Physics 750: Exercise 5

Tuesday, September 12, 2017

1. Use the `curl` command to download from the class website everything you'll need for this exercise.

```
$ WEBPATH=http://www.phy.olemiss.edu/~kbeach/
$ curl $WEBPATH/courses/fall2017/phys750/src/exercise5.tgz -O
$ tar xzf exercise5.tgz
$ cd exercise5
$ make
g++ -o integrator integrator.cpp -O2 -lm
g++ -o freefall freefall.cpp -O2 -lm
g++ -o oscillator oscillator.cpp -O2 -lm
g++ -o pendula pendula.cpp -O2 -lm
```

2. The program `integrator` computes the integral

$$I = \int_0^{5/2} dx\, \cos^2(\pi x) x e^x = \frac{e^{5/2}\pi^2(12\pi^2 - 1) + 8\pi^4 + 2\pi^2 + 1}{(1 + 4\pi^2)^2}$$

using the functions `trapezoidIntegrator` and `SimpsonIntegrator`. The exact value of the integral is $I \doteq 9.1058789328730354374605603731577$.

(a) Have a look at the integrand.

```
$ gnuplot
gnuplot> plot[0:2.5] x*exp(x)*cos(pi*x)**2
```

(b) Familiarize yourself with the source code for `integrator`. Observe that

```
double (&f)(double)
```

is the prototype for a function argument, where `f` is any function that that accepts a double and returns a double. What role does the ampersand play here?

(c) Plot $|I_{\text{trap}} - I|$ and $|I_{\text{Simp}} - I|$ versus $1/N$ on a log-log scale. Verify that the first scheme has error $O(N^{-2})$ and the second $O(N^{-4})$. (Hint: try typing `help logscale` and `help abs` from within `gnuplot`.)

(d) Change the integrand to $\cos^2(\pi x) x e^x \log x$. Why does the program fail?

(e) Devise a work around to the problem in part (d). See how close you can get to the exact value of the integral, 5.4374112923165359648655111461763. See if you can achieve better accuracy by changing the working type to `long double`.

*(f) The trapezoid and Simpson's rules are given by

$$\int_a^b dx\, f(x) \approx \frac{b-a}{2}(f(a) + f(b))$$

and

$$\int_a^b dx\, f(x) \approx \frac{b-a}{6}\Big(f(a) + 4f\Big(\frac{a+b}{2}\Big) + f(b)\Big)$$

The first is based on a linear fit to the integrand evaluated at points a and b, and the second a quadratic fit at points a, $(a+b)/2$, and b. The corresponding cubic fit leads to *Boole's rule*, which is given by the following five-point formula.

$$\int_a^b dx\, f(x) \approx \frac{b-a}{90}\Big(7f(a) + 32f\Big(\frac{3a+b}{4}\Big) + 12f\Big(\frac{a+b}{2}\Big) + 32f\Big(\frac{a+3b}{4}\Big) + 7f(b)\Big)$$

If the range of integration is broken into N uniform slices, then the relative weight at each mesh point is given by

$$
\begin{array}{cccccccc}
1 & 1 & & & & & & \\
 & 1 & 1 & & & & & \\
 & & 1 & 1 & & & & \\
 & & & & \ddots & & & \\
 & & & & & 1 & 1 & \\
 & & & & & & 1 & 1 \\
\hline
1 & 2 & 2 & 2 & 2 & \cdots & 2 & 1
\end{array}
$$

in the trapezoid case and

$$
\begin{array}{ccccccccccccc}
1 & 4 & 1 & & & & & & & & & & \\
 & & 1 & 4 & 1 & & & & & & & & \\
 & & & & 1 & 4 & 1 & & & & & & \\
 & & & & & & & \ddots & & & & & \\
 & & & & & & & & 1 & 4 & 1 & & \\
 & & & & & & & & & & 1 & 4 & 1 \\
\hline
1 & 4 & 2 & 4 & 2 & 4 & 2 & 4 & 2 & \cdots & 2 & 4 & 1
\end{array}
$$

in the Simpson case.

