STIFF EQUATIONS

A problem is stiff if $f_y(x, Y(x))$ is negative and of large magnitude, recalling that $f_y(x, y)$ plays the role of the $\lambda$ of the model equation. For systems, we consider the eigenvalues $\lambda_j \equiv \lambda_j(x)$ of $f_y(x, Y(x))$, and we assume they all satisfy

$$\text{real}(\lambda_j) \leq 0$$

The differential equation problem is called stiff if some or all of these eigenvalues have a real part that is negative and of large magnitude.

There are also problems in which the eigenvalues have $\text{imag}(\lambda_j)$ of large magnitude, and these must usually be treated by other types of methods. Stiff problems often have $\text{real}(\lambda_j)$ of greatly varying magnitude, which adds to the difficulty of their solution.
**Example.** Consider the model equation

\[ y' = \lambda y + g(x), \quad y(x_0) = Y_0 \]

For example, consider the example problem from the text (p. 405):

\[ y' = \lambda y + (1 - \lambda) \cos x - (1 + \lambda) \sin x, \quad y(0) = 1 \]

with true solution \( Y(x) = \sin x + \cos x \). Now consider the perturbed problem

\[ y' = \lambda y + (1 - \lambda) \cos x - (1 + \lambda) \sin x, \quad y(0) = 1 + \epsilon \]

with true solution

\[ Y_\epsilon(x) = Y(x) + \epsilon e^{\lambda x} \]
\[ Y_{\epsilon}(x) = Y(x) + \epsilon e^{\lambda x} \]

If we have \( \lambda < 0 \) of large magnitude, then \( Y_{\epsilon}(x) \) is essentially the same as \( Y(x) \) after a very small change in \( x \). For example, consider \( \lambda = -10,000 \). This seems a desirable property from a mathematical and physical perspective; but it proves troublesome for the behaviour of numerical methods. For the Euler method of numerical solution, we would need to have

\[
-2 < h\lambda < 0 \\
0 < h < .0002
\]
SOLVING THE BACKWARD EULER METHOD

Recall the backward Euler method for solving

\[ y' = f(x, y) \]

is given by

\[ y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}), \quad n \geq 0 \quad (1) \]

How do we solve for \( y_{n+1} \)? Consider using ordinary fixed point iteration,

\[ y_{n+1}^{(k+1)} = y_n + hf(x_{n+1}, y_{n+1}^{(k)}), \quad k = 0, 1, \ldots \quad (2) \]

To analyze the convergence,

\[
y_{n+1} - y_{n+1}^{(k+1)} = h \left[ f(x_{n+1}, y_{n+1}) - f(x_{n+1}, y_{n+1}^{(k)}) \right]
\]

\[
= h \frac{\partial f(x_{n+1}, y_{n+1})}{\partial y} \left[ y_{n+1} - y_{n+1}^{(k)} \right]
\]
If the problem is stiff, then $f_y(x_{n+1}, y_{n+1})$ is likely to be negative and of very large magnitude. Therefore, to have convergence in (2) will require a very small value of $h$. That would negate the value of using an A-stable method.

For stiff differential equations, the nonlinear equation (1) will need to be solved by other techniques. For a single equation, we might use Newton’s method or the secant method, say with an initial guess of $y_{n+1}^{(0)} = y_n$ or something better.

With a system of $m$ differential equations,

$$y' = f_y(x, y), \quad y(x_0) = Y_0$$

this becomes more of a problem. Now we want to solve

$$y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}), \quad n \geq 0$$

for the vector $y_{n+1}$. When $m$ becomes large, solving this at every step is a major cost and must be done very carefully; and much time is devoted to deciding how to do this. Newton’s method is described in the text, on pages 413-414.
Recall the tools on interpolation we used in deriving the Adams families of multistep methods. Let $P_p(x)$ interpolate $Y(x)$ at the node points $x_{n+1}, x_n, \ldots, x_{n-p+1}$

These are exactly the node points used in defining the Adams-Moulton method of order $p+1$. We can write this polynomial in its Lagrange form:

$$P_p(x) = \sum_{j=-1}^{p-1} Y(x_{n-j}) \ell_j(x)$$

$$\ell_j(x) = \prod_{\substack{i=-1 \atop i \neq j}}^{p-1} \left( \frac{x-x_{n-i}}{x_{n-j}-x_{n-i}} \right)$$
With this definition, \( \text{deg}(\ell_j) = p \) and
\[
\ell_j(x_{n-i}) \equiv \delta_{i,j} = \begin{cases} 
1, & i = j \\
0, & i \neq j 
\end{cases}
\]

We have
\[
Y(x) \approx P_p(x)
\]

For example, with \( p = 1 \):
\[
P_1(x) = \left( \frac{x - x_{n+1}}{x_n - x_{n+1}} \right) Y(x_n) + \left( \frac{x - x_n}{x_{n+1} - x_n} \right) Y(x_{n+1})
\]

