MANCHESTER 1824

The University of Manchester

Cockcroft Institute Postgraduate Lectures
Numerical Methods and Lattice Design

Lecture 1: Introduction to Numerical Methods for Computational Physics

Hywel Owen

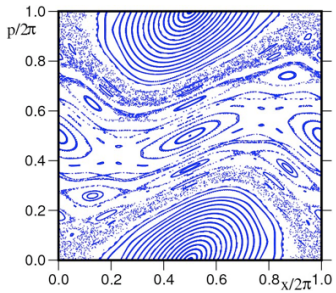
School of Physics and Astronomy, University of Manchester

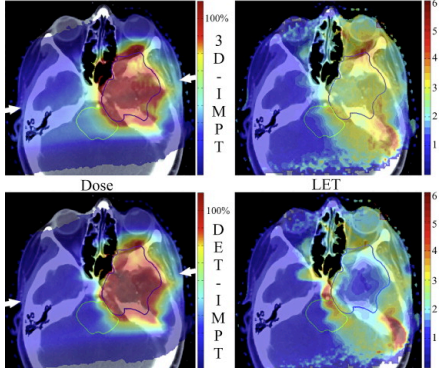


MANCHESTER 1824

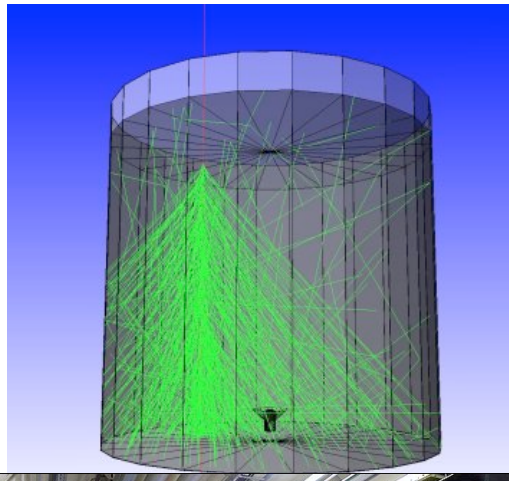
The University of Manchester

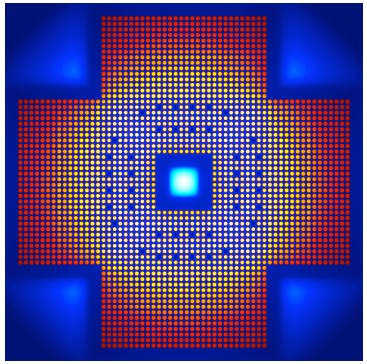
What is this course?





- This is a course on Numerical Methods:
  - The use of algorithms and numerical methods to perform data analysis and simulation in support of experimental and theoretical physics
- There are 3 pillars of physics:
  - Theory
  - Experiment
  - Simulation
- We will concentrate obviously on particle accelerators





Course Information

- The course website:
  - http://theory.physics.manchester.ac.uk/~hywel/teaching/cockcroft/
- People:
  - Hywel Owen (hywel.owen@manchester.ac.uk)
  - Bruno Muratori (bruno.muratori@stfc.ac.uk)
- Course structure:
  - 6 lectures: Sep 1/Sep 22/Sep 24
  - 3 projects: Oct 29-31

- You will need to do some background reading and work before the projects

- This course is deliberately a smorgasbord, with some later focus on practical issues in lattice design
  - The idea is to expose you to the important ideas to keep track of
- We assume you will do some other study/course on linear optics (but I will recap!)

The Course Syllabus and Projects

- Recap on programming languages for physics; MATLAB and Python; summary of commands;
- Introduction to numerical computing; errors in computer calculations;
- Numerical integration methods; Euler's method; higher-order methods;
- Precision vs. accuracy; validation;
- Phase space; conserved quantities;
- Introduction to mappings and nonlinear systems;
- *Example: Methods for solving the linear and non-linear simple harmonic oscillator.*
- Introduction to Monte Carlo methods; Monte Carlo integration; classical problems;
- Pseudorandom and quasirandom sampling; methods of sampling; generation of distributions;
- Particle transport simulation; nuclear cross sections; particle histories; applications of Monte Carlo transport;
- *Example: Simulation of penetration of neutrons through shielding.*
- From mappings to linear optics; the concept of lattices;
- Transfer matrices and periodic solutions; propagation of linear optics parameters;
- Classic optical systems: the FODO, the double-bend achromat;
- Matching and optimisation; penalty/objective functions;
- Hill-climbing methods: Cauchy's method, Nelder-Mead, simulated annealing;
- Variables and constraints; under- and over-constrained problems;
- *Example: MAD8 matching of FODO Twiss values;*
- Multiple-configuration methods; genetic algorithms and evolutionary algorithms;
- A bestiary of codes; choosing the right code;
- Common pitfalls;
- *Example: Particle tracking in MAD8;*

Advice on Programming

- The language you use does not matter, per se
  - Numerical algorithms and methods are translatable (for the most part) from one language to another
- However, you will want to use the most convenient language
  - Pre-existing knowledge
  - Compatibility with colleagues
  - Availability of pre-written code (libraries, tools, programs)
- Typically, this means the following languages will be encountered/used:
  - C++, Python, MATLAB
  - Older codes will expose you to Fortran
  - Locally quite a few people use Mathematica
  - More on this later...

  - I will show examples using MATLAB, but if you don't have it use something like Python (Anaconda or Canopy packages are recommended)

Reminders: Variables in MATLAB

- (Reminder: A variable is a unit of data with a name, which is available to the program)

- Simple variables
  - `comment = 'This is a string'; % Can put comments after the definition`
  - `a = 1; % Integers`
  - `b = 2.883; % Reals`
  - `c = 1.2*a + b; % Formulae`
  - `x = 1; y = 2; % Multiple definitions per line`
  - `v = [1 5 2 6]; % vector(row vector)—square brackets!`
  - `m = [1 5;3 8]; % 2x2 matrix`
  - `n = [0 1+7]; % Expression in a definition`
  - `p = [y(2) 7 y(3)]; % Array indexing`

Reminders: Defining and indexing arrays

```
x = 1:2:10 % First:increment:last

x =   1     3     5     7     9

g = 1:4; % Row vector
h = g'; % Transpose makes column vector
i = [1;3;4;5]; % Or define column vector explicitly
```

