Modelling of lifting of Dragon Capsule after splash down - Numerical algorithms Exercise 1

Ramana Bharathi G*
Department of Aerospace Engineering, IIST
(Dated: September 10, 2021)

After formulating the problem, some time is devoted to understand the numerical alogorithms used by solvers that will be used to analyse the mathematical model to understand the implications of the solution and the errors associated with it.

I. INTRODUCTION

Obtaining analytical solutions to differential equations is hard. Most of the differential equations have no known analytical solutions, and finding analytical solutions (elementary forms) for different kinds of differential equations can't be generalized as an algorithm to be replicated using computation (unless the particular kind of differential equation is well-known, in which case it can be solved symbolically using pre-defined general solutions)[1]. Hence numerical techniques are used to solve the differential equation (numerically) to obtain a solution within error tolerance specified by us. Most physical phenomena are modelled as a differential equations or rather a system of differential equations and solving them is paramount to make any scientific progress. Simulating such multi-physical phenomena is done using several commercial and open-source packages (ADAMS/ANSYS/LSDyna/ ABAQUS/ Nastran/COMSOL/Star-CMM+/OpenFOAM/SU2) and such solvers use sopisicated numerical algorithms to obtain the solution. A brief introduction to the numerical algorithms has been discussed below.

II. NUMERICAL INTEGRATION TECHNIQUES

Numerical integration techniques are a broad class of algorithms that help to approximately calculate a definite integral and are also used to obtain numerical solution of differential equations. [1] A wide range of methods exist to solve differential equations numerically.

II.1. Stiffness of an equation and stability of a solution

Singularities are points on the state space of the differential equation at which the solution doesn't exist alongwith the general solution (they form envelopes for the general solution curves). Movable singularities depend on the initial conditions of the differential equation, while immovable singularities are due to the equation in itself. Hence stability of solutions has to be discussed while dealing with singularities (equilibrium points). Stability of solutions of differential equations has been defined in many ways (Routh-Hurwitz criterion, Lyapunov stability, A-stability, B-stability, L-stability etc). Stability functions are defined according to the numerical technique and the definition is given according to it. Each definition is used to check the stability in diffferent situations.

Stiffness of a equation is a phenomenon wherein certain numerical methods (their solutions) computed for the given equation is numerically unstable. While numerically integrating a differential equation, step size can be varied according to the slope of the solution curve in order to not diverge. But for stiff equations, the step size has to be very small to converge to a solution, so small that it makes the computation run longer than it should take. Depending on the stiffness of the equation, the algorithm will vary in their efficiency and error to compute the solution. Although stiffness is not exactly characterisable (although stiffness ratio gives an rough idea), there are stiffness detection algorithms that help the solver select the appropriate algorithm. [2]

II.2. Implicit and explicit methods

Methods used for numerical analysis of time-dependent differential equations can be explicit or implicit. Explicit methods calculate the state of the system at a later time using the state at the current time while implicit methods solve an equation containing both the states (at later time and current time) to obtain the state at later time.[]
Explicit method: $Y(t+\delta t) = F(Y(t))$
Implicit method: $G(Y(t+\delta t), Y(t)) = 0$

While explicit methods are ideal to solve non-stiff problems, they need time steps (δt) to be very small for solving stiff problems otherwise the solution will diverge easily (unbounded error). Implicit methods though harder to implement and require extra computation, the solution obtained is more numerically stable so for stiff problems larger time steps can be used. Simplest explicit method is the forward Euler method while simplest implicit method is the backward Euler method. [3]

* ramana.sc19b028@ug.iist.ac.in

II.3. Error control

Error control is one of the highly decisive aspects of a numerical algorithm. Every numerical algorithm will have errors and it is important to quantify them to understand the convergence of the solution and to analyze the performance of the algorithm on the given differential equation and to decide an appropriate algorithm. Converge tests are performed on the solution at every step and the current state is compared with the previous state to arrive at the error. There are basically two types of errors- absolute error (atol) and relative error (rtol)- which are self explanatory from their names.

Excerpt from the documentation of the ODE solver odeint() from scipy.integrate: [4]
" *The input parameters 'rtol' and 'atol' determine the error control performed by the solver. The solver will control the vector, e, of estimated local errors in y, according to an inequality of the form "max-norm of (e/ewt) <= 1", where ewt is a vector of positive error weights computed as "ewt = rtol * abs(y) + atol". rtol and atol can be either vectors the same length as y or scalars. Defaults to 1.49012e-8.* " [**?**]

e is the vector of estimated local errors while *ewt* is a vector of positive error weights The inequality:

$$\left\| \frac{e}{ewt} \right\| \leq 1$$

where *ewt* is defined as:

$$ewt = rtol * \mid y \mid + atol$$

II.4. differential integration algorithms

We will mainly look at Runge-kutta methods.

Runge-Kutta methods: It is a family of implicit and explicit numerical methods that help to solve ordinary diferential equations defined on time(t). The classical Runge-Kutta is given by (for first order ODE):

$$y_{n+1} = y_n + \frac{1}{6} h \left(k_1 + 2k_2 + 2k_3 + k_4 \right),$$
$$t_{n+1} = t_n + h$$

Where y_{n+1} is the RK4 approximation of the $y(t_{n+1})$ and $h > 0$ is the time step size.

$$k_1 = f(t_n, y_n),$$
$$k_2 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2} \right),$$
$$k_3 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2} \right),$$
$$k_4 = f\left(t_n + h, y_n + hk_3 \right).$$

Where
k_1 is the slope at the beginning of the interval, using y (Euler's method);
k_2 is the slope at the midpoint of the interval, using y and k_1
k_3 is the slope at the midpoint, using y and k_2
k_4 is the slope at the end of the interval, using y and k_3.

