Lecture III : Realistic motion in one dimension

I. GUIDELINES TO WRITING GOOD CODE

- Your code should be readable by others. For this,
  - Write plenty of comments.
  - Choose variable names that are self-explanatory.
  - Use subroutines instead of a large piece of code.
  - Sacrifice compactness for clarity. Don't avoid extra variables and lines of code if it makes it more readable.
- Don't begin to write code immediately after you are faced with a problem. Make an outline thinking about what variables you'll need and what algorithm you will use. If necessary, write on a piece of paper a *pseudocode*.
- Most of the time, you will find yourself drawing graphs of the data that your program outputs. Format your output in a way you can use with your preferred plotting program.
- Test your program throroughly using well-known limits, special cases or when available with exact results.
- Have a good understanding of the approximations you make. Do a thorough error analysis.

II. CYCLIST

A. No air drag

If you have ever ridden a bicycle, you will know that air resistance and wind play a very important role in determining your speed. In the absence of friction (air or contact friction) and wind, determining the speed of a cyclist with a constant power output is very straightforward and can be done analytically. The equation that we need to solve is simply Newton's second law :

$$a = \frac{F}{m} \qquad \Rightarrow \qquad \frac{dv}{dt} = \frac{F}{m} \tag{1}$$

Let's assume that the power, P, generated by the cyclist is constant in time. The definition of power is the energy output per time, namely,

$$\frac{dE}{dt} = P \tag{2}$$

and because we are ignoring friction and assuming we are on a flat road, the power output of the cyclist is completely converted into propelling forward the bicycle and himself, i.e. into kinetic energy only,

$$\frac{dE}{dt} = \frac{\left(\frac{1}{2}Mv^2\right)}{dt} = Mv\frac{dv}{dt} = P. \tag{3}$$

where $M = m_b + m_c$ is the total mass of the bicycle, m_b and the cyclist, m_c. Substituting Eq. 3 into Eq. 1, we obtain, for the time evolution of the velocity,

$$\frac{dv}{dt} = \frac{P}{Mv}. \tag{4}$$

If the initial velocity of the cyclist is v_0, this equation can be easily solved analytically to assume the form,

$$v(t) = \sqrt{\frac{2Pt}{M} + v_0^2}. \tag{5}$$

As expected, without friction, the cyclist will accelerate indefinitely without reaching a limiting velocity.

Although the analytical solution of this problem is trivial, let's now discuss how we would handle it computationally. That is to say, given the problem parameters such as mass , power and the initial velocity, how would we determine computationally the velocity of the cyclist at a given time, t?

Numerical computation usually deals with discrete variables rather than continuous ones. Because we cannot treat the velocity computationally as a continuous variable, we evaluate it at discrete time intervals, $\Delta t, 2\Delta t, 3\Delta t, \cdots$. The smaller we make Δt, the closer we are to the real solution.

In order to express the time derivative of velocity in terms of Δt, we remember the definition of the derivative from elementary calculus classes

$$\frac{dv}{dt} = \lim_{\Delta t \to 0} \frac{v(t+\Delta t) - v(t)}{\Delta t}. \tag{6}$$

Notice that the derivative is essentially the difference between the velocity at the next time step and the present velocity. This difference should converge to the exact derivative at the limit of very small Δt. Substituting this in Eq. 18, we obtain

$$\frac{v(t+\Delta t) - v(t)}{\Delta t} = \frac{P}{Mv(t)}. \tag{7}$$

Eq. 7 is *iterative*, meaning that given the velocity at any time, the velocity at the next time step can be obtained. This then gives us a way to iteratively calculate the velocity for all times given the initial velocity. For ease of translation into a programming language, we can denote each time step with an index, for example, n,

$$\frac{v_{n+1} - v_n}{\Delta t} = \frac{P}{Mv_n} \tag{8}$$

where $v_n = v(t_n)$ and $\Delta t = t_{n+1} - t_n$. Isolating v_{n+1} in Eq. 8 gives

$$v_{n+1} = \frac{P}{Mv_n}\Delta t + v_n \tag{9}$$

Let's now think of an outline of a program to do the iterations. In lab, we are going to learn how to cast this equation in the form of an `Octave` script.

