ASP3012
2009

Stars Lab 1: Solving ODEs on Computers – The Lane-Emden Equation

1 Introduction

Much of theoretical astrophysics involves the numerical solution of various differential equations, both ordinary DEs (ODEs) and partial DEs (PDEs). Our aim here is to solve the Lane-Emden equation from lectures. This is a second order ODE. Most of this lab will teach you the steps along the way to solving this equation. We will assume only minimal knowledge about numerical methods and teach you as we go.

The computing environment will be linux. We realise that few of you know much about linux yet and even less about programming. Hence for these labs you will essentially run existing codes rather than write new codes. These codes are written in FORTRAN which was invented specifically for scientific programming. The tutor will help you with the linux commands you need, but everything will be quite simple. Trust me.

You will also be given a set of notes on ODEs from a course I used to teach years ago. These should be used as a reference for more info if required. There are some nice examples there.

2 The General First Order ODE Problem

Our general problem is to solve

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

with the initial condition that at x_0 we have $y = y_0$. See the ODE notes for more info. We will assume that we are to integrate this equation from $x = a$ to $x = b$ with the initial condition thus being that $y(a) = y_0$.

3 The Euler Method

One can derive the Euler method geometrically, as is done in section 11.1 of the ODE notes. An alternative derivation is possible from the Taylor Series:

$$y(x + h) = y(x) + hy'(x) + \frac{h^2}{2}y''(x) + \dots \tag{1}$$

If we neglect terms of order h^2 (i.e. we "truncate"the series here, so that the "(local) truncation error" is $O(h^2)$):

$$y(x + h) = y(x) + hy'(x) \tag{2}$$

Note that if we define discrete points x_i and y_i by

$$x_i = a + ih \tag{3}$$
$$y_i = y(x_i) \tag{4}$$

then we can write

$$y_{i+1} = y_i + hf(x_i, y_i). \tag{5}$$

We will typically integrate this equation from $x = a$ to $x = b$ with n steps of size $h = (b - a)/n$. Note some salient features:

- at *each step* we introduce an error due to truncation of the series, called the "local truncation error", which is $O(h^2)$ in this method.
- the complete integration involves n steps, so that the "global" error is $\sim nh^2 \sim (nh)h$, and since $nh = (b - a) =$ constant, then the "global" error is $O(h)$.

3.1 An Analytical Example

Now its time to show we know what we are doing. You will be given various things to do and these must be handed in to your tutor at the end of the lab.

Consider the ODE

$$\frac{dy}{dx} = y$$

with initial values $x_0 = 0$ and $y_0 = 1$.

Q1(a) Find the analytic solution.

Q1(b) Using a step length of $h = 0.2$ integrate the equation between $x = 0$ and $x = 1$. Present your results in a table with column headings i, x_i, y_i.

3.2 A Computed Example

Now log on to the computer and your tutor will explain how to get set up so you can run the exercises in this lab.

Run the program ODE1 by typing `ODE1`. This code will solve the DE $\frac{dy}{dx} = y$ with the Euler Method (as well as others we will discuss below). Enter 1 (and hit `ENTER`) to tell the code to use the Euler method.

It next requests the number of intervals to be used for the integration. Lets use 10 to start with. Just enter 10 and hit `ENTER`.

It then requests the initial and final x-value for the integration, as well as the initial y-value. Enter 0 1 1 and hit `ENTER`.

The code gives you the x and y values. It also compares these to the exact value, which for this DE and the given initial conditions is simply $y = e^x$.

3.3 Errors

Now we will have a look at the errors. We truncated the Taylor Series at terms involving h^2 so the truncation error *at each stage* is of order h^2 which we write as $O(h^2)$. So the error will accumulate at each step. We may be lucky and have the error change sign and hence decrease again. But that would be luck rather than anything else! We cannot rely on that happening.

Run the program ODE1err (ie type `ODE1err`) and enter the same inputs as before. Now you also get the percentage error (ignore the final column for now). The error does indeed increase in this case.

We can check the analysis above. If we are adding an error of $O(h^2)$ at each step i then if we divide the error by ih^2 then we should see something roughly constant. We need the factor i because there is something of $O(h^2)$ added at *each* step. The last column of your output does this and indeed we see that we get something very nearly constant.