Generalize the weights to the third order case and write a function `BooleIntegrator`. What requirement is there on the value of N? Check your results against the other integrators. Show that the error scales as $O(1/N^6)$.

**(g) Devise and implement a nonuniform mesh scheme to compute the integral

$$\int_{-\infty}^{\infty} \frac{dx}{1+x^2} = \pi \doteq 3.14159265358979323846264338327 95$$

Hint: You want to construct the mesh that is finest around $x = 0$ and becomes progressively coarser out near the tails.

3. The file `particle.hpp` contains the specification for a class `particle` that encapsulates the mass, position, velocity, and time data for a single particle. The class also provides a method `evolve`, which advances the particle in time according to a given model of acceleration and a given finite-difference integrator method.

   The programs `freefall` and `oscillator` use the `particle` class to simulate a particle falling under gravity and a particle oscillating under a linear restoring force. In both cases, the `Euler` scheme is used to update each time step.

   (a) Read over the definition of the `particle` class and make sure you understand its structure and the C++ syntax. There are some rather subtle points to take note of.

   - Class arguments are generally passed by reference (indicated by an ampersand) rather than by value. The arguments `(particle &p)` and `(const particle &p)` differ in that the former allows for changes to `p` within the function. The `const` keyword prevents changes.
   - `*this` is how a class object refers to itself.
   - A `friend` function is one that is not a class member but which is affiliated with a class and thus has access to its data. `operator<<` is a friend function that defines how `particle` objects interact with output streams via the << operator. In other words, the definition makes `particle p; cout << p;` a legal statement.

(b) The `freefall` program simulates a point particle falling for 100 s under gravity (acceleration $g = 9.80665\,\text{m/s}^2$) from a height of 50 km. Run the program and redirect its output to a file. Plot the position as a function of time and overlay the exact analytical expression. Quantify the accuracy of the numerical results for $dt = 0.01$. Is the accuracy still good for $dt = 0.1$?

(c) The `oscillator` program simulates a harmonic oscillator whose true behaviour is characterized by $x = 3\cos(t)$ and energy $E = \frac{1}{2}mv^2 + \frac{1}{2}Kx^2 = 9$. Run the provided `gnuplot` script:

```
$ gnuplot -persist oscillator.gp
Simulating harmonic oscillator with time step dt = 0.1
Simulating harmonic oscillator with time step dt = 0.05
Simulating harmonic oscillator with time step dt = 0.025
Simulating harmonic oscillator with time step dt = 0.0125
Press ENTER to continue
Press ENTER to continue
```

Three plots will be shown. Deduce their meaning by reading over `oscillator.gp`. Comment on the quality of the numerical results. Why isn't energy conserved?

(d) Using `Euler` as a template, write functions `EulerCromer` and `Verlet` that implement those update schemes. (Use the self-starting version of the Verlet.) Apply these to the harmonic oscillator.

4. The program `pendula` simulates a collection of particles obeying Hooke's Law, subject to a variety of inital conditions.

(a) Swap out the `HookesLaw` function for one named `pendulum` that implements the correct nonlinear restoring force felt by a simple pendulum of length L (with $g/L = 1$).

(b) Simulate 150 pendula simultaneously and observe their evolution in the p-x phase space. A `gnuplot` script is provided.

```
$ ./pendula 150
$ gnuplot
gnuplot> count=0
gnuplot> load 'phase_space.gp'
```

The simple pendulum is a Hamiltonian system. What can you say qualitatively about the symplectic symmetry (conservation of the enclosed phase space area)?

*(c) Modify the program so that it outputs in two-column format the current time and the enclosed phase space area. The latter should be calculated by summing the wedges $dA = \frac{1}{2}R^2\,d\theta$. Plot the area versus time.

(d) Switch out the `Euler` update method for the `Verlet` integrator (or `RungaKutta`, if you are particularly ambitious). Is the symplectic symmetry better respected?