PHYS 410: Computational Physics Fall 2022
Project 1—Charges on a Sphere (The Thomson Problem)
Due: Wednesday, October 19, 11:59 PM
PLEASE report all bugs, comments, gripes etc. to Matt: `choptuik@physics.ubc.ca`

1. OVERVIEW

Important! *Read this document carefully and completely before you start the project.*

In this project we consider collections of N identical charges which are confined to the surface of a sphere, but which are able to move over that surface under the influence of their mutual electrostatic interactions. The goal of your study will be to determine *equilibrium* configurations of the charges for any given value of N; i.e. configurations in which all electrostatic forces on any charge balance (so that the component of the net force which is tangential to the sphere vanishes for any charge), and which are stable to small perturbations. You will find these equilibria *dynamically*, by direct finite-difference solution of the full equations of motion (EOM). You will add a velocity-dependent retarding force to the EOM so that, with luck, any initial distribution of charges will eventually reach some equilibrium configuration. Your primary interest will be to catalogue these equilibria, and to attempt to characterize their symmetry in a way that will be described below.

2. EQUATIONS OF MOTION

Consider a collection of N identical point charges, with charges q_i and identical masses m_i. Without loss of generality we can set both the masses and charges equal to 1:

$$m_i \equiv 1 \ , \qquad i = 1, 2, \ldots N \, , \tag{1}$$

$$q_i \equiv 1 \ , \qquad i = 1, 2, \ldots N \, . \tag{2}$$

Although it may not seem obvious at first glance (due to the spherical geometry of the setup), it is most convenient to solve the problem in (x, y, z) (Cartesian) coordinates. We set the radius of the sphere to 1, and centre it at the origin, $(0, 0, 0)$.

Associated with any charge q_i is a position vector $\mathbf{r}_i(t)$

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

and because every charge is confined to the surface of the unit-radius sphere we have

$$r_i \equiv |\mathbf{r}_i| \equiv \sqrt{x_i^2 + y_i^2 + z_i^2} = 1 \ , \qquad i = 1, 2, \, \ldots \, N \, .$$

Here and in the following, $|\mathbf{w}|$ denotes the magnitude of a 3-vector:

$$|\mathbf{w}| \equiv \sqrt{w_x^2 + w_y^2 + w_z^2} \, . \tag{3}$$

In computing the electrostatic interaction between the charges we will need separation vectors $\mathbf{r}_{ij}$ given by

$$\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i \, ,$$

with magnitudes

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

and associated unit vectors, $\hat{\mathbf{r}}_{ij}$:

$$\hat{\mathbf{r}}_{ij} \equiv \frac{\mathbf{r}_j - \mathbf{r}_i}{r_{ij}} = \frac{\mathbf{r}_{ij}}{r_{ij}} \, . \tag{4}$$

We can now write down the equations of motion for the charges. We have

$$m_i \mathbf{a}_i = m_i \frac{d^2 \mathbf{r}_i}{dt^2} = -k_e \sum_{j=1, j \neq i}^{N} \frac{q_i q_j}{{r_{ij}}^2} \hat{\mathbf{r}}_{ij} - \gamma \mathbf{v}_i \ , \qquad i = 1, 2, \ldots N \ , \qquad 0 \leq t \leq t_{\max} \, ,$$

where $\mathbf{a}_i \equiv d^2\mathbf{r}_i/dt^2$ and $\mathbf{v}_i \equiv d\mathbf{r}_i/dt$ are the acceleration and velocity of the i-th particle, respectively, k_e is the Coulomb constant, γ is an adjustable parameter that controls the magnitude of the velocity-dependent dissipative force, $-\gamma\mathbf{v}_i$, and $t_{\max}$ is the final time of the evolution. Note that the sum in this equation accounts for the fact that every charge experiences an electrostatic force from every other charge, so that there are a total of $O(N^2)$ pairwise forces. Using (1) and (2) and also setting $k_e = 1$, the above becomes

$$\frac{d^2\mathbf{r}_i}{dt^2} = -\sum_{j=1,j\neq i}^{N} \frac{\hat{\mathbf{r}}_{ij}}{{r_{ij}}^2} - \gamma\frac{d\mathbf{r}_i}{dt}\,, \qquad i = 1, 2, \ldots N\,, \qquad 0 \le t \le t_{\max}\,.$$

For calculational purposes, it is convenient to use (4) in the above to get

$$\frac{d^2\mathbf{r}_i}{dt^2} = -\sum_{j=1,j\neq i}^{N} \frac{\mathbf{r}_{ij}}{{r_{ij}}^3} - \gamma\frac{d\mathbf{r}_i}{dt}\,, \qquad i = 1, 2, \ldots N\,, \qquad 0 \le t \le t_{\max}\,. \tag{5}$$

These are our desired equations of motion that we will solve using FDAs.