Now use
\[
P_p'(x_{n+1}) \approx Y'(x_{n+1}) = f(x_{n+1}, Y_{n+1})
\]

Continuing the example with \( p = 1 \),
\[
\frac{Y(x_{n+1}) - Y(x_n)}{x_{n+1} - x_n} \approx f(x_{n+1}, Y_{n+1})
\]

Solving for \( Y(x_{n+1}) \), we have
\[
Y(x_{n+1}) \approx Y(x_n) + hf(x_{n+1}Y_{n+1})
\]

This is just the backward Euler method.
With $p = 2$, we write

$$
P_2(x) = \left( \frac{x - x_n}{x_{n+1} - x_n} \right) \left( \frac{x - x_{n-1}}{x_{n+1} - x_{n-1}} \right) Y_{n+1} + \left( \frac{x - x_{n+1}}{x_n - x_{n+1}} \right) \left( \frac{x - x_{n-1}}{x_n - x_{n-1}} \right) Y_n + \left( \frac{x - x_{n+1}}{x_{n-1} - x_{n+1}} \right) \left( \frac{x - x_n}{x_{n-1} - x_n} \right) Y_{n-1} = \frac{(x - x_n)(x - x_{n-1})}{2h^2} Y_{n+1} - \frac{(x - x_{n+1})(x - x_{n-1})}{h^2} Y_n + \frac{(x - x_{n+1})(x - x_n)}{2h^2} Y_{n-1}$$

$$
P_2'(x_{n+1}) = \frac{3}{2h} Y_{n+1} - \frac{2}{h} Y_n + \frac{1}{2h} Y_{n-1}$$
This leads to the approximation

\[ \frac{3}{2h} Y_{n+1} - \frac{2}{h} Y_n + \frac{1}{2h} Y_{n-1} \approx Y'(x_{n+1}) = f(x_{n+1}, Y_{n+1}) \]

Solving for \( Y_{n+1} \), we have

\[ Y_{n+1} \approx \frac{4}{3} Y_n - \frac{1}{3} Y_{n-1} + \frac{2h}{3} f(x_{n+1}, Y_{n+1}) \]

The numerical method is

\[ y_{n+1} = \frac{4}{3} y_n - \frac{1}{3} y_{n-1} + \frac{2h}{3} f(x_{n+1}, y_{n+1}), \quad n \geq 1 \]

This is a two-step method of order 2, with

\[ T_n(Y) = \frac{2}{9} h^3 Y'''(\xi_n) \]

This is also an A-stable method.
For general $p \geq 1$,

\[
Y(x) \approx P_p(x) = \sum_{j=-1}^{p-1} Y(x_{n-j})\ell_j(x)
\]

\[
Y'(x) \approx P'_p(x) = \sum_{j=-1}^{p-1} Y(x_{n-j})\ell'_j(x)
\]

\[
Y'(x_{n+1}) \approx P'_p(x_{n+1}) = \sum_{j=-1}^{p-1} Y(x_{n-j})\ell'_j(x_{n+1})
\]

Using $Y'(x_{n+1}) = f(x_{n+1}, Y_{n+1})$, we have

\[
\sum_{j=-1}^{p-1} Y(x_{n-j})\ell'_j(x_{n+1}) \approx f(x_{n+1}, Y(x_{n+1}))
\]
Solve for the term $Y(x_{n+1})$ on the left side, obtaining something of the form

$$Y_{n+1} \approx \alpha_0 Y_n + \ldots + \alpha_{p-1} Y_{n-p+1} + \beta h f(x_{n+1}, Y_{n+1})$$

Values of these coefficients for $1 \leq p \leq 6$ are given on p. 411. This leads to the multistep method

$$y_{n+1} = \alpha_0 y_n + \ldots + \alpha_{p-1} y_{n-p+1} + \beta h f(x_{n+1}, y_{n+1})$$

For $p \leq 6$, these are useful in solving stiff differential equations.

For all of these cases, the region of absolute stability contains the entire negative real axis, meaning that the interval

$$-\infty < h \lambda < 0$$

is contained in the region of absolute stability. Portions above and below this interval are also contained in the region of absolute stability.
Consider solving for a function $U(x,t)$ which satisfies the equations

$$U_t = c^2 U_{xx} + G(x,t), \quad 0 < x < 1, \quad t > 0 \quad (3)$$

$$U(0,t) = d_0(t)$$
$$U(1,t) = d_1(t) \quad t \geq 0 \quad (4)$$

$$U(x,0) = f(x), \quad 0 \leq x \leq 1 \quad (5)$$

The equation (3) is an example of a parabolic partial differential equation (a parabolic) or an equation of diffusion type; and it is also called the heat equation. The equations (4) give the boundary values of $U(x,t)$ at the boundaries of the region $[0,1]$ on which the function $U$ is being sought, and the final equation (5) gives the initial value of $U$ at time $t = 0$. 
A PHYSICAL EXAMPLE

As a physical example for which this is the mathematical model, imagine a metal rod of length 1; and assume it is well insulated along its length so that the heat that escapes does so only at its ends (at $x = 0$ and $x = 1$). The function $U(x, t)$ represents the temperature of the rod at position $x$ at time $t$. The equation (3) gives the governing law for the movement of heat in the rod; and $G(x, t)$ is a source term. The initial condition (5) gives the initial temperature of the rod; and (4) gives the forced temperatures at the ends of the rod.

