International Journal of Education and Social Development
ISSN: 3078-9931 (Print), ISSN: 3078-2287 (Online) | Vol. 8, No. 2, 2026

Design of Stable Adaptive Step Size Controller for ODE Numerical Solutions Based on Condition Number Optimization

Jia Dong

School of Tianjin Foreign Studies University, Tianjin, 300204, China

Abstract: A numerical method to solve ordinary differential equations (ODEs) is used in computational science for many kinds of complex dynamic systems in all fields of engineering and science. Conventional adaptive step-size controllers are mainly concerned with bounding the local truncation error to maintain solution accuracy. However, for stiff systems, implicit numerical methods need to solve non-linear algebraic equations at each step and usually employ Newton-Raphson iterations. Therefore, at this time, the condition number of the iteration matrix affects both the stability and speed of convergence in practice. A new adaptive step-size controller that considers the condition number optimisation in the control loop is proposed in this paper. Continuously monitor and bound the condition number of the Jacobian-derived iteration matrix to adjust the step size dynamically in the proposed dual-loop controller, thus avoiding divergence of Newton iteration and reducing the amplification of floating-point errors. Based on the above theoretical analysis and control system construction, it has been shown that adding condition number constraints can enhance both the stability and efficiency of numerical integration for stiff ODEs. The above results provide an all-encompassing system for the development of high-accuracy and algebraically stable numerical solvers.

Keywords: Ordinary Differential Equations; Adaptive Step Size; Condition Number Optimization; Numerical Stability; Jacobian Matrix; Control Theory.

1. Introduction

Numerical solutions of ordinary differential equations have provided support for the current times of computational science and engineering in applied mathematics. Dynamical systems with complex behaviour, such as celestial mechanics and molecular dynamics, electrical circuit simulation, and epidemiological modelling, are all fundamentally described by ordinary differential equations. Due to the absence of a simple closed-form analytical expression for non-linear or high-dimensional systems, numerical integration methods must be used to approximate the time-varying behaviour of such systems by researchers and engineers. Recently, due to the high-computational-cost nature of many problems, research efforts to improve the efficiency, accuracy and stability of numerical solvers have been intensified. An early example of adaptive step-size methods for numerical integration is a change away from fixed-step methods. Dynamically adjust the temporal discretization interval based on the local behaviour of the solution and thus reduce computational cost in areas where it is not needed. Small time steps are used for rapid changes in transients, and larger time steps are employed in relatively smooth and slowly varying areas to reduce computational costs without sacrificing accuracy.

Adaptive step-size controllers have been performing well in recent years, but there are still problems with stiff differential equations. Stiffness is a mathematical property of a dynamical system that indicates the presence of widely varying time scales in different components of that system. In such systems, the very fast transient components decay quickly and are only followed by slow dynamics that determine the long-term behaviour of the solution. Although explicit numerical methods are theoretically able to handle stiff systems, they are restricted by absolute stability conditions and need to take very small step sizes even when the active solution is smooth. Therefore, the typical method for stiff problems is an implicit numerical method, and since it is unconditionally stable, its step size needs only to meet the accuracy requirement.

Implicit methods also have an additional problem of computational complexity, and at each integration step, a system of nonlinear algebraic equations needs to be solved. Newton-Raphson iteration and its modified forms are used almost exclusively for this purpose. The convergence and stability of these iterative algebraic solvers are largely determined by the characteristics of the iteration matrix, which itself is obtained from the Jacobian of the system. If the step size is too large, the iteration matrix will be very ill-conditioned. An ill-conditioned matrix will amplify floating-point round-off errors and significantly reduce the convergence speed of the Newton solver; thus, it may fail to converge and subsequently reject a step. Repeatedly reject a step; otherwise, it will be slow to compute or fail entirely.

Most current state-of-the-art adaptive controllers focus almost exclusively on bounding the local truncation error and have neglected the algebraic stability of the underlying implicit solver. When a step fails due to non-convergence, conventional controllers only employ arbitrary heuristic reductions in step size and do not address the underlying mathematical reason. This paper addresses the above deficiency by introducing a new kind of stable adaptive step-size controller that optimizes the condition number.

