Numerical Methods Appendix to GES 554

Charles O'Neill

Lessons 37-40

1 Numerical Methods

Now that the govering equations are in a numerical form, there are two major numerical methods required: time integration and matrix inversion.

1.1 Time Integration

Time integration involves integrating a temporal differential equation forward in time. The general governing equation is

$$\frac{dy}{dt} = f(y)$$

with initial conditions

$$y(0) = y_0$$

1.1.1 Order Analysis

We have two types of governing equations: convection and diffusion. Given an arbitrary solution in Fourier space

$$y = ae^{ikx}$$

The convection equation expands to

$$\frac{d\left(\frac{(\rho u)^2}{\rho}\right)}{dx} = \frac{d}{dx}\left(abe^{2ikx}/ce^{ikx}\right) = ike^{ikx}$$

This gives an eigenratio of

$$\lambda = k$$

The dissipation equation expands to

$$\frac{1}{Re}\frac{d^2(\rho e)}{dx^2} = \frac{d}{dx}\left(ae^{ikx}\right) = -\frac{k^2}{Re}e^{ikx}$$

for an eigenratio of

$$\lambda = \frac{k^2}{Re}$$

This indicates that when the Reynolds number is greater than k, then the limiting timestep is convection. Boyd[2] states that "There is little advantage to treating the nonlinear terms implicitly because a timestep longer than the explicit advective stability limit would be too inaccurate to be acceptable."

1.1.2 Literature

Time integration advances a solution when temporal derivatives are known. In general, we are interested in 1st order ordinary differential equations of the form

$$\frac{dy}{dt} = f\left(t, y\right)$$

with

$$y(t_0) = y_0$$

The field of differential equation integration has evolved tremendously in the last few decades, so the well-known methods commonly seen in engineering textbooks[8] must not be prematurely selected. The state of the art in the late 20th century is represented by the *Solving Ordinary Differential Equations* books in two volumes [5, 6]. For reasons to be discussed, volume two[6] is a primary reference for this paper.

The Galerkin formulation of the Navier-Stokes equations are "very stiff" according to Boyd[2]. This implies either an implicit solver or an explicit solver with small timesteps. Increasing the spatial solution order increases the solution stiffness. We should seriously consider an implicit scheme of approximately the same temporal order as spatial order.

Implicit solvers have advantages for boundary conditions and solution assurance. First, the implicit solvers have known residuals. These residuals are easy to watch. Explicit solvers usually do not have as simple of a visual quality indicator. However, too small of a timestep reduces higher order implicit schemes to an equivalent backwards Euler scheme (with all of the disadvantages of such).

For implicit iteration, Hairer and Wanner[6] say:

> For a general nonlinear differential equation the system... has to be solved iteratively. In the stone-age of stiff computation (i.e., before 1967) people were usually thinking of fixed-point iteration. But this transforms the algorithm into an explicit method and destroys the good stability properties.

Traditional numerical method for implicit ode solution is based on Newton's method which requires a Jacobian matrix. Our FE equations are not easily decomposed into an explicit Jacobian, nor is a numerical approximation of the Jacobian appropriate with array sizes in the millions. The most difficult part of an implicit FE solver is not the time advancement scheme but solving the linear equation resulting from the scheme. State of the art for implicit ODE solutions does not yet match the complexity of FE solvers. The issue is that while the mass matrix is linear, the force vector is not.

$$M\frac{da}{dt} = F$$

F is neither trivial to calculate nor trivial to decompose into linear components necessary for the Jacobian dF/da. Of course, Newton's method is preferred over an iterative Krylov or Jacobi method simply for the convergence rate. Press[8] states

> Even when Newton-Raphson is rejected for the early stages of convergence..., it is very common to "polish up" a root with one or two steps of Newton-Raphson, which can multiply by two or four its number of significant figures!

1.1.3 Predictor Corrector

Predictor corrector (PC) methods are a traditional[8] integration method with the form

$$y_j = hb_i f\left(t_i, y_i\right)$$

Expanded for order $p+1$, this is

$$y_{n+1} = y_n + h\beta_0 f\left(t_{n+1}, y_{n+1}\right) + h\beta_1 f\left(t_n, y_n\right) + \cdots + h\beta_p f\left(t_{n-p+1}, y_{n-p+1}\right)$$

Predictor methods omit the implicit β_0 term to get started; corrector methods include the β_0 term for higher accuracy. Increasing order is obtained by adding more past derivatives. Thus, changing step size h either requires restarting with a lower order approximation or deriving a series of special β terms for the step size propagation.

