ORDINARY DIFFERENTIAL EQUATIONS

Ari Maller

FIRST ORDER DIFFERENTIAL EQUATIONS

- One of the main uses of numerical techniques is for the solving of differential equations which are notoriously difficult to solve analytically. An ordinary differential equation is something of the form

$$\frac{dx}{dt} = \frac{2x}{t}$$

- This can be solved by separation of variables. But what if instead we had something like the following

$$\frac{dx}{dt} = \frac{2x}{t} + \frac{3x^2}{t^3}$$

- this is not separable nor is it linear. Nonlinear equations can rarely be solved analytically, but numerically it doesn't matter if the equation is linear or not.

FIRST ORDER DIFFERENTIAL EQUATIONS WITH ONE VARIABLE

- The general form of a first order one variable differential equation is

$$\frac{dx}{dt} = f(x,t)$$

- Note we have been choosing t as the independent variable. In physics we are often interested in the time evolution of a system, but the independent variable could be spatial or something else.
- To calculate a solution for a differential equation we also need a boundary condition, the value of x at some value of t. Like $x(0)=0$. We will assume we are given boundary conditions for the solutions we discuss.

EULER'S METHOD

- If we have a differential equation and boundary condition at x(t) we can write the value of x a short time later using a Taylor expansion.

$$x(t+h) = x(t) + h\frac{dx}{dt} + \frac{1}{2}h^2\frac{d^2x}{dt^2} + ...$$
$$= x(t) + hf(x,t) + O(h^2)$$

- To first order we now have a way to calculate x(t). Start with our boundary condition, then get x(t')=x(t+h)=x(t)+h f(x,t). Then get x(t'')=x(t'+h) until you have covered the range of t you want to cover.

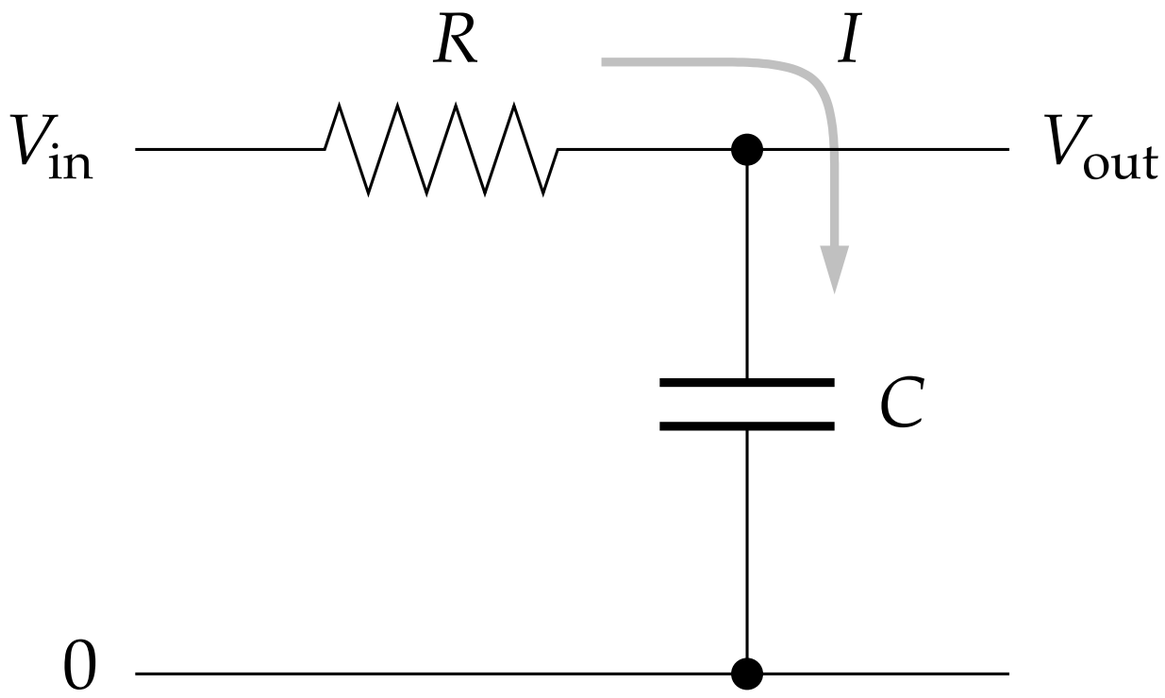
- In general Euler's method is not bad; however, we never actually use it because with a little extra effort we can get a method, Runge-Kutta that is more accurate and often faster.

EULER'S METHOD

- The error in Euler's method goes as 1/2 h^2 d^2x/dt^2, but this is for each step. If we are performing our calculation from a to b with step size h, that that requires N=(b-a)/h steps. The total error then would be:

$$\sum_{k=0}^{N-1} \frac{1}{2} h^2 \frac{d^2x}{dt^2} = \frac{1}{2} h \sum_{k=0}^{N-1} h \frac{df}{dt} \simeq \frac{1}{2} h \int_a^b \frac{df}{dt} dt = \frac{1}{2} h [f(x(b), b) - f(x(a), a)]$$

- The total error goes as h instead of h^2. This means that if we choose h small enough we should be able to get reasonable results, but h might have to be very small and therefore N very large. If h is too small we have to worry about roundoff errors.



- Here is a simple electronic circuit with one resistor and one capacitor: This circuit acts as a low-pass filter: you send a signal in on the left and it comes out filtered on the right.
- Using Ohm's law and the capacitor law and assuming that the output load has very high impedance, so that a negligible amount of current flows through it, we can write down the equations governing this circuit as follows. Let I be the current that flows through R and into the capacitor, and let Q be the charge on the capacitor. Then:

$$IR = V_{\text{in}} - V_{\text{out}}\,, \qquad Q = CV_{\text{out}}\,, \qquad I = \frac{dQ}{dt}.$$

EXERCISE 8.1 (MODIFIED)

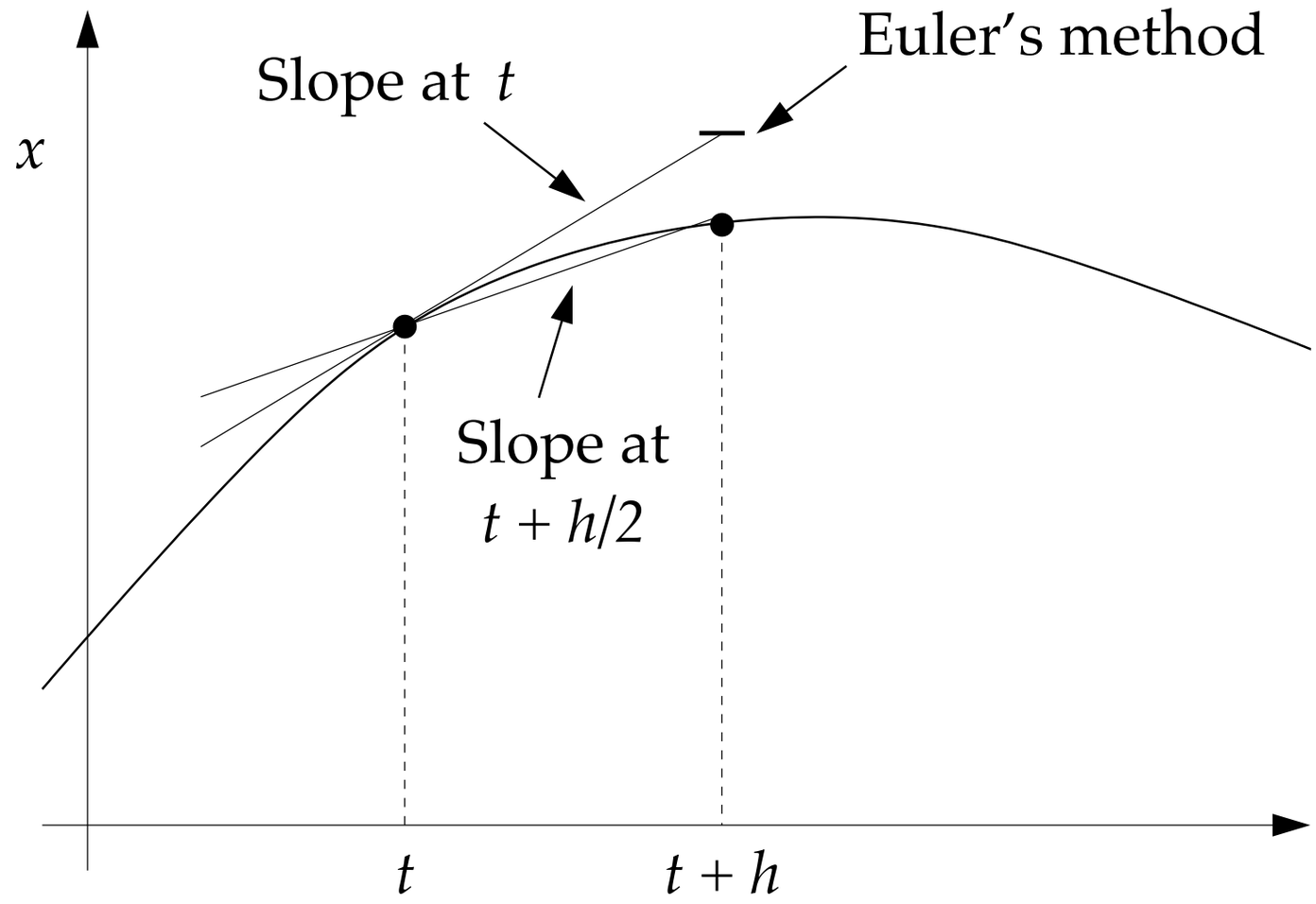
- Substituting the second equation into the third, then substituting the result into the first equation, we find that

$$\frac{dV_{\text{out}}}{dt} = \frac{1}{RC}\left(V_{\text{in}} - V_{\text{out}}\right).$$

- Write a program to solve this equation for $V_{out}(t)$ using Euler's method when in the input signal is a square-wave with frequency 1 and amplitude 1:
- $V_{in}(t) = 1$ if np.floor(2t) is even, -1 if np.floor(2t) is odd
- Use the program to make plots of the output of the filter circuit from t=0 to t=10 when RC=0.01, 0.1, and 1, with initial condition $V_{out}(0)=0$. You will have to make a decision about what value of h to use in your calculation. Small values give more accurate results, but the program will take longer to run. Try a variety of different values and choose one for your final calculations that seems sensible to you.
- Based on the graphs produced by your program, describe what you see and explain what the circuit is doing.
- A program similar to the one you wrote is running inside most stereos and music players, to create the effect of the “bass” control. In the old days, the bass control on a stereo would have been connected to a real electronic low-pass filter in the amplifier circuitry, but these days there is just a computer processor that simulates the behavior of the filter in a manner similar to your program.

RUNGE-KUTTA

- We can think of Euler's method graphically. Then what we did was take the slope at the point t, and then used that line to extrapolate what the function would be at t+h.
- The idea of Runge-Kutta is that instead we calculate the slope at t+h/2 and then extrapolate this slope to get x(t+h). This usually gives a better estimate.



RUNGE-KUTTA

- To see how accurate such a method would be let us again perform a Taylor expansion, but now around x(t+h/2)

$$x(t+h) = x(t+\frac{h}{2}) + \frac{1}{2}h\frac{dx}{dt} + \frac{1}{8}h^2\frac{d^2x}{dt^2} + O(h^3)$$

- we can also derive an expression for x(t)

$$x(t) = x(t+\frac{h}{2}) - \frac{1}{2}h\frac{dx}{dt} + \frac{1}{8}h^2\frac{d^2x}{dt^2} + O(h^3)$$

- if we subtract these two equations we get

$$x(t+h) = x(t) + h\frac{dx}{dt} + O(h^3) = x(t) + hf(x(t+\frac{1}{2}h), t+\frac{1}{2}h) + O(h^3)$$

- This is an expression for x(t+h) that is accurate to h^2, the error is $O(h^3)$. Thus it is a second order scheme instead of first order. Similar to how Simpson's rule improved over the trapezoidal rule by an order.

