Chapter 8
Stiffness

Stiffness is a general (though somewhat fuzzy) term to describe systems of equations which exhibit phenomena at widely-varying scales. For the ODEs we have been studying, this means widely-varying timescales.

28 Self-Assessment

Before reading this chapter, you may wish to review...

- Eigenvalues [18.03 Lecture 25 Video]
- Eigenvalue stability

After reading this chapter you should be able to...

- describe stiffness of an ODE
- discuss the tradeoff between accuracy and computational cost
- describe what role stiffness plays in the eigenvalue stability of a method for a given problem
- implement and effectively utilize the backward Euler method
- implement and use the trapezoidal integration method
- calculate numerically and describe the meaning of the spectral condition number
- describe how finite difference discretization of the heat equation leads to a stiff system of ODEs

29 Multi-scale phenomena

One way for stiffness to arise is through a difference in timescales between a forcing timescale and any characteristic timescales of the unforced system. For example, consider the following problem:

$$u_t + 1000u = 100\sin t, \qquad u(0) = 1. \tag{68}$$

The forcing term oscillates with a frequency of 1. By comparison, the unforced problem decays very rapidly since the eigenvalue is $\lambda = -1000$. Thus, the timescales are different by a factor of 1000.

Suppose we are only interested in the long time behavior of $u(t)$, not the initial transient. We would like to take a timestep that would be set by the requirements to resolve the $\sin t$ forcing. For example, one might expect that setting $\Delta t = 2\pi/100$ (which would result in 100 timesteps per period of the forcing) would be sufficient to have reasonable accuracy. However, if the method does not have a large eigenvalue stability region, this may not be possible. In this case, the Δt set by stability requirements is much smaller than what we need for accuracy. A small Δt means that many iterations are needed, which makes the simulation computationally expensive.

For example, if a forward Euler method is applied to this problem, eigenvalue stability would limit the $\Delta t \leq 0.002$ (since the eigenvalue is $\lambda = -1000$ and the forward Euler stability region crosses the real axis at -2). The results from simulations for a variety of Δt using forward Euler are shown in Figure 29. For $\Delta t = 0.001$, the solution is well behaved and looks realistic. For $\Delta t = 0.0019$, the approach of eigenvalue instability is evident as there are oscillations during the first few iterations which eventually decay. For $\Delta t = 0.002$, the oscillations no longer decay but remain throughout the entire simulation. Finally, for $\Delta t = 0.0021$, the oscillations grow unbounded. A zoomed image of these results concentrating on the initial time behavior is shown in Figure 29.

