Ordinary Differential Equation Models: Numerical Integration of Initial-Value Problems

Although the concept of ordinary differential equations (ODEs) as modelling formalism is quite old (see *Simulation Modelling Formalism: Ordinary Differential Equations*), only the advent of modern computer technology has made ODEs solvable in a very general sense. In this article we shall try to classify some of the more popular numerical integration schemes. As there does not exist any numerical integration algorithm which is equally well suited to all types of ODE models, we shall also try to classify the models to some extent.

1. General

Given a set of ODEs represented as

$$\dot{x} = f(x, t) \qquad x(t_0) = x_0 \quad t \in [t_0, t_f] \quad x \in \mathbb{R}^n \tag{1}$$

which is the representation most commonly used in numerical integration, we want to find the solution $x(t)$ at times $t \in [t_0, t_f]$, given a consistent set of initial values specified at time $t = t_0$. In this case, we talk of an initial-value problem, as opposed to a boundary-value problem (Yen 1979) which we obtain when specifying consistent sets of boundary values at several different instants of time $t_i \in [t_0, t_f]$. As these boundary-value problems require quite different solution techniques, we shall restrict further discussion to the treatment of initial-value problems only.

All methods used have in common that they apply extrapolation techniques to extrapolate the solution at a given time t, say $t = t^*$, to the solution some small amount of time later:

$$x(t = t^*) \rightarrow x(t = t^* + \Delta t) \tag{2}$$

The most common approach is by expanding a Taylor series around time t^*:

$$x(t^* + \Delta t) = x(t^*) + \Delta t \cdot \dot{x}(t^*) + \frac{\Delta t^2}{2!} \cdot \ddot{x}(t^*) + \cdots \tag{3}$$

which is an infinite-order polynomial in Δt. It is quite evident that the higher-order terms in this polynomial lose their importance when Δt is made sufficiently small. That is, to compute the polynomial approximation sufficiently accurately, we can either consider many terms of the Taylor series (high-order approximation) with a large step size, or a few terms with a small step size.

Except for a few "exotic" approaches, all currently used algorithms follow this basic idea. However, it is not the only conceivable one and this approach is perhaps not even the most natural one. Thus we could argue along the following lines: as Δt is made smaller and smaller, the nonlinearities in the state-space description also become more and more negligible:

$$\dot{x} = f(x, t) \rightarrow \dot{x} = A(t) \cdot x \tag{4}$$

It is therefore sensible to search for an algorithm which solves the *linear* initial-value problem

$$\dot{x} = A \cdot x \qquad x(t_0) = x_0 \qquad x \in \mathbb{R}^n \tag{5}$$

in an optimal way. For this problem, however, we know the analytic solution. If all eigenvalues λ_k of the matrix A are distinct, we may write

$$x(t) = \sum_{k=1}^{n} c_k \exp(i\lambda_k t) \qquad c_k \in \mathbb{R} \quad \lambda_k \in \mathbb{C} \tag{6}$$

For multiple eigenvalues, we obtain some additional terms of the form $c_k t^l \exp(i\lambda_k t)$, where $l \in [0, 1, \ldots, (L-1)]$ with L a multiple of eigenvalue λ_k. By this approach, we can compute the true solution of the linear problem by a finite rather than an infinite sum. For nonlinear systems, we may use the same approach by linearizing around a working point. The linear part, of which we have to compute the eigenvalues, is the Jacobian of the system:

$$J(t) = \left.\frac{\partial f(x, t)}{\partial x}\right|_{x = x(t)} \tag{7}$$

In this approach, the solution is approximated by a sum of (complex) exponentials rather than by a polynomial sum. Besides the classical methods for eigenvalue computation (Smith *et al.* 1974, 1977), there also exist some iterative methods (Peters and Wilkinson 1970) which are probably better for our purpose, as the eigenvalues change only slightly from one step to the next.