A useful diagnostic quantity for our simulation of charges on a sphere is the total potential energy, $V(t)$, for the collection of charges. Given (2) and our choice $k_e = 1$, this is given by

$$V(t) = \sum_{i=2}^{N}\sum_{j=1}^{i-1}\frac{1}{r_{ij}}\,. \tag{6}$$

For a stable equilibrium solution, $\lim_{t\to\infty} V(t)$ will be a minimum. This minimum may be either local or global depending on the number of charges and the initial conditions.

3. FINITE DIFFERENCING

To solve (5) via finite differencing we replace the continuum values of t, $0 \le t \le t_{\max}$, with a temporal mesh that we will define using an integer-valued level parameter, ℓ. Specifically, we take

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

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

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

and introduce the usual finite difference notation for a grid function, $\mathbf{r}_i^n$:

$$\mathbf{r}_i^n \equiv \mathbf{r}_i\left(t^n\right)\,.$$

We then discretize (5) by replacing the first and second time derivatives that appear with our usual $O(\Delta t^2)$ accurate FDAs:

$$\left.\frac{d\mathbf{r}_i}{dt}\right|_{t=t^n} \to \frac{\mathbf{r}_i^{n+1} - \mathbf{r}_i^{n-1}}{2\Delta t}\,,$$

$$\left.\frac{d^2\mathbf{r}_i}{dt^2}\right|_{t=t^n} \to \frac{\mathbf{r}_i^{n+1} - 2\mathbf{r}_i^{n} + \mathbf{r}_i^{n-1}}{\Delta t^2}\,.$$

Important: Here and below it is to be understood that definitions, equations etc. with a "free" subscript i hold for all charges, i.e. all values of i (so that the range specification $i = 1, 2, \ldots, N$ is suppressed).

Substituting these last expressions in (5), we can then solve for the advanced-time value, $\mathbf{r}_i^{n+1}$ explicitly. This I will leave for you to do. Then, provided that we know $\mathbf{r}_i^1$ and $\mathbf{r}_i^2$—and these will come from the initial conditions—we can use the result that you get to determine the values $\mathbf{r}_i^n$, $n = 3, 4, \ldots n_t$.

3.1 Normalization of positions

For all time steps we will require that the positions of the charges are properly normalized so that all charges are on the surface of the unit sphere. After application of the discrete version of (5) at any time step you will find that the charges are *not* on the sphere (there's nothing in the equation of motion that constrains them to remain on the surface). One straightforward way of dealing with this issue is to project the charges along the radial lines that connect their positions with the origin of the coordinate system. Specifically, let the provisional position of the i-th charge after a finite difference update be denoted $\tilde{\mathbf{r}}_i^{n+1}$:

$$\tilde{\mathbf{r}}_i^{n+1} = \left[\tilde{x}_i^{n+1}, \tilde{y}_i^{n+1}, \tilde{z}_i^{n+1}\right] .$$

We then project the charge by taking

$$\tilde{\mathbf{r}}_i^{n+1} \to \frac{\tilde{\mathbf{r}}_i^{n+1}}{|\tilde{\mathbf{r}}_i^{n+1}|} = \frac{\left[\tilde{x}_i^{n+1}, \tilde{y}_i^{n+1}, \tilde{z}^{n+1}\right]}{\sqrt{\left(\tilde{x}_i^{n+1}\right)^2 + \left(\tilde{y}_i^{n+1}\right)^2 + \left(\tilde{z}_i^{n+1}\right)^2}} . \tag{7}$$

3.2 Initial data

Given the goal of determining an (approximate) equilibrium configuration for any given number of charges, a natural way of assigning initial conditions is to let the positions of the charges be distributed randomly on the surface of the sphere. This can be done in many ways. The recommended approach assigns values of x, y and z drawn randomly from the range $[-1, 1]$, followed by a renormalization (projection) as discussed above. This will set the values r_i^1. The values r_i^2 can be initialized using $r_i^2 = r_i^1$, which has the interpretation that the velocities of the charges vanish at the initial time. In the convergence test discussed below you will use a convergence analysis to deduce the actual accuracy of this procedure.