2. Theoretical Framework of Adaptive Step Size Control

Adaptable step-size control mechanisms are, in essence, feedback-control systems for the domain of numerical integration. A general adaptive solver sets an upper limit on the local truncation error to ensure that this value does not

exceed a set threshold. Usually, two approximations of the solution at each time step are computed simultaneously: a high-order approximation and a lower-order approximation. The difference between the two calculated values is asymptotically accurate for the local truncation error. If the estimated error is smaller than the tolerance, then a step is taken; otherwise, the step size of the next integration interval will be increased. Conversely, if the estimated error exceeds the tolerance, then the step is rejected and the solver needs to recompute the current interval using a reduced step size.

The first applications of adaptive stepping used simple multiplicative factors based on the asymptotic behaviour of the truncation error. However, these basic methods often led to irregular sequences of step sizes that were either too large or too small at various times. Oscillating behaviour is generally referred to as step-size chattering, and it reduces the efficiency of a numerical solver considerably. To address the above instability, research has been carried out to introduce control-theoretic methods for step-size selection in explicit Runge-Kutta methods [1]. A digital control loop was established for the selection of step size, and therefore, traditional linear control theory could be applied to analyze and optimise the dynamics of the step size sequence.

Based on the above, the proportional-integral (PI) and proportional-integral-derivative (PID) controllers have been adopted as the standard in automatic control and adaptive time-stepping [4]. The proportional component of a PI controller is determined based on the current local truncation error, and the Integral component accumulates the error of the previous step. Thus, this historical awareness reduces the oscillation of the step size sequence and achieves a relatively stable and efficient integration process. The mathematical expression of the PI controller contains tuning parameters that determine how quickly the step size changes; thus, solvers need to be selected according to the characteristics of the differential equations being solved. The precise mathematical analysis of the above control loops is based on digital signal processing theory. By considering the sequence of step sizes and error estimates as discrete-time signals, Z-transform can be used to find the transfer functions of controllers. Frequency-domain analysis can be used to determine the stability and transient behaviour of the adaptive mechanism asymptotically.

Although PI controllers are very good at reducing the local truncation error locally, they are not suitable for implicit methods and highly stiff systems. The size of the local truncation error in a stiff system is not the only bound for a step size. Management of the convergence properties of the Newton iteration is also required for the stability of differential-algebraic equations or coupled algebraic systems [8]. The standard PI controller does not know about the algebraic health of the implicit solver. They will increase the step size rapidly as long as the truncation error is small and inadvertently move the iteration matrix into an ill-conditioned region. When the algebraic solver is forced to fail, the PI controller cannot react in time, so it reduces the step size and wastes the work done by the solver. A reactive paradigm is being introduced; thus, more all-encompassing control strategies need to be developed to prevent algebraic instability proactively.

3. The Role of Condition Number in Numerical Stability

The condition number of a matrix is a relatively small value, and therefore, small perturbations in the input data will not lead to large changes in the solution of the linear system. In the field of numerical linear algebra, it offers a reliable upper bound on the amount of error that can be magnified by various means, such as floating-point round-off or approximation inaccuracies, during matrix inversion and linear system solving. For the accuracy and stability of the numerical algorithm to be maintained, a relatively small condition number must be achieved; otherwise, if the matrix is severely ill-conditioned, the computed solution will be unreliable due to catastrophic cancellation and error amplification [5].

Implicit numerical methods for stiff ordinary differential equations are employed here, and in calculating the next state vector, a nonlinear system needs to be solved. It is generally achieved by a modified Newton's method, and the iteration matrix is given by M = I - h * gamma * J, where I is the identity matrix, h is the integration step size, gamma is a method-specific parameter, and J is the Jacobian matrix of the system of differential equations. The speed of convergence and stability of the Newton iteration depend on the spectrum and condition number of this iteration matrix. As the step size h increases, the term h * gamma * J becomes dominant over the identity matrix, and thus the condition number of M approaches the condition number of the Jacobian itself [3]. If the original physical system is extremely stiff, then the Jacobian will have a large spread of eigenvalues, and a highly ill-conditioned iteration matrix will result at large step sizes.