Press[8] states "We suspect that predictor-corrector integrators have had their day, and that they are no longer the method of choice for most problems in ODEs.... There is one exceptional case: high-precision solutions of very smooth equations with very complicated right-hand sides...." Even worse, the predictor corrector's stability domain shrinks as the integration order increases[6]. Since the time integrator should roughly match the domain expansion order, a decreasing stability domain is certainly not wanted. Innately, the predictor corrector requires that the mapping $f(t, y)$ does not change.

1.1.4 Runge Kutta

Runge Kutta methods refer to both implicit and explicit multi-stage time integration. Iserles's[7] book provides a valuabe reference for Runge Kutta schemes.

The general form of a Runge-Kutta type integrator is

$$y_{n+1} = y_n + hb_i k_i$$

$$k_i = f\left(t_n + hc_i,\, y_n + ha_{ij}k_j\right)$$

$$t_i = t_o + hc_i$$

Increasing order is obtained by adding more k terms. Changing step size h is possible at each step. This form is often displayed as a tableau

$$\begin{array}{cccc} c_1 & a_{11} & \cdots & a_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ c_p & a_{p1} & \cdots & a_{pp} \\ \square & b_1 & \cdots & b_p \end{array}$$

Explicit methods consist of a lower triangular a where all $a_{ij} = 0$ when $j \geq i$; implicit methods have at least one non-zero a_{ij} term where $j \geq i$. Implicit RK requires iteration. RK properties include: $c_i = \sum_j a_{ij}$ and $\sum_j b_j = 1$. For the typical application, these properties are usually confined to transcription error identification.

Each RK step is completely independent of previous steps. More importantly, the mapping $f(t, y)$ can change space. For CFD applications, RK allows for a completely different computational grid at each timestep.

We will describe some of the common RK integrators below.

Forward and Backward Euler The forward Euler, a 1st order method, is a simple integrator.

$$y_{n+1} = y_n + hf\left(t_n, y_n\right)$$

The tableau is

$$\begin{array}{cc} 0 & 0 \\ & 1 \end{array}$$

By comparison, the backward Euler is an implicit 1st order method

$$y_{n+1} = y_n + hk$$

$$k = f\left(t_n + h, y_n + hk\right)$$

Notice that y_n does not form a closure; iterations and stopping criteria are required. Its tableau is

$$\begin{array}{cc} 1 & 1 \\ & 1 \end{array}$$

Crank Nicholson Crank Nicholson (C-N) is an implicit second order method often seen in finite difference codes. The tableau is

$$\begin{array}{ccc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ & \frac{1}{2} & \frac{1}{2} \end{array}$$

IRK2 One possible second order implicit RK2 method is

$$\begin{array}{ccc} 0 & \frac{1}{4} & -\frac{1}{4} \\ \frac{2}{3} & \frac{1}{4} & \frac{5}{12} \\ & \frac{1}{4} & \frac{3}{4} \end{array}$$

Expanded, this is

$$t_1 = t_o$$

$$t_2 = t_o + \frac{2}{3}\Delta t$$

$$z_1 = y_n + \Delta t \left(\frac{1}{4} F\left(z_1, t_1\right) - \frac{1}{4} F\left(z_2, t_2\right) \right)$$

$$z_2 = y_n + \Delta t \left(\frac{1}{4} F\left(z_1, t_1\right) + \frac{5}{12} F\left(z_2, t_2\right) \right)$$

$$y_{n+1} = y_n + \Delta t \left(\frac{1}{4} F\left(z_1, t_1\right) + \frac{3}{4} F\left(z_2, t_2\right) \right)$$

This looks ripe for iteration; however, this is exactly the situation Haier warns about using fixed point iteration rather than fully implicit inversion.

Hammer-Hollingsworth This is another implicit RK2 method.

$$\begin{array}{ccc} \frac{3-\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{3+\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\ & \frac{1}{2} & \frac{1}{2} \end{array}$$

RK4 The canonical Runge-Kutta integrator is the explicit RK4. The tableau is

$$\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

Adaptive RK Adaptive RK typically indicates two RK methods where one lower-order method is a subset of the higher-order method. Adaptive methods allow for fast error estimates for calculating step sizes. Famous methods are Cash-Karp, RKF, etc[8].

