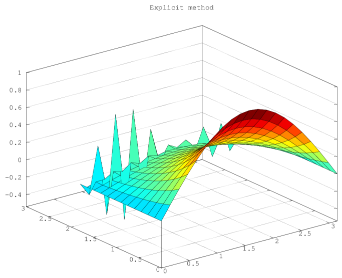
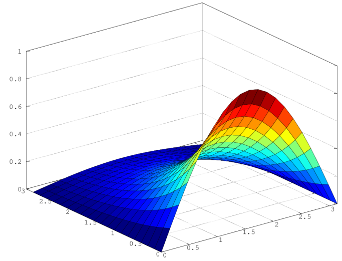




MA385 Part 2: Initial Value Problems

2.6: From IVPs to Linear Systems

Dr Niall Madden

October 2025

1. If we had more time...

In this final section, we highlight some of the many important aspects of the numerical solution of IVPs that are **not** covered in detail in this course:

- Systems of ODEs;
- Higher-order equations;
- Implicit methods; and
- Problems in two dimensions.

We have the additional goal of seeing how these methods related to the earlier section of the course (nonlinear problems) and next section (linear equation solving).

2. Systems of ODEs

So far we have solved only single IVPs. However, many interesting problems are coupled systems: find functions y and z such that

$$y'(t) = f_1(t, y, z),$$

$$z'(t) = f_2(t, y, z).$$

This does not present much of a problem to us. For example the Euler Method is extended to

$$y_{i+1} = y_i + hf_1(t, y_i, z_i),$$

$$z_{i+1} = z_i + hf_2(t, y_i, z_i).$$

2. Systems of ODEs

Example 2.6.1

In pharmokinetics, the flow of drugs between the blood and major organs can be modelled

$$\frac{dy}{dt}(t) = k_{21}z(t) - (k_{12} + k_{\mathrm{elim}})y(t).$$
$$\frac{dz}{dt}(t) = k_{12}y(t) - k_{21}z(t).$$
$$y(0) = d, \quad z(0) = 0.$$

where y is the concentration of a given drug in the blood-stream and z is its concentration in another organ. The parameters k_{21}, k_{12} and k_{elim} are determined from physical experiments.

2. Systems of ODEs

Example 2.6.2

$$\frac{dy}{dt}(t) = k_{21}z(t) - (k_{12} + k_{\mathrm{elim}})y(t).$$
$$\frac{dz}{dt}(t) = k_{12}y(t) - k_{21}z(t).$$
$$y(0) = d, \quad z(0) = 0.$$

Euler's method for this is:

$$y_{i+1} = y_i + h\big(-(k_{12} + k_{\mathrm{elim}})y_i + k_{21}z_i\big),$$
$$z_{i+1} = z_i + h\big(k_{12}y_i + k_{21}z_i\big).$$

3. Higher-Order problems

So far we've only considered **first-order** initial value problems. However, the methods can easily be extended to high-order problems:

$$y''(t) + a(t)y'(t) = f(t, y); \quad y(t_0) = y_0, y'(t_0) = y_1.$$

We do this by converting the problem to a system: set $z(t) = y'(t)$. Then:

$$\begin{aligned} z'(t) &= -a(t)z(t) + f(t, y), & z(t_0) &= y_1, \\ y'(t) &= z(t), & y(t_0) &= y_0. \end{aligned}$$

Now apply any one-step method to this system.

3. Higher-Order problems

Example 2.6.3

Transform the following 2nd-order IVP as a system of 1st order problems, and write down the Euler method for the resulting problem:

$$y''(t) - 3y'(t) + 2y(t) + e^t = 0,$$

$$y(1) = e,\ y'(1) = 2e.$$

3. Higher-Order problems

Example

Let $z = y'$, then

$$\begin{aligned} z'(t) &= 3z(t) - 2y(t) + e^t, \quad z(0) = 2e \\ y'(t) &= z(t), \qquad y(0) = e. \end{aligned}$$

Euler's Method is

$$\begin{aligned} z_{i+1} &= z_i + h\big(3z_i - 2y_i + e^{t_i}\big), \\ y_{i+1} &= y_i + hz_i. \end{aligned}$$

4. Implicit methods

Although we won't dwell on the point, there are many problems for which the one-step methods we have seen will give a useful solution only when the step size, h, is small enough. For larger h, the solution can be very unstable.
Such problems are called "**stiff**" problems. They can be solved, but are best done with so-called "implicit methods", the simplest of which is the Implicit Euler Method:

$$y_{i+1} = y_i + hf(t_{i+1}, y_{i+1}).$$

Note that y_{i+1} appears on both sizes of the equation. To implement this method, we need to be able to solve this non-linear problem. The most common method for doing this is Newton's method.