2ND ORDER RUNGE – KUTTA

- The only problem with this expression is that it includes the value of the function evaluated at t+h/2 which we don't know. We get around this by using Euler's method to estimate x(t+h/2) and then using that estimate in the 2nd order Runge-Kutta formula to get x(t+h).
- So a 2nd order Runge-Kutta works in the following way.

$$k_1 = hf(x,t)$$

$$k_2 = hf(x + \frac{1}{2}k_1, t + \frac{1}{2}h)$$

$$x(t+h) = x(t) + k_2$$

- Euler's method gives us a first stab at the solution and then we use that for our better guess.

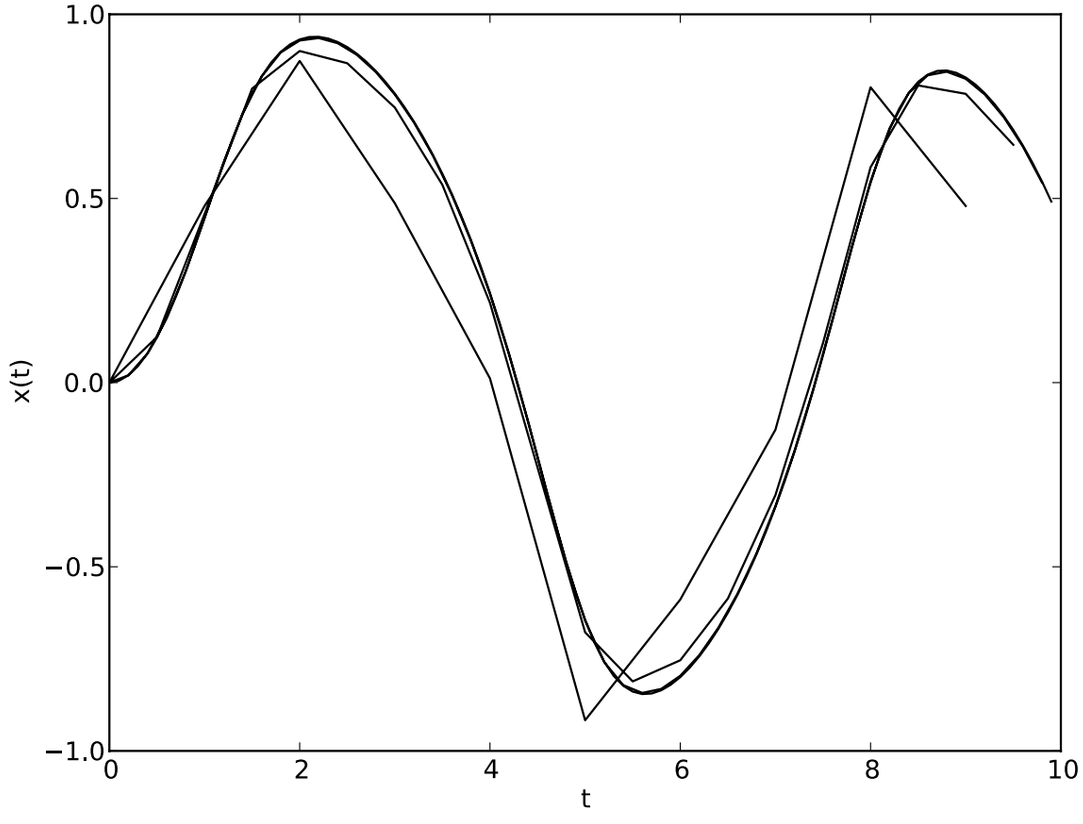
2ND ORDER RUNGE-KUTTA

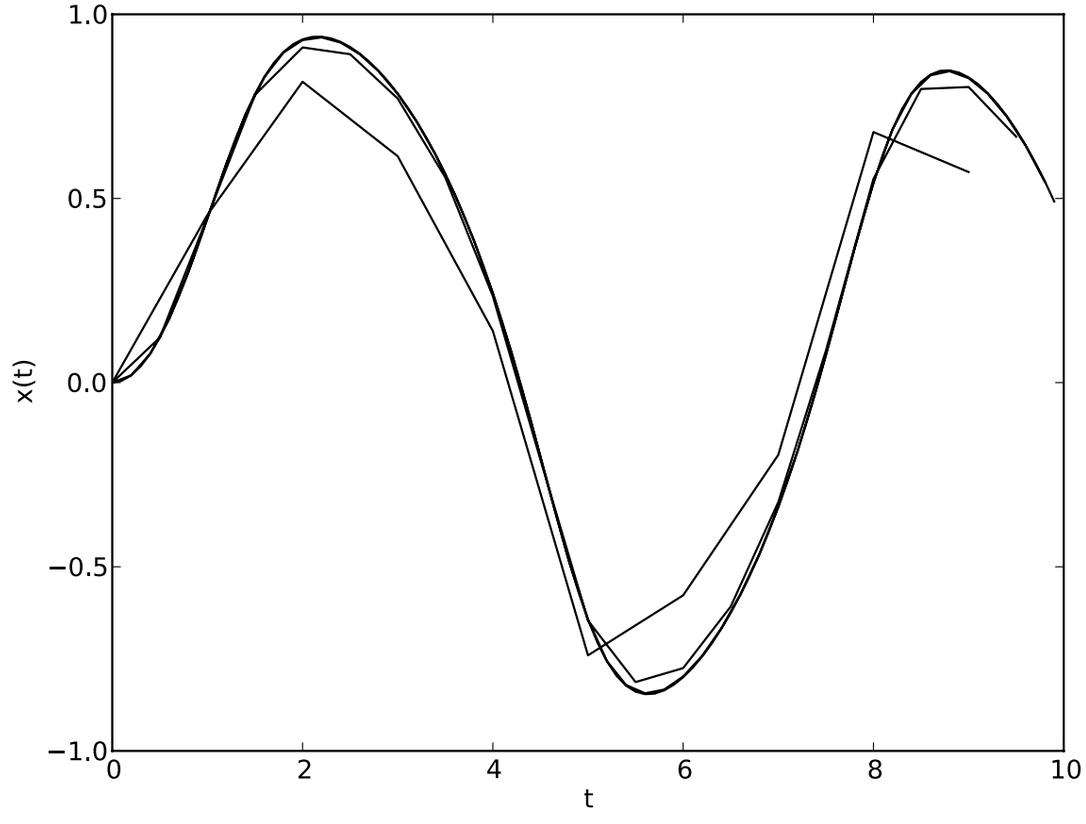
- The method described above is called 2nd order Runge-Kutta. Note that the order refers to a single step, when we add up N steps we expect the error to increase by one order.
- However, we used Euler's method in order to get the value as h/2 and Euler's method is first order. You can show that this additional error only contributes the third order so the method is still 2nd order.
- However, there is no need to stick with 2nd order. We can perform more Taylor series around more points and cancel out terms till we have a scheme with the accuracy that we desire. In fact 4th order Runge-Kutta is usually considered the 'best' choice.

4TH ORDER RUNGE-KUTTA

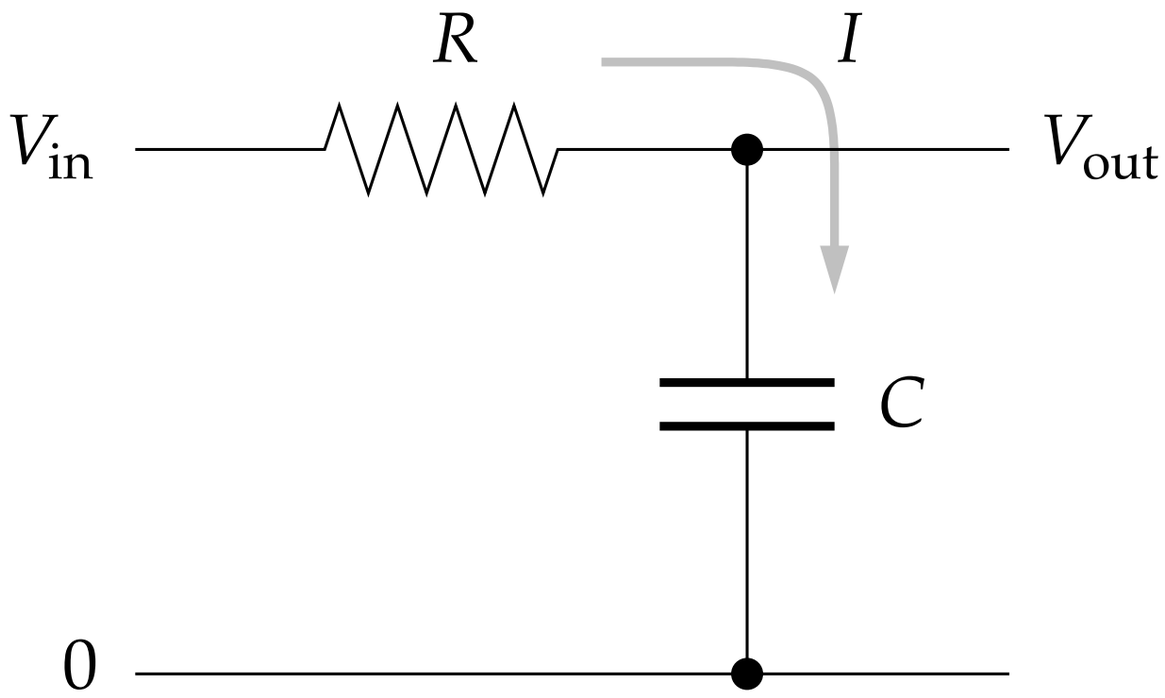
- Fourth order Runge-Kutta is considered the gold standard for solving differential equations. That is it is often considered the right compromise between accuracy and complexity to program.
- Performing the Taylor expansions and algebra what one gets at the end is the following algorithm.

$$
\begin{aligned}
k_1 &= hf(x,t) \\
k_2 &= hf(x+\frac{1}{2}k_1, t+\frac{1}{2}h) \\
k_3 &= hf(x+\frac{1}{2}k_2, t+\frac{1}{2}h) \\
k_4 &= hf(x+k_3, t+h) \\
x(t+h) &= x(t) + \frac{1}{6}(k_1+2k_2+2k_3+k_4)
\end{aligned}
$$





Solutions calculated using 2nd order Runge-Kutta on the left and 4th order on the right using 10,20,50, and 100 steps. You can see by 20 steps one has a reasonably good approximation to the differential equation. One thing to note about Runge-Kutta, the method is so accurate that if you make a mistake in your coding of the algorithm, it may not at all be obvious. Thus one must take extra care in coding it up.



