CPSC 533D Assignment 1 *Time Integration: 15%* **Due January 31, 2005**

1 Programming

In this assignment you will animate a set of particles linked by springs.

The main program `compute_springs` will take a configuration text file as input, which includes the following numbers in order:

- T: the final time in seconds (for the simulation to stop at)
- r: the frame rate in frames per second
- K: the spring stiffness constant
- D: the spring damping constant
- L: the spring rest length
- M: the mass of each particle
- n: the number of springs (a positive integer)
- x_0, y_0, z_0: the initial position of particle 0
- u_0, v_0, w_0: the initial velocity of particle 0
- ...
- x_n, y_n, z_n: the same for particle n
- u_n, v_n, w_n

The particles will be linked together with springs in a chain (i.e. a spring between particle 0 and 1, then a spring between particle 1 and 2, ..., and a spring between particles $n-1$ and n). After reading this file in, the main program will compute the simulation with a variety of different time integrators: Forward Euler, Staggered Symplectic Euler, Backward Euler, and the Implicit Compromise method from the lectures. For these last two implicit methods, you can implement them semi-implicitly, i.e. with just one linear solve instead of doing a full Newton iteration. The time step for each method will be equal to the frame time. The program will output the positions of the particles at each frame in a text file (one file for each integrator).

I have provided some code to start you off, in the directory `compute_springs`. See the README for instructions on what you need to add, how to compile, etc.

There is also a directory `opengl_springs`, containing a simple program to visualize the output from the simulations. See the README in that directory for instructions on how to use it. You do not need to add any code here: this is purely for your convenience in debugging and analyzing what's going on.

Finally, there is a directory `renderman_springs`, for a program which will produce high quality output for your animations. You will need to learn about RenderMan to complete this part of the assignment—see the course web-page for some resources to start learning about this. See the README as always for more detailed instructions.

2 Analysis

(1) Comment on the results of your simulations, for a variety of different K's and D's.

(2) Prove that RK4 applied to $dx/dt = -x$ is never oscillatory no matter how large the time step is (even if it is going unstable).

(3) Consider a simple spring system (in one dimension), i.e. the test equation for position-dependent forces:

$$\begin{aligned} \frac{dx}{dt} &= v \\ \frac{dv}{dt} &= -Kx \end{aligned}$$

For the initial condition $x = 1$, $v = 0$, what is the exact solution? What does Trapezoidal Rule calculate? For time steps much larger than $1/\sqrt{K}$ do you think this would look reasonable in an animation?

3 Handing It In

You need to hand in the code you write for `compute_springs` and `renderman_springs`, an animation you generate from your code, and answers to the questions in the analysis section.

To hand in the animation you generate for an assignment, put it on the web and email me the URL.

To hand in the programming part of an assignment, tar and gzip the directory containing your source code (but not output data, object files or executables!). If you have changed how the code is compiled or run, or sone anything you want to comment on, include a README file. Then email me the .tar.gz file as an attachment.

To hand in the written part of an assignment, either email it to me (preferably in plain text or PDF), give it to me in person, or slide it under my office door.

I understand it is sometimes difficult to turn in assignments on time due to unforeseen emergencies. If you have a good reason you can't turn something in on time, please contact me as soon as possible about it and we can work something out. Otherwise, there will be a 20% per day late penalty, starting when I come to my office in the morning after the due date.