Computational Physics (ZCE 111)

Yoon Tiem Leong
USM School of Physics, Academic Year 2011/12, 2nd semester

Note

- The lecture notes are adopted from Giodarno and Nakanishi, Second edition.
- Programming tool: Mathematica
- Install your own Mathematica. For instruction, visit the course webpage.
- Visit the course webpage and to read more about what's computational physics.
- Read Giordano.

Chapter 1

A First Numerical Problem

1.1 Radioactive Decay

- A large sample of nuclei decays according to the first order equation
$$\frac{dN_U}{dt} = -\frac{N_U}{\tau}$$
- The analytical solution is given by
$$N_U = N_U(0)\, e^{-t/\tau}$$
- where τ is the mean lifetime for a particular nuclear species.
- Plot the graph $N_U = N_U(0)\, e^{-t/\tau}$ using *Mathematica*'s Plot command.
- In order to plot the graph, numerical values, you must supply the 'initial values', $N_U(0)$ and τ.
- For example, U-238, $\tau = 10^9$ yr.
- Note that to plot the function, you must not use the symbol "N" to name the function, since "N" is the symbol reserved for *Mathematica.* Use the symbol "Nu" instead of "N" (it's just a label anyway).

Plotting $N/N_U(0)$ vs. x

- You can also plot $N/N_U(0)$ as a function of time.
- This let you monitor the fraction of nuclei number remains as a function of t/τ.
- To do so, define $x = t/\tau$
- $\hat{N} = N / N_U(0) = e^{-\frac{t}{\tau}} = e^{-x} \Rightarrow$
- x is the measure of time in unit of τ.

Estimating the half life using Solve

- Can you estimate the time (in unit of τ) when the half of the original nuclei has decayed?
- Theoretically, this time is called the half-life, given by $e^{-t/\tau} = 1/2$, i.e., $t \equiv t_{1/2} = \tau \ln 2$, or $x = t/\tau = \ln 2$.
- How to obtain $t_{1/2}$ using *Mathematica*? This could be solved using the function Solve.
- We ask the Mathematica to solve the equation for the value of t such that $f(t) = e^{-t/\tau} = ½$.

Estimating the half life using Solve (cont.)

- What is the time t' taken for 75% of the total nuclei has decayed?
- This means only 35% = 0.35 of the nuclei remains at time t'.
- Solve for t' using Mathematica using Solve.
- In general, you can write a code using Solve to calculate what is the time taken for the sample to decay to any given fraction (specified by $\hat{N}$)

NDSolve

- *Mathematica*'s NDSolve function can be used to solve the equation $dN/dt=-(N/\tau)$.
- This is a numerical operation.
- To use NDSolve, numerical values of the initial or boundary conditions for the differential equation must be supplied, e.g., Nu[0]==N0.

Euler's method

- Now, we will write our own numerical program to solve the equation dN/dt=-(N/t), instead of using NDSolve, NSolve.

Our goal is to obtain N_U as a function of t.

Given the value of N_U at one particular value of t

we want to estimate its value at later times.

Taylor expansion for N_U

$$N_U(\Delta t) = N_U(0) + \frac{dN_U}{dt}\Delta t + \frac{1}{2}\frac{d^2N_U}{dt^2}(\Delta t)^2 + \cdots$$

$$N_U(\Delta t) \approx N_U(0) + \frac{dN_U}{dt}\Delta t$$

$$\frac{dN_U}{dt} \equiv \lim_{\Delta t\to 0}\frac{N_U(t+\Delta t) - N_U(t)}{\Delta t} \approx \frac{N_U(t+\Delta t) - N_U(t)}{\Delta t}$$

$$N_U(t+\Delta t) \approx N_U(t) + \frac{dN_U}{dt}\Delta t$$

The difference equation used to determine N_U at the next time step

Given that we know the value of N_U at some value of t,

$$\boxed{N_U(t + \Delta t)} \approx \boxed{N_U(t) - \frac{N_U(t)}{\tau}\Delta t}$$

LHS, the value for the next time step is calculated based on the RHS

RHS, values from previous time steps already stored in computer's memory

we can use (1.7) to *estimate* its value a time Δt later.

Discretising the differential equation

- The differential equation dN/dt=-(N/τ) is said to be discretised into a difference equation, $N_U(t+\Delta t) \approx N_U(t) - \frac{N_U(t)}{\tau}\Delta t$ which is suitable for numerical manipulation using computer.
- There are many different way to discretise a differential equation.
- The method used here is known as Euler method, where the differentiation of a function at time t is approximated as

$$\frac{df(t)}{dt} \simeq \frac{f(t+\Delta t) - f(t)}{\Delta t} = \frac{f(t_{i+1}) - f(t_i)}{\Delta t}, \qquad t_i = i\Delta t, \quad t_{i+1} - t_i = \Delta t,$$

$$\frac{df(t_i)}{dt} \simeq \frac{f(t_{i+1}) - f(t_i)}{\Delta t}; t_i = i\Delta t \Rightarrow f(t_{i+1}) = f(t_i) + \frac{df(t_i)}{dt}\Delta t = f(t_i) + G(t_i)\Delta t$$

$\Rightarrow f(t_{i+1}) = f(t_i) + G(t_i)\Delta t$,where $G(t_i)$ is the function appearing in the E.o.M in $\frac{df(t)}{dt} = G(t)$

(in our case here, $G(t) \equiv -\frac{N(t)}{\tau}$, $f(t) \equiv N(t)$)

General structure of a code

- Initialisation
- Calculation
- Output

The code's structure

- Initialisation: Assign $N_u(t=0)$, τ, Δt.
- In principle, the finer the time interval Δt is, the numerical solution becomes more accurate.
- The global error is of the order O ~ Δt
- Also define when to stop, say tfinal= 10τ.
- Calculate $N_u(\Delta t)= N_u\,(0)+\ \Delta t(N_u(0)/\tau)$.
- Then calculate $N_u(2\Delta t)= N_u(\Delta t) + \Delta t(N_u(\Delta t)/\tau)$
- $N_u(3\Delta t)= N_u(2\Delta t)+\Delta t\, N_u(2\Delta t)/\tau$ ...
- Stop when t= tfinal= 10τ.
- Number of steps, Nstep = 1+IntegerPart[tfinal/Δt]
- Plot the output: $N_u(t)$ as function of t.