- Here is a simple electronic circuit with one resistor and one capacitor: This circuit acts as a low-pass filter: you send a signal in on the left and it comes out filtered on the right.
- Using Ohm's law and the capacitor law and assuming that the output load has very high impedance, so that a negligible amount of current flows through it, we can write down the equations governing this circuit as follows. Let I be the current that flows through R and into the capacitor, and let Q be the charge on the capacitor. Then:

$$IR = V_{\text{in}} - V_{\text{out}}\,, \qquad Q = CV_{\text{out}}\,, \qquad I = \frac{dQ}{dt}.$$

EXERCISE 8.1

- Substituting the second equation into the third, then substituting the result into the first equation, we find that

$$\frac{dV_{\text{out}}}{dt} = \frac{1}{RC}\left(V_{\text{in}} - V_{\text{out}}\right).$$

- Write a program to solve this equation for $V_{out}(t)$ using the fourth-order Runge-Kutta method when in the input signal is a square-wave with frequency 1 and amplitude 1:
- $V_{in}(t) = 1$ if np.floor(2t) is even, -1 if np.floor(2t) is odd
- Use the program to make plots of the output of the filter circuit from t=0 to t=10 when RC=0.01, 0.1, and 1, with initial condition $V_{out}(0)=0$. You will have to make a decision about what value of h to use in your calculation. Small values give more accurate results, but the program will take longer to run. Try a variety of different values and choose one for your final calculations that seems sensible to you.
- Based on the graphs produced by your program, describe what you see and explain what the circuit is doing.
- A program similar to the one you wrote is running inside most stereos and music players, to create the effect of the “bass” control. In the old days, the bass control on a stereo would have been connected to a real electronic low-pass filter in the amplifier circuitry, but these days there is just a computer processor that simulates the behavior of the filter in a manner similar to your program.

EXAMPLE 8.3 RK4.PY

```
from math import sin
from numpy import arange

def f(x,t):
    return -x**3 + sin(t)

a = 0.0
b = 10.0
N = 10
h = (b-a)/N

tpoints = arange(a,b,h)
xpoints = []
x = 0.0

for t in tpoints:
    xpoints.append(x)
    k1 = h*f(x,t)
    k2 = h*f(x+0.5*k1,t+0.5*h)
    k3 = h*f(x+0.5*k2,t+0.5*h)
    k4 = h*f(x+k3,t+h)
    x += (k1+2*k2+2*k3+k4)/6
```

SOLUTIONS OVER INFINITE RANGES

- So we now have a good technique for solving ODEs. However, if we want to solve our ODE out to t equals infinity then we are out of luck, because we can never solve an infinite number of steps h.
- To deal with this problem we use the same approach as when we needed to integrate functions out to infinity, a suitable change of variables.
- If we define u= t/(1+t) then when t->∞ u -> 1. Using the chain rule on dx/dt=f(x,t) gives.

$$\frac{dx}{du}\frac{du}{dt} = f(x,t) \qquad \text{or} \qquad \frac{dx}{du} = \frac{dt}{du} f(x, \frac{u}{u-1}) \qquad \text{but} \qquad \frac{dt}{du} = \frac{1}{(1-u)^2}$$

$$\text{so} \qquad \frac{dx}{du} = (1-u)^{-2} f(x, \frac{u}{u-1})$$

SOLUTIONS OVER INFINITE RANGES

- But this is just an ordinary differential equation, but now we only want to evaluate it out to 1.0. If we define

$$g(x,u) = (1-u)^{-2} f(x, \frac{u}{u-1})$$

- then we just have $\frac{dx}{du} = g(x,u)$

- solving this will give us x(u) which we can then map back using u(t) to x(t).

- There are other changes of variables besides the one shown here. Good choices can make the function g(x,u) simpler and easier to work with. Or make the solution more accurate in a region of interest.

DIFFERENTIAL EQUATIONS WITH MORE THAN ONE VARIABLE

- So far we have considered ODEs with only one dependent variable. But there are plenty of situations in physics where one has more than one dependent variable.
- In these cases we have simultaneous differential equations, where the derivatives can depend on any of the variables. For example, $\frac{dx}{dt} = xy - x \qquad \frac{dy}{dt} = y - xy - \sin^2 \omega t$
- Note there is only one dependent variable, t. There are still ordinary differential equations, not partial differential equations.

DIFFERENTIAL EQUATIONS WITH MORE THAN ONE VARIABLE

- A general form for two first order differential equation is

$$\frac{dx}{dt} = f_x(x, y, t) \qquad \frac{dy}{dt} = f_y(x, y, t)$$

- for an arbitrary number of variables we could write this in vector form

$$\frac{d\mathbf{r}}{dt} = \mathbf{f}(\mathbf{r}, t)$$

- solving these simultaneous differential equation analytic is likely to be very hard, but numerical it is really just the same Runge-Kutta scheme. We perform the Taylor expansion

$$\mathbf{r}(t + h) = \mathbf{r}(t) + h\frac{d\mathbf{r}}{dt} + O(h^2)$$

- Thus we get the same expressions for Euler's method and Runge-Kutta the only difference being we have a vector of equations at each step instead of just one.

SECOND ORDER DIFFERENTIAL EQUATIONS

- We have been discussing first-order differential equations, but first-order equations are rare in physics. Many if not most of the differential equations in physics are second order.
- Let's consider the simpler case of only one dependent variable. Then a second order differential equation would look like

$$\frac{d^2x}{dt^2} = f(x, \frac{dx}{dt}, t)$$

- Now lets define a new quantity y where

$$y = \frac{dx}{dt} \qquad \frac{dy}{dt} = f(x, y, t)$$

- But now you'll notice that our second-order equation is now two first order equations which we have already discussed how to solve.

HIGHER ORDER DIFFERENTIAL EQUATIONS

- The same trick can be used for any order of differential equation. For example a third order equation

$$\frac{d^3x}{dt^3} = f(x, \frac{dx}{dt}, \frac{d^2x}{dt^2}, t)$$

- we define two new variables, y and z to get

$$\frac{dx}{dt} = y \qquad \frac{dy}{dt} = z \qquad \frac{dz}{dt} = f(x, y, z, t)$$

- This can be generalized to more than one dependent variables simply by performing the same action but on the vectors.

$$\frac{d^2\mathbf{r}}{dt} = \mathbf{f}(\mathbf{r}, \frac{\mathbf{r}}{dt}, t) \qquad \frac{d\mathbf{r}}{dt} = \mathbf{s} \qquad \frac{d\mathbf{s}}{dt} = \mathbf{f}(\mathbf{r}, \mathbf{s}, t)$$

EXAMPLE 8.6 THE NONLINEAR PENDULUM

- Let's consider the pendulum which we generally treat as linear, but in reality in nonlinear. Say we have a pendulum with a length l and a mass m at the end of it. The equation of motion for the pendulum will be

$$ml\frac{d^2\theta}{dt^2} = -mgsin\theta$$

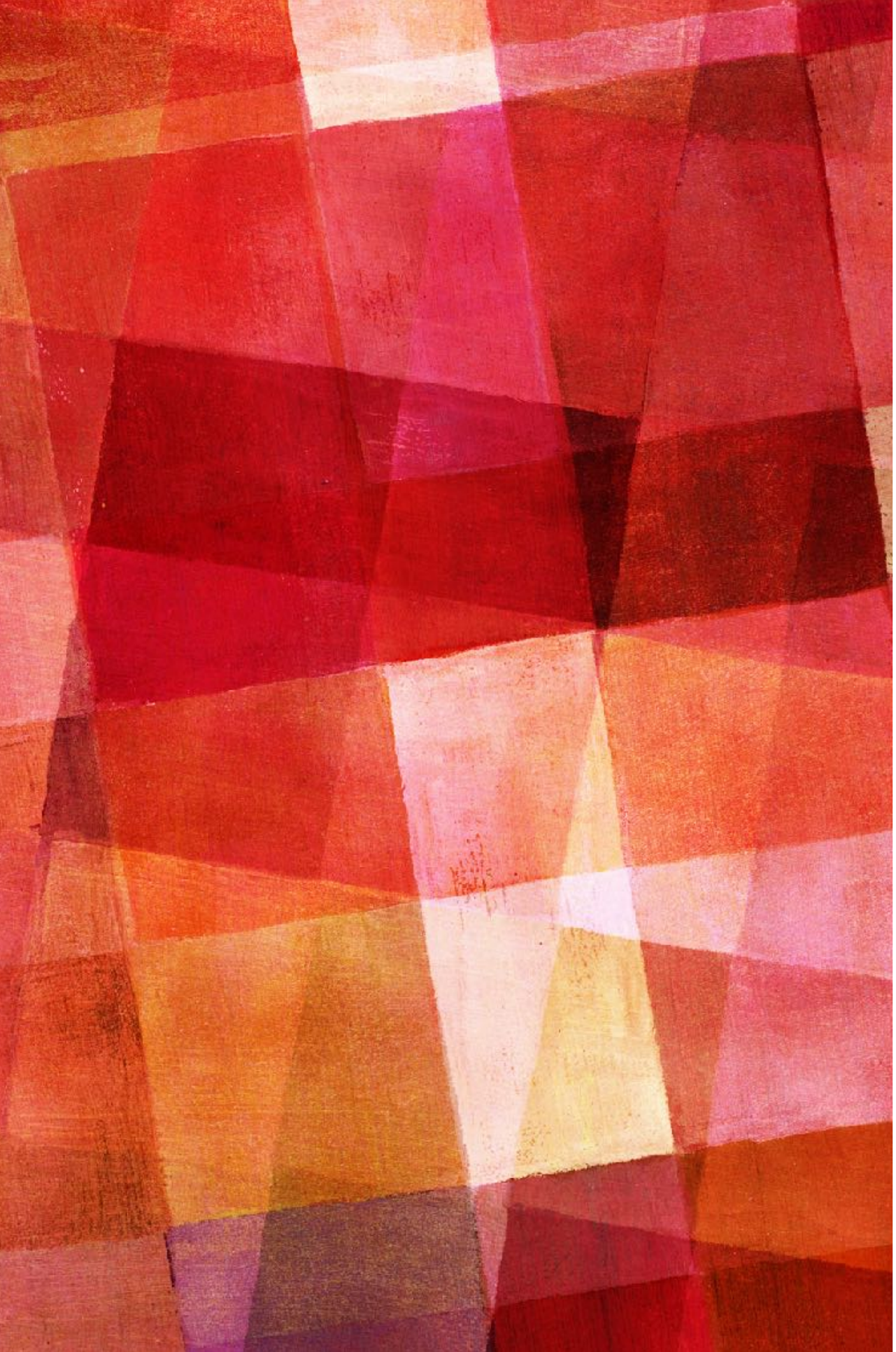
- a second order differential equation. Now let us use our trick

$$\frac{d\theta}{dt} = \omega \qquad \frac{d\omega}{dt} = -\frac{g}{l}\sin\theta$$

- we now have two first-order equations. So let's combine them into vectors, $\mathbf{r}=(\theta,\omega)$. The $\mathbf{f}$ will be the vector function that acts on them.