The severe effect of a large condition number in computational linear algebra can be traced back to the basic representation of real numbers in current computer systems. According to the IEEE 754 standard for floating-point arithmetic, all mathematical operations inherently introduce a small, bounded round-off error. When the iteration matrix is well-conditioned, the local errors are safely absorbed and bounded. However, when the condition number is large, the matrix is a mathematical magnifying glass that amplifies small floating-point errors exponentially; at this time, these errors may become too large for the computed solution vector. In the context of the Newton-Raphson method for implicit ODE solvers, this enlarged error shows up as a perturbed Newton update direction. Instead of moving closer to the true root of the nonlinear algebraic system, the solver takes a very large step into a mathematically invalid area of the state space and thus loses the convergence speed of the iteration.

In the past, many robust software packages have failed to consider the conditioning of the iteration matrix. For example, the original legacy code and the organised collection of ODE solvers frequently exhibited unexplained convergence failures when applied to certain classes of highly nonlinear stiff problems [7]. These failures were often mistakenly attributed to abnormalities in the physical model or errors in the error estimator, but they were, in fact, the direct mathematical result of unbounded growth in the condition number. When the condition number of M is greater than 1/machine precision, the linear solvers used in the Newton iterations will be unable to compute a stable update direction. The Newton method fails to converge, thus cannot reduce the algebraic residual, and is subsequently marked as a step rejection by the top-level solver logic.

4. Controller Design Based on Condition Number Optimization

Given the severe constraint of a badly conditioned iteration matrix, this paper introduces an optimisation loop for the condition number directly inside the adaptive step size control structure. The first idea of this design is that the step size needs to meet two separate requirements at the same time: a maximum error bound from local truncation errors, and a stability bound that prevents instability due to the condition number of the iteration matrix. A dual-loop controller is constructed to choose between the two restrictions at different times in the numerical solver and ensure both accuracy and algebraic stability are met.

The first control loop is the same as a typical PI error controller. It is based on the difference between the embedded Runge-Kutta stages to determine the local truncation error and sets a new step size according to this error. The primary loop in this classical nonstiff problem is sufficient on its own, and due to the explicit nature of the methods, algebraic iteration matrices do not need to be introduced [2]. However, the second control loop continuously monitors the conditioning of the implicit formulation. As the computation of the exact condition number of a large iteration matrix through singular value decomposition is too expensive, the proposed controller uses a computationally inexpensive iterative 1-norm condition number estimator. These estimators need only a few matrix-vector products and provide a very stable approximation of the true condition number.

The secondary control loop uses the estimated condition number to calculate a stability-based step size. The control law of this secondary loop is designed to keep the condition number well below a specific safety limit, which is generally determined by the machine precision and the required convergence properties of the Newton solver. The general integration of adaptive time-stepping and computational stability is achieved by setting the final proposed step size as the minimum of the accuracy-based step size and the stability-based step size [6]. If the system is highly transient, the local truncation error will be large, and thus the accuracy loop will limit the step size. On the other hand, if the system has reached a stable, stiff state, then the error will decay exponentially; at this time, the condition number loop will take over to avoid an excessively large step size due to algebraic instability. A relatively high-end software structure is needed to realise the application of the above dual-loop condition-number controller without sacrificing the original speed advantage.

The above two-loop structures are highly suitable for partitioning large, complex systems of convection-diffusion-reaction equations solved with additive Runge-Kutta schemes [9]. The stiff reaction term is treated implicitly in the additive scheme, and the non-stiff convection term is treated explicitly. Only the condition number controller observes the implicit iteration matrix to prevent the stiff parts from causing instability in the whole coupled integration. The realised controller design is also very compatible with the current computational framework. Feature-rich ecosystems for solving differential equations can easily support this dual-loop logic via callback interfaces and modular controller plugins [10]. With a first-class control variable of the condition number, contemporary solvers have been able to reduce the need for heuristic or arbitrary step rejection in the simulation of complicated stiff dynamical systems. This design is also interpretable; that is, every rejected or reduced step can be traced back to either a differential accuracy constraint or an algebraic conditioning constraint, and not regarded as an unexplained numerical failure.

