My additional goal was to simulate planets approaching a star from infinity. At t = 0 five planets are placed at y = 1 AU and x = 1.2 AU – 2.0 AU, in 0.2 AU steps. The planets all have the same initial velocity, which points in the negative y direction. The subsequent trajectory, energy and angular momentum of each planet is then calculated, and stored in files. These can then be plotted using gnuplot to create an animation of the trajectories and a plot of energy or angular momentum against time.

The program calculates the motion of each planet using two methods: Euler’s method and the Leapfrog method. As each method uses almost the same steps, functions were created to be used for each method. The functions were written in separate files, and all the files were combined using a makefile. The function files were: ‘calcR.cc’, which calculates the distance from the planet to the star, ‘force.cc’, which calculates the gravitational force on the planet, ‘pos.cc’, which calculates the position of the planet, ‘vel.cc’, which calculates the velocity of the planet, and ‘tim.cc’, which increments the total time. The position of each planet are stored in a file ‘euler.txt’ for Euler’s method and ‘leapfrog.txt’ for the Leapfrog method. These files are accessed by gnuplot scripts – ‘plot.gnu’ and ‘planets.gnu’ for Euler’s, ‘lfplot.gnu’ and ‘planets.gnu’ for Leapfrog – which produce png images of the total path traced by each planet as time progresses. These images can then be collected into a gif. The energy and angular momentum of each planet are calculated using functions in files ‘ene.cc’ and ‘ang.cc’. These are then stored in ‘eeng.txt’ and ‘eang.txt’ for Euler’s, and ‘leng.txt’ and ‘lang.txt’ for Leapfrog. These can be plotted in gnuplot using ‘energy.gnu’ and ‘angmom.gnu’ for Euler’s and ‘lfenergy.gnu’ and ‘lfangmom.gnu’. The details for each planet (mass, position, velocity, and force), were stored in an array of structs. The declaration of this struct, as well as the declaration of each function, and all global constants, were written in a header file ‘planets.h’.

The system being simulated is five planets travelling in parallel towards a star. The planets were all given the mass of the Earth, and the star the mass of the Sun, as I thought that using a real-life gravitational system as a basis would help determine whether any issues were with the maths or with the coding. The planets all took hyperbolic trajectories. This is to be expected: the planets would not be able to form bound orbits without a secondary impulse, which the program did not provide, and there is a very slim chance of the parameters being right for a parabolic trajectory. Despite starting parallel, the final trajectories are not parallel. This is a result of the initial energies of each system being the same. The closes approach of each planet will therefore not be in a straight line (proportion of kinetic and potential energy of each planet will be different), causing the trajectories to cross. Interestingly the energy and angular momentum plots of the two methods are different, despite the animations being more or less the same. The plots for the Leapfrog method were straight lines. This should not be the case, as the force on the planet changes over time. There may be a bug in the code, otherwise the Leapfrog method should not be used for this purpose. The energy graph with the Euler’s method showed a potential well that decreased in depth and appeared later as the initial distance increases. If the kinetic energy is neglected (the initial velocities were quite small) then this plot works well to describe the potential energy of the planets: the shift in well is due to the force on each planet decreasing as initial distance increases, and the asymptote at *E* = 0 is indicative of an escaped particle. The angular momentum graph shows an ‘S’ curve for each planet. The sharp increase makes sense as the planets are quickly swung by the star’s gravity, but the non-zero plateau’s do not make sense as the planets do not start and end the simulation orbiting. This is likely due to the angular momentum being measured relative to the star, which makes little sense for an object not orbiting it.

Overall this project has helped teach me a lot about hyperbolic orbits. I have had little experience with these in the past, and this project has helped me to visualise the effects. The programming has taught me how to use structs and most importantly how to split code into separate files. The practice should help reduce debugging times in future projects.

My solution was in files named ‘planets.cc’, ‘calcR.cc’, ‘force.cc’, ‘pos.cc’, ‘vel.cc’, ‘tim.cc’, ‘ene.cc’, ‘ang.cc’, and ‘planets.h’. These were combined with a file named ‘Makefile’. The gnuplot scripts were in files named ‘planets.gnu’, ‘plot.gnu’, ‘lfplanets.gnu’, ‘lfplot.gnu’, ‘energy.gnu’, ‘angmom.gnu’, ‘lfenergy.gnu’, and ‘lfangmom.gnu’.