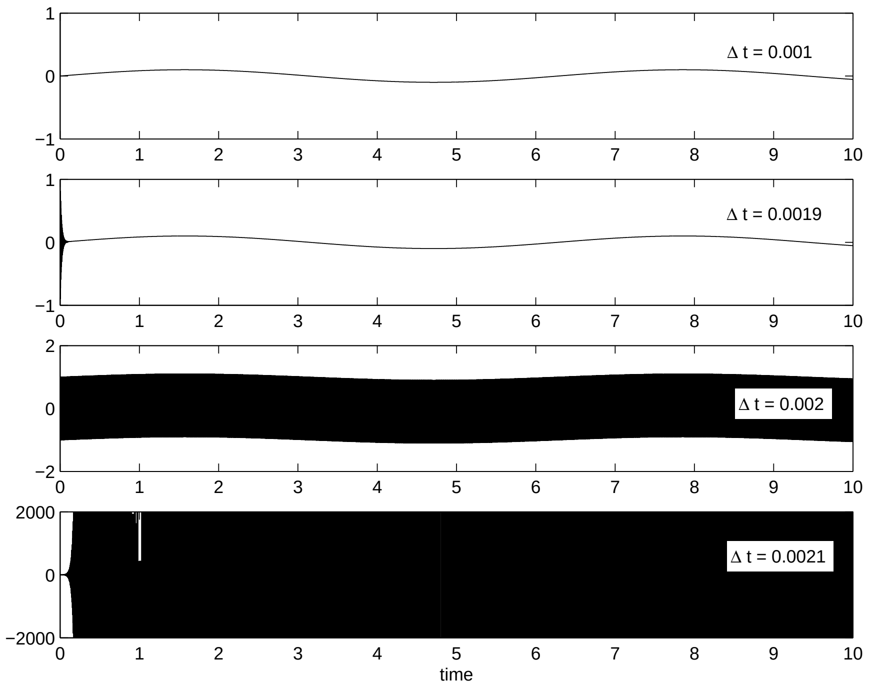

Fig. 6 Forward Euler solution for $u_t + 1000u = 100\sin t$ with $u(0) = 1$ at $\Delta t = 0.001$, 0.0019, 0.002, and 0.0021.

30 Backward Euler method

A more efficient approach to numerically integrating this stiff problem would be to use a method with eigenvalue stability for large negative real eigenvalues. Implicit methods often have excellent stability along the negative real axis. Recall from Lecture 3 that an implicit method is one in which the new value v^{n+1} is an implicit function of itself through the forcing function f. The simplest implicit method is the backward Euler method,

$$v^{n+1} = v^n + \Delta t f(v^{n+1}, t^{n+1}). \tag{69}$$

The backward Euler method is first order accurate ($p = 1$). The amplification factor for this method is,

$$g = \frac{1}{1 - \lambda \Delta t} \tag{70}$$

When λ is negative real, then $g < 1$ for all Δt. The eigenvalue stability region for the backward Euler method is shown in Figure 30. Only the small circular portion in the right-half plane is unstable while the entire left-half plane is stable.

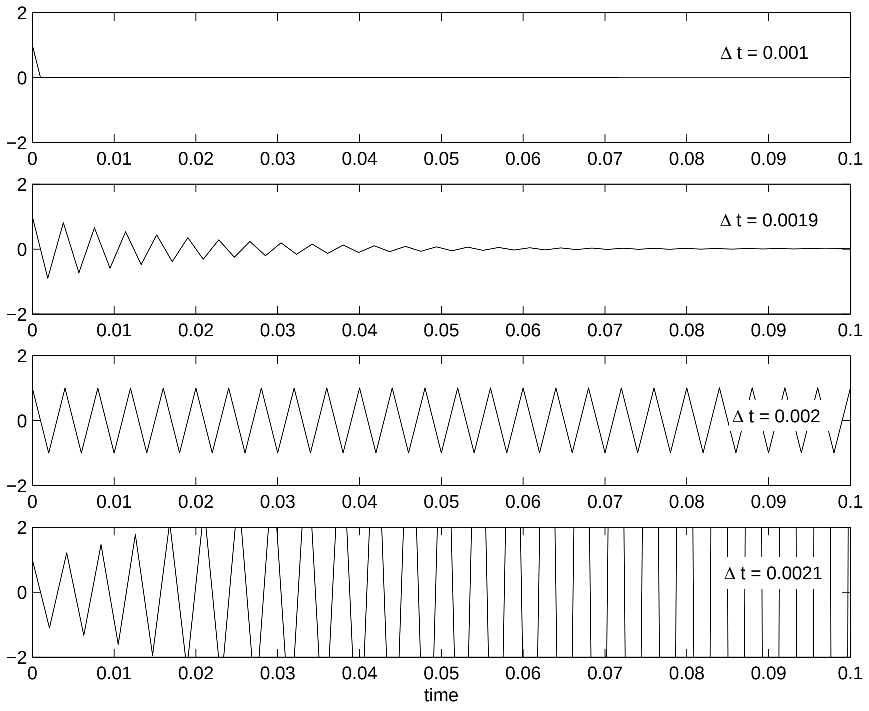

Fig. 7 Forward Euler solution for $u_t + 1000u = 100\sin t$ with $u(0) = 1$ at $\Delta t = 0.001$, 0.0019, 0.002, and 0.0021. Zoom in for smaller t.

