Exercise 2
Understanding Various Numerical Integration Methods
AE332 - Modelling and Analysis Lab

Subrahmanya V Bhide (SC18B030)
Indian Institute of Space Science and Technology
Department of Aerospace Engineering
(Dated: 28 September 2020)

This is a report based on the study done on various types of numerical integration algorithims for solving Ordinary Differential Equations. This report summarizes the different numerical methods available for solving IVP's and also discusses their merits and demerits and their applications in commercial softwares and packages.

INTRODUCTION

Numerical Integration methods become important tools for engineering applications due to the fact that, natural phenomena and most of the systems which engineers develop are governed by differential equations. To produce an effective engineering solution we need to know the solutions (atleast be able to obtain approximate solutions) for the governing differential equations.
Here we discuss some of the numerical methods available for solving Ordinary Differential Equations with initial conditions. Since any n^{th} order differential equation can be converted to a set of 1^{st} order differential equations, numerical methods are explained considering a 1^{st} order ODE with initial conditions as in Eqn. 1.

$$y'(t) = f(t, y(t)) \quad ; \quad y(t_0) = y_0 \tag{1}$$

CLASSIFICATION OF NUMERICAL METHODS AND ODES

The numerical methods can be classified in the following ways:

- Implicit and Explicit:
  For Explicit methods the current state of the system depends on, at most, all the previous states of the system.

  $$y(t_n) = F\left[y(t_{n-1}), y(t_{n-2}), y(t_{n-3}), ..., y(t_0)\right]$$

  For Implicit Methods the current state is found out by solving an equation of the form:

  $$G\left[y(t_n), y(t_{n-1}), y(t_{n-2}), y(t_{n-3}), ..., y(t_0)\right] = 0$$

- Stiff and Non-Stiff Problems:
  Stiff problems are those, for which obtaining a reliable solution requires the use of very small, almost impratctical timesteps and thus leads to an increase in the computation time. This problem arises due to the reason that the time scales in the problem vary by a large factor.

- Linear Multistep Methods and Runge-Kutta Methods:
  The various numerical methods can be broadly classified into these two categories. These categories are explained in detail in the following sections.

LINEAR MULTISTEP METHODS

Conceptually, a numerical method starts from an initial point and then takes a short step forward in time to find the next solution point. The process continues with subsequent steps to map out the solution. If the solution at a point is considered to be dependant on only one point in its neighbourhood it is reffered to as a single step method. If the solution at a point is considered to be dependant on more than one point in its neighbourhood then they are reffered to as multistep methods. The methods are called linear multistep because the dependance is assumed to be a linear combination of solution points in the neighbourhood.[1]
A general linear multistep method for the ODE given by Eqn. 1 is given below which can be represented in compact form as in Eqn. 2.

$$y_{n+s} + a_{s-1} \cdot y_{n+s-1} + a_{s-2} \cdot y_{n+s-2} + ... + a_0 \cdot y_n =$$

$$h \cdot [b_s \cdot f(t_{n+s}, y_{n+s}) + b_{s-1} \cdot f(t_{n+s-1}, y_{n+s-1}) + ... + b_0 \cdot f(t_n, y_n)]$$

here h is the common difference i.e. $h = t_{n+1} - t_n$.

$$\sum_{j=0}^{s} a_j y_{n+j} = h \sum_{j=0}^{s} b_j f(t_{n+j}, y_{n+j}) \tag{2}$$

Here we can observe that if b_s is zero then the method becomes explicit and y_{n+s} can be directly computed, else if $b_s \neq 0$ it makes the method implicit and the

values for y for each step have to be solved iteratively.
Different methods are charecterised by the coefficients, a_i and b_i.
Some methods are discussed in the following sections.

Forward Euler Method

The equation to be solved for this method is Eqn. 3.

$$y_{n+1} = y_n + h \cdot f(t_n, y_n) \tag{3}$$

This is a one step method as y_{n+1} depends on its previous state y_n.
This is an explicit method as solution at a state is completely determined by known quantities of the previous state. To find the solution, one has to solve Eqn. 3 recursively starting from the initial state progressing forward in t.