1.1.5 Efficient Implicit Formation

Directly implementing one of the above RK routines is not especially efficient. For the implicit RK routines, there are stability and major efficiency issues with stagewise iteration. The conceptual issue is that the mass matrix only contains the temporal information about the governing equations. From theory, this restricts both the rate of convergence and the timestep, since the product of timestep and right-hand-side eigenvalues must be small. We need Jacobian information.

A generic form of a RK stage is

$$z_i = y_o + \Delta t\, a_{ij} \left(\frac{dz}{dt} \right)_j$$

This implies a matrix inverse operation

$$z_i = y_o + \Delta t\, a_{ij} \left(M^{-1} B \right)_j$$

where

$$M \frac{dz}{dt} = B$$

contains the fluid governing equations. However, this form is not especially efficient.

Instead, the equation is premultiplied by the conceptual mass matrix to form

$$M z_i = M y_o + \Delta t\, a_{ij} B_j$$

Even further simplification occurs when expanding z_i

$$z = y_o + \Delta z$$

Also, B is expanded into a 1st order Taylor series as

$$B\left(z_i\right) = B\left(y_0\right) + \frac{dB\left(y_0\right)}{dz}\left(z_i - y_0\right) = B\left(y_0\right) + \frac{dB\left(y_0\right)}{dz}\Delta z$$

Combining gives

$$M \Delta z_i = \Delta t\, a_{ij} \left(B\left(y_0\right) + \frac{dB\left(y_0\right)}{dz} \Delta z \right)$$

Now, forcing function information is available for stage iterations

$$\left(M - \frac{dB}{dz} \right) \Delta z_i = \Delta t\, a_{ij} B_o$$

This appears to be a distinct disadvantage since dB/dz, the Jacobian term, is complicated. This is where a non-direct matrix inversion method allows a simplification for an already multiplied Jacobian and vector term. Following Gear and Saad[4], a numerical Jacobian approximation is typically sufficient

$$Jv \approx \frac{1}{\epsilon}\left(B\left(z+\epsilon v\right)-B\left(z\right)\right)$$

The time integration equations are now in a canonical form for numerical inversion.

As expected, the multi-stage RK routines are now coupled where previously they were block independent. A two stage RK routine ready for numerical inversion has the form

$$\begin{bmatrix} [M-\Delta t\, a_{11}J_1] & [-\Delta t\, a_{12}J_2] \\ [-\Delta t\, a_{21}J_1] & [M-\Delta t\, a_{22}J_2] \end{bmatrix} \begin{pmatrix} \Delta z_1 \\ \Delta z_2 \end{pmatrix} = \Delta t \begin{bmatrix} a_{11}B(y_0)+a_{12}B(y_0) \\ a_{21}B(y_0)+a_{22}B(y_0) \end{bmatrix}$$

whereas the non-Jacobian form was

$$\begin{bmatrix} [M] & \\ & [M] \end{bmatrix} \begin{pmatrix} \Delta z_1 \\ \Delta z_2 \end{pmatrix} = \Delta t \begin{bmatrix} a_{11}B(z_1)+a_{12}B(z_2) \\ a_{21}B(z_1)+a_{22}B(z_2) \end{bmatrix}$$

This reflects the change to a true implicit iterative scheme that satisfies the Hairer and Wanner[6] fixed-point iteration stability comment. The disadvantage is a *tremendous* increase in the computational requirement (i.e., computing $J\Delta z$ at each stage and step).

Expansion Point Expanding around a different point is instructive.

$$z = \bar{z} + \Delta z$$

so that

$$\Delta z = z - \bar{z}$$

The 1st order Taylor series is

$$B\left(z_i\right) = B\left(\bar{z}_i\right) + \frac{dB\left(\bar{z}_i\right)}{dz}\left(z_i - \bar{z}\right) = B\left(\bar{z}_i\right) + \frac{dB\left(\bar{z}_i\right)}{dz}\Delta z_i$$

Combining as before gives

$$M\left(z_i - y_0\right) = \Delta t\, a_{ij}\left(B\left(\bar{z}_i\right) + \frac{dB\left(\bar{z}_i\right)}{dz}\Delta z_i\right)$$