5. Conclusion

In short, the design of a stable adaptive step size controller based on condition number optimisation has achieved a new level of performance for the numerical integration of ordinary differential equations. Only the traditional adaptive algorithm has access to a local truncation error estimate and thus cannot handle the severe algebraic instability that occurs in the integration of highly stiff systems. By permitting an unbounded step size in the smooth region, error-based controllers may inadvertently drive the implicit iteration matrices into a state of severe ill-conditioning and thus cause Newton iteration to diverge or become computationally infeasible.

The new dual-loop control structure can directly correct for the above mathematical flaw by selecting the condition number as the first control factor. Dynamically estimate the condition number of the iteration matrix at each step and proactively limit the increase in step size before algebraic instability occurs. The two constraints restrict both the error in differentiation and algebraic tractability; therefore, most step rejections due to non-computability will not occur, and the non-linear solver will exhibit stable, monotonic convergence. A new model of all-around, dual-constraint optimisation, rather than one focused on errors, will probably solve the empirical problems of the past few decades in high-performance computing.

As mathematical models in all fields of science continue to increase in complexity, dimension and stiffness, the corresponding numerical engines need to be enhanced to ensure high-precision and stable calculations independently. The addition of condition number monitoring provides an 'algebraically-aware' mathematical system for the numerical solver to reduce blind computation. In addition to the first two, this architecture has opened up new directions for deep mathematical research on the coupled dynamics of differential approximation and algebraic resolution. Future research will likely expand this control framework to cover tightly coupled multi-physics simulations, and the connection between time steps and algebraic conditioning will be relatively far-reaching.

References

[1] Gustafsson, K. (1991). Control theoretic techniques for stepsize selection in explicit Runge-Kutta methods. ACM Transactions on Mathematical Software, 17(4), 533-554. https://doi.org/10.1145/210232.210242

[2] Hairer, E., Nørsett, S. P., & Wanner, G. (1993). Solving ordinary differential equations I: Nonstiff problems (2nd ed.). Springer-Verlag. https://doi.org/10.1007/978-3-540-78862-1

[3] Hairer, E., & Wanner, G. (1996). Solving ordinary differential equations II: Stiff and differential-algebraic problems (2nd ed.). Springer-Verlag. https://doi.org/10.1007/978-3-642-05221-7

[4] Söderlind, G. (2002). Automatic control and adaptive time-stepping. Numerical Algorithms, 31(1-4), 281-310. https://doi.org/10.1023/A:1021160023092

[5] Higham, N. J. (2002). Accuracy and stability of numerical algorithms (2nd ed.). SIAM. https://doi.org/10.1137/1.9780898718027

[6] Söderlind, G., & Wang, L. (2006). Adaptive time-stepping and computational stability. Journal of Computational and Applied Mathematics, 185(2), 225-243. https://doi.org/10.1016/j.cam.2005.03.008

[7] Hindmarsh, A. C. (1983). ODEPACK, a systematized collection of ODE solvers. In R. S. Stepleman et al. (Eds.), Scientific Computing (pp. 55-64). North-Holland.

[8] Ascher, U. M., & Petzold, L. R. (1998). Computer methods for ordinary differential equations and differential-algebraic equations. SIAM. https://doi.org/10.1137/1.9781611971392

[9] Kennedy, C. A., & Carpenter, M. H. (2003). Additive Runge-Kutta schemes for convection-diffusion-reaction equations. Applied Numerical Mathematics, 44(1-2), 139-181. https://doi.org/10.1016/S0168-9274(02)00138-1

[10] Rackauckas, C., & Nie, Q. (2017). DifferentialEquations.jl: A performant and feature-rich ecosystem for solving differential equations in Julia. Journal of Open Research Software, 5(1), Article 15. https://doi.org/10.5334/jors.151