Stiff Differential Equations

Let us consider the IVP

$$\ddot{x} + c\dot{x} + kx = 0, \quad x(0) = 0, \ \dot{x}(0) = 1, \qquad (1)$$

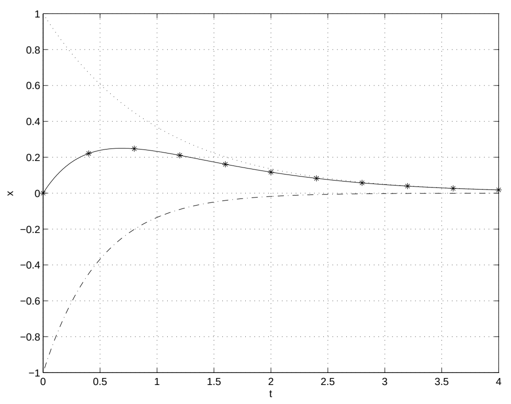
of a simple mass-spring-damper system, the response of which we want to study. We consider the following three cases of the parameter values. The obvious solutions are also given alongside.

(a) $c = 3$, $k = 2$: $\;x = e^{-t} - e^{-2t}$,

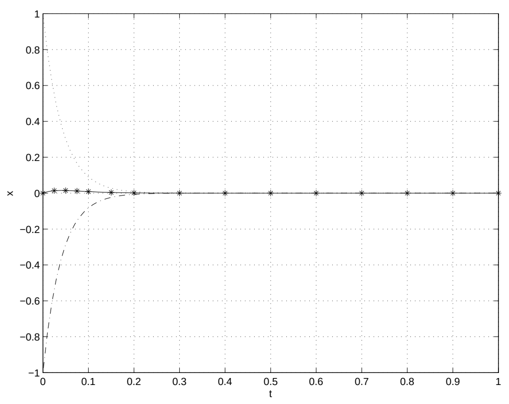
(b) $c = 49$, $k = 600$: $\;x = e^{-24t} - e^{-25t}$, and

(c) $c = 302$, $k = 600$: $\;x = \frac{e^{-2t} - e^{-300t}}{298}$.

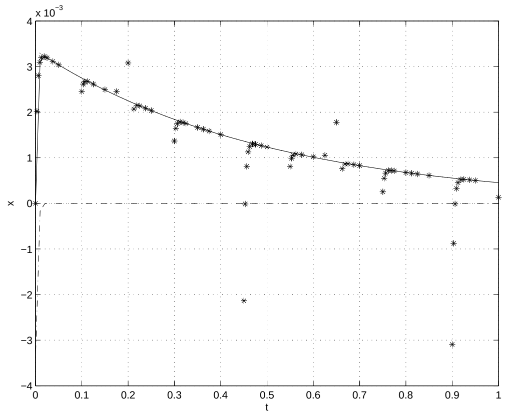
Fig. 1 shows these solutions (by solid lines), their components (by dashed and dotted lines) and the nodes of an adaptive RK4 solution marked with asterisks. [1]



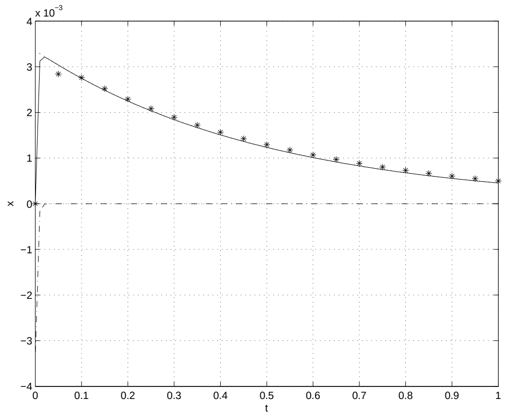
(a) Case of $c = 3$, $k = 2$



(b) Case of $c = 49$, $k = 600$



(c) Case of $c = 302$, $k = 600$



(d) Case (c) with implicit method

Figure 1: Solutions of a mass-spring-damper system

In case (a), the rate of decay of each component as well as the complete solution is moderate. Therefore, RK4 solves the case quite casually with the originally suggested mesh of ten intervals, without subdividing any interval, which is evident from the uniform spacing of the asterisks falling on the solution curve. [2]

In case (b), the spring is 'stiffer' compared to case (a). So, the response changes more rapidly. In the example, the damping coefficient c is also increased so as to keep the solution exponential, to make a reasonable comparison. Even if c is kept small, the solution would still change rapidly, except that it would oscillate. In any case, in order to capture a rapidly changing solution (or a component of it), it is reasonable to expect adaptive RK4 to reduce the step size, as noticeable from the higher density of asterisks in the plot of Fig. 1(b). [3]