4. EQUIVALENCE CLASSES OF CHARGES

We can partially characterize the symmetry of the equilibrium configurations that we generate by considering the pairwise distances between charges at the final time, t^{n_t}. Here, for simplicity of notation let $\mathbf{r}_i$ denote the final positions of the charges and then define

$$d_{ij} = |\mathbf{r}_j - \mathbf{r}_i| \qquad i, j = 1, 2, \ldots, N ,$$

where you are reminded that the magnitude, $|\mathbf{w}|$ of any 3-vector $\mathbf{w}$ is given by (3). In MATLAB it is natural to store the values d_{ij} as a two-dimensional array (matrix).

Now define $\bar{d}_i$ $i = 1, 2, \ldots, N$ to be the values of d_{ij} sorted in ascending order: that is, for any i—or row of the matrix storing d_{ij}—$\bar{d}_i$ is the vector of values resulting from sorting that row in ascending order. The sort is readily accomplished using MATLAB's `sort` function.

We will say that two charges, labelled by i and i', respectively, are in the same equivalence class if

$$\bar{d}_i = \bar{d}_{i'}$$

where the equality is to hold element-wise and for all elements of the two vectors of sorted lengths. In practice we will only require equality up to a tolerance ϵ_{ec}, i.e. so that

$$|\bar{d}_i - \bar{d}_{i'}| \le \epsilon_{\text{ec}} ,$$

where again the inequality is to be understood to apply to all elements of the vector of distance differences, $\bar{d}_i - \bar{d}_{i'}$.

We will assert without proof that if two charges are in the same equivalence class then they are indistinguishable in the equilibrium configuration.

Your solution of the project must include code that analyzes d_{ij} to determine a vector v_{ec} that defines the number of charges in each equivalence class that you identify. For example, for the case $N = 13$ you might find

$$v_{\text{ec}} = 4 \quad 2 \quad 2 \quad 2 \quad 2 \quad 1 ,$$

which means there are 4 charges in the first class, 2 charges in each of the second through fifth classes and 1 charge in the sixth class. The ordering of the counts is arbitrary, but I have chosen to sort the values of $v_{\rm ec}$ in descending order. You should do the same.

Important: *Writing the code to determine the equivalence classes may be the most challenging part of the project for you. If you find yourself unduly stuck, don't be afraid to ask for hints.*

5. IMPLEMENTATION REQUIREMENTS, SUGGESTIONS AND NOTES

5.1 Implementation/representation of $\mathbf{r}_i^n$ (requirement)

Implement the positions of the charges as a three dimensional array, whose dimensions can be defined by the following initialization statement:

```
r = zeros(nc, 3, nt)
```

where `nc` is the number of charges (i.e. N) and `nt` is the number of timesteps. Then we have the following identifications:

$$\texttt{r(i, 1, n)} \equiv x_i^n\,,$$
$$\texttt{r(i, 2, n)} \equiv y_i^n\,,$$
$$\texttt{r(i, 3, n)} \equiv z_i^n\,,$$

and

$$\texttt{r(i, :, n)} \equiv \mathbf{r}_i^n\,.$$

Note that in defining `r` in this fashion the implication is that we store the entire dynamical evolution of the charges.

