac{d^2\mathbf{r}_i}{dt^2} = \sum_j \frac{m_i m_j}{r_{ij}^3}\,\mathbf{r}_{ij}$$

and then dividing both sides of the above equation by m_i, we have

$$\frac{d^2\mathbf{r}_i}{dt^2} = \sum_j \frac{m_j}{r_{ij}^3}\,\mathbf{r}_{ij} \tag{4}$$

- Equation (4) is a system of second-order-in time differential equations for the *vector* quantities, $\mathbf{r}_i(t)$

- We note that using a form of the equations of motion in which we have divided by m_i allows us to integrate the equations for the case that some of the particles are *massless*

- In order to compute a specific solution, we must supply initial conditions, which in this case are the initial positions and initial velocities of the particles, i.e.

$$\mathbf{r}_i(0) = \mathbf{r}_{0i} \qquad i = 1, 2, \ldots, N \tag{5}$$

$$\mathbf{v}_i(0) \equiv \frac{d\mathbf{r}}{dt}\,(0) = \mathbf{v}_{0i} \qquad i = 1, 2, \ldots, N \tag{6}$$

where $\mathbf{r}_{0i}$ and $\mathbf{v}_{0i}$, $i = 1, \ldots, N$ are specified vectors (total of $6N$ numbers)

1.2 Solution via Finite Difference Approximation

1.2.1 Discretization: Step 1—Finite Difference Grid

- Continuum domain is

$$0 \le t \le t_{\max}$$

- We will assume that we can proceed using a uniform time mesh (i.e. constant time step) as usual: may *not* be a good assumption, particularly if particles start "clumping"

- For the purposes of development and convergence testing it is convenient to specify the mesh via *level* parameter, ℓ

$$n_t = 2^\ell + 1$$

$$\Delta t = \frac{t_{\max}}{n_t - 1} = 2^{-\ell} t_{\max}$$

$$t^n = (n-1)\Delta t, \quad n = 1, 2, \ldots, n_t$$

For production runs, however, it is certainly acceptable to fix the set of discrete times by giving, for example, $t_{\max}$ and Δt.

1.2.2 Discretization: Steps 2 and 3—Derivation and Solution of the FDAs

- Continuum equations $\to$ discrete equations
- FD notation

$$\mathbf{r}_i^n \equiv \mathbf{r}_i(t^n)$$

  where we use a superscript, rather than subscript, n, since we are using a superscript to enumerate the particles.

- Need approximation for second time derivative, use usual second order centred formula

$$\left.\frac{d^2\mathbf{r}(t)}{dt^2}\right|_{t=t^n} \approx \frac{\mathbf{r}^{n+1} - 2\mathbf{r}^n + \mathbf{r}^{n-1}}{\Delta t^2}$$

- Substituting in (4), we have

$$\frac{\mathbf{r}_i^{n+1} - 2\mathbf{r}_i^n + \mathbf{r}_i^{n-1}}{\Delta t^2} = \sum_j \frac{m_j}{\left(r_{ij}^n\right)^3}\left(\mathbf{r}_j^n - \mathbf{r}_i^n\right), \quad n+1 = 3, 4, \ldots, n_t \tag{7}$$

- We view this as an equation for the advanced-time values, $\mathbf{r}_i^{n+1}$, assuming that the values $\mathbf{r}_i^n$ and $\mathbf{r}_i^{n-1}$ are known
- We can solve (7) explicitly for $\mathbf{r}_i^{n+1}$, and will leave that to the reader
- As usual for a problem in dynamics, we need to deal with the initial conditions and, since we are using a three-time-level scheme, we thus need to determine values for $\mathbf{r}_i^1 = \mathbf{r}_i(0)$ and $\mathbf{r}_i^2 = \mathbf{r}_i(\Delta t)$
- This can be done in a manner that precisely parallels the analogous calculation for the nonlinear pendulum. This computation will also be left to the reader.

1.3. Energy Quantities and Energy Conservation

- For the gravitational N-body problem we have the following (again with $G = 1$)
- Total kinetic energy

$$T(t) = \sum_{i=1}^{N} \frac{1}{2} m_i \, {v_i}^2 \tag{8}$$

- Total potential energy

$$V(t) = -\sum_{i=1}^{N} \sum_{j=1}^{i-1} \frac{m_i \, m_j}{r_{ij}} \tag{9}$$

- **Important:** Note the the second summation in the above is limited to values of j that are strictly *less* than i

  If we summed over all values of j—i.e. so that the upper limit of the sum was N—we would "double count" the potential energy contributions (think, e.g., of the two-particle case where there is only one contribution)

- Total conserved energy

$$E(t) = T(t) + V(t) \tag{10}$$

- We can compute discrete versions of these quantities, and especially for small numbers of particles, a test for convergence of

$$dE(t) = E(t) - E(0)$$

  is one way of establishing code correctness

1.4. MATLAB Implementation Suggestions

- Use multi-dimensional arrays to store discrete positions
- Ideally, store entire solution (i.e. all time steps) as we did with pendulum example
- For example, create and "zero" 3-dimensional array `r` via

```
r = zeros(N, 3, nt);

   % N:   number of particles
   % nt:  total number of time steps
```

- Then would have the following

$$\begin{aligned}\mathtt{r(i,1,n)} &\equiv x_i^n \\ \mathtt{r(i,2,n)} &\equiv y_i^n \\ \mathtt{r(i,3,n)} &\equiv z_i^n\end{aligned}$$

- Consider writing an acceleration-computing function with a header such as

```
function [a] = nbodyaccn(m, r)

% m:  Vector of length N containing the particle masses
% r:  N x 3 array containing the particle positions
% a:  N x 3 array containing the computed particle accelerations
```

1.5. Suggested test case

- A good, non-trivial configuration that you can use to develop and test your implementation describes two particles with arbitrary masses in mutual circular orbit about their center of mass, and in the x-y plane.

- **EXERCISE:** Let the particle masses be m_1 and m_2, respectively, and let the particles be separated by a distance r. Let the initial position and velocity vectors be

$$\begin{aligned}\mathbf{r}_1(0) &= (r_1, 0, 0)\\ \mathbf{r}_2(0) &= (-r_2, 0, 0)\\ \mathbf{v}_1(0) &= (0, v_1, 0)\\ \mathbf{v}_2(0) &= (0, -v_2, 0)\end{aligned}$$

  where r_1, r_2, v_1 and v_2 are all positive quantities, so that the particle separation is given by $r = r_1 + r_2$. Show that if

$$\begin{aligned}r_1 &= \frac{m_2}{m} r\\ r_2 &= \frac{m_1}{m} r\\ v_1 &= \frac{\sqrt{m_2 r_1}}{r}\\ v_2 &= \frac{\sqrt{m_1 r_2}}{r}\end{aligned}$$

  where $m = m_1 + m_2$ is the total mass of the system, then the particles *will* execute circular orbits about the center of mass. (Once more, recall that $G = 1$.)

- **NOTE:** If you *do* use this configuration to develop/test your code, I expect that you will include the verification (or derivation) of the above results in your writeup.