Backward Euler Method

The equation to be solved for this method is Eqn. 4.

$$y_{n+1} = y_n + h \cdot f(t_{n+1}, y_{n+1}) \tag{4}$$

This is a one step method as y_{n+1} depends on its previous state y_n.
This is an implicit method. Thus to find y_{n+1} we have to solve an equation. This is done usually using iterative methods such as the Newton Raphson method or other iterative methods.

Adam Bashforth Methods

This is a family of linear multistep methods where, the coefficients are $a_{s-1} = -1$ and $a_{s-2} = \cdots = a_0 = 0$ and b_i are taken to be the coefficients of the polynomial interpolation of the solution points (t, y). b_s is taken to be equal to zero.
The Adam Bashforth method of order 2 ($s = 2$) is given in Eqn. 5.

$$y_{n+2} = y_{n+1} + h\left(\frac{3}{2}f(t_{n+1}, y_{n+1}) - \frac{1}{2}f(t_n, y_n)\right) \tag{5}$$

The Adam Bashforth method of order 3 ($s = 3$) is given in Eqn. 6.

$$y_{n+3} = y_{n+2} + h\Big(\frac{23}{12}f(t_{n+2}, y_{n+2}) - \frac{16}{12}f(t_{n+1}, y_{n+1}) + \frac{5}{12}f(t_n, y_n)\Big) \tag{6}$$

The Adam Bashforth method of order 3 ($s = 3$) is given in Eqn. 6.

The coefficients b_j turn out to be given by:

$$b_{s-j-1} = \frac{(-1)^j}{j!(s-j-1)!}\int_0^1 \prod_{\substack{i=0 \\ i\neq j}}^{s-1}(u+i)\,du,$$

for $j = 0, \ldots, s-1$. s being the order of the method.

Adam Moulton Method

This is a family of linear multistep methods where, the coefficients are $a_{s-1} = -1$ and $a_{s-2} = \cdots = a_0 = 0$ and b_i are taken to be the coefficients of the polynomial interpolation of the solution points (t, y). The restriction of $b_s = 0$ is removed in this method and thus this method becomes an implicit method.
The Adam Moulton method of order 1 ($s = 1$) is given in Eqn. 7.

$$y_{n+1} = y_n + \frac{1}{2}h\left(f(t_{n+1}, y_{n+1}) + f(t_n, y_n)\right) \tag{7}$$

The Adam Moulton method of order 2 ($s = 2$) is given in Eqn. 8.

$$y_{n+2} = y_{n+1} + h\Big(\frac{5}{12}f(t_{n+2}, y_{n+2}) + \frac{2}{3}f(t_{n+1}, y_{n+1}) - \frac{1}{12}f(t_n, y_n)\Big) \tag{8}$$

The coefficients are given by:

$$b_{s-j} = \frac{(-1)^j}{j!(s-j)!}\int_0^1 \prod_{\substack{i=0 \\ i\neq j}}^{s}(u+i-1)\,du,$$

for $j = 0, \ldots, s$

. s being the order of the method.

Backward differentiation formula

The general formula for the BDF can be written as in Eqn. 9.

$$\sum_{k=0}^{s} a_k y_{n+k} = h\beta f(t_{n+s}, y_{n+s}) \tag{9}$$

The Coefficients a_k and β are obtained through polynomial interpolation of the points $(t, y), \ldots, (t_{n+s}, y_{n+s})$

The two step ($s = 2$) BDF is given in Eqn. 10.

$$y_{n+2} - \tfrac{4}{3}y_{n+1} + \tfrac{1}{3}y_n = \tfrac{2}{3}hf(t_{n+2}, y_{n+2}) \quad (10)$$

The single step BDF is same as the Backward Euler method. The BDF methods provide acceptable results only upto 7 step methods i.e. for $s > 7$ the BDF is unstable.

RUNGE-KUTTA METHODS

Explicit Runge Kutta Method

The general form of the Explicit Runge Kutta Method can be given as in Eqn. 11.

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i \quad (11)$$

Where,
$k_1 = f(t_n, y_n)$,
$k_2 = f(t_n + c_2 h, y_n + h(a_{21}k_1))$,
$k_3 = f(t_n + c_3 h, y_n + h(a_{31}k_1 + a_{32}k_2))$,
$\vdots$
$k_s = f(t_n + c_s h, y_n + h(a_{s1}k_1 + a_{s2}k_2 + \cdots + a_{s,s-1}k_{s-1}))$
In contrast to the linear multistep methods, the Runge-Kutta Scheme considers that y_{n+1} depends on yn and the weighted average of s increments, where each increment is the product of the size of the interval, h, and an estimated slope specified by function f on the right-hand side of the differential equation. The widely known RK scheme is the RK4 which considers 4 increments.