➤ We can code up **f** as the following:

```
def f(r,t):
    theta=r[0]
    omega=r[1]
    ftheta = omega
    fomega = -(g/l)*sin(theta)
    return np.array([ftheta,fomega],float)
```



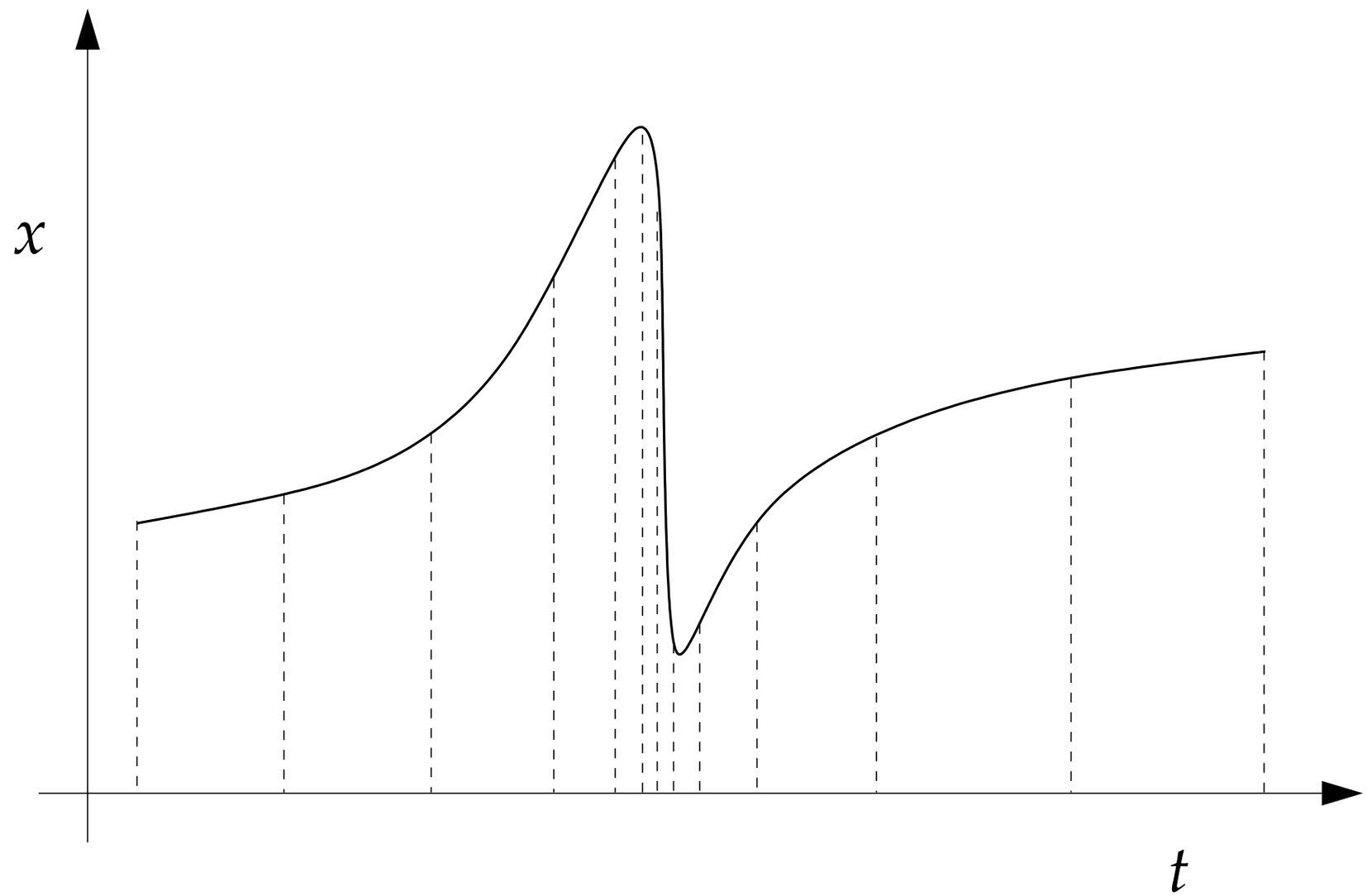
EXERCISE 8.4

➤ Write a program to solve the two first-order equations, using the fourth-order Runge--Kutta method for a pendulum with a 10 cm arm. Use your program to calculate the angle θ of displacement for several periods of the pendulum when it is released from a standstill at $\theta = 179°$ from the vertical. Make a graph of θ as a function of time.

VARYING THE STEP SIZE

➤ So far in all of our discussion we have taken the step size, h, to be fixed. But what if our function looked like the one below.

➤ We would need very small step sizes to resolve the drop in the middle part. But those steps would be over kill for the rest of the function.

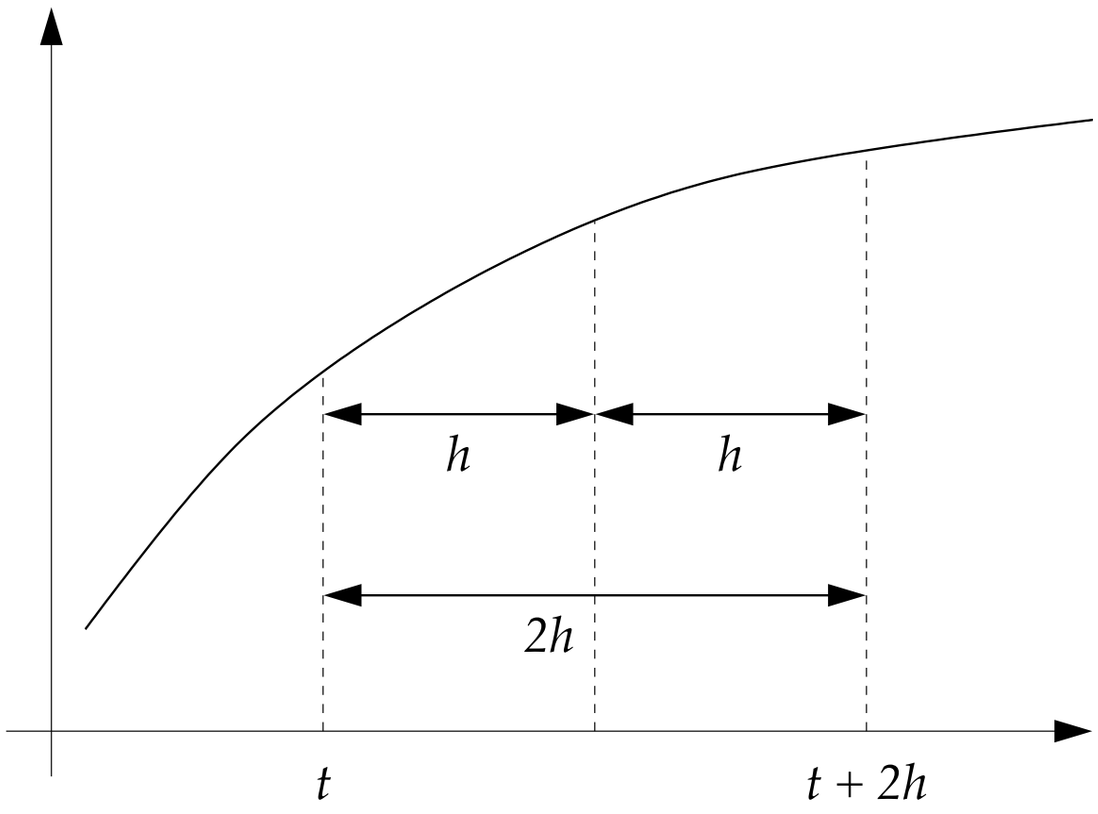


ADAPTIVE STEP SIZE

- The basic idea of adaptive step size is to very the step size such that the error in each step is roughly the same.
- In practice the adaptive step size method has two parts. First we have to estimate the error on our steps and then we compare the error to the accuracy we would like to have.
- Let's look at how this would work for 4th order Runge-Kutta.
- We start by choosing an initial value for h and make two steps, $x(t+h)$ and $x(t+2h)$.
- Then we make the same evaluation in just one step. By comparing the results we get as estimate of the error.

ADAPTIVE STEP SIZE

- The method is accurate to fourth order so we expect the error to go as h^5.
- If we make two steps the error will be roughly $2ch^5$. With one step twice as big the error would be $c(2h)^5 = 32ch^5$.



- So the difference between our two estimates, $x_1 - x_2 = 30ch^5$.
- The error $\varepsilon = ch^5$ would then be

$$\epsilon = \frac{1}{30}(x_1 - x_2)$$

ADAPTIVE STEP SIZE

- But we still don't know what the correct step size is. Let us take the correct step size to be h'. Then from our formula

$$\epsilon' = ch'^5 = ch^5\left(\frac{h'}{h}\right)^5 = \frac{1}{30}(x_1 - x_2)\left(\frac{h'}{h}\right)^5$$

- Let the target accuracy for our calculation be δ, then the accuracy for each step would need to be h'δ. We want the value of h' so that the actual accuracy equals the target accuracy or

$$\frac{1}{30}(x_1 - x_2)\left(\frac{h'}{h}\right)^5 = h'\delta$$

- solving for h' gives $h' = h\left(\frac{30h\delta}{|x_1 - x_2|}\right)^{1/4} = h\rho^{1/4}$
- and we now have a formula that gives us h' for a trial h.

ADAPTIVE STEP SIZE ALGORITHM

1. First perform two steps of size h and one step from the same starting point of size 2h. This gives two estimates of x(t+2h), x_1 and x_2.
2. Calculate the ratio $$\rho = \frac{30h\delta}{|x_1 - x_2|}$$
3. If greater than one then are accuracy is better than our target accuracy so keep the value of x_1, use formula to get correct larger h' step size and move on starting again with step 1.
4. IF less than one then plug into previous equation to get correct step size h' and use that to determine x(t+h'). The move on starting again at step 1.

ADAPTIVE STEP SIZE

- Note it is possible by bad luck that x_1 will equal x_2, making h' infinite or close to infinite. Commonly, one places an upper limit on h' so that one ignores the calculated value if it becomes so large that it seems problematic.
- A common rule of thumb is to never let it increase by more than a factor of two between steps.
- One might think that requiring a repeat if the accuracy in one step is large as being overly cautious, it is after all only one step. But errors add, so if a thousand steps had an error that was too big, that may significantly increase the error above the targeted amount.

ADAPTIVE STEP SIZE FOR MULTIPLE EQUATIONS

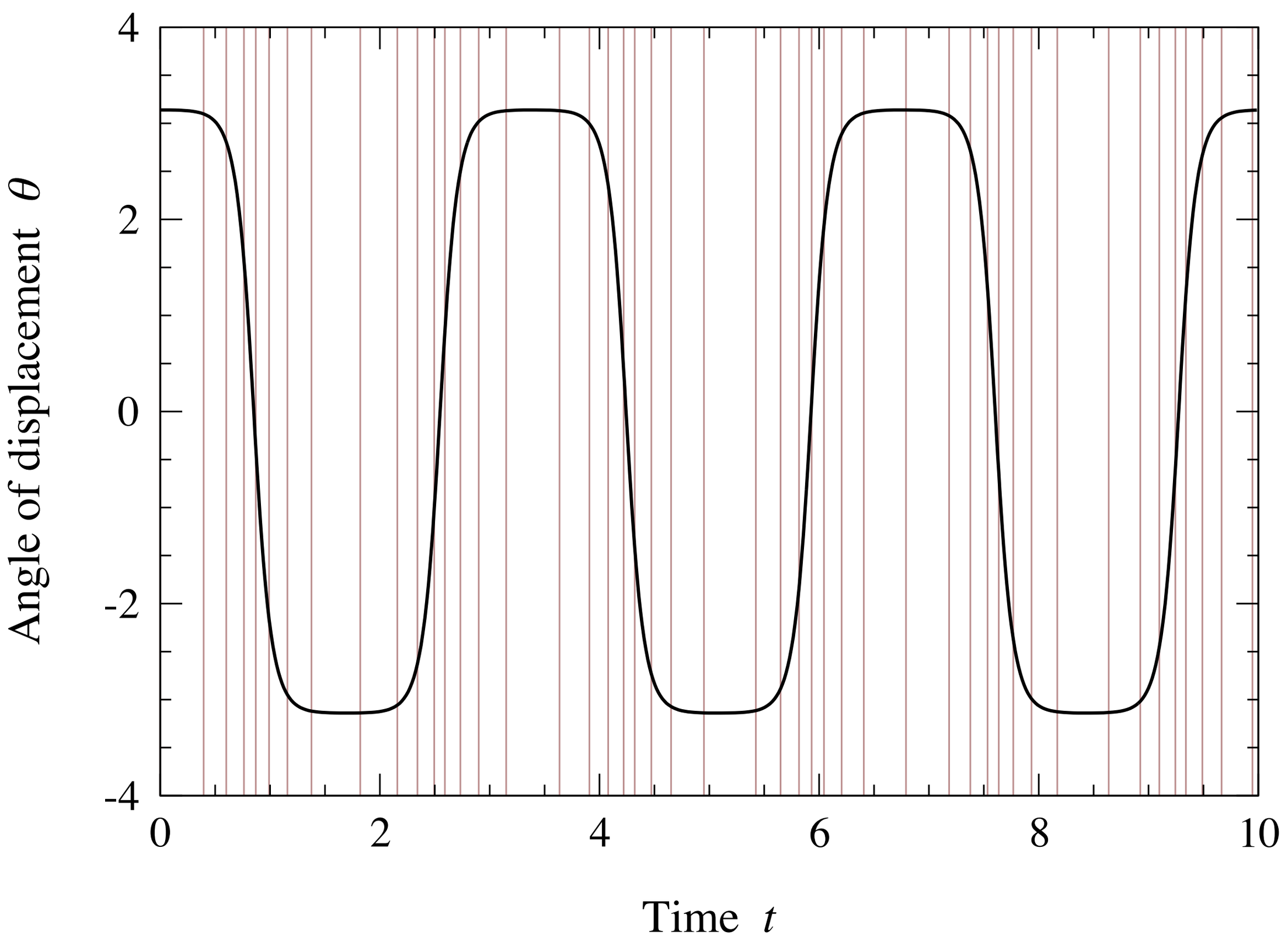
- If we have multiple equations the derivation can be duplicated to show we get