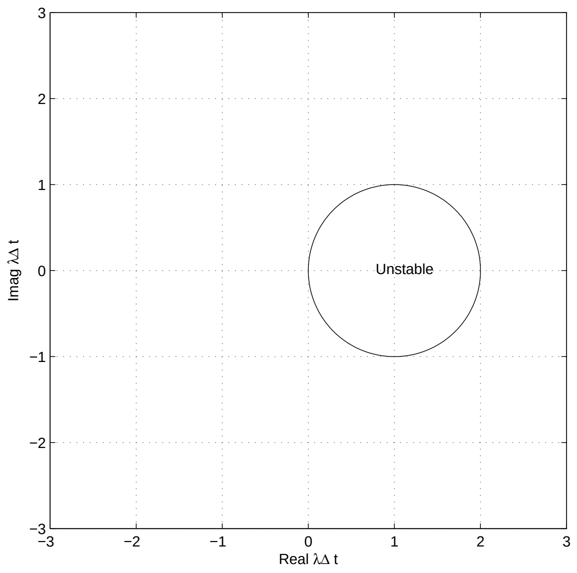

Fig. 8 Backward Euler stability region

Exercise 1. Implement the backward Euler method for Equation 68 as a Matlab function. The function should take as inputs the initial condition `u0`, the timestep `dt`, and the number of steps `N` (in that order). It should output an `N`-by-1 vector of the state `u` where each element is the value of u at each time step, excluding the initial condition (i.e., $u(1) = u^1$, $u(2) = u^2$, ...).

31 Trapezoidal integration method

Another popular implicit method is trapezoidal integration,

$$v^{n+1} = v^n + \frac{1}{2}\Delta t \left[f(v^{n+1}, t^{n+1}) + f(v^n, t^n) \right]. \tag{71}$$

Trapezoidal integration is second-order accurate ($p = 2$). The amplification factor is,

$$g = \frac{1 + \frac{1}{2}\lambda \Delta t}{1 - \frac{1}{2}\lambda \Delta t}. \tag{72}$$

The stability boundary for trapezoidal integration lies on the imaginary axis (see Figure 31). Again, this method is

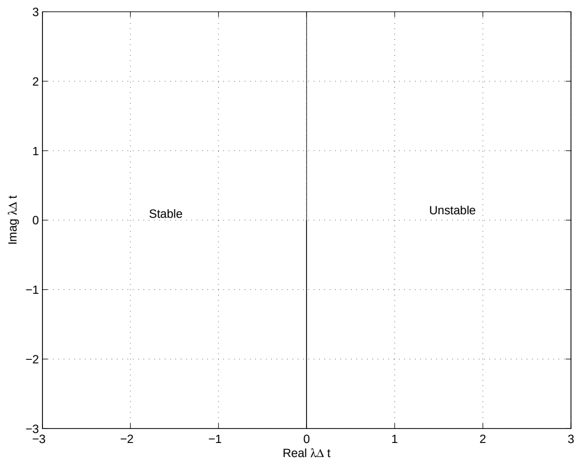

Fig. 9 Trapezoidal integration stability region

stable for the entire left-half plane thus it will work well for stiff problems.

The accuracy of the forward Euler, backward Euler, and trapezoidal integration methods are compared in Figure 31 for Equation 68. The error is computed as the maximum across all timesteps of the difference between numerical and exact solutions,

$$E = \max_{n=[0,T/\Delta t]} |v^n - u(n\Delta t)|.$$

These results show that the forward Euler method is first order accurate (since the slope on the log-log scaling is 1) once the Δt is small enough to have eigenvalue stability (for $\Delta t > 0.002$ the algorithm is unstable and the errors are essentially unbounded). In contrast, the two implicit methods have reasonable errors for all Δt's. As the Δt become small, the slope of the backward Euler and trapezoidal methods become essentially 1 and 2 (indicating first and second order accuracy). Clearly, if high accuracy is required, the trapezoidal method will require fewer timesteps to achieve this accuracy.