A third approach may be to look for algorithms which make use of a Fourier series decomposition rather than a Taylor series expansion. This approach was tried quite early by Brock and Murray (1952). It had little success, however, as the recomputation of the Fourier coefficients after each step was very expensive. However, at that time the fast Fourier transform (FFT) (Brigham

1974) was not yet developed. This is a method which also permits a high degree of parallelization (pipelining) for improved efficiency (Bergland 1972). The time appears to have come to elaborate on that approach once more. In particular, this method has some potential for the solution of oscillatory problems (complex λ_k), which we shall discuss in Sect. 7.

2. Numerical Stability

Let us return to the linear problem of Eqn. (5). The values of A for which this problem has a stable solution, that is,

$$\lim_{t\to\infty} x(t) \to 0$$

are given by the condition that all eigenvalues of A must have a negative real part, that is, all eigenvalues must lie in the open left-hand complex half-plane.

Is this fact also reflected by the numerical solution? Only if this is the case is the numerical solution said to converge to the true solution. Let us look at the linear approximation of the Taylor series first:

$$x(t^* + \Delta t) = x(t^*) + \Delta t \cdot \dot{x}(t^*) \tag{8}$$

which is known as Euler's integration rule. Substituting in Eqn. (5),

$$\begin{aligned} x(t^* + \Delta t) &= x(t^*) + \Delta t \cdot A \cdot x(t^*) \\ &= (I + \Delta t \cdot A)x(t^*) \end{aligned} \tag{9}$$

A stable solution is found for all A such that $|I + \Delta t \cdot A| < 1$ that is, all λ_k multiplied by Δt must lie in a circle of radius 1 around the point -1 (Fig. 1). Therefore, given a stable system described by matrix A with eigenvalues λ_k, the step size Δt must be chosen sufficiently small to keep all $\lambda_k \Delta t$ within the domain of numerical stability. Outside this domain, the numerical algorithm is unstable.

Let us now discuss what happens when we modify Euler's integration rule such that the derivative is used at point $(t^* + \Delta t)$ instead of t^*:

$$x(t^* + \Delta t) = x(t^*) + \Delta t \cdot \dot{x}(t^* + \Delta t) \tag{10}$$

Substituting in Eqn. (5), we obtain

$$x(t^* + \Delta t) = (I - \Delta t \cdot A)^{-1} \cdot x(t^*) \tag{11}$$

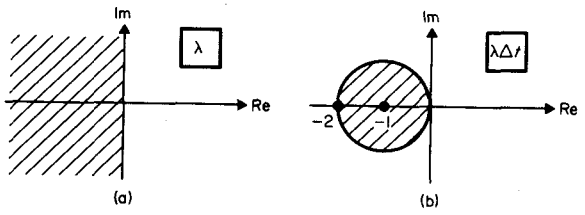

Figure 1
(a) Domain of analytic stability; (b) domain of numerical stability of Euler's integration rule

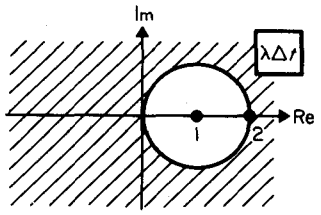

Figure 2
Domain of stability of modified Euler's integration rule

Here we obtain the domain of stability shown in Fig. 2. This time we have derived an algorithm which exhibits stability throughout the negative half-plane (the so-called A-stable algorithm). Unfortunately, we may now also observe a numerically stable solution in the case of algebraically unstable problems.

Each numerical integration scheme has a stability domain in the $(\lambda \cdot \Delta t)$ plane which is characteristic of it. A numerical algorithm that reflects the analytic stability correctly cannot exist, as the numerical algorithm cannot distinguish between the points $+\infty$ and $-\infty$. Therefore, either both points exhibit stability behavior or both exhibit instability.

More about numerical stability can be found in any book dealing with numerical initial value problems (e.g., Björck and Dahlqvist 1974, Gear 1971, Lambert 1973).