Outline :

- Choose constants for the problem. Make sure you stick to a consistent set of units be it SI, a.u. or cgs. Let's think of a reasonable set of constants :

Parameter	`Octave symbol`	Value	Unit
Constant power output of biker	`P`	400	W
Mass of a race bike	`mb`	15	kg
Mass of the cyclist	`mc`	55	kg
Initial velocity	`v0`	4	m/sec
Time increment	`dt`	0.1	sec

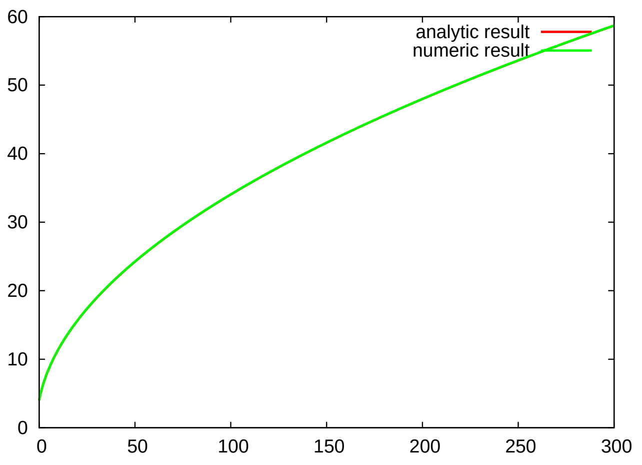
- Choose a suitable Δt.
- Start with the initial velocity, v_0.
- Substitute v_n into Eq. 9 to get v_{n+1}.
- Collect all the velocities and write to file.

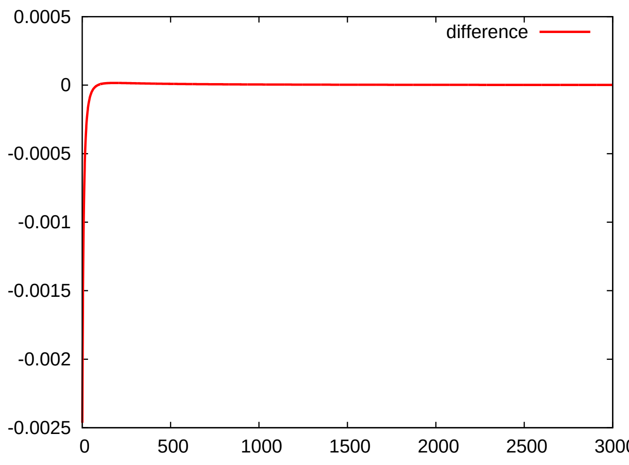
Let's now cast this into `Octave` code :

```
hande@439a:~$ cat bicycle_nodrag.m

## Velocity profile of a cyclist in the absence of drag

## Define the constants of the problem in the correct units
v0 = 4;             ## initial velocity of the cyclist [m/s]
```

```
P  = 400;           ## power output of the cyclist [Watts]
mc = 55;            ## weight of cyclist [kg]
mb = 15;            ## weight of bike [kg]
m = mb+mc;          ## total weight [kg]
ttotal = 300;       ## total time of simulation [sec]
dt = 0.1;           ## time step [sec]

## Calculate the number of steps that need to be taken to cover a time
## interval of ttotal. Make it an integer by rounding down (or up, it
## doesn't make a difference)
Nsteps = floor(ttotal/dt);

## Initialize the velocity array. At the moment it's a single number but it
## will become an array as we loop
v=v0;

## Iterate the finite difference loop
for n=1:Nsteps
  v = [v; P*dt/(m*v(n))+ v(n)];
endfor

## For plotting purposes, define the time axis.
t=[0:Nsteps]*dt;

## We'd like to plot out result against the analytical result
vanalytic = sqrt(2*P*t/m+v0^2);

## Make two plots
##    1) vanalytic and v superimposed
##    2) vanalytic-v for better visualization

figure(1);
plot(t,vanalytic,';analytic result;',t,v,';numeric result;');
figure(2);
plot(diff(vanalytic-v'),';difference;');
```