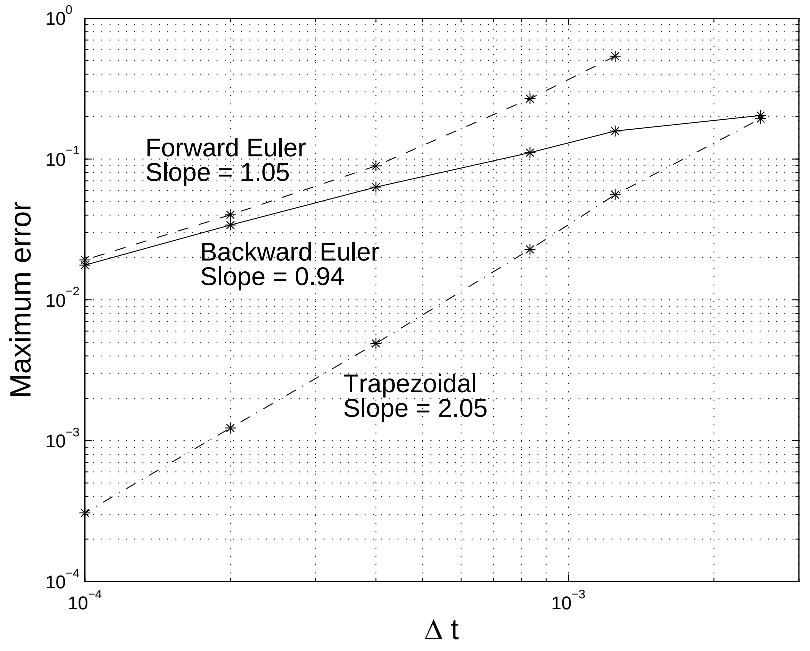

Fig. 10 Comparison of error for forward Euler, backward Euler, and trapezoidal integration versus Δt for $u_t + 1000u = 100\sin t$ with $u(0) = 1$.

32 Spectral condition number

Stiffness can also arise in linear or linearized systems when eigenvalues exist with significantly different magnitudes. For example,

$$u_t = Au, \qquad A = \begin{pmatrix} -1 & 1 \\ 0 & -1000 \end{pmatrix}.$$

The eigenvalues of A are $\lambda = -1$ and $\lambda = -1000$. Since the timestep must be set so that both eigenvalues are stable, the larger eigenvalue will dominate the timestep. The spectral condition number is the ratio of the largest to smallest eigenvalue magnitudes,

$$\text{Spectral Condition Number} = \frac{\max|\lambda_j|}{\min|\lambda_j|}$$

When the spectral condition number is greater than around 1000, problems are starting to become stiff and implicit methods are likely to be more efficient than explicit methods.

Exercise 2. What is the spectral condition number of the matrix

$$A = \begin{bmatrix} 1032 & 18 \\ 54 & -200 \end{bmatrix}.$$

Example 1. One of the more common ways for a stiff system of ODE's to arise is in the discretization of time-dependent partial differential equations (PDEs). For example, consider a one-dimensional heat diffusion problem that is modeled by the following PDE for the temperature T:

$$T_t = \frac{k}{\rho c_p} T_{xx}.$$

where ρ, c_p, and k are the density, specific heat, and thermal conductivity of the material, respectively. Suppose the physical domain of length L from $x = 0$ to $x = L$. A finite difference approximation in x might divide the physical domain into a set of equally spaced nodes with distance $h = L/(N-1)$ where N is the total number of nodes including the endpoints. So, node i would be located at $x_i = ih$. Then, at each node, T_{xx} is approximated using a finite difference derivative. For example, at node i we might use the following approximation,