Finally, over the n steps we will accumulate an error which is of order nh^2. But nh is just $b - a$ which is a constant. So the final error, at the final integration point $x = b$ will be $O(h)$. Lets check that this is true.

Q1(c) Run ODE1err with 10, 20 and 30 steps. Prepare a column of the errors at $x = 2$. Using these values, and the values of h used for the different n, show that the error at $x = 2$ is indeed proportional to h (or very nearly so).

4 The Improved Euler Method

The main source of error in the Euler method is that we ignored curvature across the intervals of width h. By making h smaller we do better. But is there another way? In the Improved Euler method we use an average gradient between x_i and x_{i+1} to step across the interval of width h. See the ODE notes, section 11.2.

We will use an average gradient across the interval. Hence we use:

$$y_{i+1} = y_i + h\left[\frac{f(x_i, y_i) + f(x_{i+1}, y_{i+1})}{2}\right]. \tag{6}$$

The problem with this equation is that it is *implicit*, with y_{i+1} appearing on both sides of the equation. To avoid this problem we will **predict** a value of y_{i+1} to use on the right-hand-side, and then use this to provide a **corrected** value. We predict by using the Euler method:

$$\bar{y}_{i+1} = y_i + hf(x_i, y_i) \tag{7}$$

and then use this in the above formula for y_{i+1}:

$$y_{i+1} = y_i + h\left[\frac{f(x_i, y_i) + f(x_{i+1}, \bar{y}_{i+1})}{2}\right]. \tag{8}$$

Lets look at the same example as before, $\frac{dy}{dx} = y$, with $x(0) = 1$. Run ODE1 and this time enter 2 for the Improved Euler method. Again use 10 intervals and $a = 0$, $b = 1$, $y(0) = 1$.

Q2(a) Compare the values of $y(2)$ using the Euler and Improved Euler with $n = 10$. Use ODE1err to determine the percentage errors in the two cases.

So the Improved Euler method has done a much better job, as expected. One can show (see the ODE notes) that the local truncation error of this method is $O(h^3)$. Hence the global error, after n steps, should be of $O(h^2)$ since we accumulate n errors of $O(h^3)$, but nh is a constant. The last column in the output from ODE1err for the Improved Euler method is the error divided by ih^3 and it is indeed very nearly constant.

Q2(b) Use ODE1Err to show that the global error for the Improved Euler method is proportional to h^2.

5 The 4th order Runge-Kutta Method

The Euler and Improved Euler methods are particular examples of a class of techniques known as "Runge-Kutta" methods, which have the general form

$$y_{i+1} = y_i + h\phi \tag{9}$$

where ϕ is some approximation to the slope. For example:

1. $\phi = k_1$ where $k_1 = f(x_i, y_i)$ is the Euler Method.
2. $\phi = \frac{1}{2}(k_1 + k_2^\star)$ where $k_2^\star = f(x_i + h, y_i + hk_1)$ is the Improved Euler Method.

The most popular method of this class is the 4^{th} Order Runge-Kutta Method (hereafter RK4), which has a global error of order h^4 which means that the local truncation error is $O(h^5)$. In this method we take:

$$k_1 = f(x_i, y_i) \tag{10}$$
$$k_2 = f(x_i + \tfrac{h}{2}, y_i + \tfrac{hk_1}{2}) \tag{11}$$
$$k_3 = f(x_i + \tfrac{h}{2}, y_i + \tfrac{hk_2}{2}) \tag{12}$$
$$k_4 = f(x_i + h, y_i + hk_3) \tag{13}$$

and

$$y_{i+1} = y_i + h\phi$$

where

$$\phi = \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right).$$

ODE1 also has the RK4 method coded into it so you can easily use this method as well. Run ODE1 with n=10 and look at how close it is to the exact solution! Not bad, eh?

Run ODE1err and see the errors. Note that the truncation error is proportional to h^5 so the final column gives the error divided by ih^5, and shows it to be very nearly constant, as expected.

Q3 Use ODE1Err to show that the global error for the RK4 method is proportional to h^4.

6 Computational Efficiency

We have seen that the RK4 method can be very accurate. We could try it on other DEs, but trust me, its quite a good method to use. One question though is the computational efficiency. Each step of the RK4 method has involved a lot more work (for the computer!) than each step in the Euler method. When the RK4 method is doing that extra overhead work per step we could be doing finer steps with the Euler method. So the question arises: for the same amount of computer time, which is the best method? Indeed, I was reviewing a book on numerical methods once and the author wrote "With fast modern computers there is no need to go to higher order methods. We are just as well off to use a low order method, like Euler, and use many steps." Is this true?