The resulting plots should look like :





The plot on the left clearly shows that for `dt=0.1`, the analytic and the numeric results are too close to make a distinction by eye at this scale. On the right-hand side the difference is plotted to give us a better feel. You can repeat the calculation for different `dt`. How does the error depend on `dt`?

Needless to say, the velocity profile calculated in this way is only an approximation to the exact solution. Thus, there will always be an error, which stems from the finite difference form of the derivative that we utilize in Eq. 6

$$\begin{aligned} v(t+\Delta t) - v(t) &= v(t) + v'(t)\Delta t + \frac{1}{2}v''(t)(\Delta t)^2 + \frac{1}{3!}v'''(t)(\Delta t)^3 + \cdots - v(t) \\ &= v'(t)\Delta t + \frac{1}{2}v''(t)(\Delta t)^2 + \frac{1}{3!}v'''(t)(\Delta t)^3 + \cdots . \end{aligned} \tag{10}$$

In the language of discrete steps,

$$v'(t_n) \approx \frac{v_{n+1} - v_n}{\Delta t} + \mathcal{O}(\Delta t) \tag{11}$$

t o leading order in Δt. Substituting Eq. 11 in Eq. 8, we have

$$v_{n+1} = \frac{P}{Mv_n}\Delta t + v_n + \mathcal{O}((\Delta t)^2). \tag{12}$$

Thus the lowest-order error that we make is of second order in Δt. The magnitude of the error also depends on the derivatives of the velocity function itself as can be seen from the Taylor expansion in Eq. 10.

B. In the presence of air drag

Simple as it might be, our picture of an ever-accelerating cyclist is a highly idealized one. Even the best-trained cyclist with the most highly-optimized race bike will not be able to stand the air friction forever. This is why bike producers put a lot of effort into designing bikes that help the cyclist assume a position during the race so as to minimize the air drag. Air friction is also the reason why cyclists ride in packs during a race. This way the racers in front "cut through" the air lessening the friction for the ones who are behind them.

The force due to air drag is a highly complicated function of velocity. Although we don't know the exact dependence, one thing we know is that its direction is exactly opposite that of the motion. We approximate it with a few elements of a Taylor series

$$F_{drag} = -B_1 v - B_2 v^2, \tag{13}$$

where B_1 and B_2 are positive coefficients. For very small speeds, we can take the drag force to depend on the velocity. But for any reasonable speed and for larger object (such as a cyclist), we can omit the linear dependence (known as the Stokes term) and take only the quadratic term. As mentioned above, we do not know the exact value of B_2 but an intelligent guess could be made :

- The origin of the air drag is the energy necessary to move the air encountered by a moving object *out of the way.*
- The mass of air displaced as an object moves with velocity v in a very small time interval dt is

  $$m_{air} \approx \rho A v dt \tag{14}$$

  where ρ is the density of air and A is the cross-section area of the object.
- The mass of air thus displaced is given a kinetic energy of

  $$E_{air} \approx \frac{1}{2} m_{air} v^2, \tag{15}$$

  which is provided by the drag force, F_{drag}. In writing this, we assume that the air is accelerated to a velocity of order v in a time interval dt.
- From the definition of work, we have

  $$\begin{aligned} F_{drag} v dt &= -E_{air} \\ F_{drag} v dt &= -\frac{1}{2}\rho A v^3 dt \\ \Rightarrow F_{drag} &= -\frac{1}{2}\rho A v^2 \end{aligned} \tag{16}$$