This needs one more step

$$\left(z_i - y_0\right) = \left(z_i - y_0 + \bar{z} - \bar{z}\right) = \Delta z_i + \left(\bar{z} - y_0\right)$$

Combining gives

$$M\Delta z_i = \Delta t\, a_{ij}\left(B\left(\bar{z}_i\right) + \frac{dB\left(\bar{z}_i\right)}{dz}\Delta z_i\right) - M\left(\bar{z} - y_0\right)$$

Nicely, the initial iteration residual is

$$R = \Delta t \, a_{ij} B\left(z_i\right)$$

and the linear term is

$$Ax = M \Delta z_i$$

This form should be more robust when the Jacobian dB/dz is not exact. The objective would be to reduce the converged Δz to zero. Otherwise, this form reduces to the previous form.

Time Integration Experiment The concepts introduced above are tested for a known solution. The differential equation is

$$\frac{dy}{dt} = -1 - Cy$$

with the initial condition

$$y(0) = 1$$

The solution is

$$y(t) = -\frac{1}{C} + \left(1 + \frac{1}{C}\right) e^{-Ct}$$

The objective is to compare simple one-stage time integration methods for explicit (Forward Euler), fixed-point implicit (Backwards Euler), and Jacobian-coupled implicit (Backwards Euler) routines. Figure 1 plots the one-step prediction for an increasing timestep and increasing time constant C. Repeated, this experiment only considers one step with a varying timestep Δt from 0 to 2. Forward Euler behaves as expected with a linear prediction based on the solution derivative at $y(0)$. Fixed point iteration of the backwards Euler method converges for small timesteps and diverges for larger timesteps; this situation is what Hairer and Wanner mean by fixed-point iteration stability. The fixed point scheme becomes unstable for timesteps larger than approximately $1/C$. This is consistent with the previous assertion that timestep multiplied by eigenvalues must be small. The Backwards Euler method with Jacobian information converges for all timesteps.

The point to take away is that just because a scheme is iterative does not mean that it is guaranteed to converge. Nor does an arbitrary iterative scheme always allow larger timesteps than an explicit scheme. Fixed point iteration does indeed destroy the stability advantages of an implicit scheme. As Boyd illustrates[2], implicit methods track the slow manifold.

1.2 Numerical Matrix Inversion

Matrix inversion is a critical operation for effective PDE solver design. The canonical form for matrix inversion is