One good way to estimate the amount of work done by a method is to count the number of times that the function $\frac{dy}{dx}$ is evaluated. In the Euler method it was once per step. For the Improved Euler it was twice (a predictor step and a corrector step). For the RK4 method it was four times. (In general, an nth order method requires n function evaluations per step, but we will not prove this!) So that would imply that one step of the Euler method is about half as much work as one step of the Improved Euler. And that, similarly, one step of the RK4 method is about as much work as 4 steps of the Euler or 2 steps of the Improved Euler.

So how do the accuracies compare if we adjust the number of zones to allow for the work involved in the different methods?

I have written a program called PlotODE1 which solves our favourite DE $\frac{dy}{dx} = y$ with our usual initial condition ($x(0) = 1$) between $x = 0$ and $x = 1$. It uses n steps of the RK4 method, $2n$ steps of the Improved Euler and $4n$ steps of the Euler. So all use about the same amount of computer time. Which is the most accurate?

Run the program PlotODE1. You will be asked for the n value to use for the RK4 integration. The code will then use double that value for the Improved Euler and double that again for the Euler method. I would suggest using something like 4 initially. The values of a, b and $y(a)$ are the usual: 0, 1 and 1. This then outputs some numbers. Lets have a look at them.

First it gives the column of x values. Then it gives the y-values found using the Euler method. Next are the values given by the Improved Euler. But since we used half as many points, the values are given for every second x-value. The next column gives the RK4 values, and there is a value here only for every fourth x-value in the table, as we used a step-length of four times the Euler method. The last column gives the exact value, at all points.

As before you will notice that the RK4 method is giving excellent answers despite such a large value of $h = 0.25$. Compare that with the Improved Euler method. Despite using twice the number of steps ($h = 0.125$) it is clearly not as accurate. Now go to the Euler method and you can see substantial errors even though it is using four times as many steps as the RK4 method!

The next block of numbers gives the fractional errors and the log of the errors, for each of the three methods. Euler has typical errors of about 2% in this case. The Improved Euler has errors of about 0.2%,whereas the RK4 method has errors of about 0.002%. What a triumph! (Sorry. Got a bit carried away there...)

You will notice that the program is displaying a message from the plotting routine. It wants to know where to plot the graph, which is a plot of the different y-values for each method. The default is to draw it in a window on your screen so just hit `ENTER` or type in `/XWIN`. The white line and points are the Euler method results, the Improved Euler is in Red, the RK4 in green and the exact solution is in blue. You cannot see a lot here, except that the white curve is pretty poor compared to the rest.

The second graph plots log of the errors. You will need to move the "focus" back into the original window where you ran the program. Move the mouse over that window and click, then hit `ENTER`. Another `ENTER` will draw the plot for you. Note how low the errors are for the RK4 curve (green), at about 10^{-5} or 0.001%. To exit, move the focus back to the original window and hit `ENTER`.

Q4 Summarise what you have just seen, and make a comment on the claim made earlier in a textbook that higher order methods are not needed becasue we are just as well off using something like the Euler method but with lots of steps. How many steps does it take, in our example, for the Euler method to get to an error that matches the RK4 method with 10 steps? (Use ODE1err to determine this.)

7 The Runge-Kutta-Fehlberg Method

Wouldn't it be nice if we could just specify the maximum error we would tolerate and let the code choose the step it should use? That is the idea behind the Runge-Kutta-Fehlberg Method, hereafter RKF. The idea here is

that we use two approximations, one with truncation error of $O(h^5)$ and one of $O(h^6)$ (and hence global error of orders h^4 and h^5 respectively). It is then possible, combining these two estimates, to control the errors.

To see how this is done, consider running the Euler method twice with two different steps h and $h/2$. We know that the global error is proportional to the step-length to the power one. That is:

$$y(b) = Y(b) + \alpha h \tag{14}$$

where y is the Euler approximation, Y is the exact value, h is the step-length, b is the last point in the integration region, and α is the unknown constant of proportionality. Now suppose we repeat the calculation using $h/2$ as the step. Then clearly we have a new approximation $z(x)$ say with:

$$z(b) = Y(b) + \alpha h/2. \tag{15}$$

Subtracting these gives

$$y(b) - z(b) = Y(b) + \alpha h - Y(b) - \alpha h/2 = \alpha h/2 \tag{16}$$

But we know $y(b)$ and $z(b)$ and h so we can solve for the value of α. Hence we can improve either estimate for $Y(b)$ by using this value of α in either of the equations (14) or (15) above.