- In reaching the expression for F_{drag} in Eq. 16, we have made several assumptions including the velocity to which the air was accelerated and the geometry-independence of drag(except the area). Taking into account these assumptions and empirical observations, air drag is proportional to a geometry dependent constant, C.

$$\Rightarrow F_{drag} = -\frac{1}{2} C \rho A v^2 \tag{17}$$

Adding the drag force (only the v^2 term) to the expression in Eq. 18, we obtain

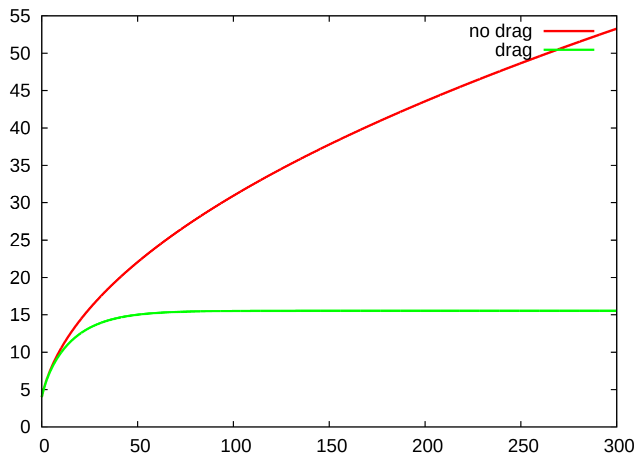
$$\frac{dv}{dt} = \frac{P}{Mv} - \frac{1}{2} \frac{C \rho A v^2}{M}, \tag{18}$$

which is much harder to evaluate analytically than Eq. 18. However, it requires almost no work to add this to our iterative expression in Eq. 9 and solve numerically.

$$v_{n+1} = v_n + \frac{P}{M v_n} \Delta t - \frac{C \rho A v_n^2}{2M} \Delta t \tag{19}$$

We can include the drag term to our code without the drag easily. First, we copy the old program file into a new file, say, `bicycle_drag.m`. We then modify the term that updates the velocity and also add a few constants to the header where we declare the constants.

```
hande@439a:~$ cat bicycle_drag.m

## Velocity profile of a cyclist in the absence of drag

## Define the constants of the problem in the correct units
v0 = 4;              ## initial velocity of the cyclist [m/s]
P  = 400;            ## power output of the cyclist [Watts]
mc = 70;             ## weight of cyclist [kg]
mb = 15;             ## weight of bike [kg]
m = mb+mc;           ## total weight [kg]
ttotal = 300;        ## total time of simulation [sec]
dt = 0.1;            ## time step [sec]
C = 0.5;             ## drag coefficient(unitless)
A = 0.33;            ## frontal area of bike + cyclist [m^2]
rho = 1.29;          ## density of air at sea level [kg/m^3]

## Calculate the number of steps that need to be taken to cover a time
## interval of ttotal. Make it an integer by rounding down (or up, it
## doesn't make a difference)
Nsteps = floor(ttotal/dt);

## Initialize the velocity array. At the moment it's a single number but it
## will become an array as we loop
v=v0;

## Iterate the finite difference loop
for n=1:Nsteps
  v = [v; P*dt/(m*v(n))+ v(n)-0.5*C*A*rho*v(n)*v(n)*dt/m];
endfor

## For plotting purposes, define the time axis.
t=[0:Nsteps]*dt;

## The no-drag velocity profile (analytical)
vanalytic = sqrt(2*P*t/m+v0^2);

## Plot the no-drag case and the drag case together
plot(t,vanalytic,';analytic result;',t,v,';numeric result;');
```



As expected, the drag force prevents the cyclist to accelerate indefinitely. His velocity converges to a limit value of about 15.5 m/sec. You can modify the parameters in the problem to see how much they effect the terminal velocity.