4. Implicit methods

As recently as 2010, high-order implicit methods were considered to be only of theoretical interest: they were to challenging to code, and the non-linear solve could be slow to converge.

However, that has changed.
Many libraries are not available, which leverage recent advances in programming abstraction and last solvers. These include

- `RungeKutta.jl` for Julia;
- `solve_ivp` in scipy/Python.
- My favourite: **Irksome** for Firedrake/Python. This provides time-stepping, for any Butcher Tableau, when solving a PDE by finite element methods (FEMs). See `https://www.firedrakeproject.org/Irksome/irksome.html`

For more about FEMs: see MA378.
For more about PDEs: keep listening...

5. Towards PDEs

So far, in MA385, we've only considered *ordinary* differential equations: these are DEs which involve functions of just one variable. In our examples above, this variable was time.

However, many physical phenomena vary in space *and* time, and so the solutions to the differential equations the model them depend on two or more variables. The derivatives expressed in the equations are *partial derivatives* and so they are called *partial differential equations* (PDEs).
We will take a brief look at how to solve these (and how not to solve them). This will motivate the following section, on solving systems of linear equations.

5. Towards PDEs

Recall (again) the Black-Scholes equations for pricing an option:

$$\frac{\partial V}{\partial t} - \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} - rS\frac{\partial V}{\partial S} + rV = 0.$$

With a little effort, (see, e.g., Chapter 5 of "*The Mathematics of Financial Derivatives: a student introduction*", by Wilmott et al.) this can be transformed to the simpler-looking *heat equation*:

$$\frac{\partial u}{\partial t}(t,x) = \frac{\partial^2 u}{\partial x^2}(t,x), \quad \text{for} \quad (x,t) \in [0,L] \times [0,T],$$

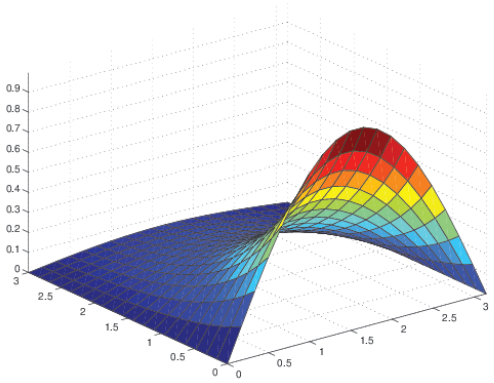
and with the initial and boundary conditions

$$u(0,x) = g(x) \qquad \text{and } u(t,0) = a(t), u(t,L) = b(t).$$

5. Towards PDEs

Example 2.6.4

If $L = \pi$, $g(x) = \sin(x)$, $a(t) = b(t) \equiv 0$ then $u(t,x) = e^{-t}\sin(x)$.



5. Towards PDEs

This problem can't be solved explicitly for arbitrary g, a, b, and so a numerical scheme is used. Suppose we somehow know $\partial^2 u/\partial x^2$, then we could just use Euler's method:

$$u(t_{i+1}, x) = u(t_i, x) + h\frac{\partial^2 u}{\partial x^2}(t_i, x).$$

Although we don't know $\frac{\partial^2 u}{\partial x^2}(t_i, x)$ we can *approximate* it, using a **finite difference method.**

5. Towards PDEs

1. Divide $[0, T]$ into N_t intervals of width h_t, giving the grid $\{0 = t_0 < t_1 < \cdots < t_{N-1} < t_{N_t} = T\}$, with $t_i = t_0 + ih_t$.