5.2 Top-level function for simulation (requirement)

For your simulation of the charge dynamics, code a top level function with header

```
function [t, r, v, v_ec] = charges(r0, tmax, level, gamma, epsec)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  charges: Top-level function for solution of charges-on-a-sphere
%  problem.
%
%  Input arguments
%
%  r0:      Initial positions (nc x 3 array, where nc is the number of
%           charges)
%  tmax:    Maximum simulation time
%  level:   Discretization level
%  gamma:   Dissipation coefficient
%  epsec:   Tolerance for equivalence class analysis
%
%  Output arguments
%
%  t:       Vector of simulation times (length nt row vector)
%  r:       Positions of charges (nc x 3 x nt array)
%  v:       Potential vector (length nt row vector)
%  v_ec:    Equivalence class counts (row vector with length determined
%           by equivalence class analysis)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Note that the initial conditions are to be supplied as an input, and that the number of charges does not explicitly appear in the input arguments since it can be determined from the size of `r0`.

In addition to performing the main simulation, which defines the array `r`, this function is also responsible for computing and storing the potential at each time step (the computation can be done in a separate function if you wish), and for analyzing the charge positions at the final time to determine the equivalence class counts (this can also be done in a separate function).

5.3 Developing your code

Begin by having `charges` focus exclusively on the task of simulating the charge dynamics, including the computation of the potential. Use simulation parameters for a very brief integration of a small number of charges, taking, for example, `nc = 2`, `tmax = 5`, `level = 3` and `gamma = 1`. Use the instructor-supplied function `charges_plot.m`, available from the course Homework page, or plotting facilities of your own design, to visualize the results of the simulation. (`charges_plot` should be called at the end of the simulation).

Include tracing statements in your code (use `printf`) to periodically output the value of the potential.

Remove and restore semi-colons at the end of statements as necessary to trace the flow of your computation.

Once it appears that your simulation is reasonable, compute with `tmax = 100` and `level = 8` until the final configuration has the two charges located directly opposite one another on the sphere, and the final value of V is 0.5.

At this point you can start (gradually) increasing the number of charges, as well as increasing the final time and discretization level. In all instances aim to have V equilibrate by the end of the simulation.

Once you are confident that your simulations are working correctly (including the convergence test described below), implement the equivalence class analysis. I suggest that you use `epsec =` 10^{-5}, and note that the analysis will be quite sensitive to how close your final positions are to a true equilibrium. You should find that for some values of N the equilibrium is quite easy to determine (in terms of `tmax`), while for others it is more difficult. You will not be penalized if your analysis does not agree with the key for particularly tricky cases.

5.4 Initializing charge positions randomly

Since the charges lie on a unit-radius, origin-centred sphere, their x_i, y_i and z_i coordinates satisfy

$$-1 \leq x_i,\ y_i,\ z_i \leq 1\,.$$

To randomly initialize coordinates in the interval $[-1, 1]$ the following code can be used

```
r0 = 2*rand(nc,3) - 1;
```

The coordinates will then need to be normalized using (7).

5.5 A useful Wikipedia page

If you google "Thomson problem" you will find a Wikipedia page that will certainly be of use to you. In particular, in the section "Configurations of smallest known energy" is a table that lists the value of V (E_1 in the page's notation) for various values of N. Ideally, you will make minimal reference to this page until you have successfully implemented your algorithm but, of course, I cannot stop you from "reading ahead"!

Note that, for certain values of N, you may find values of $V(t_{\max})$ which are in disagreement with the table in the Wikipedia page. Assuming that your simulation is working correctly this will likely be due to the fact that for those values of N there is more than one equilibrium configuration and your code has found *local* minima in V rather than the *global* minima quoted in the Wikipedia table. This is fine and not something you need to be concerned with. However, if you have the time and inclination, you might want to investigate those values of N in more detail, using a number of different random initial conditions in an attempt to find the global minima.

5.6 Simulation runtime

For

```
nc = 60
tmax = 500
level = 12
```

my implementation of `charges` takes about 3 seconds on my home PC (a relatively new model). You should expect comparable runtimes up to a factor of a few or so. If you find that a run with these parameters takes, say, of order a minute or greater, there are probably one or more inefficiencies that you need to weed out.

6. CALCULATIONS TO PERFORM

This section describes the minimum set of calculations that you are to perform and describe in your writeup. You are free (and encouraged) to go beyond the minimum set and your successful efforts in this regard will be recognized in the form of a certain number of bonus marks that will applied against any shortcomings in the basic part of your implementation, results and writeup (but the total mark for the project will be capped at 100%).

6.1 4-level convergence test

Code a script source file, `convtest.m`, that performs a sequence of 4 calculations with

```
nc = 4
tmax = 10
gamma = 1
epsec = 1.0e-5
```

with initial conditions (non-random) given by