This is all quite different from other languages

Reminders: Indexing

```
x = [1.1 -2.2 3.3 -4.4 5.5];

x(3) is 3.3
x(1:2) is [1.1 -2.2]
x(1:2:5) is [1.1 3.3 5.5]
```

```
m = [1 2 3;-2 -3 -4;3 4 5];
m(6) is 4 – weird but true…
m(2,3) is -4
m(3,:) is [3 4 5]
m(:,2) is [2;-3;4]
m(1:2,3:4) is [2 3;-3 -4]
```

```
1     2     3
-2    -3    -4
3     4     5
```

```
1     2     3
-2    -3    -4
3     4     5
```

Reminders: Functions

There are lots of functions and constants available to you:

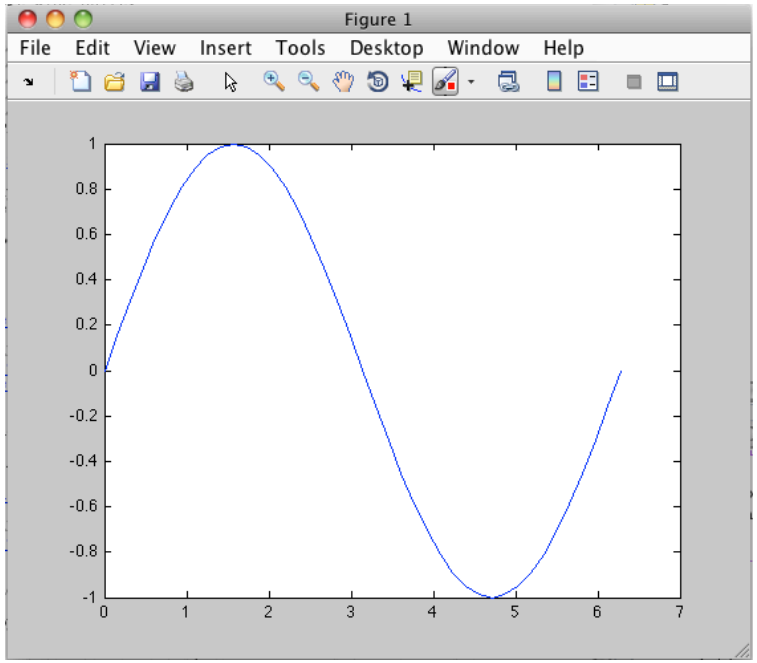
Math:	+ - * / ^ ()	
Comparison:	== < > ~=	(these return 1/0 for 'true'/'false')
Logical:	& |	(also return 1/0 for 'true'/'false')
Numerical:	Inf j NaN pi	
Functions:	abs exp log log2 log10 mod real imag round sign	
Trig:	sin cos tan asin acos atan (etc.)	
Specialised:	besselj factorial legendre fft (etc.)	

Use the Quick Reference Booklets

BIG TIP: Someone will probably have wanted to do the same job as you. That function already exists, you just have to FIND IT.

Reminders: Let's plot some data

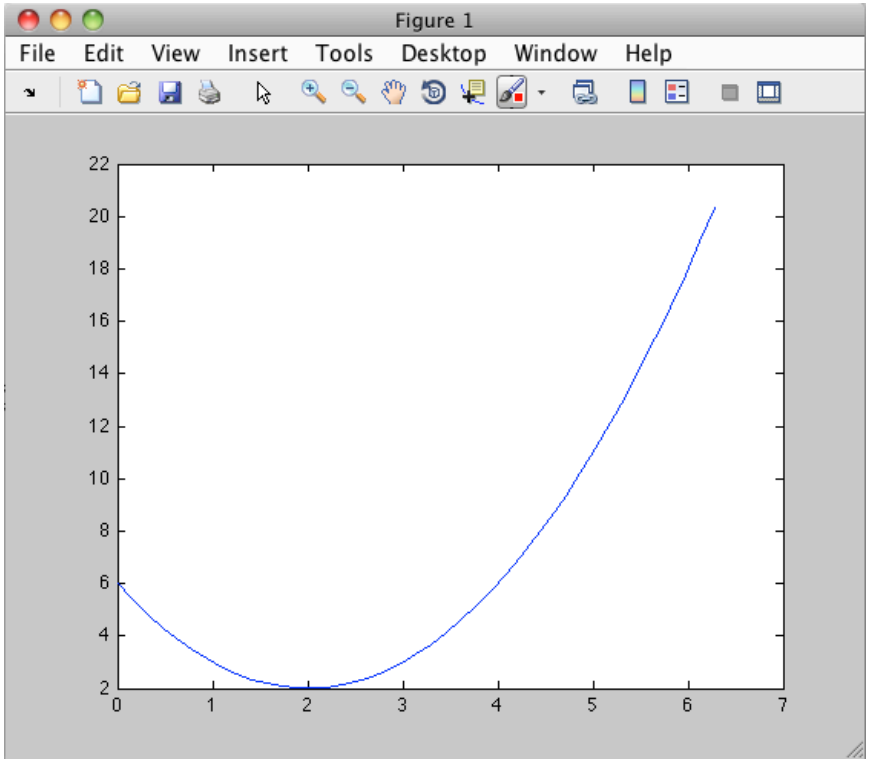
```
x = 0:pi/20:2*pi; % Make some x values
y = sin(x); % Make some y values as a fn. of x

plot(x,y);
```



Hint: also try linspace

Reminders: How about another one?

```
x = 0:pi/20:2*pi; % Make some x values
z = (x-2).^2+2; % Make some z
% (x-2)^2+2 doesn't work!
plot(x,z); % Over-writes first figure
```



Matrix and Array Multiply/Power – 'a Gotcha'

This is something *very* particular to MATLAB:

- * is used for matrix multiplication
- .* is used for element-by-element (array) multiplication
- ^ is used for matrix powers
- .^ is used for element-by-element (array) powers

```
>> a = [1 2;3 4]
a =
     1     2
     3     4
```

```
>> a^2
ans =
     7    10
    15    22
>> a.^2
ans =
     1     4
     9    16
```

a^2 is equivalent to a*a (matrix multiplication)