$$\epsilon_x = \frac{1}{30}(x_1 - x_2) \qquad \epsilon_y = \frac{1}{30}(y_1 - y_2)$$

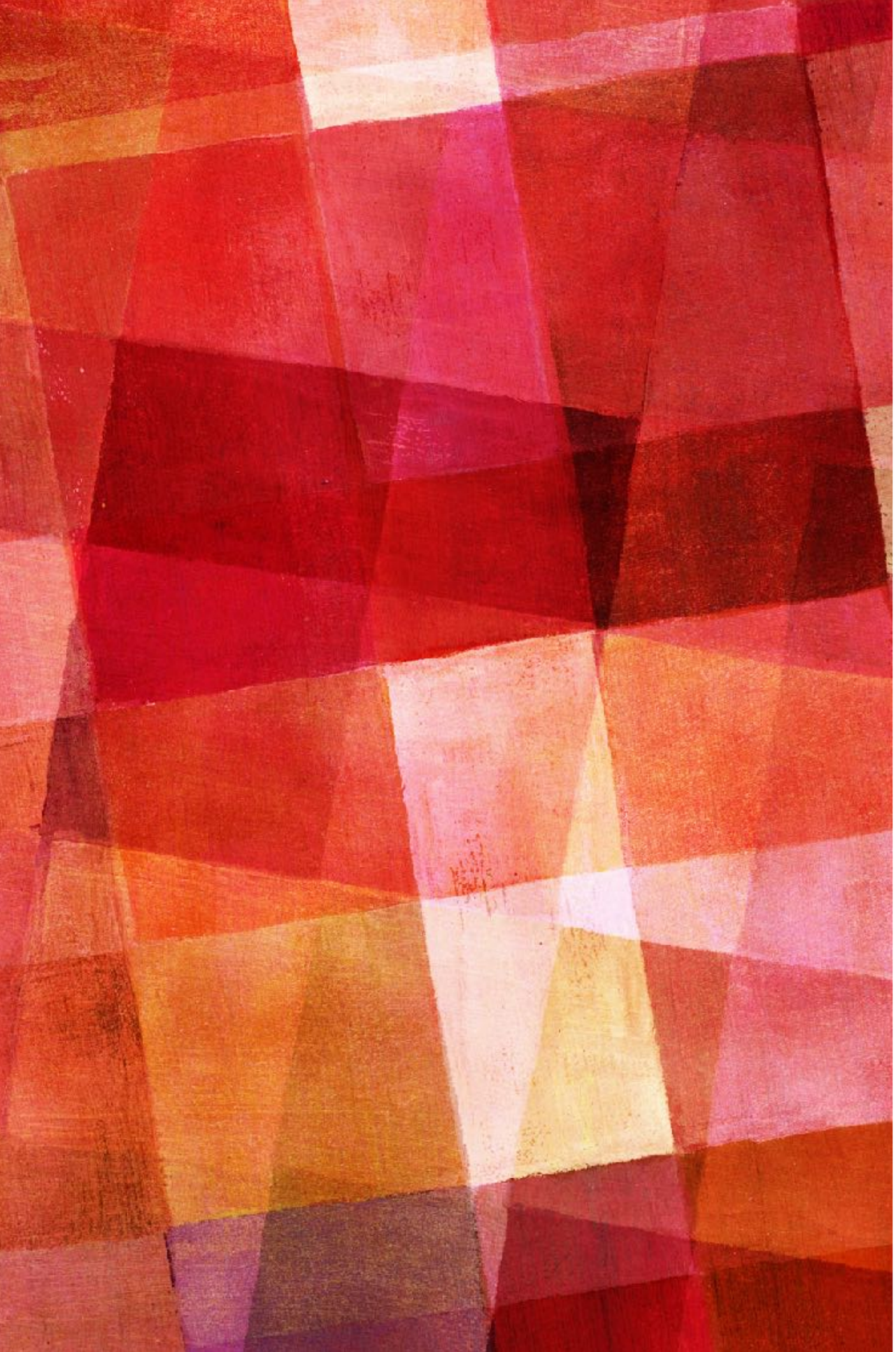
- The only complication is in what accuracy we want. We might want the sum of the square of the errors

$$\epsilon = \sqrt{\epsilon_x^2 + \epsilon_y^2}$$

- However it is a 2nd order differential equation where we have simply defined y=dx/dt then we might only care about the error in x and not the error in y at all.
- So with multiple equations some thought as to the accuracy of what is needed in determining adaptive step sizes.



Adaptive step size calculation of the nonlinear pendulum we looked at earlier. As one can see the steps are more finely spaced where the function changes quickly.



OTHER METHODS

- Leapfrog Method
- Verlet Method
- Modified Midpoint Method
- Burlisch-Stoer Method

LEAPFROG METHOD

- The leapfrog method is very similar to Runge-Kutta, but has certain advantages for certain problems.
- In Runge-Kutta we evaluate x half way to wear we want and then use that to go the full step.

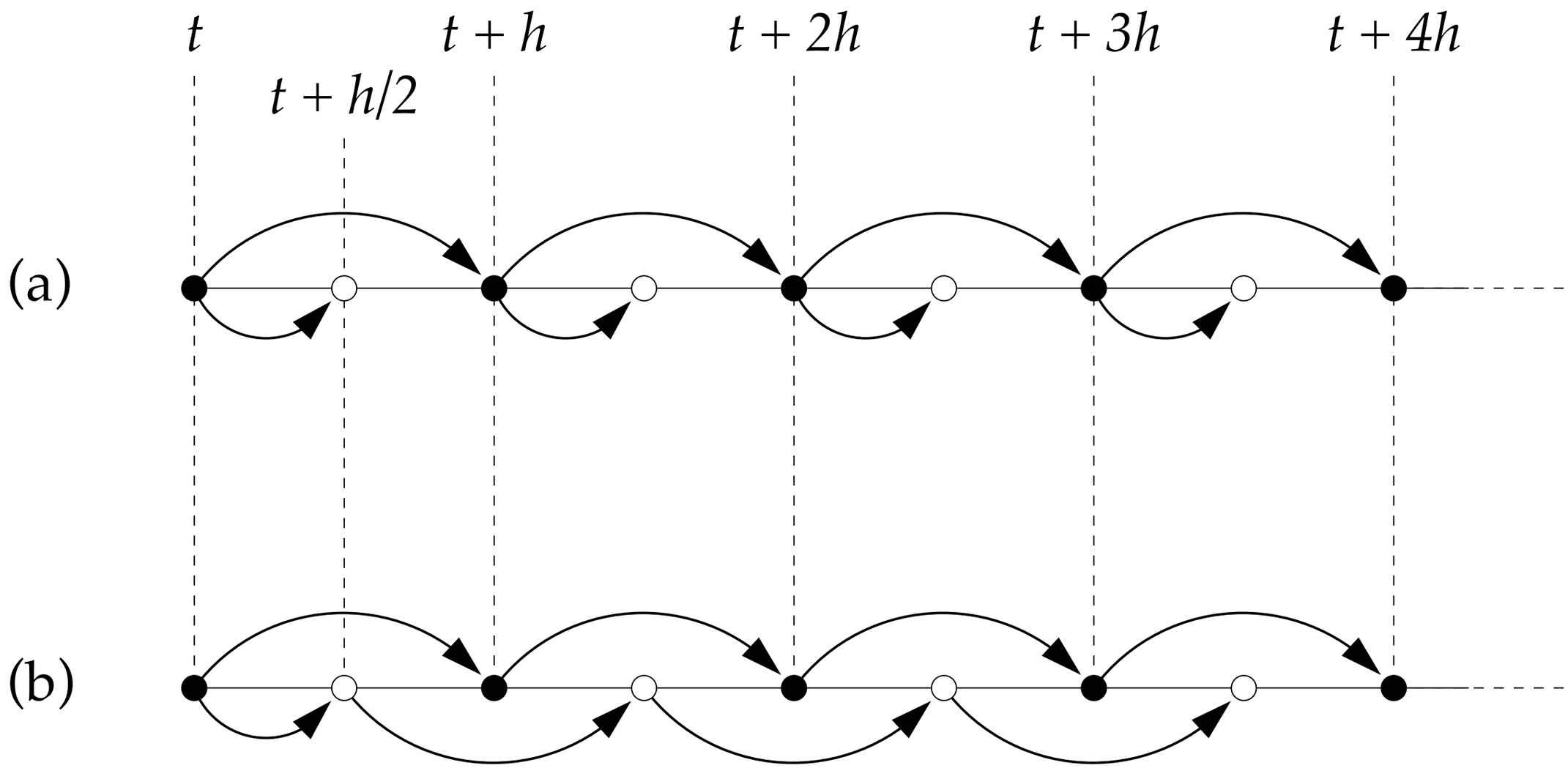
$$x(t+\frac{1}{2}h) = x(t) + \frac{1}{2}hf(x,t)$$

$$x(t+h) = x(t) + hf(x(t+\frac{1}{2}h), t+\frac{1}{2}h)$$

- The leapfrog method instead uses the previous half step to calculate the next 3/2h step.

$$x(t+\frac{3}{2}h) = x(t+\frac{1}{2}h) + hf(x(t+h), t+h)$$

$$x(t+2h) = x(t+h) + hf(x(t+\frac{3}{2}h), t+\frac{3}{2}h)$$



The top figure shows the Runge-Kutta method where after each step one uses a half step to calculate the next step. In the leapfrog method one takes a full step from the starting half step and uses each of those middle steps to evaluate the full steps.

LEAPFROG METHOD

- This method can be extended to multiple variables and functions just like Runge-Kutta and do higher order derivatives. So the obvious question is what do we gain over Runge-Kutta by using the leapfrog method.
- The leapfrog method has two significant features that make it interesting:

1. The method is time-reversal symmetric. This can be useful for physics problems where energy conservation is important.
2. The error is even in step size h, which makes it an ideal starting point for Richardson extrapolation.