```
r0 = [ [1, 0, 0]; [0, 1, 0]; [0, 0, 1]; (sqrt(3)/3) * [1, 1, 1] ]
```

and with discretization levels

```
level = 10, 11, 12, 13
```

Let the level-ℓ values of the x-coordinate of the first charge be denoted, x_l. x_l is a vector of length n_{t_l} where n_{t_l} is the number of time steps in the level-ℓ calculation.

Now consider the level-to-level differences

$$\delta x_{10} = x_{11} - x_{10}\,,$$
$$\delta x_{11} = x_{12} - x_{11}\,,$$
$$\delta x_{12} = x_{13} - x_{12}\,,$$

where it is to be understood that the subtraction involves that set of the finer grid function values (higher level) that coincides with the set of coarser grid function times (i.e. every second element of the higher level grid function).

If we graph on a single plot

$$\delta x_{10}\,, \quad \rho\,\delta x_{11}\,, \quad \rho^2 \delta x_{12}$$

as a function of t, and where ρ is either 2 or 4, then we should see near-coincidence of the curves for:

1. $\rho = 2$ if our FDA is first-order accurate,
2. $\rho = 4$ if our FDA is second-order accurate.

Make plots for both $\rho = 2$ and $\rho = 4$ and include them in your writeup. Discuss what the plots tell you about the convergence of your FDA and, to the best of your ability, provide some rationale for what you see. The plotting should be done within the script file, `convtest.m`.

6.2 Time evolution of potential for 12-charge calculation

Code a script source file, `plotv.m`, that performs a simulation with

```
nc = 12
tmax = 10
level = 12
gamma = 1
epsec = 1.0e-5
```

and random initial positions.

The script file should make a plot of $V(t)$ vs t: include the plot in your writeup.

6.3 Survey of $V(t_{\max}; N)$ and $v_{\text{ec}}(N)$ for various values of N

Only complete this part of the project when you are confident that your simulations are working properly and with reasonable efficiency.

Code a script source file, `survey.m`*, that performs a survey for $N = 2, 3, \ldots, 59, 60$. Include tables of $V(t_{\max}; N)$ and $v_{\text{ec}}(N)$ in your writeup.*

The script should make the tables—two columns of values for $V(t_{\max}; N)$ (N and V) and varying numbers of columns of values for $v_{\text{ec}}(N)$ (N and v_{ec})—as simple text files directly from your MATLAB computations (use `fopen`, `fprintf` and `fclose`) and include them verbatim in your report (i.e. the formatting doesn't have to be anything fancy).

Specifically, and to facilitate grading, ensure that your survey results are written to text files named `vsurvey.dat` and `ecsurvey.dat` for the final potential values and equivalence class counts, respectively. For the case of `vsurvey.dat`, use a `fprintf` statement with a format specification precisely as follows:

```
fprintf(fid_v, '%3d %16.10f\n', nc, v(end));
```

Here `nc` is the number of charges and `v(end)` is the final value of the potential.

This formatting should display $V(t_{\max}; N)$ to 10 digits following the decimal place.

For `ecsurvey.dat`, use the following sequence of `fprintf` statements:

```
fprintf(fid_ec, '%3d ', nc);
fprintf(fid_ec, '%d ', v_ec);
fprintf(fid_ec, '\n');
```

Here `v_ec` is the vector of equivalence class counts. Note that there is a space before the final quote in the first two `fprintf` statements.

It is up to you to determine appropriate values for the arguments to `charges` in order to produce an accurate survey (but keep in mind that you won't be penalized for incorrect $v_{\text{ec}}(N)$ for "troublesome" values of N).

Write a few sentences about what trends you see in $v_{\text{ec}}(N)$. Also briefly discuss the extent to which your results for $V(t_{\max}; N)$ do or don't agree with those listed on the Wikipedia page.

6.4 Video of sample evolution

Make an AVI (or MPEG-4 if you have the codec installed) video of a simulation of your choice. You can use the instructor-supplied function `charges_video.m` for this purpose. Note that 15 frames (time steps) will produce 1 second of video so adjust the number of time steps in your simulation to generate a video of reasonable length (say 1 minute or less). Feel free to customize `charges_video.m` as you wish.