Given RK4 is a fourth order method, hence errors are of the order $O(h^4)$. Fig. 1 shows the slopes used in the above definition.

Runge-kutta methods encompass most of the famous numerical intgration techniques. 1st order RK method (RK1) is Euler's method. 2nd order RK method (RK2) with two stages is midpoint method. 4th order RK method (RK4) is simpson's rule. ode45 in MATLAB basically uses 4th order RK in 5 stages. RK method can also be defined implicitly to help solve stiff equations (as explicit RK methods are not A-stable i.e not atable enough for stiff equations). Adaptive Rk methods truncate the step size accordingly and optimize the performance. RK method is characterized into different submethods (or different order and stages) using Butcher tableau. RK methods and their variants are highly versatile in solving many types of differential equations. [5]

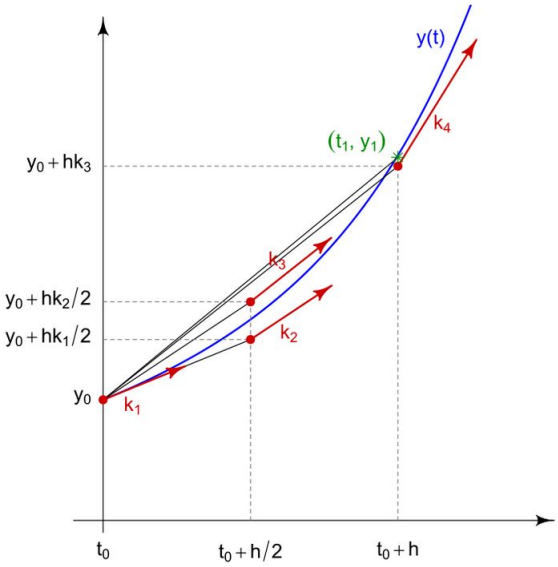

FIG. 1. slopes used by classical Runge-Kutta method [5]

There are plenty of algorithms for solving different types of differential equations. This link shows all the numerical algorithms used by the solvers in DifferentialEquations.jl in julia. https://diffeq.sciml.ai/

stable/solvers/ode_solve/#ode_solve

III. NUMERICAL ALGORITHMS FOR SIMULATIONS

For simulations, spacial-temporal non-linear PDEs have to be solved in most cases. The spacial aspect brings in the non-linear component usually. In general, the spacial and the temporal (if applicable) is split into small systems to be analysed separately. This process is called discretization (meshing). System of Differential-algebraic equations is produced using different approximations and then they are solved using numerical solution algorithms. Various numerical solution algorithms can be classified into two broad categories; direct and iterative solvers. These algorithms are designed to exploit the sparsity of matrices that depend on the choices of variational formulation and discretization strategy. After solving, post-processing is done to obtain necessary information from the data produced by the solution and to estimate the error associated with the solution.

Prominent numerical algorithms inlcude:

Finite Element Method: [6] [7] FEM is one the most widely used method for solving PDE in 2 or 3 dimensions (Boundary value problems). Finite elements are generated by spacial discretization (meshing). The finite elements are then mathematically modelled individually (by defining shape functions on the elements) and this results in a system of Differential-Algebraic equations. Steady-state problems produce a system of linear equations, which are solved using numerical linear algebra methods and transient problems produce a system of ordinary differential equations (ODEs) which are solved using numerical integration techniques (e.g. Runge-Kutta methods). The unknown function is found locally and then a global estimate is calculated. The solution is obtained my minimizing the error function using calculus of variations (e.g. Galerkin method). THere are many types of FEM (AFEM,hpkPEM,XFEM etc.).FEM can easily handle coplex geometries and it is more accurate. FEM is the method of choice for analyses in structural mechanics (static/dynamic/modal/stability).

Finite Difference Method: [8] FDM is one of the easiest and the oldest numerical techniques to solve 3-D PDEs. Domain is discretized in space and in time and approximations are calculated at those points. The derivative relation between the adjacent nodes is approximated as differential gradients (from Taylor's series) and hence the system of linear equations is generated which is solved using numerical linear algebra. The discretization error is obtained by combining the error in taylor series approximation in all the nodes. FDM is very easy to implement and compute but its basic form is restricted to simple rectangular/block meshes. Unstructured meshes are hard to handle.

Finite Volume Method: [9] In FVM, the spacial domain is discretized into smaller control volumes. Volume integrals are converted to surface integrals using divergence theorem. These are calculated as fluxes at the surfaces of the smaller control volumes.Using interpolation/extrapolation, values at nodes are computed. This method is inherenty conservative as the fluxes entering a control volume is equal to the fluxes leaving the adjacent control volumes. Separate conservation checks are unnecessary. It is easily formulated for unstrutured meshes. But the method is computationally heavy.

IV. CONCLUSIONS

A brief discussion regarding the numerical techniques and algorithms used to solve differential equations and for simulations have been covered. This will help us to understand the methodology of arriving at the solution of the mathematical model that will be developed to model the physical problem.

[1] Numerical integration (2021).
[2] Stiff equation (2021).
[3] D. Fain, Freshman friendship and university connection.
[4] Scipy, scipy/odepack.py at v1.7.1 · scipy/scipy.
[5] Runge–kutta methods (2021).
[6] Finite element method (2021).
[7] What is fea: Finite element analysis? documentation (2021).
[8] Finite difference method (2021).
[9] Finite volume method (2021).