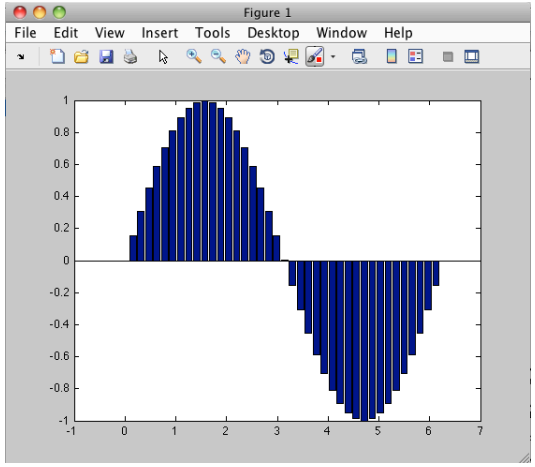
...and for multiplication

```
>> a = [1 2;3 4]

a =

     1     2
     3     4

>> b=[1 1;1 1]

b =

     1     1
     1     1
```

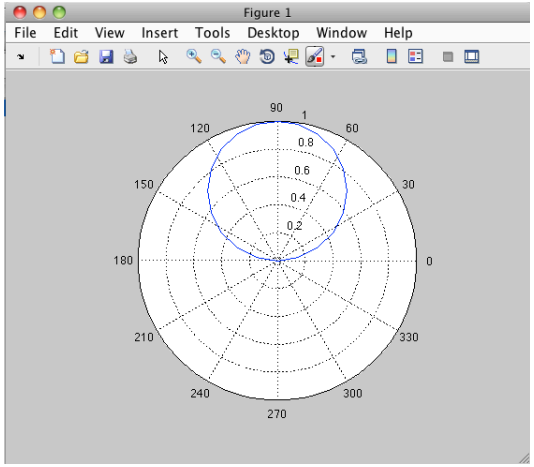
```
>> a*b

ans =

     3     3
     7     7

>> a.*b

ans =

     1     2
     3     4
```

Some other plot types

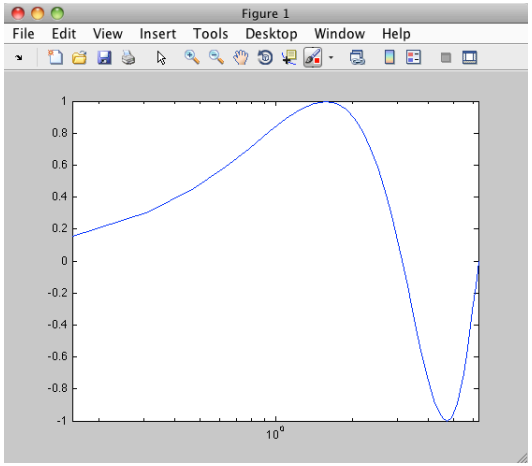
bar(x,y)



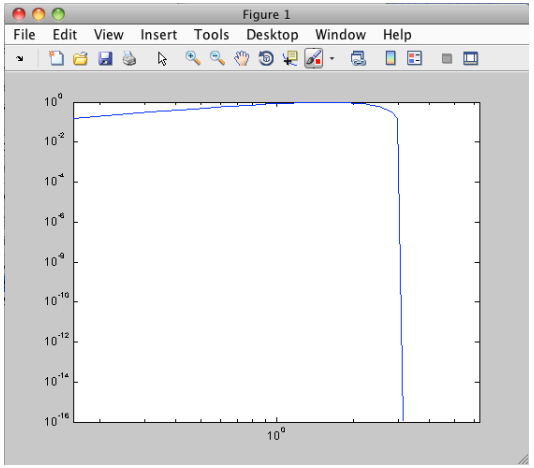
polar(x,y)



semilogx(x,y)



loglog(x,y)



- (Use the Quick Reference Booklets)

What is a Numerical Method?

- A numerical method is an algorithm that allows you to solve a problem numerically. Nowadays this is done on a computer
  - usually using some iterative procedure.
- Examples:
  - Solving systems of equations
  - Finding roots of equations (esp. non-linear)
  - Solving ordinary differential equations/partial differential equations
  - Monte-Carlo simulation of physical systems
- During numerical analysis, we must be aware that our predictions may be different from real system, e.g. to what accuracy can we believe them?
  - Round-off error
  - Range error
  - Truncation error

Representing Numbers in Computers

- Normally, computers store individual floating-point numbers in either
  - Single precision (4 bytes/32 bits)
  - Double precision (8 bytes/64 bits)/binary64 – e.g. default in MATLAB
  - If you define `a = 0.02;` then a will be in double-precision format
- A floating-point number is represented by a significand and exponent

- Significand (incorrectly) previously called mantissa
  - mantissa still used very widely

$$1.6 \times 10^{-19}$$

Significand · Base · Exponent

$$n = (-1)^s \times m \times b^q$$

Sign (0 or 1) · Number · Significand (binary number in double-precision) · Base (2 in double-precision) · Exponent

1 bit · 52 bits · 11bits

How accurate are stored numbers?

- Accuracy is determined by the number of bits in the significand:
  - Double-precision accuracy is about 16 decimal places $\frac{1}{b^l} = \frac{1}{2^{52}} \simeq 1 \times 10^{-16}$
  - Single precision accuracy is about 7 decimal places $\frac{1}{b^l} = \frac{1}{2^{23}} \simeq 1 \times 10^{-7}$

- Round-off errors can be very significant:
  - 'Subtraction of similar numbers' problem

$$\frac{10^{-20}}{(3 + 10^{-20}) - 3)}$$

Cannot be represented with 16 sd

This immediately gives catastrophic round-off error

>>10^(-20)/((3+10^(-20))-3) gives Inf (inifinity) instead of 1

  - Try it!