Implicit Runge-Kutta Method

The general form of the implicit RK scheme is given in Eqn. 12.

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i \quad (12)$$

Where,
$k_i = f\left(t_n + c_i h,\ y_n + h\sum_{j=1}^{s} a_{ij}k_j\right), \quad i = 1, \ldots, s.$

Adaptive Runge Kutta Methods

In these methods two different Runge-Kutta schemes, one with order p and the other with order $p-1$ are solved. These two methods are interwoven as they have common intermediate steps. The difference between the two obtained solutions is kept below a user defined threshold by adjusting the step size. Hence in these methods the user need not provide and spend time to come up with an appropriate step size.[2]

A GENERAL DISCUSSION ON THE METHODS DISCUSSED

Consider the IVP given in Eqn. 1, it can be written as $dy = f(t, y(t))dt$. Integrating this between y_n and y_{n+1} and t_n and t_{n+1} we get $y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(t, y(t))dt$. Now different Linear multistep methods are based on the polynomial approximation we consider for f. For instance if $f = f(t_n, y_n)$, a constant, we obtain the forward Euler method and if $f = f(t_{n+1}, y_{n+1})$ we obtain the backward Euler method. If we approximate f as a polynomial, fitting the polynomial using 2 previous timesteps i.e $(t_{n-1}, f(t_{n-1}, y_{n-1}))$ and $(t_n, f(t_n, y_n))$ then we obtain the 2 step Adam Bashforth method. Instead of $(t_{n-1}, f(t_{n-1}, y_{n-1}))$ if we use $(t_{n+1}, f(t_{n+1}, y_{n+1}))$ we obtain the corresponding Adam Moultons method.[3]
In the methods where the number of steps is more than 1, the initial conditions required are obtained by using lower order methods. For instance to use Adam-Bashforth method with $s = 2$, we can use Eulers method to estimate the value of y_1 using the value of y_0 initially available.
Apart from implicit and explicit methods there are also methods known as predictor-corrector methods where we use explicit methods first to 'predict' the value of y_{n+s}. That value is then used in an implicit formula to 'correct' the corresponding value.

STABILITY & ERRORS

The important charecteristics of a method, which define their usefullness and applicability are discussed below:

Consistency

A multistep method is said to be consistent if the local truncation error (Error caused in each step) goes to zero faster than the step size h as h tends to zero. The truncation error depends on the method we use.

Order

A method is said to be of order p if the local error is of the order $O(h^{p+1})$ as h goes to zero. Hence we can see that higher the order more close is the actual solution to the numerically calculated one.

Stability

Stability of a method is a measure of how stable a method is with respect to pertubations in the initial values. A linear multistep method is said to be zero-stable

for a certain differential equation on a given time interval, if a perturbation in the starting values of size ϵ causes the numerical solution over that time interval to change by no more than Kϵ for some value of K which does not depend on the step size h.

Convergence

A multistep method is said to be convergent if the numerical solution approaches the exact solution as h tends to zero.
For an iterative method if, ϵ_{n+1} and ϵn be the local errors at the corresponding timesteps and the Eqn. 13 is satisfied with $A > 0$ and $R > 0$ then the R is known as the order of convergence.

$$\lim_{n \to inf} \frac{\epsilon_{n+1}}{\epsilon_n^R} = A \tag{13}$$

Dahlquist's theorem also known as Dahlquist's First Barrier, states that:
The maximum order of convergence of an s-step method is:

1. s+2 for implicit methods with an even s.
2. s+1 for implicit methods with an odd s.
3. s for explicit methods.