3. Explicit versus Implicit Integration

Looking at the above two algorithms, an important difference can be seen with respect to computability. In the former case—that is, forward Euler integration—we use the (known) state x at time t_0 to compute the first derivative $\dot{x}$ at time t_0. Now, the Euler formula can be used to compute the new state x at time $(t_0 + \Delta t)$ and so on. Such a scheme is called *explicit*. In the latter case (backward Euler integration) this approach fails and we have to solve simultaneously a set of nonlinear algebraic equations:

$$\begin{aligned} x(t^* + \Delta t) &= x(t^*) + \Delta t \cdot \dot{x}(t^* + \Delta t) \\ &= x(t^*) + \Delta t \cdot f(x(t^* + \Delta t), t^* + \Delta t) \end{aligned} \tag{12}$$

We call such an algorithm *implicit*. Obviously, explicit integration is easier to perform and is therefore more commonly used in simulation. However, implicit integration generally exhibits better stability properties.

The predictor–corrector methods use an explicit integration step for predicting the next state and then correct this prediction by a second implicit scheme using the previously computed prediction on the right-hand side. For example, we may have for the explicit predictor (forward Euler)

$$\left.\begin{aligned} \dot{x}(t^*) &= f(x(t^*), t^*) \\ x^{\mathrm{P}}(t^* + \Delta t) &= x(t^*) + \Delta t \cdot \dot{x}(t^*) \end{aligned}\right\} \tag{13}$$

and for the implicit corrector (backward Euler)

$$\left.\begin{aligned} \dot{x}^{\mathrm{P}}(t^* + \Delta t) &= f(x^{\mathrm{P}}(t^* + \Delta t), t^* + \Delta t) \\ x(t^* + \Delta t) &= x(t^*) + \Delta t \cdot \dot{x}^{\mathrm{P}}(t^* + \Delta t) \end{aligned}\right\} \quad (14)$$

This is not necessarily the best algorithm to use, as a very small modification can raise this first-order approximation to a second-order approximation.

4. *One-Step versus Multistep Methods*

So far we have considered only first-order approximations of the Taylor series which can directly be evaluated from the state-space description. For more accurate evaluations, we require higher-order derivatives. These are most commonly approximated by one of two approaches.

(a) Additional function evaluations are performed at some auxiliary time instants $t_i \in [t^*, t^* + \Delta t]$. Such methods are called one-step methods, the most prominent being the Runge–Kutta methods (Fehlberg 1968).

(b) Older function values from previous steps are used to approximate the higher-order derivatives. Such methods are called multistep methods, the most prominent being the Adams methods (Lambert 1973).

Quite obviously, these two classes of algorithm behave somewhat differently in several respects.

(a) Multistep methods require precisely one function evaluation per step, whereas one-step methods require several. Multistep methods store numerical information over many steps, whereas one-step methods discard all previous information at the end of each step. Therefore, multistep methods are generally more economical for problems with constant numerical properties, that is, for systems which are linear or close to linear. One-step methods behave better for highly nonlinear systems, that is, for systems with varying numerical properties.

(b) One-step methods are self-starting, whereas multistep methods need to be started either by low-order approximations or by use of a one-step method. Therefore one-step methods are more economical for integration over a few steps, whereas multistep methods may be better for longer integration.

(c) Multistep methods need to be restarted after each discontinuity. Therefore, one-step methods are better for strongly discontinuous problems.

(d) Step-size control is easier to perform on one-step than on multistep methods. To allow nevertheless for a more or less economic step-size adjustment, modern multistep methods relate the set of $(k + 1)$ function values back to the vector of states and the first up to kth state derivatives (the so-called Nordsieck vector). In this way, a step-size adjustment results in a recomputation of new supporting function values rather than in a complete restart of the algorithm.

5. *Semianalytic Methods*

A very modern approach, which also may be strongly parallelized (Halin *et al.* 1980), is to combine numerical methods with methods of computer science. If, for instance, the normal state-space representation $\dot{x} = f(x, t)$ is explicitly given (no external driving functions in a real-time environment), one may easily compute

$$\ddot{x} = \frac{\partial f}{\partial t}(x, t) \qquad \dddot{x} = \cdots \quad (15)$$

algebraically at compile time (Joss 1976), and these are then evaluated in a straightforward manner at run time for the evaluation of the higher-order Taylor series terms. In this way, the run-time execution may be economized at the expense of a slower compilation.