- Matlab has variable eps that gives lower bound on accuracy

eps = 2.2204e-16

Round-Off Error

Example: Round-off error when calculating derivatives

$$f'(x) = \frac{f(x+h) - f(x)}{h} \qquad h \to 0$$

$$\Delta(h) = \left| f'(x) - \frac{f(x+h) - f(x)}{h} \right|$$

Exact Derivative *Approximation*

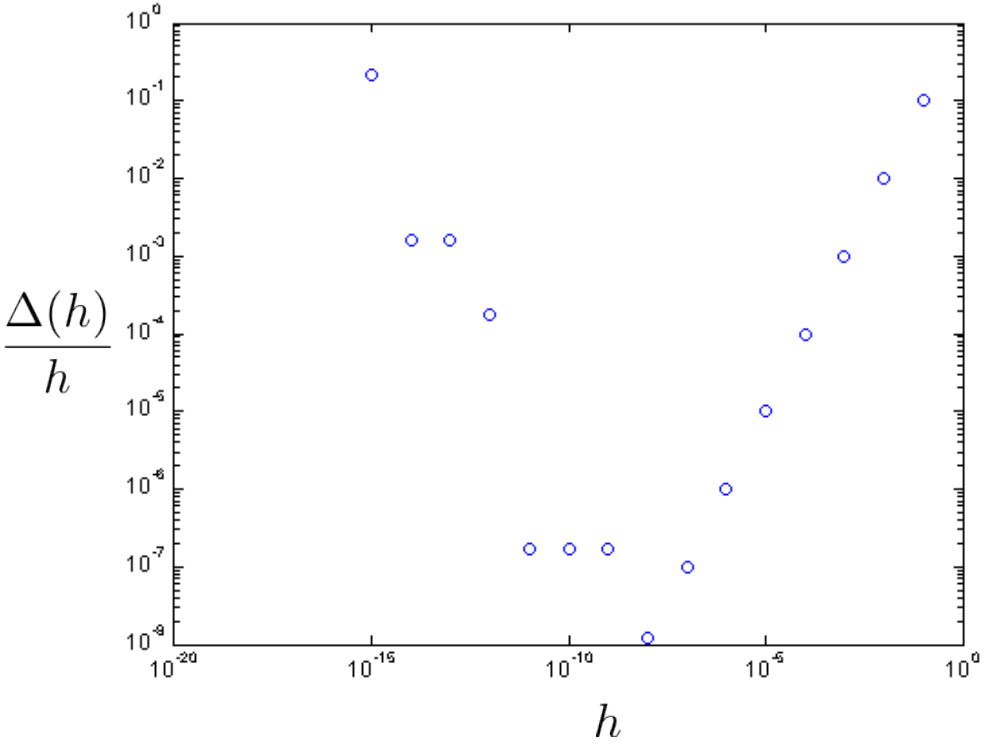
Example: $f(x) = x^2$

$$\Delta(h) = 2x - \frac{(x+h)^2 - x^2}{h}$$

An optimum value of h is often around

$$h \simeq 10^{-8}$$

But, it depends on the problem!



Range Error

- Range errors occur when you use a number outside of the exponent range
  - Smallest representable number: $2^{2^{-(l_q-1)}}$
  - Largest representable number: $2^{-2^{(l_q-1)}}$
  - Single precision: $l_q = 8 \quad 2^{\pm 2^{(l_q-1)}} \simeq 10^{\pm 38}$
  - Double precision: $l_q = 11 \quad 2^{\pm 2^{(l_q-1)}} \simeq 10^{\pm 308}$
- Exceeding the single precision range limit is not difficult in physics

$$a_0 = \frac{4\pi\epsilon_0\hbar^2}{m_e e^2} \simeq 5.2 \times 10^{-11} m$$

well within single precision limit

  - But, we would (without thinking) calculate the numerator and denominator separately in code: $4\pi\epsilon_0\hbar^2 \simeq 1.24 \times 10^{-78}$

$$m_e e^2 \simeq 2.34 \times 10^{-68}$$

oops!

  - Solution: use unit scale appropriate to the problem!

Truncation Error

Suppose we need to find the first derivative $f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$

We have seen that h cannot be chosen too small because of round-off error

Therefore $f'(x) = \frac{f(x+h) - f(x)}{h} + error$

Can we find an expression for the error?

Consider the Taylor expansion $f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \dots$

(equivalently) $f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(\zeta)$ where $x \leqslant \zeta \leqslant x+h$

Therefore $f'(x) \simeq \frac{f(x+h) - f(x)}{h} - \frac{1}{2}hf''(x)$

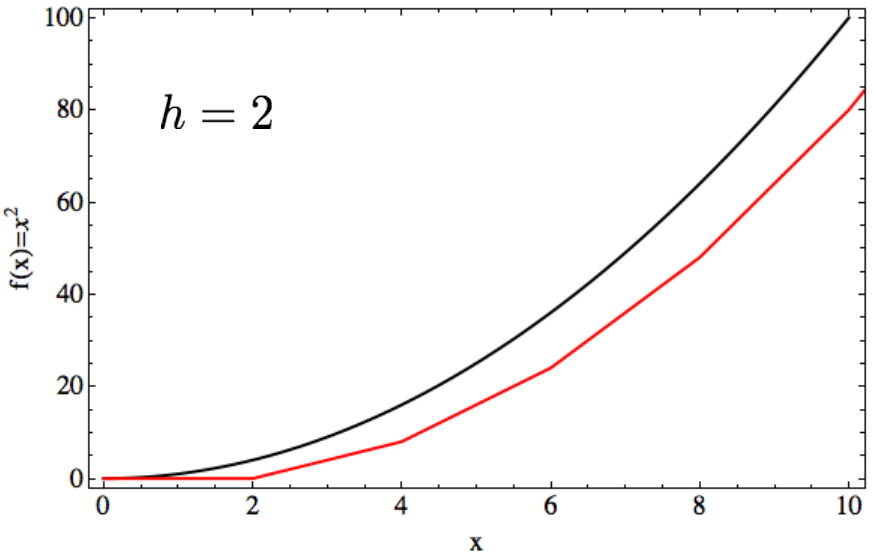
$\frac{1}{2}hf''(x) \sim O(h)$ is called the (local) truncation error

Numerical Integration

Now we are in a position to perform numerical integration

We will consider a universal equation of motion, and look at the simplest method we can imagine. This will turn out to be Euler's Method

Before we do that, let's picture what we are doing:



$$f(x) = x^2$$

$$f(x+h) = f(x) + hf'(x)$$

After n steps, we have effectively calculated the definite integral

$$\int_0^{nh} 2x dx \qquad f'(x) = 2x$$

This is a brief overview: for more detail see e.g. MIT OpenCourseWare: https://www.youtube.com/watch?v=X5-ucBtneVM

Euler's Method

$$\frac{dv}{dt} = a(x,t) \qquad \frac{dx}{dt} = v(t)$$ (2 equations, not 1!)

Using the definition of numerical derivatives, we can write

$$\frac{v(t+h) - v(t))}{h} + O(h) = a(x(t), v(t))$$

$$\frac{x(t+h) - x(t))}{h} + O(h) = v(t)$$

(forward derivatives)

Multiplying through,

$$v(t+h) = v(t) + h \cdot a(x(t), v(t)) + O(h^2)$$

$$x(t+h) = x(t) + h \cdot x(t) + O(h^2)$$

Re-writing in iterative notation,

$$v_{n+1} = v_n + h \cdot a_n$$

$$x_{n+1} = x_n + h \cdot v_n$$

Choose h and some initial conditions, and off you go!