TIME REVERSAL SYMMETRY

- The leapfrog method is time symmetric because if we started at some point and evaluated forward in time to some t' and then applied the algorithm in reverse with step size -h we would end up where we started (aside from rounding error)
- This is not true for 2nd order Runge-Kutta, if you perform the algorithm in reverse each step with have errors of $O(h^3)$ compared to what you got in the forward direction.
- Why is time reversal symmetry important, because if you have a system that should conserve energy, numerical errors will cause the energy to drift. But if your solver is time symmetric, then the energy will be conserved.

Energy of nonlinear oscillator

2nd-order Runge-Kutta

Leapfrog

Total energy

0.01
0
-0.01

0.04
0
-0.04

0
2
4
6
8
10

Time

ENERGY CONSERVATION WITH LEAPFROG METHOD

- It is important to understand how energy conservation works with the leapfrog method.
- First energy is not conserved, it will vary during the calculation because of numerical errors.
- The energy will only be conserved after the completion of a periodic event. That is an event that you could go forward to get to the next one, or backward to get to the previous one. This is the only time where the energy in conserved because of the time symmetry.
- However, if these conditions are met then the energy will return to the same value every period which can prevent drifts away from the original energy. If calculating over many many cycles this can greatly increase accuracy.

VERLET METHOD

- Suppose we want to use the leapfrog method to solve the equations of motion of a system:

$$\frac{d^2x}{dt^2} = f(x,t)$$

- We can solve this in our usual way

$$\frac{dx}{dt} = v \qquad \frac{df}{dt} = f(x,t)$$

- Now instead of our normal approach of creating a vector let us just write

$$x(t+h) = x(t) + hv(t+\frac{1}{2}h)$$

$$v(t+\frac{3}{2}h) = v(t+\frac{1}{2}h) + hf(x(t+h), t+h)$$

- Notice that x only depends on integer h while v only depends on 1/2 integer h. So in this case that is an improvement over the vector method.

VERLET METHOD

- This simplification only works for equations of motion or differential equations that have this special structure where the right-hand side of the first equation depends on v but not on x and the right-hand side of the second equation depends on x and not on v. This situation is not unusual in physics.
- One minor problem is that if you want to calculate some quantity the depends on x and v like potential energy you don't know both of them at the same step. You can however calculate v(t+h) from v(t+h/2) like this.

$$v(t+h) = v(t+\frac{1}{2}h) + \frac{1}{2}hf(x(t+h), t+h)$$

- This method is called the Verlet method. Putting the equations together one gets.

VERLET METHOD

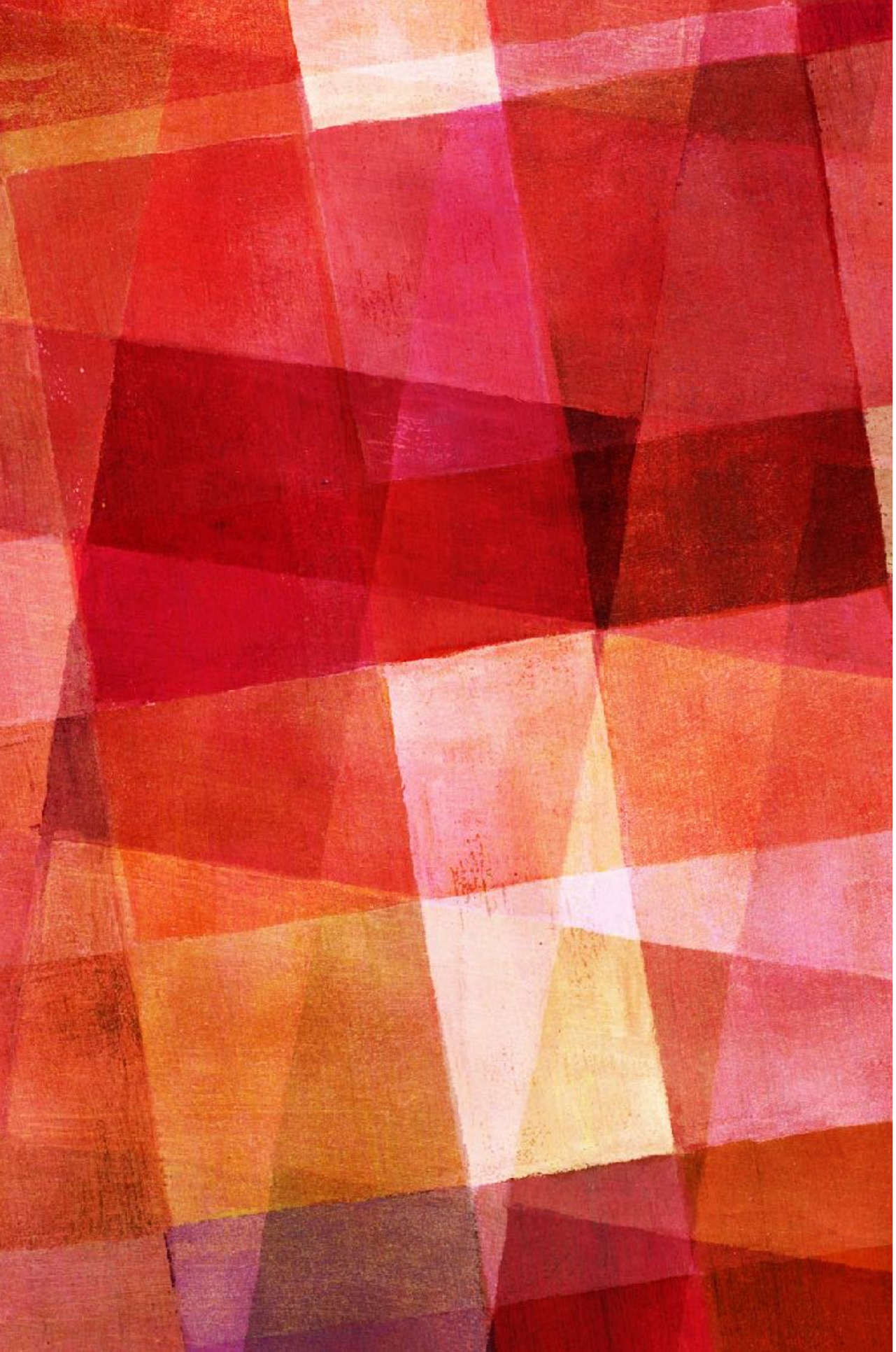
- If

$$\frac{d^2\mathbf{r}}{dt^2} = \mathbf{f}(\mathbf{r}, t)$$

- then defining $\mathbf{v}$=d$\mathbf{r}$/dt.

$$\mathbf{r}(t+h) = \mathbf{r}(t) + h\mathbf{v}(t+\frac{1}{2}h)$$
$$\mathbf{k} = h\mathbf{f}(\mathbf{r}(t+h), t+h)$$
$$\mathbf{v}(t+h) = \mathbf{v}(t+\frac{1}{2}h) + \frac{1}{2}\mathbf{k}$$
$$\mathbf{v}(t+\frac{3}{2}h) = \mathbf{v}(t+\frac{1}{2}h) + \mathbf{k}$$

- The 3rd equation is only needed if you want to know both $\mathbf{r}$ and $\mathbf{v}$ at t+h. And of course t+3/2h = t' + 1/2h so one can keep stepping forward.

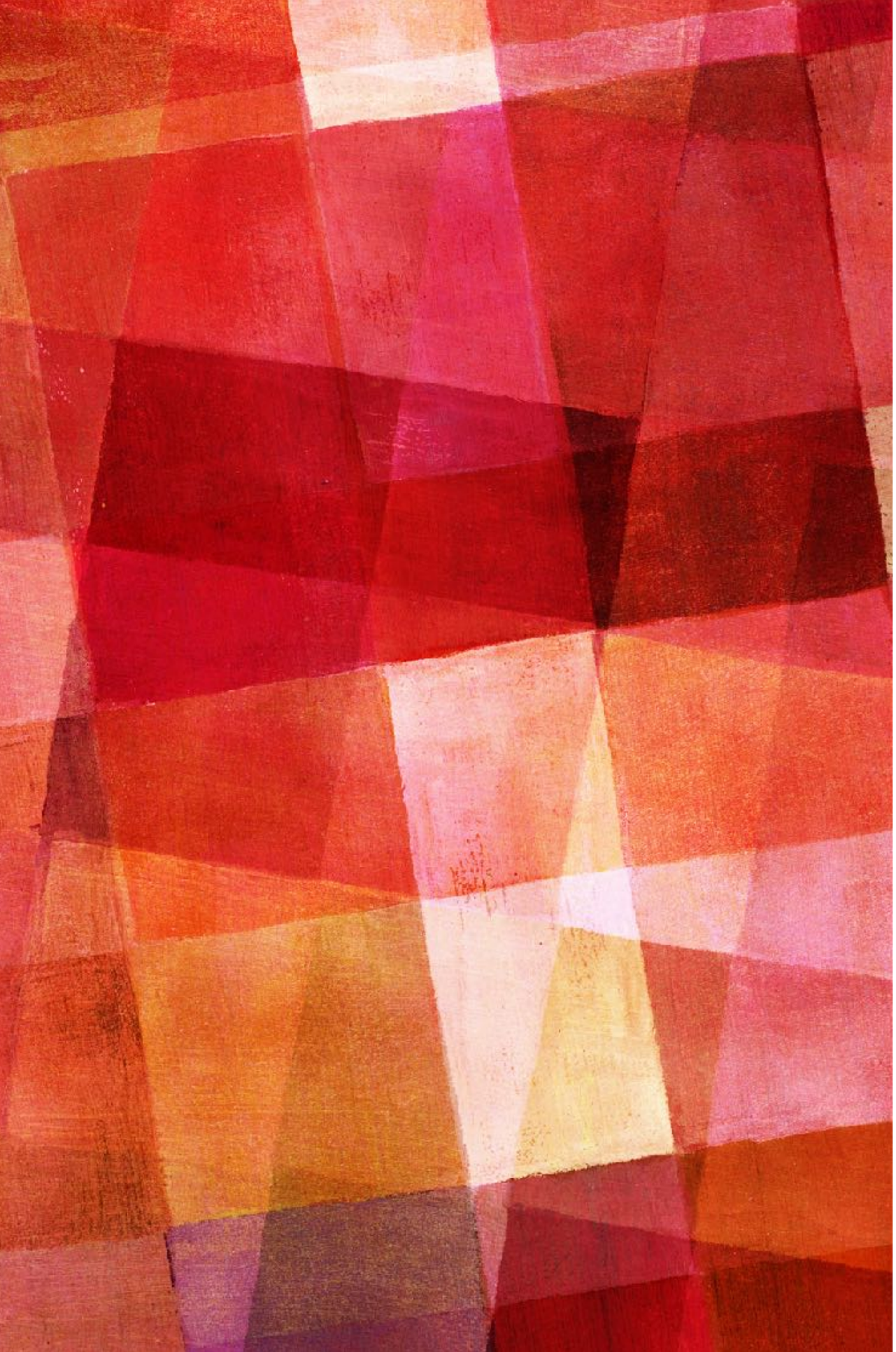


EXERCISE 8.12 ORBIT OF THE EARTH

- Use the Verlet method to calculate the orbit of the Earth around the Sun. The equations of motion for the position $\mathbf{r}=(x,y)$ of the planet are the same as those for any orbiting body. In vector form they are

$$\frac{d^2\mathbf{r}}{dt^2} = -GM\frac{\mathbf{r}}{r^3}$$

- where G=6.6738E-11, M=1.9891E30 kg.
- The orbit of the Earth is not perfectly circular. When it is at closet point it is moving precisely tangentially at a distance of 1.4710E11 m with a velocity of 3.0287E4 m/s.
- Write a program to calculate the orbit of the Earth using the Verlet method with a time step of h=1 hour. Make a plot of the orbit for several complete revolutions.