The constant $c > 0$ depends on the physical characteristics of the rod. For simplicity, we assume $c = 1$. 
THE METHOD OF LINES

Introduce a mesh on $0 \leq x \leq 1$. For an integer $m > 0$, define $\delta = 1/m$, and

$$x_j = j\delta, \quad j = 0, 1, ..., m$$

We give a method which solves for approximations to $U(x, t)$ at the node points $x_1, ..., x_{m-1}$. If you look at the domain of the function $U(x, t)$, namely

$$\{(x, t) \mid 0 \leq x \leq 1, \ t \geq 0\}$$

then we are solving for estimates of $U(x, t)$ along the lines

$$\{(x_j, t) \mid t \geq 0\}, \quad j = 1, 2, ..., m - 1$$

We approximate the PDE at the points on these lines.

We begin by approximating the term $U_{xx}(x_j, t)$. To do so, we return to a numerical differentiation formula
from Chapter 5. For a function $g(x)$,
\[
g''(x) = \frac{g(x + \delta) - 2g(x) + g(x - \delta)}{\delta^2} - \frac{\delta^2}{12} g^{(4)}(\xi)
\]
with some $x - \delta \leq \xi \leq x + \delta$ (cf. p. 318). Then
\[
U_{xx}(x_j, t) = \frac{U(x_{j+1}, t) - 2U(x_j, t) + U(x_{j-1}, t)}{\delta^2} - \frac{\delta^2}{12} \frac{\partial^4 U(\xi_j, t)}{\partial x^4}
\]
with $x_{j-1} \leq \xi_j \leq x_{j+1}$, for $j = 1, 2, \ldots, m - 1$. We will substitute this into our PDE (3), at the point $(x_j, t)$. This yields
\[
U_t(x_j, t) = \frac{U(x_{j+1}, t) - 2U(x_j, t) + U(x_{j-1}, t)}{\delta^2} - \frac{\delta^2}{12} \frac{\partial^4 U(\xi_j, t)}{\partial x^4} + G(x_j, t)
\]
(6)
We drop the truncation error to obtain our numerical method.
Introduce the functions $u_j(t)$ as the approximation we will compute for $U(x_j, t)$, for $j = 0, \ldots, m$. In fact, we take

$$u_0(t) = d_0(t), \quad u_m(t) = d_1(t)$$ (7)

Then our numerical approximation of (6) is given by

$$u_j'(t) = \frac{u_{j+1}(t) - 2u_j(t) + u_{j-1}(t)}{\delta^2} + G(x_j, t)$$ (8)

for $j = 1, \ldots, m - 1$. In addition, the initial condition (5) implies we should use

$$u_j(0) = f(x_j), \quad j = 1, \ldots, m - 1$$ (9)

The equations (7)-(9) form an initial value problem for a linear system of $m - 1$ ordinary differential equations for the unknown functions $u_1, \ldots, u_{m-1}$.

Under suitable assumptions on $u, G, d_0, d_1, f$, it can be proven that

$$\max_{0 \leq x_j \leq 1} \max_{0 \leq t \leq T} \left| U(x_j, t) - u_j(t) \right| \leq c_T \delta^2$$ (10)
Introduce

\[ \Lambda = \frac{1}{\delta^2} \begin{bmatrix} -2 & 1 & 0 & \cdots \\ 1 & -2 & 1 & 0 & \cdots \\ 0 & 1 & -2 & 1 \\ \vdots & & \ddots & \ddots \\ 0 & \cdots & 0 & 1 & -2 \end{bmatrix} \]

\[ u(t) = [u_1(t), \ldots, u_{m-1}(t)]^T \]

\[ u_0 = [f(x_1), \ldots, f(x_{m-1})]^T \]

\[ g(t) = [G(x_1, t), \ldots, G(x_{m-1}, t)]^T + \frac{1}{\delta^2} [d_0(t), 0, \ldots, 0, d_1(t)]^T \]

Then our numerical method (7)-(9) can be written as the initial value problem

\[ u'(t) = \Lambda u(t) + g(t), \quad u(0) = u_0 \quad (11) \]

How do we solve this problem?
Euler’s method (with stepsize $h$ in the time variable $t$):

$$V_{n+1} = V_n + h [\Lambda V_n + g(t_n)], \quad n \geq 0$$

with $V_0 = u_0$. We have introduced $V_n \approx u(t_n)$.

