

CA205 Assignment 2: Review Notes

Greg Gamble

email: gregg@maths.uwa.edu.au

WWW: <http://maths.uwa.edu.au/~gregg/CA205/>

October 22, 1997

1 Writing Skills

On the whole, there was a significant improvement in both the content and structure of submissions. However, many failed to grasp that the report (i.e. *Abstract* through to *Conclusion*) should be contained in the first two (or perhaps three) pages following the title page; it should contain only *text*, except for perhaps *small* figures and tables. *All* other tables, figures and computer code should be in appendices, and they should have appropriate headings, labels and numbering so that they can be referred to from the text of your report. When referring to appendicised material, particularly graphs, it is important to *interpret* the material. Don't just say '*the simulations appear as graphs X, Y and Z*'; rather say: '*the simulations appear as graphs X, Y and Z, from which we observe that there is such-and-such trend*'. A small point: though the structure as stated in the *Review Notes* for Assignment 1 stated there should be a *Body* there should be *no* actual heading *Body*.

2 Runge-Kutta vs Euler

In general, when modelling the heating/cooling of an electric iron, students observed a very small difference between the Euler and Runge-Kutta methods, but had no idea what *benchmark* to compare them to. Since the Euler and Runge-Kutta methods model the first order d.e.

$$K \frac{dT}{dt} = Q_{\text{in}} - h(T - T_{\text{amb}}),$$

the *benchmark*, to which the methods should be compared, is the closed form solution of this d.e.! Incidentally, rearranging this d.e., we obtain

$$\frac{dT}{dt} = -\frac{h}{K} \left(T - \left(\frac{Q_{\text{in}}}{h} + T_{\text{amb}} \right) \right),$$

from which you may be able to observe that T_{∞} , the limiting temperature as $t \rightarrow \infty$ is given by:

$$T_{\infty} = \frac{Q_{\text{in}}}{h} + T_{\text{amb}}.$$

In terms of this observation, the closed form solution can then be simply expressed as

$$T = T_{\infty} - (T_{\infty} - T_0)e^{-(h/k)t},$$

where T_0 is the initial value of T . This solution applies to any of the *heating* and *cooling* cycles, so long as for a given cycle, time t is measured from the beginning of the cycle and T_0 is the temperature at the beginning of that cycle. Observe also that for a *cooling* cycle that $T_{\infty} = T_{\text{amb}}$.

So what might have been an ideal comparison of the methods? . . . It would have been enough to provide computer code and graphical output (preferably with `zoom on` to show they differ) *and* provide an *interpretation* by giving the *percentage errors* of each final T of the cycle. Each *percentage error* should have been calculated as the absolute value of

$$\frac{T_{\text{final_observed}} - T_{\text{final_theoretical}}}{T_{\text{final_theoretical}} - T_0} \times 100\%.$$

3 Computer code

Students would have found the exercises far easier if they had modularised their programs by using functions, and to add features to previous programs to *generalise* their application. For example, the two lines

```
Trate = (Qin - h*(T - 20))/K;  
T = T + Trate * dt;
```

could have been replaced with

```
T = T + deltaT(T,dt);
```

where `deltaT` is a function contained in a file `deltaT.m`. This factors out the *Euler* component to another file; the program can then be converted to run an *R-K* simulation, simply by rewriting `deltaT.m`. Moreover, by modifying the original iron main program so that it itself is a function and modifying the `deltaT` line again to take an additional argument telling it which method to use, one can create a new main program that calls the old one twice – the first time using the *Euler* method and the second time using the *R-K* method. (A neat way of doing this is by using MATLAB’s `eval` function. If interested, ask me about it!) What’s more by pursuing this approach further one can obtain a program to model the spring that uses almost identical code to that of the iron (but without any on-off code). Note that because MATLAB ‘vectorises’ just about every conceivable operation possible, the main program for the spring can contain a line¹

```
X = X + deltaX(X,dt);
```

where `X` contains the components x (the spring displacement) and $y = \dot{x}$, and `deltaX` is defined by

¹Note the similarity with the iron program!

```

function [dX] = deltaX(X,dt)
k1 = eulr(X, dt);
k2 = eulr(X + k1/2, dt);
k3 = eulr(X + k2/2, dt);
k4 = eulr(X + k3, dt);
dX = (k1 + 2*(k2 + k3) + k4)/6;

```

which calls eulr which is defined by

```

function [dX] = eulr(X,dt)
global A;
dX = A * X * dt;

```

where A is defined in the main program to be

```
A = [0 1; -k/m -c/m];
```

Note that the second order d.e.

$$m\ddot{x} = -kx - c\dot{x}$$

can be rewritten as a pair of first order d.e.s by writing $\dot{x} = y$, which in matrix form is

$$\dot{\underline{\mathbf{X}}} = \begin{bmatrix} 0 & 1 \\ -k/m & -c/m \end{bmatrix} \underline{\mathbf{X}},$$

where

$$\underline{\mathbf{X}} = \begin{bmatrix} x \\ y \end{bmatrix}.$$