6. *Stiff Problems*

Let us now describe a few classes of problem which call for special integration techniques. In many problems in both engineering and physics, the eigenvalues of their Jacobian matrix exhibit a widespread range of eigenmodes.

If we remember the charts of stability domains in the $(\lambda \cdot \Delta t)$ plane, it is quite clear that stiff problems require very small step sizes to be taken to keep all $\lambda_k \cdot \Delta t$ within the stable domain, except when an A-stable method is applied. Unfortuantely, there exist no higher-order A-stable approximations. For this reason, the specifications are somewhat weakened by requiring that a method be (A, α)-stable with α only slightly less than 90° (Fig. 3).

Such methods exist and are good for all sufficiently damped "stiff" systems. These methods fail entirely when applied to highly oscillatory problems which have

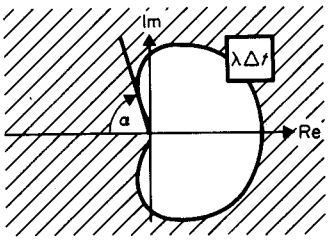

Figure 3
Domain of stability for stiff problems

eigenvalues close to the imaginary axis. The best currently known methods with such behavior are the backward-difference formulae (BDF methods), which are also often called Gear algorithms, as the first efficient implementation of such an algorithm was made by Gear (1971). BDF methods belong to the implicit multistep formulae. The most up-to-date implementations have been developed by Hindmarsh (1983).

7. *Highly Oscillatory Problems*

As we have seen, we still lack an appropriate algorithm for problems with eigenvalues close to the imaginary axis (Gear 1983). Recently, Petzold (1978) developed a pseudostroboscopic method for this purpose. The idea behind this algorithm is fairly simple. First, we integrate with one algorithm using a small step size over a few periods of the fact oscillation. During this integration we store all maxima values. Then we use these maxima as the supporting values of another problem, which no longer shows these fast oscillations (equivalent to the envelope), to extrapolate by the use of another integration algorithm with a large step size to a new point on the envelope. In this way we lose the information on the phase of the fast oscillation, but retain the information on frequency and shape. It is evident that this method can be used to sweep out precisely one pair of complex eigenvalues close to the imaginary axis. The method must fail when several such pairs of eigenvalues lie close to each other.

For this class of problems (and for multiple eigenvalues), the Fourier methods mentioned in Sect. 1 may prove beneficial in the future.

8. *Linear Problems*

It is of interest to note that hardly any currently available simulation software (apart from special-purpose programs for linear network analysis) offers special integration techniques for the analysis of linear systems. In fact, this important class of problems certainly deserves special treatment.

As stated above, explicit integration schemes are mostly preferred to implicit schemes (except for stiff systems), owing to a smaller computational overhead, although the implicit formulae exhibit far better stability behavior. In the case of linear models, however, the nonlinear function iteration reduces to the inversion of an $n \times n$ matrix. Moreover, if the model is time-invariant, this matrix needs to be inverted only once. Therefore, implicit algorithms are highly recommended for linear-system integration.

Moreover, we can make use of the linear structure to an even greater extent. Given a linear ODE system of the form

$$\left|\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned}\right| \qquad x \in \mathbb{R}^n \quad u \in \mathbb{R}^m \quad y \in \mathbb{R}^p \tag{16}$$

the question is whether there exists a set of difference equations of the form

$$\left|\begin{aligned} \hat{x}[(k+1)\Delta t] &= F\hat{x}(k\Delta t) + Gu(k\Delta t) \\ \hat{y}[(k+1)\Delta t] &= H\hat{x}[(k+1)\Delta t] + Iu[(k+1)\Delta t] \end{aligned}\right|$$

$$\hat{x} \in \mathbb{R}^n \quad u \in \mathbb{R}^m \quad \hat{y} \in \mathbb{R}^p \tag{17}$$

which represents the desired solutions correctly at each sampling point, that is:

$$\left.\begin{aligned} \hat{x}(k\Delta t) &\equiv x(k\Delta t) \\ \hat{y}(k\Delta t) &\equiv y(k\Delta t) \end{aligned}\right\} \tag{18}$$

The answer is yes (except for the variations of $u(t)$ within the sampling intervals). By use of the z transform method (Jury 1964 and *Digital Control: Practical Design Considerations*), we derive

$$\left.\begin{aligned} F &= \exp(A\Delta t) & G &= \int_0^t \exp(A\tau)B\,\mathrm{d}\tau \\ H &\equiv C & I &\equiv D \end{aligned}\right\} \tag{19}$$

which (at least for time-invariant systems) can be computed once and for all, producing an excellent method for the "integration" of such systems. A variant of this method (Tustin's integration) which is sometimes used is described by Howe (1982).