Backward Euler’s method:

$$V_{n+1} = V_n + h [\Lambda V_{n+1} + g(t_{n+1})], \quad n \geq 0$$

with $V_0 = u_0$.

Trapezoidal method:

$$V_{n+1} = V_n + \frac{h}{2} [\Lambda V_n + g(t_n)
+ \Lambda V_{n+1} + g(t_{n+1})]$$
Before proceeding with these numerical methods, first examine the system

\[ u'(t) = \Lambda u(t) + g(t), \quad u(0) = u_0 \]

In this case, \( f(t, u) = \Lambda u + g(t) \); and the Jacobian matrix is

\[ f_u(t, u) = \Lambda \]

Thus we must examine the eigenvalues of \( \Lambda \). This is in fact a well-known matrix, and its eigenvalues are

\[ \lambda_j = -\frac{4}{\delta^2} \sin^2 \left( \frac{j\pi}{2m} \right), \quad j = 1, \ldots, m - 1 \]

Thus

\[ \lambda_{m-1} \leq \lambda_j \leq \lambda_1 \]

\[ \lambda_{m-1} \approx -\frac{4}{\delta^2}, \quad \lambda_1 \approx -\pi^2 \quad (12) \]

We see that for \( \delta \) small, the eigenvalues of \( \Lambda \) can be very large in size, while being real and negative. This is a stiff system. For example, take \( m = 100 \), and thus \( \delta = 0.01 \).
EULER’S METHOD

Euler’s method is

\[ V_{n+1} = V_n + h [\Lambda V_n + g(t_n)], \quad n \geq 0 \]  (13)

For stability, it requires

\[ -2 < h \lambda_j < 0 \]

for all eigenvalues of \( \Lambda \). Using the bounds on \( \lambda_j \), this requires

\[ \frac{4h}{\delta^2} < 2 \]

\[ h < \frac{1}{2} \delta^2 \]  (14)

This is a well-known condition for stability of (13). In the case \( m = 100 \), this requires the time step \( h \) to satisfy

\[ h < .00005 \]

which is a severe restriction.
THE BACKWARD EULER’S METHOD

The method is

\[ V_{n+1} = V_n + h [ \Lambda V_{n+1} + g(t_{n+1}) ] , \quad n \geq 0 \quad (15) \]

With both this method and Euler’s method, it can be shown that

\[ \max_{0 \leq x_j \leq 1, 0 \leq t \leq T} | U(x_j, t) - V_j,n | \leq c_T \delta^2 + c_2 h \]

But unlike Euler’s method, there is no longer any step-size restriction on \( h \).
To solve (15) for $V_{n+1}$, we rewrite it as

$$(I - h\Lambda) V_{n+1} = V_n + h g(t_{n+1})$$

(16)

The matrix $I - h\Lambda$ is of tridiagonal form; and linear systems with such a form are quite easy to solve with a very low order of arithmetic operations. In this particular case, the linear system can be solved with around $5m$ arithmetic operations for each value of $n$. For the heat equation, the backward Euler method is always preferable to the Euler method.
NUMERICAL EXAMPLE

We choose the true solution to be

\[ U(x, t) = e^{-1.1t} \sin(\pi x), \quad 0 < x < 1, \quad t > 0 \]

The functions \(G(x, t), d_0(t), d_1(t), f(x)\) are determined accordingly.

For the Euler method, we choose \(m = 4, 8, 16;\) and we choose \(h = \frac{1}{2} \delta^2.\) This means using

\[ h = .031, .0078, .0020 \]

For the backward Euler method, we again use \(m = 4, 8, 16;\) but now we use simply \(h = 0.1.\)
THE TRAPEZOIDAL METHOD

The trapezoidal method is given by

\[ V_{n+1} = V_n + \frac{h}{2} \left[ \Lambda V_n + g(t_n) + \Lambda V_{n+1} + g(t_{n+1}) \right] \]  \hspace{1cm} (17)

It can be shown that

\[ \max_{0 \leq t \leq T} \left| U(x_j, t) - V_{j,n} \right| \leq c_T \delta^2 + c_2 h^2 \]

To solve the equation (17) for \( V_{n+1} \), we have

\[ \left( I - \frac{1}{2} h \Lambda \right) V_{n+1} = \left( I + \frac{1}{2} h \Lambda \right) V_n + \frac{h}{2} [g(t_n) + g(t_{n+1})] \]

The matrix \( I - \frac{1}{2} h \Lambda \) is again tridiagonal, and we can solve this system quite inexpensively. This is known as the Crank-Nicolson method when used to solve parabolic PDEs.