Q5a Determine an Euler estimate for the solution of $\frac{dy}{dx} = y$ at $x = 2$ given $x(0) = 1$. Use the code ODE1 to do this, using step-lengths of $h = 0.1$ and 0.05 to solve for α in equations (14) and (15) and hence determine a better answer than the one given by the code. Compare with the exact solution $y = e^x$.

Q5b Using the same principle (ands the same DE and initial conditions) use ODE1 again but this time with the Improved Euler method to improve on the solutions for $y(2)$ determined by the code. Note that the global error here is different to that quoted in equation (14)! You will need to derive a new version of equations (14) - (16) for the Improved Euler method (see the discussion following question 2(a)).

It is this sort of analysis that enables the RKF method to estimate the local error. Because the global error of the RKF method uses a 4th and a 5th order estimate it is often called the RKF45 method. One would normally require four function evaluations for a fourth order method (as we found for RK4) and a further five for a fifth order method. The one step involves nine function evaluations, and would be relatively expensive in computer time (per step). But Fehlberg found a way to get both estimates with only six function evaluations. So its rather efficient!

OK. Enough of the theory. Lets run an example.

Run program RKF45-1a, which will solve our standard example, $\frac{dy}{dx} = y$ with our usual initial conditions $x(0) = 1$ on the interval $0 \leq x \leq 1$. First you enter $a = 0$, $b = 1$ and $y(a) = 1$. It then asks for the maximum error you will accept. Lets use 1.e-6. It also wants a maximum h. I suggest 0.1 for now. The code will start with h equal to the specified maximum error (in this case 1e-6).

With those values you will see that the error is always below the specified 1.e-6, even when h reaches our maximum specified value of 0.1, which is used for the last few steps.

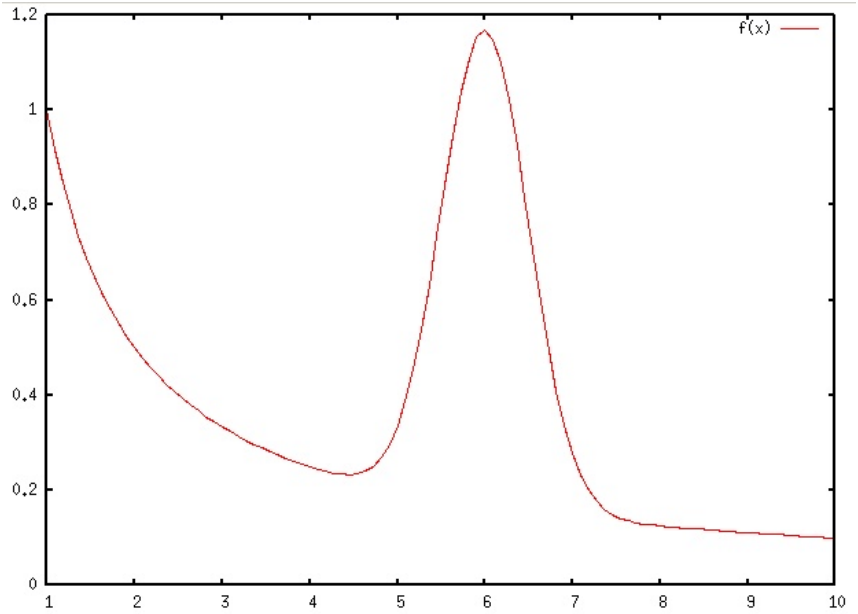
Lets try it again, but with a tougher function this time. Lets have a look at trying to solve the following DE:

$$\frac{dy}{dx} = \frac{1}{x^2} - 4(x-6)e^{-2(x-6)^2} \tag{17}$$

which has the analytic solution

$$y = \frac{1}{x} + e^{-2(x-6)^2}. \tag{18}$$

Note that this equation has a large bump at $x = 6$, after varying very smoothly prior to that. How will the RKF45 method go at handling this sudden change?