Higher-Order Methods

An easy way to improve on Euler's method is to use one extra term in the Taylor expansion for the derivative. This is Heun's method:

$$v_{n+1} = v_n + h \cdot a_n$$

$$x_{n+1} = x_n + h \cdot v_n + \frac{h^2}{2} \cdot a_n \qquad error \sim O(h^3)$$

(cf.) $x(t+dt) = x(t) + x'(t)dt + x''(t)\frac{(dt)^2}{2} + \ldots$

Verlet's Method uses the concept of centred derivatives:

$$\frac{d^2x}{dt^2} = a \qquad \frac{dx}{dt} = v$$

$$f'(t) = \frac{f(t+h) - f(t-h)}{2h} - \frac{1}{6}h^2 f'''(\xi)$$

$$f''(t) = \frac{f(t+h) + f(t-h) - 2f(t)}{h^2} - \frac{1}{12}h^2 f^{(4)}(\xi)$$

re-arranging the 2nd of these gives

$$x_{n+1} = 2x_n - x_{n-1} + h^2 \cdot a_n + O(h^4)$$

Note: quartic in h!

Wow! This should be much more accurate!
But notice that (n+1) term requires n and (n-1) terms – not self-starting.
Must use Euler method for first step.

Solvable system – Spring-Mass with Damping

General SHM with damping

$$m\ddot{x} = -b\dot{x} - kx + F(t)$$

$$a(t) = -\frac{b}{m}v(t) - \frac{k}{m}x(t) + \frac{F(t)}{m}$$

$$a_n = -\frac{b}{m}v_n - \frac{k}{m}x_n + \frac{F(t)}{m}$$

Solvable if:

$$F(t) = 0$$

(and for certain functions)

Euler

$$a_n = -\frac{b}{m}v_n - \frac{k}{m}x_n$$

$$v_{n+1} = v_n + h \cdot a_n$$

$$x_{n+1} = x_n + h \cdot v_n$$

Imp. Euler

$$a_n = -\frac{b}{m}v_n - \frac{k}{m}x_n$$

$$v_{n+1} = v_n + h \cdot a_n$$

$$x_{n+1} = x_n + h \cdot v_n + \frac{1}{2}h^2 \cdot a_n$$

Verlet

$$x_{n+1} = A \cdot x_n + B \cdot x_{n-1}$$

$$A = 2\frac{(2m - kh^2)}{D} \quad B = \frac{(bh - 2m)}{D}$$

$$D = 2m + bh$$

Q: Why are we looking at an analytically solvable system?

Simple Harmonic Motion: Exact Solution

$m\ddot{x} + b\dot{x} + kx = 0$ has solution $x(t) = Ae^{(i\omega - \frac{\gamma}{2})t}$ where $\gamma = \frac{b}{m}$ & $\omega_0 = \sqrt{\frac{k}{m}}$

$$\omega^2 = \omega_0^2 - \frac{\gamma^2}{4} = \frac{k}{m} - \frac{b^2}{4m^2}$$

Quality factor $Q = \frac{\omega_0}{\gamma} = \sqrt{\frac{km}{b^2}}$

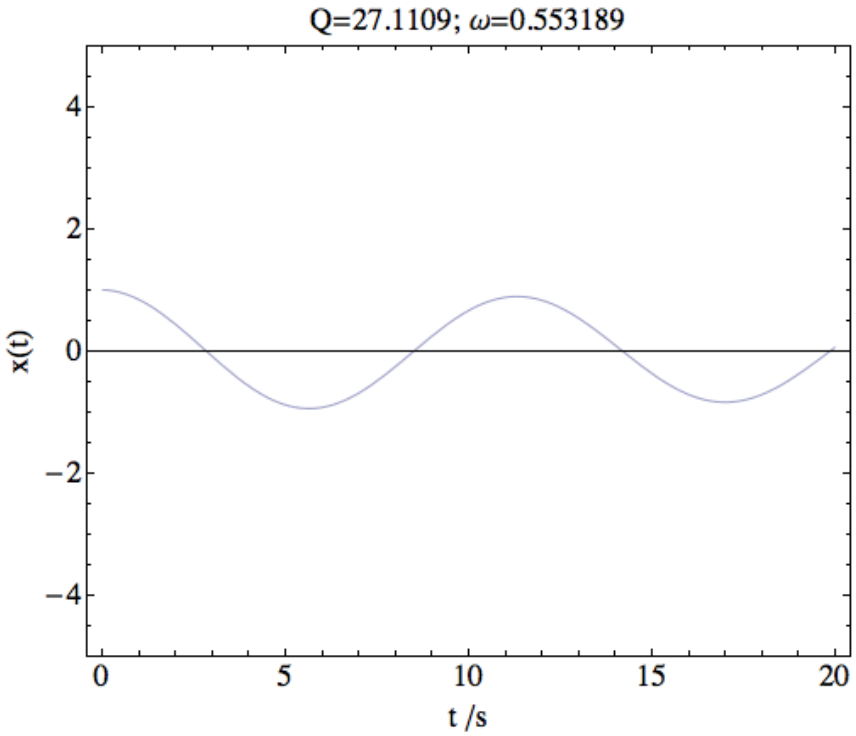
Critical damping is defined as $\omega = 0$, therefore $\omega_0 = \frac{\gamma^2}{4}$ and so $b = 2\sqrt{km}$

Substituting into equation for quality factor gives $Q = \frac{1}{2}$

$b_{cr} = 2\sqrt{km}$ $\quad b = \frac{1}{2}b_{cr} \quad Q = 1$

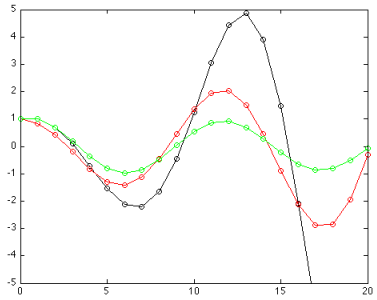
$b = 2b_{cr} \quad Q = \frac{1}{4}$

Analytic versus numerical simulation

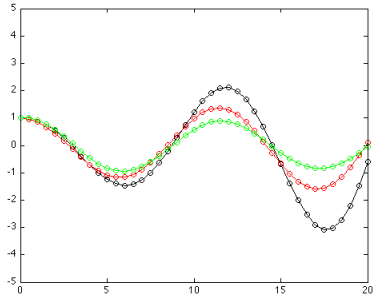


Analytical solution

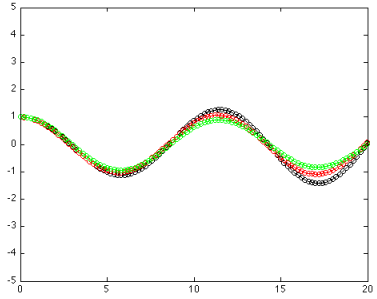
The important point here is to test your algorithm in the case where the solution is analytically known: then you know how good it is.