Note that the term 'stiffness' in the jargon of differential equations owes its name to the stiffness of this spring. Still, it can be argued that case (b) in this example is qualitatively no different from case (a), except for a change in the time scale. For example, if you measure time in a different unit (say, in 'deciseconds', rather than seconds), such that $\tau = 10t$, then the ODE becomes $\frac{d^2x}{d\tau^2} + 4.9\frac{dx}{d\tau} + 6x = 0$, not very different from case (a) in terms of the order of magnitude of the coefficients. Therefore, in a true sense, the 'differential equation' in case (b) is not really stiff, even if the 'spring' is. Adaptive RK4 correctly reflects this fact by its successful solution with reduced step size. Such cases do not pose a great problem, because all components of the solution changing equally fast also brings down the total time span of the simulation that is relevant. Similar is the situation with a single first order ODE, any stiff behaviour of which results purely from a gross difference of order between its characteristic time scale and the duration for which it is being integrated. [4]

The true colours of a stiff differential equation are exhibited in case (c). The rapidly varying component, $-\frac{e^{-300t}}{298}$, dies down extremely fast and there is a slowly changing component, $\frac{e^{-2t}}{298}$, that keeps on slogging and constitutes the complete solution in the long run. Here, the actual solution, and its accuracy, depends upon the slowly varying component, but the step size is decided by the eigenvalue $\lambda_2 = -300$ corresponding to the rapidly varying component, even after the 'magnitude' of that component has died off. Apart from increasing the computational cost, extremely small steps promote round-off errors and collect a lot of garbage in the solution. This is why adaptive RK4 produces innumerable small steps in this case and fails to capture the actual solution, as evident from a high density of asterisks in Fig. 1(c), most of them in wrong places. In this trivial case, we clearly know the source of the trouble. In an actual (nonlinear) problem, it may not be possible to decouple the components, so there is no question of solving the components separately. [5]

Note that a change in the time scale will not help in case (c). But, an implicit method will. To complete this study, observe the solution of the equation of case (c) by backward Euler's method in Fig. 1(d). In this case, the accuracy is reasonable (with a step size of 0.05) and stability is perfect. [6]

Thus, to summarize, a stiff system of differential equations is characterized by solution components with widely varying rates of change, or widely varying orders of magnitude of the eigenvalues of the Jacobian $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$. For such a system, the slowly varying components make it necessary to solve the system for a sufficiently large domain, while the rapidly varying components put a severe limitation on the allowable step size of explicit methods. As a result, the solution process becomes extremely inefficient on one hand, while round-off errors spoil the solution quality on the other. Therefore, explicit methods are not suitable for solving stiff ODE systems. Implicit methods are found stable and fairly insensitive to the stiffness of a problem. [7]

Then, what do you do with your old RK4? Should you pack it off from your desk? Quite the contrary. For most of the routine problems that you encounter, adaptive RK4 is likely to be applicable and efficient. Therefore, if you continue using it for the usual ODE solutions as a routine matter, you enjoy our full moral support. But, when you encounter a stiff system, if you still insist on executing a billion steps of picoseconds, then we revise our moral support. In such a situation, you should switch over to an implicit method, the higher cost of which 'per step' will bring a huge payoff. [8]

In actual serious computational research, you are likely to use professional library routines for solving your ODE systems. The programmers of those routines know the programs better. But you know your system. Depending upon the nature of your problem and also upon the diagnosis of test runs, you need to call the correct library routines, in the correct manner. For example, an implicit method needs the Jacobian, and a corresponding library routine may ask for it. It may not be very difficult to estimate the Jacobian by finite differencing $\mathbf{f}(x, \mathbf{y})$ itself, but it will be of overall advantage, if you can provide a more efficient and accurate Jacobian, possibly based on analytical derivatives. [9]

Professional stiff ODE solvers do not treat the modest backward Euler's method as the last word, of course. Implicit generalizations of Runge-Kutta and Bulirsch-Stoer methods are typically employed for the purpose. Among multi-step methods, Gear's backward differentiation method is found suitable for stiff problems. See [**?**, **?**, **?**, **?**], if you are interested in details. [10]

Before we close the topic, we need to discuss one important point. How to choose an appropriate step size in an implicit method? As you can see from Fig. **??**, there is no upper limit on the step size from the stability requirement. But, accuracy suffers if a large step size is chosen. Besides, large steps may not be efficient, either. With a large step size, Newton's method may need more iterations to solve Eqn. **??**, and its convergence properties may also suffer. Thus, there is a trade-off between the number of steps and the necessity of keeping Newton's method in the safe 'local' zone, besides the *lower limit* on step size in the presence of any positive eigenvalue. As you know, there is no free lunch. We simply look for good and cheap restaurants. [11]