$$Ax = b$$

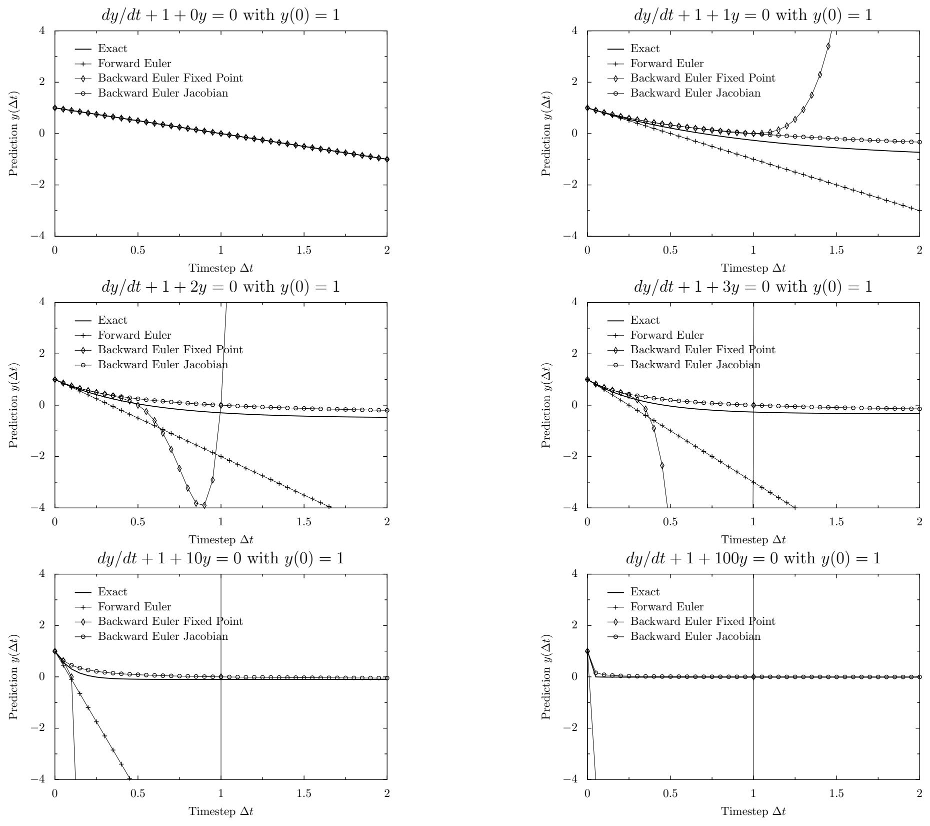

Figure 1: Time Integration Experiment

In residual form for iteration, the canonical form is

$$r = b - Ax$$

The objective is to reduce the residual r to zero.

The numerical matrix inversion literature is large and continuously evolving. The *Templates* book[1], Boyd's book[2], and [9] are useful starting points for investigating iterative methods. Preconditioning and other advanced routines[3] are known to improve convergence rates.

1.2.1 Generic Jacobi and Krylov Iteration

Jacobi iteration updates the state vector with the residual scaled for stability.

$$x_{i+1} = x_i + \alpha_i r_i$$

where α is chosen based on an approximation to A's eigenvalues[11].

1.2.2 Richardson Residual Minimization

The Richardson Residual Minimization method[2] uses Jacobi iteration with

$$\alpha_i = \frac{\sum r_i q_i}{\sum q_i q_i} = \frac{\sum r_i r_i}{\sum r_i q_i}$$

where q_i is calculated as

$$q_i = A r_i$$

This method is a linear equation residual minimization along the steepest descent direction. A should be positive semi definite but not necessarily symmetric[11].

1.2.3 Conjugate Gradient

The conjugate gradient (CG) method is popular with nice convergence properties at the expense of more storage. CG methods also require a positive semi-definite symmetric A. Shewchuk[10] provides an excellent foundation to the various CG methods. Press[8] shows Polak-Ribiere correction for β as

$$\beta_i = \frac{(r_{i+1} - r_i) \cdot (r_{i+1})}{r_i \cdot r_i}$$

This correction gracefully adapts the CG to a soft restart. Computing the correction is however not quite as graceful.

1.2.4 Preconditioning

Preconditioning the inversion improves the iterative process. The general idea is that the inverse of A is difficult, but an approximation to A is easy to invert. So premultiply by the approximation P^{-1}

$$P^{-1}Ax = P^{-1}B$$

Naturally, if $P^{-1}A = I$, then there is no need to iteratively invert A. Yet, when P contains some fundamental portions of A's eigenvectors, then $P^{-1}A$ becomes more diagonal. The tradeoff is finding a sufficiently complex but invertible approximation to A.

References

[1] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition.* SIAM, Philadelphia, PA, 1994.

[2] John P. Boyd. *Chebyschev and Fourier Spectral Methods.* Dover, Mineola, NY, 2 edition, 2000.

[3] W. La Cruz and M. Raydan. Residual iterative schemes for large-scale nonsymmetric positive definite linear systems. *Comput. Appl. Math.*, 27(2), 2008.

[4] C. W. Gear and Y. Saad. Iterative solution of linear equations in ODE codes. *SIAM J. Sci. Stat. Comput.*, 4(4):583–601, December 1983.

[5] Ernst Hairer, Syvert Paul Nørsett, and Gerhard Wanner. *Solving ordinary differential equations I: Nonstiff problems*, volume 1. Springer-Verlag, Berlin, 2nd edition, 2008.

[6] Ernst Hairer and Gerhard Wanner. *Solving ordinary differential equations II: Stiff and differential-algebraic problems*, volume 2. Springer-Verlag, Berlin, 2nd edition, 2010.

[7] Arieh Iserles. *A First Course in the Numerical Analysis of Differential Equations.* Cambridge University Press, Cambridge UK, 1996.

[8] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, Cambridge, 2 edition, 1992.

[9] Yousef Saad. *Iterative Methods for Sparse Linear Systems.* SIAM, Philadelphia, PA, 2nd edition, 2003.

[10] J.R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[11] Gilbert Strang. *Introduction to Applied Mathematics.* Wellesley-Cambridge, Wellesley, MA, 1986.