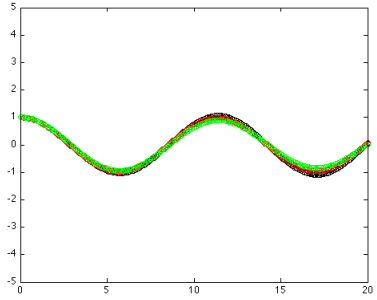
$h = 1s$



$h = 0.5s$

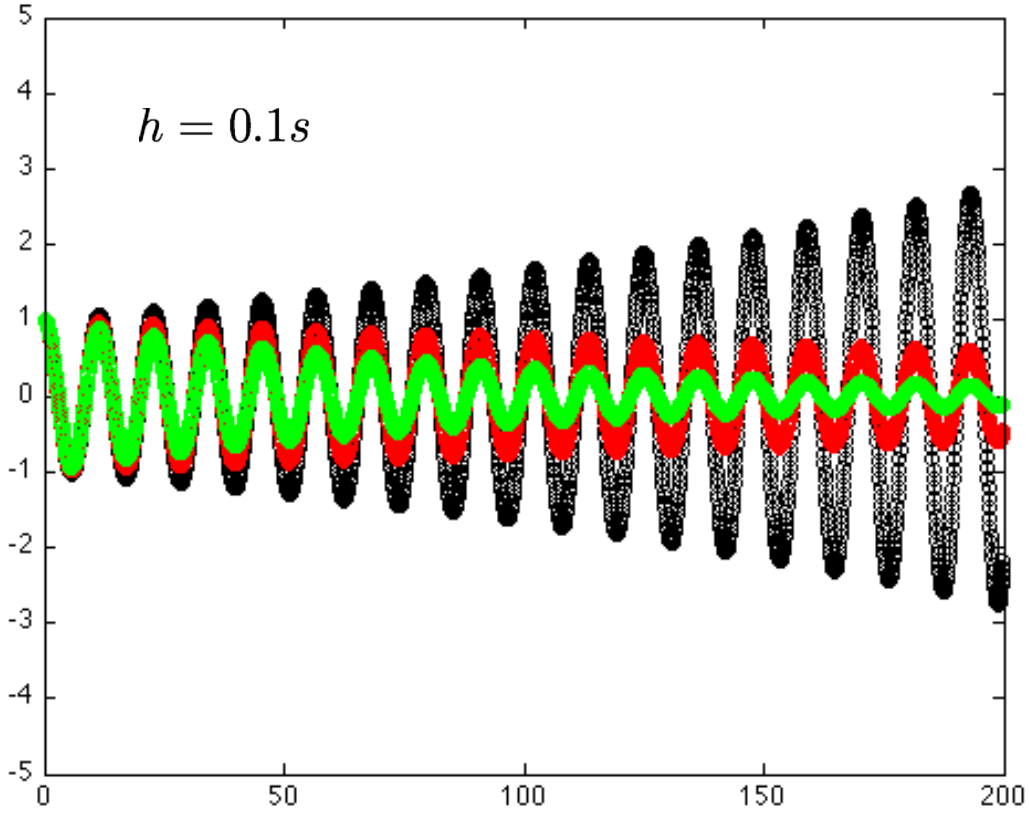


$h = 0.2s$



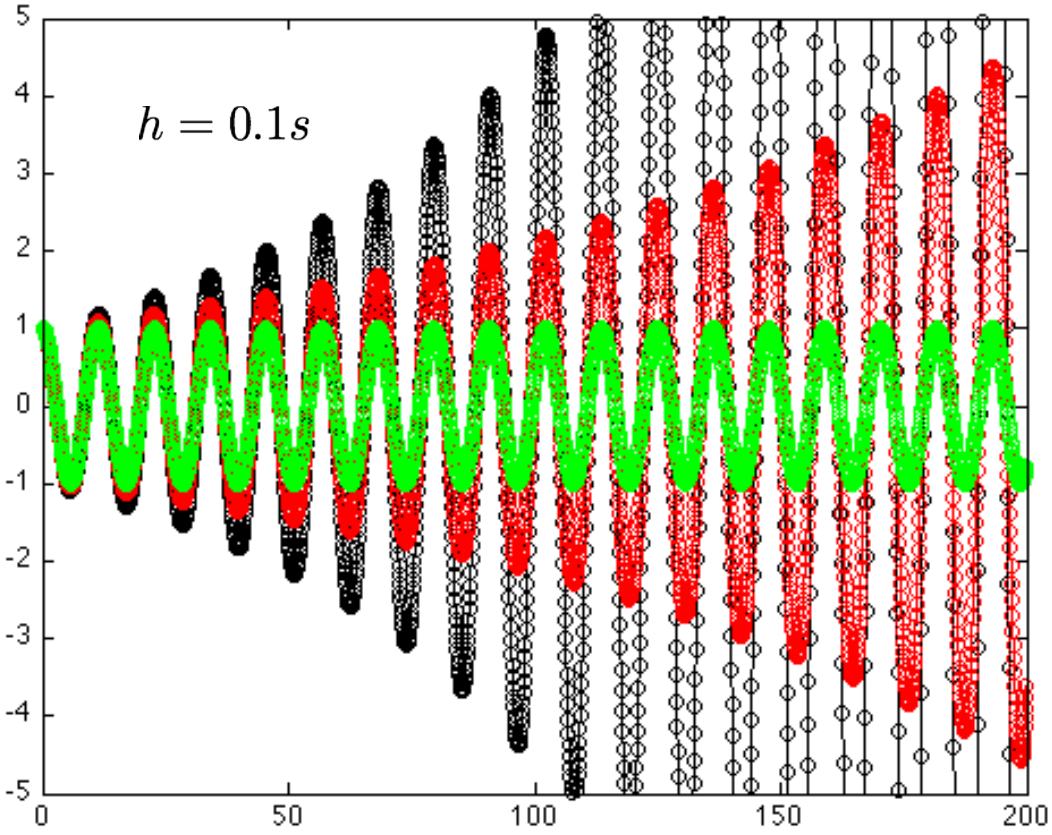
$h = 0.1s$

Numerical Integration Over Long Times



Artificial addition of energy to system – numerical artefact.
These integrators are non-*symplectic*, one of them badly so!