EXERCISE 8.12 ORBIT OF THE EARTH

- The gravitational potential energy of the Earth is -GMm/r where m = 5.9722E24 kg is Earth's mass. It's kinetic energy is $1/2mv^2$. Modify your program to calculate both of these at each step along with the total energy and plot the three as a function of time. You should find that the potential and kinetic energy vary visibly during the course of an orbit, but the total energy remains constant.

- Now plot the total energy alone without the others and you should be able to see a slight variation over the course of an orbit. Because you are using the Verlet method, however, which conserves energy in the long term, the energy should always return to its starting value.

MODIFIED MIDPOINT METHOD

- The leapfrog method offers another advantage, the total error after summing all steps is an even function of the step size h.
- Since the leapfrog method is time symmetric that means if we go a step forward and then back the error should be the same but opposite in sign. $\epsilon(h) = -\epsilon(-h)$
- Thus it must be an odd function, so only containing odd terms in h.

$$\epsilon(h) = c_3 h^3 + c_5 h^5 + c_7 h^7 + ...$$

- Now when we sum up over many steps we lose a power of h so the error should contain only even terms in h starting with h^2. The one problem with this is the first step we took which uses Euler's method and thus has odd powers of h.

MODIFIED MIDPOINT METHOD

- To solve this problem we can use what is called Gragg's method or more commonly the modified midpoint method.
- Let us assume we want to solve our differential equation from some time t to t+H where H is not small. We use n steps of size h=H/n each. We define

$$x_0 = x(t)$$

$$y_1 = x_0 + \frac{1}{2}hf(x_0, t)$$

- then

$$x_1 = x_0 + hf(y_1, t + \frac{1}{2}h)$$

$$y_2 = y_1 + hf(x_1, t + h)$$

$$x_2 = x_1 + hf(y_2, t + \frac{3}{2}h)$$

MODIFIED MIDPOINT METHOD

- More generally $$y_{m+1} = y_m + hf(x_m, t + mh)$$ $$x_{m+1} = x_m + hf(y_{m+1}, t + (m + \frac{1}{2})h)$$

- The last two points in the solution will be y_n = x(t+H - 1/2h) and x_n=x(t+H). But we can also calculate a final value from y_n. $$x(t + H) = y_n + \frac{1}{2}hf(x_n, t + H)$$

- We now have two ways to calculate x(t+H). We can combine the two and get. $$x(t + H) = \frac{1}{2}[x_n + y_n + \frac{1}{2}hf(x_n, t + H)]$$

- If we use this equation to calculate the final term. Then odd terms of h that came from the first step are cancelled out.

BURLISCH-STOER METHOD

- The modified midpoint method is rarely used alone since it doesn't do anything better than the leapfrog method. However, combined with Richardson extrapolation it becomes the powerful Burlisch-Stoer method.
- Let's start with a simple ODE $dx/dt = f(x,t)$ with a $x(t)$ and we want to solve the equation to $x(t+H)$.
- First we calculate the solution using the modified midpoint method to $t+H$ using one step (that is $h=H$). Let's call this solution $R_{1,1}$ (R for Richardson).
- Now we perform the same calculation with half the step size $H/2$ and call that calculation $R_{2,1}$.

BURLISCH-STOER METHOD

- Now since we've calculated the same thing they should be equal within the error

$$x(t+H) = R_{1,1} + c_1 h_1^2 + O(h_1^4) = R_{2,1} + c_2 h_2^2 + O(h_2^4)$$

- since $h_1 = 2h_2$ we get $$c_1 h_2^2 = \frac{1}{3}(R_{2,1} - R_{1,1})$$
- we can use this to get a new estimate of x(t+H)

$$R_{2,2} = R_{2,1} + \frac{1}{3}(R_{2,1} - R_{1,1})$$

- We can now use a step size of $h_3 = H/3$ to get R3,1 and then use the same argument to get

$$R_{3,3} = R_{3,2} + \frac{16}{65}(R_{3,2} - R_{2,2})$$

- our error is now of order h^6 because the modified midpoint method only has even powers in the error.

BURLISCH-STOER METHOD

- We can continue this as many times as we want each time reducing the error by h^2. As a general formula we have

$$x(t+H) = R_{n,m} + \frac{R_{n,m} - R_{n-1,m}}{[n/(n-1)]^{2m} - 1}$$

- Where the error goes as $O(h_n^{2m+2})$
- We can calculate the remaining error at each step and then stop decreasing our step size once we have the accuracy we desire.
- If this reminds you of Romberg integration that is because it is exactly the same Richardson expansion that we use to get the increased accuracy.

BURLISCH-STOER METHOD

- In practice then the method is the following. Choose and accuracy δ and then divide the interval you want to solve over into N regions of length H. Apply the following steps to get a solution in each region.

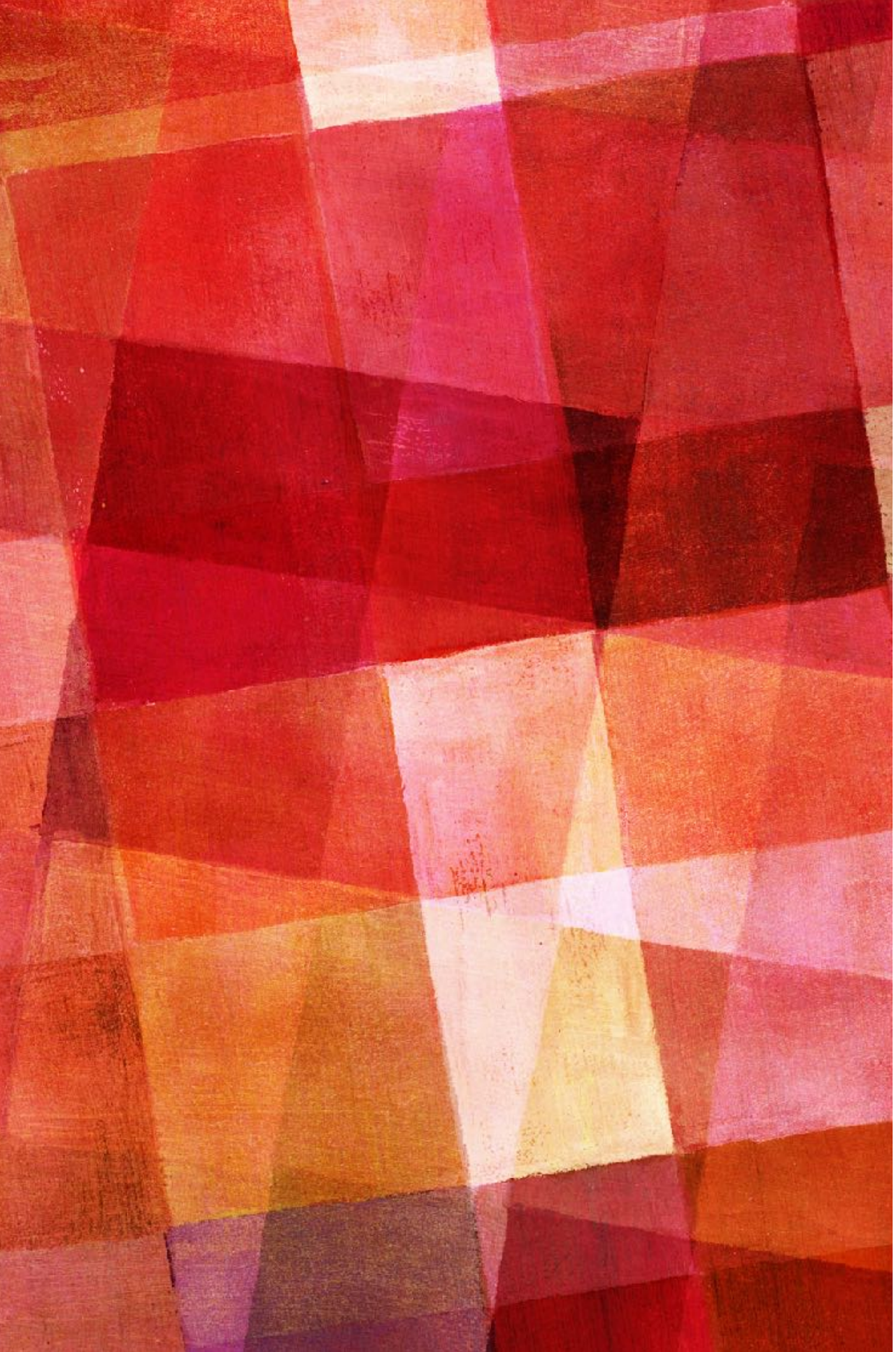
1. Set n=1 and used modified midpoint to estimate $R_{1,1}$ using just one step.
2. Increase n by 1 and use a new modified midpoint with that many steps.
3. Use the equation to calculate Rn,2…Rn,n for that step.
4. Compare the error with the target accuracy δH, if bigger go back to step 2.

BURLISCH-STOER METHOD

- This method can provide very accurate results, even better than 4th order Runge-Kutta with adaptive step sizes. It is however more difficult to code.
- One thing to note, the method will only work will if the error converges fairly quickly with each step.
- Poorly converging solutions will take many steps and thus give no performance increases over Runge-Kutta.
- Usually this will depend on how well behaved is the solution. If it is smoothly varying you should be fine. IF it has pathologies like discontinuities, infinities, etc. then this method is unlikely to give good results.

CHOOSING THE STEP SIZE

- For the Burlisch-Stoer method we only want to do 8-10 steps in each interval H. Thus how to choose the size of H is of some importance.
- What is usually done is one starts with some initial guess for H. Then one performs Burlisch-Stoer for that region, but a maximum refinement is chosen like 8. If the accuracy hasn't been reached by this level, instead of performing another step one halves the original size of the region H to H/2.
- Repeating this each time one find a value of H for each region that need not be the same that gets one to the desired accuracy within ones maximum defined refinement criteria.