To test the stability of a method we replace f(t,y) with $\lambda y(t)$ where in general λ is a complex number. The method is said to be stable in the regions λh, where the growth factor ($\frac{y_{n+1}}{yn}$) is less than unity.
Now if the stability region includes the entire complex half-plane with negative real part then the method is said to be A-Stable. Also, as the stability regions get smaller, we have to take smaller time steps i.e. h must be decreased , which increases the computational time.
Dahlquist's Second Barrier states that:
There are no explicit A-stable linear multistep methods. Further, the implicit A-stable linear mulitstep methods have order of convergence at most s = 2.
In contrast to this A-Stable Runge-Kutta Methods can have arbitrarily higher order.[1, 2, 4]

USE OF NUMERICAL METHODS IN COMMERCIAL SOFTWARES

In commericial softwares such as Abaqus, Nastran, Ansys, Comsol ...etc, the analysis possible in these softwares can be broadly divided into two categories:

1. FEA refers to finite element analysis and is used to solve governing partial differential equations along with boundary conditions. FEA helps us to do variety of structural and thermal analyses. Here the domain of the problem is divided into small elements. The variables of the problem for Eg. displacement, temperature ...etc, must be determined at the end points of the elements called nodes. The governing differential equation is multiplied by a function known as the weight function and integrated to get an algebraic equation involving the unknowns in the problem. When these equations are written for all the nodes we get a matrix equation. The boundary conditions also either get converted into an algebraic equation or get applied through the value of the variables at the boundaries. Equation 14, 15 and 16 are the matrix equations obtained for linear, non-linear and dynamic problems respectively.

$$[K]\{u\} = \{f\} \tag{14}$$

$$[K(u)]\{u\} = \{f\} \tag{15}$$

$$[M]\{u''\} + [C]\{u'\} + [K]\{u\} = \{f\} \tag{16}$$

Here K is called the Stiffness matrix, M is the Mass Matrix, C is the damping matrix and f is called the force vector.
These equations can be solved using two methods namely, direct method where the matrix K is inverted and multiplied with f. Since even what we tend to believe is a simple problem the matrix K would be quite large and it would require huge amount of memory. Thus the matrix is represented as a sum of matrices which are easier to operate on, such as diagonal, triangular and sparse matices. This is known as decomposition. Also sometimes the equation is multiplied by another matrix known as the preconditioner matrix which makes the inversion process easier. There exist many variations and routines through which the solution can be obtained, such as UMFPACK, SPOOLES, TAUCS ...etc. Also there are many kinds of preconditioner methods available such as Geometric multigrid, SSOR, Diagonal scaling (Jacobi) ...etc. Each routine is suited for particular types of problems.
Ansys uses what is called as the Ansys Frontal solver which is based on Cholesky factorization algorithm (TAUCS) and a Sparse solver is also available. Abaqus uses a sparse, Gauss elimination method. Nastran uses Cholesky method and some tweakability is also provided to the user. Direct solvers in COMSOL are the MUMPS, PARDISO, and SPOOLES solvers.[5–8]
Apart from direct solvers we can also have iterative solvers, where, a solution is first guessed and then it is modified and improved based on the results it provides. Some solvers available for this approach are GMRES, Conjugate Gradients ...etc. In Ansys the iterative solver is known as the AMG (Algebraic multigrid method) and it also has a Jacobi preconditioned conjugate gradient solver (JCG). The

solver in Abaqus is based on Domain decomposition method. For Non-Linear systems as in Eqn. 15 methods such as Newton Raphson or Fixed point integration methods are used to obtain the solution.[9, 10]
The implicit backward Euler method is also used for solving the matrix equations. For matrix equations as in Eqn. 16, Abaqus provides options for using either the Backward Euler method or the Hilber-Hughes-taylor method. Ansys provides the Arbitrary Lagrange-Euler method that combines the advantages of Lagrange and Euler methods while eliminating some of their shortcomings. It also includes methods such as the PCG Lanczos method and their variations for different kinds of problems. Nastran provides Cholskey method for direct solution of the matrix equation and a method known as FBS for iterative routines.[11, 12]

2. CFD refers to computational fluid dynamics. This deals with fluid flows and hence we need to solve the Navier Stokes Equation which is the governing equation for fluid flows. In FEA the system is divided into elements but here we divide the region of interest into finite volumes and conservation laws are applied on these finite volumes to get equations in terms of the variables like u, v, w the velocity components in x, y and z directions their gradients ... etc. The Integral form of the Navier Stokes Equation is discretized and the integral is replaced by the summation over all the finite volumes. There cant be a direct method to solve these equations as the pressure variable present in the equation is dependant on velocity at cell/volume centers hence these equations are solved iteratively. In each iteration a matrix equation has to be solved.[13]