No damping



Question: How would you check the green one was really constant amplitude?

Energy Errors in Euler's Method

Ignoring for simplicity an initial velocity, we may write an expression for the damped SHM motion

$$x(t) = Ae^{-bt/2m}\cos(\sqrt{(\frac{k}{m} - \frac{b^2}{4m^2})}t)$$

The energy is just

$$E(t) = \frac{1}{2}kx^2(t) + \frac{1}{2}mv^2(t)$$

Substituting and expanding out, we get (eventually)

$$E(t) = \frac{A^2e^{-bt/m}}{8m}(4km + b^2\cos(\frac{\sqrt{4km - b^2}t}{m}) + b\sqrt{4km - b^2}\sin(\frac{\sqrt{4km - b^2}t}{m}))$$

Hint: can do this expansion straightforwardly using Mathematica or other computer algebra system

Euler's Method with No Damping

Start by writing out expression for Euler step

$$\begin{aligned} x_{n+1} &= x_n + hv_n \\ v_{n+1} &= v_n - \frac{kh}{m}x_n \end{aligned}$$

Then substitute into expression for energy

$$E_{n+1} = \frac{1}{2}kx_n^2 + \frac{1}{2}mv_n^2 + \frac{1}{2m}h^2k^2x_n^2 + \frac{1}{2}h^2kv_n^2$$

and collect terms to give

$$E_{n+1} = E_n(1 + \frac{k}{m}h^2)$$

In other words, our simulation predicts a steadily increasing energy!

The Euler-Cromer Method

Let's make a slight change to Euler's method

$$\begin{aligned} x_{n+1} &= x_n + h v_{n+1} \\ v_{n+1} &= v_n - \frac{kh}{m} x_n \end{aligned}$$

(note: you will have to put the damping part back in yourself!)

Expanding out the expression for energy again, we obtain (eventually)

$$E_{n+1} = E_n - \frac{1}{2}h^2\left(\frac{k^2 x_n^2}{m} - k v_n^2\right) - h^3 \frac{k^2 x_n v_n}{m} + h^4 \frac{k^3 x_n^2}{2m^2}$$

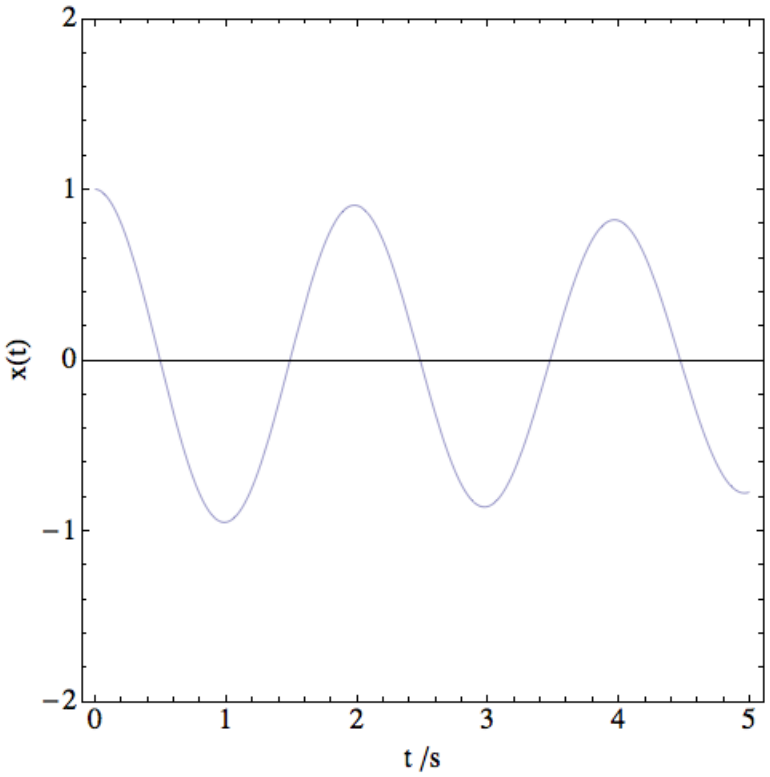
The second term averages out over one oscillation. The result is that the overall energy is conserved.

But… there are oscillations about this average, compared to the true value of the energy. Better than Euler though!

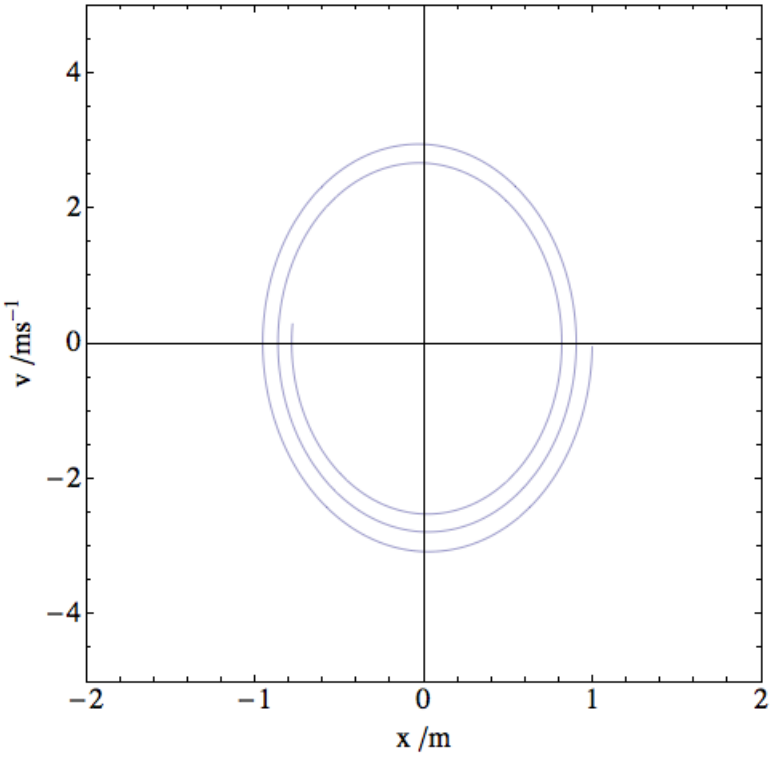
Euler-Cromer is a Symplectic Integrator, i.e. it is energy-preserving