9. *Noisy Systems*

Problems in electrical engineering often exhibit some stochastic properties which are not negligible. Such problems provide difficulties since there are no algorithms coping reasonably well with such problems. All polynomial extrapolation methods must fail, as white noise is everywhere discontinuous, whereas polynomials are everywhere continuous functions. Any step-size control mechanism or order-adjustment algorithm must fail to produce anything meaningful, as all these algorithms are based inherently on deterministic behavior. We must resort to the traditional fixed-step, fixed-order algorithms with a "sufficiently small" step size and hope for the best. Cellier (1979) gives a recipe for the selection of an appropriate step size. However, this method also fails when the signal–noise ratio is too low. Analog computation may then still be the best answer.

See also: Ordinary Differential Equation Models: Symbolic Manipulation; Moving-Boundary Models: Numerical Solution

Bibliography

Bergland G D 1972 A parallel implementation of the fast Fourier transform algorithm. *IEEE Trans. Comput.* **21**(4), 366–70

Björck Å, Dahlqvist G 1974 *Numerical Methods*. Prentice-Hall, Englewood Cliffs, New Jersey, p. 576

Brigham E O 1974 *The Fast Fourier Transform*. Prentice-Hall, Englewood Cliffs, New Jersey, p. 252

Brock P, Murray F J 1952 The use of exponential sums in step-by-step integration. *Math. Tables Aids Comput.* **6**, 63–78

Cellier F E 1979 Combined continuous/discrete system simulation by use of digital computers: Techniques and tools. Ph.D. thesis, ETH Zurich, p. 266

Fehlberg E 1968 Classical 5th-, 6th-, 7th-, and 8th- Order Runge-Kutta Formulas, Report NASA TR R-287. National Aeronautics and Space Administration, Washington, DC, p. 82

Gear C W 1971 *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall, Englewood Cliffs, New Jersey, p. 253

Gear C W 1983 Stiff software: What do we have and what do we need? In: Aiken R (ed.) *Stiff Computation.* Oxford University Press, Oxford

Halin H J *et al.* 1980 The ETH multiprocessor project: parallel simulation of continuous systems. *Simulation* **35**(4), 109–23

Hindmarsh A C 1983 Stiff system problems and solutions at LLNL. In: Aiken R (ed.) *Stiff Computation.* Oxford University Press, Oxford

Howe R 1982 Digital simulation of transfer functions. In: *Proc. Summer Comput. Simul. Conf.* AFIPS Press, Arlington, Virginia, pp. 57–63

Joss J 1976 Algorithmisches differentieren. Ph.D. thesis, ETH Zurich, p. 69

Jury E I 1964 *Theory and Application of the z-Transform Method.* Wiley, New York, p. 330

Lambert J D 1973 *Computational Methods in Ordinary Differential Equations.* Wiley, New York, p. 278

Peters G, Wilkinson J H 1970 $Ax = \lambda Bx$ and the generalized eigenproblem. *SIAM J. Numer. Anal.* **7**, 479–92

Petzold L R 1978 An Efficient Numerical Method for Highly Oscillatory Ordinary Differential Equations, Report UIUCDCS-R-78-933. University of Illinois, Urbana-Champaign, p. 131

Smith B T, Garbow B S *et al.* 1974, 1977 *Matrix Eigensystem Routines—EISPACK Guide*, Lecture Notes in Computer Science, Vol. 6. p. 387, Vol. 51. p. 343. Springer, Berlin

Yen T 1979 *Computational Methods in Engineering Boundary Value Problems.* Academic Press, New York, p. 309

F. E. Cellier