APPLICATIONS, ADVANTAGES and DISADVANTAGES Of NUMERICAL METHODS DISCUSSED

Direct solvers require more memory compared to iterative methods, but iterative methods require more computational time. Explicit methods are not unconditionally stable and hence care must be taken to select the timesteps appropriately. On the other hand Implicit methods though being unconditionally stable require more computational time as compared to explicit methods. For stiff systems adaptive RK methods are more suitable as the step size is arrived at by the solver on its own based on the tolerance specified by the user, other RK or Linear multistep methods can also be used for stiff systems. The FEA and CFD discussed in the previous section corresponds to situations where the governing equations are PDE's along with boundary conditions. For simpler problems governed by ODEs a large range of methods are available in different programming languages. Scipy includes functions which are based on Adaptive Runge-Kutta Methods for Python under the function odeint and includes Adams method and BDF under the function ode, Sundials provides solvers based on Adams and BDF methods along with RK methods. It is available for C++ and also provides wrappers for python. In matlab RK method is available through the routine ode45. ode113 offers an Adam method, ode15's offer multistep methods.[14]

APPENDIX

Newton Raphson Method

Newton Raphson method is an iterative method used to find the solution for an equation, usually non linear. Let the equation be as given in Eqn. 17.

$$y = f(x, y) \tag{17}$$

If we need the values of x for which $y = y_0$, then, initially a value for x is guessed as x_0. The value of $y_0 - f(x_0, y_0)$ is found out. Ideally this value must be zero, and is not zero if x_0 is not the solution for the Eqn. 17. Now the guess value is changed to the x intercept of the tangent drawn at x_0. This process is repeated until the difference $y_0 - f(x_0, y_0)$ becomes conveniently close to zero.
The updation for value of x can be done as in Eqn. 18.[15]

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{18}$$

[1] https://en.wikipedia.org/wiki/Linear_multistep_method; Accessed on 16 September 2020 4:28 pm.
[2] https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods; Accessed on 16 September 2020 6:05 pm.
[3] https://www.youtube.com/watch?v=fp6n7x55tkQ; Accessed on 17 September 2020 9:37 pm.
[4] https://www.math.utah.edu/~vshankar/5620/LinearMultistepII.pdf; Accessed on 25 September 2020 3:50 pm.
[5] https://lost-contact.mit.edu/afs/nada.kth.se/amdlinks/pkg/femlab/3.1x/doc/guide/ugsolve9.htm; Accessed on 27 September 2020 10:30 am.
[6] https://www.comsol.com/blogs/solutions-linear-systems-\equations-direct-iterative-solvers/; Accessed on 27 September 2020 11:08 am.
[7] https://abaqus-docs.mit.edu/2017/English/SIMACAEANLRefMap/simaanl-c-dynamic.htm#hj-top; Accessed on 27 September 2020 11:23 am.

[8] https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.8.4470&rep=rep1&type=pdf#:~:text=3.1.&text=The%20original%20solver%20used%20in,direct%20method%20for%20many%20years.; Accessed on 26 September 2020 1:22 pm.
[9] https://abaqus-docs.mit.edu/2017/English/SIMACAETHERefMap/simathe-c-nonlinearsol.htm; Accessed on 26 September 2020 1:41 pm.
[10] https://dianafea.com/manuals/d943/Analys/node431.html; Accessed on 26 September 2020 2:14 pm.
[11] https://support.ansys.com/staticassets/ANSYS/staticassets/resourcelibrary/brochure/dynamics-11.pdf; Accessed on 26 September 2020 3:01 pm.
[12] https://docs.plm.automation.siemens.com/data_services/resources/nxnastran/12/help/custom/en_US/numerical/numerical.pdf; Accessed on 28 September 2020 2:33 pm.
[13] https://ttu-ir.tdl.org/bitstream/handle/2346/ETD-TTU-2010-12-1156/TAN-DISSERTATION.pdf?sequence=1&isAllowed=y; Accessed on 28 September 2020 5:24 pm.
[14] http://www.stochasticlifestyle.com/comparison-differential-equation-solver-suites-matlab-rc-fortran/; Accessed on 28 September 2020 9:59 pm.
[15] https://en.wikipedia.org/wiki/Newton%27s_method; Accessed on 28 September 2020 8:51 pm.