Visualising Integrators

A=1,k=10,b=0.1,m=1



Time Plot



Phase Space Plot

Things you ought to find out for yourself

- Order of the method
- Symplectic vs non-symplectic integrators
- Most people use:
  - RK4 – 4th-order Runge Kutta (non-symplectic)
  - 4th-order Yoshida integrator (symplectic)
- There are many such methods, sometimes with overlapping names
  - RK2 = Heun's Method = Improved Euler

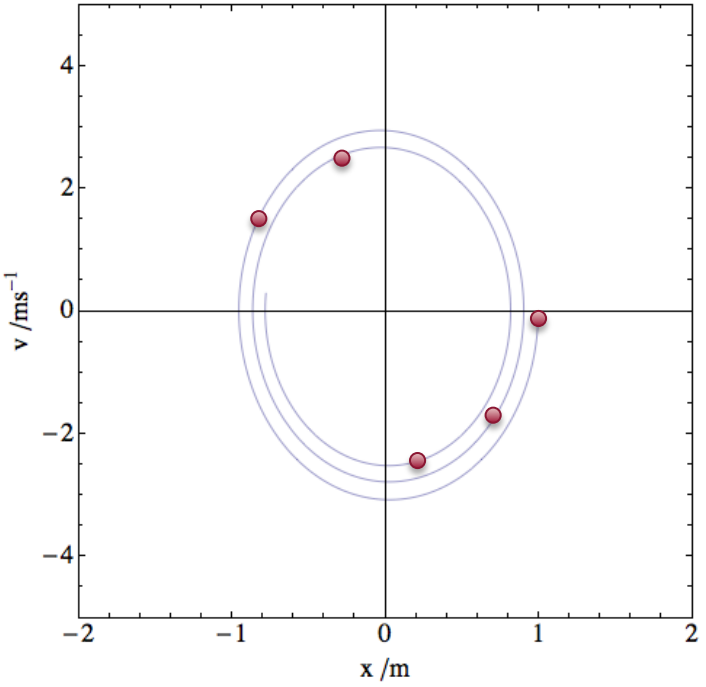
- We haven't covered some things yet. These include:
  - Butcher Tableaux (simple way of classifying methods)
  - Stability of methods; stiffness
  - Extension to multiple dimensions
  - Simplifying methods, e.g. mappings
  - Nonlinearities and chaotic dynamics

Main points

- We pay attention to those quantities in a problem which ought to be conserved – this is a way of testing our algorithm/code
- We try to test a code against a problem/situation with a known solution
  - This concept is known as validation
- Validation is a very important and very under-utilised concept in simulation
  - All your simulations should be validated in some form

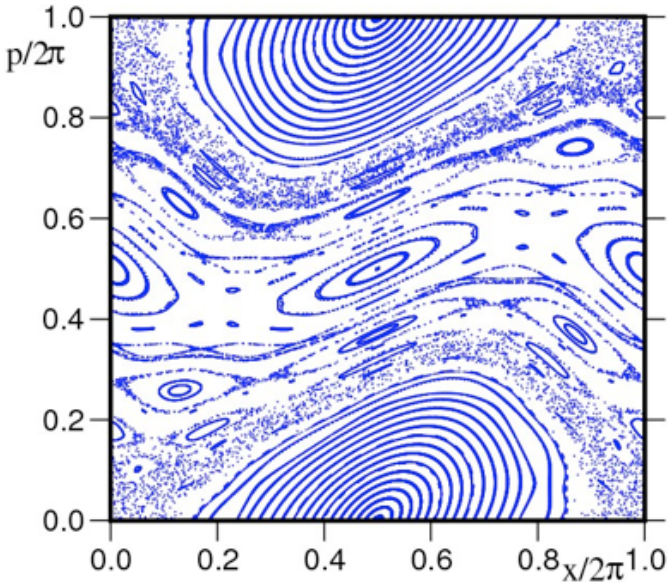
Introduction to mappings

- A linear SHO without damping has a known analytic solution – we don't need to integrate it
- Instead, we can calculate the state of the system at some later time using an evaluation of that solution
  - This is a form of mapping
  - We of course find that linear systems are quite dull!
  - Let's look at some nonlinear systems



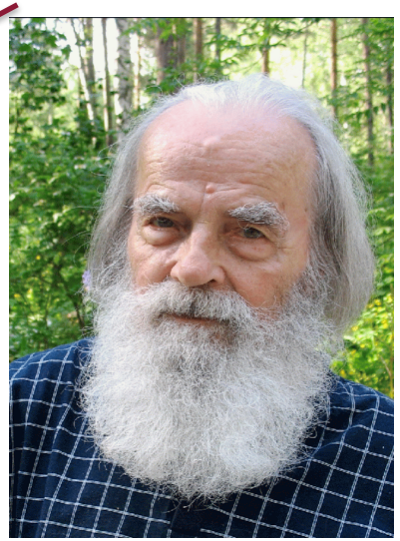
The Chirikov Map

- The Chirikov map is akin to a kicked pendulum
  - It has a linear part described by a mapping (from one kick to the next)
  - And it has a nonlinear kick (of some form)



$$p_{n+1} = p_n + K \sin \theta_n$$
$$\theta_{n+1} = \theta_n + p_{n+1}$$

Nonlinear part!



Boris Chirikov
(1928-2008)

Example: Nonlinear SHM

Some nonlinear spring with damping

$$m\ddot{x} = -b\dot{x} - kx - k_2x^2$$

$$a_n = -\frac{b}{m}v_n - \frac{k}{m}x_n - \frac{k_2}{m}x^2$$

- Since the acceleration can be expressed at a given time, it may be used within an integrator just the same as if the system was linear

Project 1

- Project 1 Tasks
  - Write Euler/Euler-Cromer and at least one higher-order method (e.g. RK4) for integrating a linear SHO with damping
  - Validate your simulation and present suitable results and visualisations
  - Examine the nonlinear oscillator and obtain interesting results; present with suitable parameters