This function has been programmed into the code RKF45-1b, which you should run. The code requests the initial and final x-range for the integration, a and b as well as the initial value for $y(a)$. We will use $a = 1$, $b = 10$ and $y(1) = 1$ so enter: 1 10 1. For the maximum error try 1.e-6 and a maximum step-length of 1.

Now have a look at the output. You can see the error start to increase and when we get to $x = 3.38278$ the errors are determined to be too big with the current value of $h = 0.524$ so h is reduced. This stops the error getting above the 1.e-6 that we specified. Note that the error stays at about 1e-7 or 1e-8. The code continues to choose its value of h as determined by the RKF45 method, to keep the errors below 1e-6.

PlotRKF45-1b is essentially the same code except that it will plot the solution and the numerical approximation. Run it. From the first graph that the code draws, you can see that the RKF45 method does very well. The green line is the exact solution. One thing to consider of course is whether the spacing used by the RKF45 method is fine enough. That depends on the use you have for your solution. If you need a finer grid of (x, y) values then you can specify the maximum h so that you get something that you want. In any event, the RKF45 method does well. The second graph shows the errors, and they are indeed below the 1e-6 that we specified.

8 Higher-Order DEs

There is one more thing to do before we can tackle the Lane-Emden equation. So far we have been solving just one first order ODE. We need to generalise to handle an arbitrary number (well, two for the Lane-Emden equation, but the principle is the same and will allow you to solve any system or and arbitrarily high order ODE).

8.1 2nd Order ODEs with the Euler Method

The general second order DE is of the form

$$\frac{d^2y}{dx^2} = f(x, y, \frac{dy}{dx}) \tag{19}$$

We can write this as two first order DEs by defining

$$z = \frac{dy}{dx}. \tag{20}$$

Then our general second order DE is now a first order DE in three variables:

$$\frac{d^2y}{dx^2} = \frac{dz}{dx} = f(x, y, z) \tag{21}$$

These two equations (20) and (21) are equivalent to the second order DE but they are simply two first order DEs. We can solve them using whichever method we choose. Lets look at how to do this with the Euler method,

because it is the simplest. To make things slightly more general lets look at the two first order DEs:

$$\frac{dy}{dx} = f(x,y,z) \tag{22}$$
$$\frac{dz}{dx} = g(x,y,z). \tag{23}$$
$$\tag{24}$$

with the initial conditions

$$x = a \tag{25}$$
$$y = y_0 \tag{26}$$
$$z = z_0. \tag{27}$$

We will integrate from $x = a$ to $x = b$ with n steps of size $h = (b-a)/n$. We use the standard Euler iterative scheme:

$$x_{i+1} = x_i + h \tag{28}$$
$$y_{i+1} = y_i + hf(x_i, y_i, z_i) \tag{29}$$
$$z_{i+1} = z_i + hg(x_i, y_i, z_i) \tag{30}$$

The only subtle thing to note is that we must update y_i and z_i at the same time. We cannot evaluate all the y_i and then go and evaluate all the z_i. That is inferred in the equations above anyway, but I mention it for completeness.

To make our arbitrary system equivalent to a 2nd order DE we just take

$$\frac{dy}{dx} = f(x,y,z) \quad = z \tag{31}$$
$$\frac{dz}{dx} = g(x,y,z) \quad = \frac{d^2y}{dx^2} \tag{32}$$

Program ODE2 solves the simple harmonic motion equation:

$$\frac{d^2y}{dx^2} + y = 0 \tag{33}$$

which has solution

$$y = A\sin(x) + B\cos(x).$$

Run ODE2. You can enter 4 to see solutions from Euler, Improved Euler and RK4 all at once. Take $n = 20$ intervals and enter the values $a = 0$, $b = 3.14159$, $y(0) = 0.0$ and $\frac{dy}{dx}(0) = z(0) = 1.0$. The code will provide the estimates from the 3 different methods as well as the analytic solution ($y = \sin(x)$ for the initial conditions used). As usual, the RK4 method does the best.

Q6 Now run PlotODE2. Use $n = 20$ for the RK4 method (the others will be increased so that a similar amount of computer time is spent on each method). Enter the values $a = 0$, $b = 6.29$, $y(0) = 0.0$ and $\frac{dy}{dx}(0) = z(0) = 1.0$ so that you integrate over an entire period of 2π. Why are the errors so large near $x = \pi$ and $x = 2\pi$?