BOUNDARY VALUE PROBLEMS

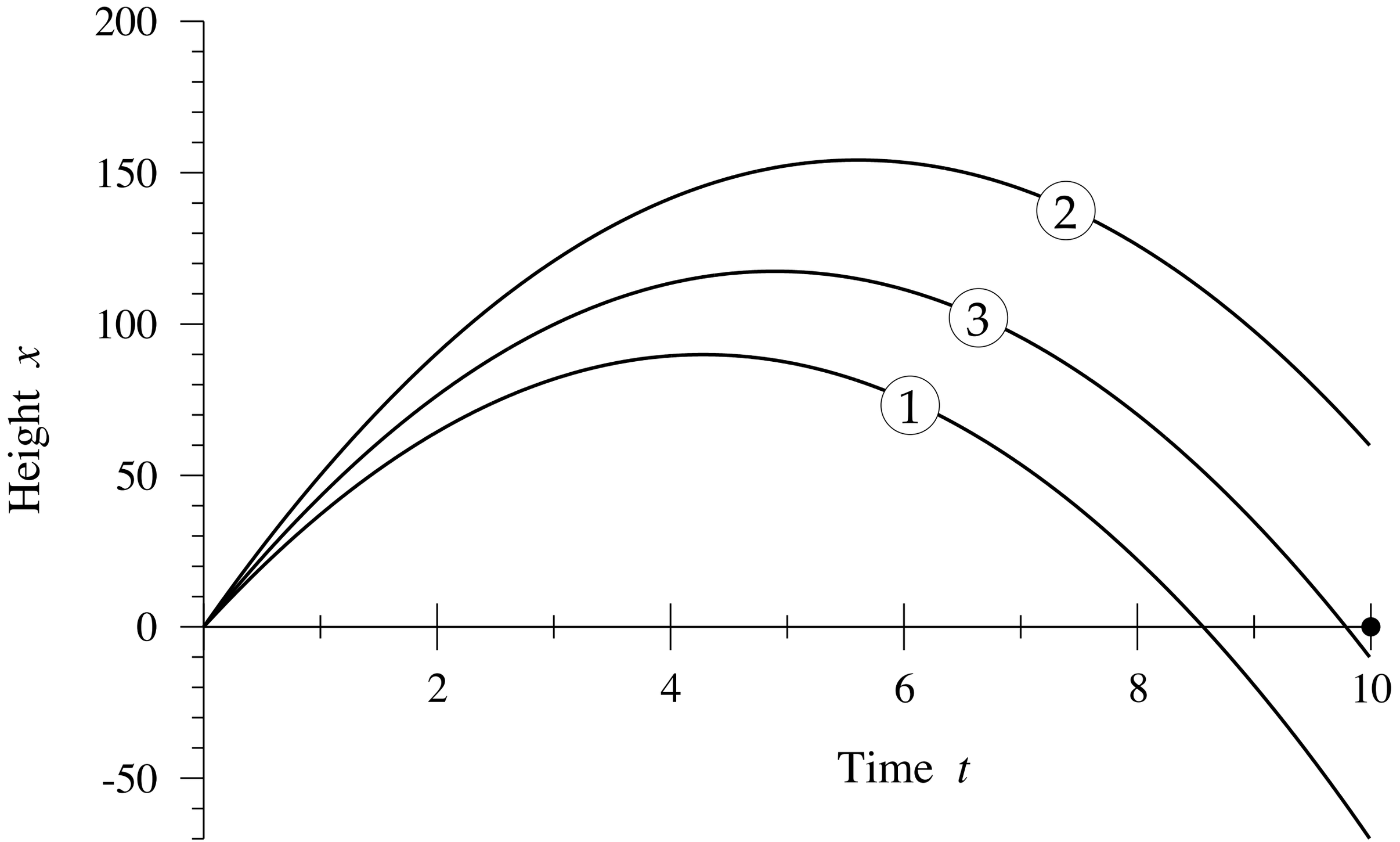
- Shooting Method
- Relaxation Method
- Eigenvalue Problems

BOUNDARY VALUE PROBLEMS

- So far we have discussed *initial value problems*, meaning solving ODEs where we know the initial values of the problem.
- Now we turn to *boundary value problems*, ODEs where we don't know the initial condition.
- Consider for example the differential equation governing free fall.
- $$\frac{d^2x}{dt^2} = -g$$
- To solve this equation we need two conditions because it is 2nd order. We could start with the initial position and velocity of an object say a ball. This would be initial value.
- However, our condition could also be a position at time t_1 and a position at time t_2. Then we are asking for what initial velocity would result in the ball being at the required position at t_2.

THE SHOOTING METHOD

- The shooting method is a trial-and-error method that searches for the correct values of the initial conditions that match a given set of boundary conditions.
- Let's consider the thrown ball in free fall. We start by guessing an initial velocity for the ball. We then see where it would be at t_2. It is unlikely to be at the correct position, it will probably overshoot or undershoot the right answer. So all we need to do is modify our first guess and try again.
- How do we do this? There is some function $x=f(v)$ that gives the balls height as a function of the initial velocity. If we want the ball to have a height of zero then we want to find the roots of the function $f(v)$.


200
150
100
50
0
-50
Height x
2
4
6
8
10
Time t
1
2
3

THE SHOOTING METHOD

- So to use the shooting method we use a root finding method like binary search. We have a starting v and we use an ODE method like Runge-Kutta to solve f(v).
- We then use the binary search to get a new guess for v and Runge-Kutta to solve f(v).
- After enough iterations we find the v that gives $f(v)=0$ which is the solution we are looking for.
- Any of the root finding methods and ODE methods can be used for this, binary search and Runge-Kutta are only mentioned as examples.

RELAXATION METHOD

- There is another method for solving boundary value problems called the relaxation method. With this method one starts with a guess of the shape of the solution that meets the boundary conditions.
- The function isn't an actual solution to the equation. What one does is then adjust (relax) the guess of the solution until it gets closer and closer to an actual solution to the ODE.
- This method is used more commonly for solving partial differential equation, so we will delay discussing it till we cover PDEs.

EIGENVALUE PROBLEMS

- A special type of boundary value problem occurs when the equations being solved are linear and homogenous, meaning every term in linear in the dependent variable. An example of this is the time-independent Schrondinger equation:

$$-\frac{\hbar}{2\pi}\frac{d^2\psi}{dx^2} + V(x)\psi(x) = E\psi(x)$$

- In this case every term in the equation is linear in $\psi(x)$.
- Let's consider the problem of a particle in a square potential well with infinitely high walls.

$$V(x) = 0 \quad \text{for} \quad 0 < x < L$$
$$V(x) = \infty \quad \text{elsewhere}$$

- The particle can't be where $V=\infty$ so this seems like a normal boundary value problem $\psi(0)=\psi(L)=0$ we could try to solve it using the shooting method.

EIGENVALUE PROBLEMS

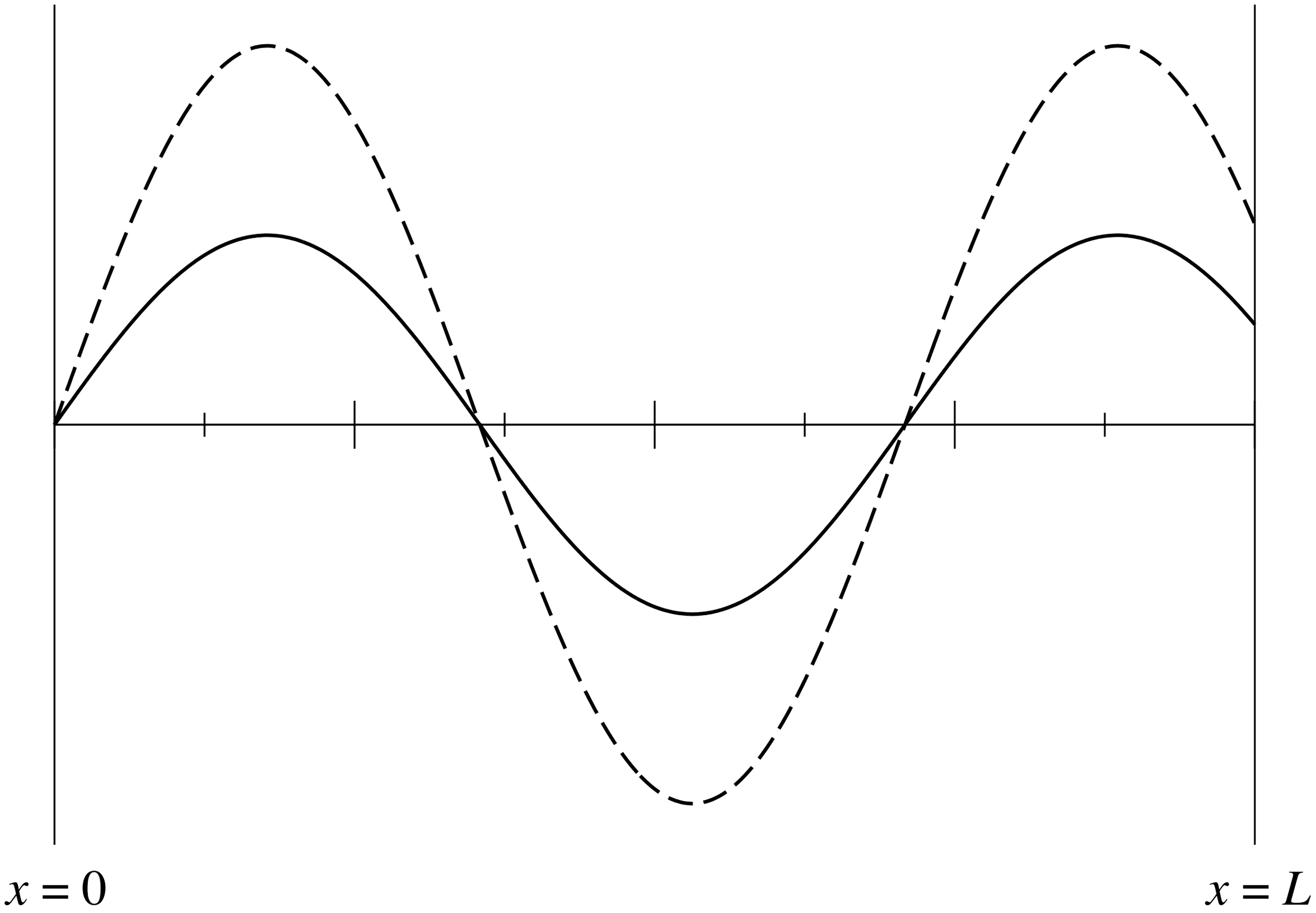
- We would start by turning it into 2 first-order equations:

$$\frac{d\psi}{dt} = \phi \qquad \frac{d\phi}{dt} = \frac{2m}{\hbar^2}[V(x) - E]\psi$$

- For our first guess we need two values, we know $\psi(0)=0$ and we can guess some value for φ.

- We want $\psi(L)=0$, if our first guess didn't work (which it doesn't) we would then want to vary φ. However, it turns out that won't work. No value of φ will give us $\psi(L)=0$.

- That is because the Schrondinger equation is a linear equation. If we double our value of φ we will get double the value of $\psi(L)$. But no multiplicative factor will make $\psi(L)=0$.


$x = 0$
$x = L$

EIGENVALUE PROBLEMS

- The fundamental problem in this case is that there is no general solution to this problem. There is no general solution that satisfies the boundary conditions.
- For this case there are only solutions for specific values of the energy E, a common occurrence in quantum mechanics. Thus we want to solve for the eigenvalues of E where solutions exist. Once we have the eigenvalues we can solve for $\psi(x)$.
- One way to do this is to use something like the shooting method, but to vary E instead of varying the dependent parameters.

EIGENVALUE PROBLEM

- So we can think of the solution of the Schrondinger equation as giving us a function f(E) equal to the value of the wave function at x=L. We then want to find the value of E that makes the wave function zero at L. That is we want to find the roots of the function f(E).
- What about the other unknown boundary condition $\varphi = d\psi/dt$? It doesn't matter, if we have a solution that is zero at x=L, it is zero for any value of φ. Traditionally this factor is fixed by requiring the wave function to be normalized so that the integral over the wave function is one (like a probability). If this is our case we can normalize it like this which gives a value for φ.

SCIPY INTEGRATE

- The scipy integrate subpackage contains ode solvers, in particular Runge-Kuta, but also implicit solvers. These are all for initial value problems.
- The basic wrapper is the function is solve_ivp(fun, (ti,tf), y0, method = 'RK45'). This will solve an ode of form dy/dt = fun(t,y), over the interval ti to tf where y(ti)=y0. fun can be a vector function but you must set keyword vectorized=True.
- Note that our more physics relevant ode solvers (leapfrog, verlet) are not part of these standard libraries. We are getting to a level of complexity where standard solution are starting to become discipline specific.