$$T_{xx}|_i = \frac{T_{i+1} - 2T_i + T_{i-1}}{h^2}.$$

Note: we will discuss finite difference discretizations of PDE's in detail in Lecture 13. Using this in the heat diffusion equation, we can find the time rate of change at node i as,

$$T_t|_i = \frac{k}{\rho c_p} \frac{T_{i+1} - 2T_i + T_{i-1}}{h^2}.$$

Thus, the T_t at node i depends on the values of T at nodes $i-1$, i, and $i+1$ in a linear manner. Since each node in the interior of the domain will satisfy the same equation, we can put the finite difference discretization of the heat diffusion problem into our standard system of ODE's form,

$$u_t = Au + b, \qquad u = [T_2, T_3, T_4, \ldots, T_{N-1}]^T,$$

where A will be a tri-diagonal matrix (i.e. only the main diagonal and the two neighboring diagonals will be non-zero) since the finite difference approximation only depends on the neighboring nodes. The vector b will depend on the specific boundary conditions.

The question is how do the eigenvalues of A behave, in particular, as the node spacing h is decreased. To look at this, we arbitrarily choose,

$$\frac{k}{\rho c_p} = 1 \qquad L = 1$$

since the magnitudes of these parameters will scale the magnitude of the eigenvalues of A by the same value but not alter the ratio of eigenvalues (the ratio is only altered by the choice of h/L). Figure 1 shows the locations of the eigenvalues for $h/L = 0.1$ and $h/L = 0.05$. The eigenvalues are negative real numbers. The smallest magnitude eigenvalues appear to be nearly unchanged by the different values of h. However, the largest magnitude eigenvalues appear to have increased by a factor of 4 from approximately -400 to -1600 when h decreased by a factor of 2. This suggests that the ratio of largest-to-smallest magnitude eigenvalues (i.e. the spectral condition number) is $O(1/h^2)$. Table 1 confirms this depends for a range of h/L values and also confirms that the smallest eigenvalue changes very little as h decreases.

h/L	$\min\lvert\lambda\rvert$	$\max\lvert\lambda\rvert$	$\max\lvert\lambda\rvert / \min\lvert\lambda\rvert$
0.001	9.87	3999990	405284
0.01	9.86	39990	4052
0.1	9.79	390	40

Table 2 Minimum and maximum magnitude eigenvalues for one-dimensional diffusion

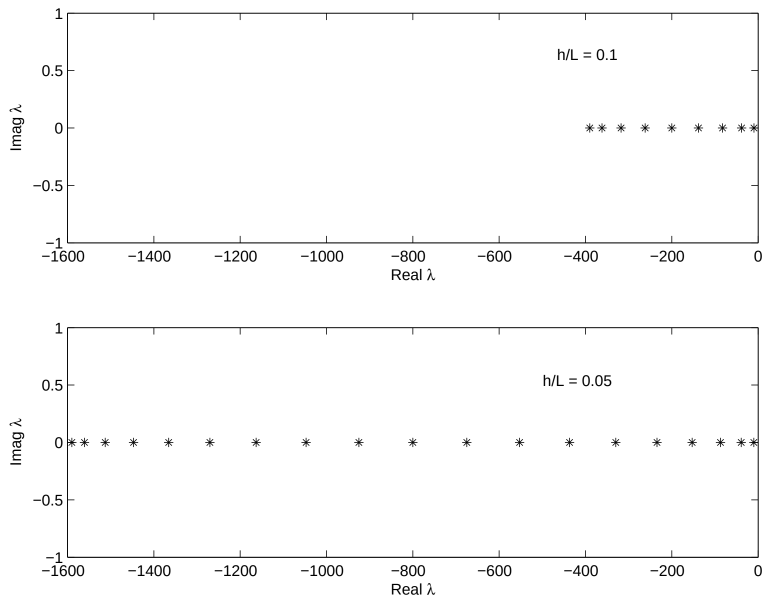

Fig. 11 Eigenvalues for discretization of one-dimenionsal diffusion equation for $h/L = 0.1$ and $h/L = 0.05$.