9 The Lane-Emden Equation

So, all the previous work was meant as an introduction so that you will understand what we are doing to solve the Lane-Emden equation! But it is good revision and also teaches you some good methods!

Recall from notes that after scaling, the dimensionless Lane-Emden equation is

$$\frac{d}{d\xi}\left(\xi^2\frac{d\theta}{d\xi}\right) = -\xi^2\theta^n \tag{34}$$

Here ξ is the scaled radius and θ is the scaled temperature, which also determines the pressure and density because of the assumed polytropic equation of state $P = K\rho^\gamma$ with the polytropic index $n = 1/(\gamma - 1)$. We will look at the scaling and how to relate this to real stars in next week's lab. For now, lets just look at the solution.

We first need to write the 2nd order Lane-Emden equation as two first order equations. In the code we will use x for ξ, y for θ and z for $\frac{d\theta}{d\xi}$. The Lane-Emden equation is then

$$\frac{d}{dx}\left(x^2\frac{dy}{dx}\right) = -x^2y^n \tag{35}$$

If we expand the Lane-Emden equation we get:

$$x^2\frac{d^2y}{dx^2} + 2x\frac{dy}{dx} + x^2y^n = 0 \tag{36}$$

We write this as two first order equations by setting $z = \frac{dy}{dx}$. The two equations thus become:

$$\frac{dy}{dx} = z \tag{37}$$

$$\frac{dz}{dx} = -\frac{1}{x^2}\left(2xz + x^2y^n\right) \tag{38}$$

The program `le` uses these equations and a RKF45 method to solve the Lane-Emden equation. It begins with a series solution, as derived in lectures, to start the integrations. One specifies a tolerance or maximum error. This is also used as the starting value for x. You must also specify a maximum step-length h. Note that for polytropes with large radi (ie those with large n) you will need to make h larger or it can take a long time to complete the integration! For n larger than about 4 you should use h_{max} of about 1.

The code integrated the equations until y goes negative. It then steps back to the last positive value and decreases h and continues. It does this until h is as small as the maximum error. Then it stops and prints ou the final radius, and a couple of other useful things.

So now run `le`. Enter the polytropic index n. Lets start with $n = 1$ which has an analytical solution (recall from lectures that in this case $y = \sin(x)/x$) and the radius is π. You are also asked if you want to see the iteration data and how much of it. For now, lets look at it all. Enter 1 to see every step. Maximum step-length can be 1.0 or so as the code will keep it smaller when it needs to be to stay below the tolerance, which is the last thing you specify. Use 1.e-3 for now.

Note that the code did not need to decrease h to match the specified error. The final solution for the dimensionless radius is given at the bottom as is the central concentration, measured by $\frac{\rho_c}{\bar{\rho}}$.

Q7a We know that the analytic solution is $y = \sin(x)/x$ and the radius is the first root of the function y. So the dimensionless radius should be π. Is it? Discuss why/why not. How would you get a better answer?

Q7b You should now have determined a value for the maximum error that gives a good value, at least for $n = 1$. Now determine the radius and central concentration for polytropes with $n = 0, 0.5, 1, 1.5, 2, 3, 4, 4.5$. Present your results as a table of values.

Q7c You can use the program `gnuplot` to plot these results. Type the following commands:

```
> gnuplot
> set xlabel 'Polytropic Index n'
> set ylabel 'Dimensionless Radius xi'
> set title 'Sexy Graph'
> unset key
> plot [0:5] [0:32] '-' with linespoints pointsize 5
```

GNUPLOT will then expect you to enter data pairs (x,y) in the format `0 2.4494` with one x-value and one y-value per line, with a space between them. When you have finished simply enter 'e' on the last line and the graph will appear.
The [0:5] specifies the x-range. Feel free to adjust it.
The [0:32] specifies the y-range. Feel free to adjust it.
Using GNUPLOT, generate a graph of the radius vs polytropic index. Your tutor will show you how to save a copy of the graph for you to print and include with your report. You can enter the data values into a datafile and have GNUPLOT read it, if you prefer. Ask your tutor or type `HELP` in GNUPLOT.
Also generate a graph of central concentration vs polytropic index. Prepare one with linear axes, up to $n = 3$ and then one with a logarithmic y-axis to cover the values up tp $n = 4.5$.