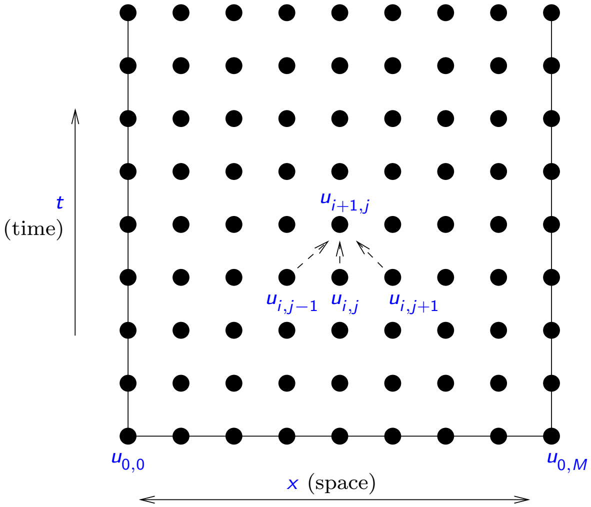
2. Divide $[0, L]$ into N_x intervals of width h_x, giving the grid $\{0 = x_0 < x_1 < \cdots < x_{N_x} = L\}$ with $x_j = x_0 + jh_x$.
3. Denote by $u_{i,j}$ the approximation for $u(t, x)$ at (t_i, x_j).
4. For each $i = 1, 2, \ldots, N_t$, use the approximation:

$$\frac{\partial^2 u}{\partial x^2}(t_i, x_j) \approx \delta_x^2 u_{i,j} = \frac{1}{h_x^2}\big(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}\big),$$

   for $k = 1, 2, \ldots, N_x - 1$.
5. Now set: $u_{i+1,j} := u_{i,j} - h_t \delta_x^2 u_{i,j}$.

5. Towards PDEs

This scheme is called an **explicit method**: if we know $u_{i,j-1}$, $u_{i,j}$ and $u_{i,j+1}$ then we can explicitly calculate $u_{i+1,j}$.



5. Towards PDEs

Unfortunately, this method is not very stable: huge errors occur in the approximation. (Example: see `Heat.py`).
Explaining why this can fail is a little technical...

- For the method to have **stability** we must be able to ensure that the solutions are always non-negative.
- That in turn requires that $h_t \leq Ch_x^2$ for some constant C.
- This is quite an onerous condition to satisfy.

5. Towards PDEs

Instead one might use an **implicit method**:

$$u(t_{i+1}, x) = u(t_i, x) + h\frac{\partial^2 u}{\partial x^2}(t_{i+1}, x).$$

In practice, if we know $u_{i-1,j}$, we compute $u_{i,j-1}$, $u_{i,j}$ and $u_{i,j+1}$ simultaneously:

$$u_{i,j} - h_t\delta_x^2 u_{i,j} = u_{i-1,j}$$

This is actually a set of simultaneous equations:

$$\begin{aligned} u_{i,0} &= a(t_i), \\ \alpha u_{i,j-1} + \beta u_{i,j} + \alpha u_{i,j+1} &= u_{i-1,k}, \quad k = 1, 2, \dots, M-1 \\ u_{i,M} &= b(t_i), \end{aligned}$$

where $\alpha = -\frac{h_t}{h_x^2}$ and $\beta = \frac{2h_t}{h_x^2} + 1$.

5. Towards PDEs

This could be expressed more clearly as the matrix-vector equation:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ \alpha & \beta & \alpha & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & \alpha & \beta & \alpha & \dots & 0 & 0 & 0 & 0 \\ & \vdots & & & \ddots & & & \vdots & \\ 0 & 0 & 0 & 0 & \dots & \alpha & \beta & \alpha & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & \alpha & \beta & \alpha \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_{i,0} \\ u_{i,1} \\ u_{i,2} \\ \vdots \\ u_{i,n-2} \\ u_{i,n-1} \\ u_{i,n} \end{pmatrix} = \begin{pmatrix} a(0) \\ u_{i-1,1} \\ u_{i-1,2} \\ \vdots \\ u_{i-1,n-2} \\ u_{i-1,n-1} \\ b(T) \end{pmatrix}.$$

So "all" we have to do now is solve this system of equations. That is what the next section of the course is about.

6. Exercises

Exercise 2.6.1 (You can safely ignore this question; it won't be on the exam)

Write down the Euler Method for the following 3rd-order IVP

$$y''' - y'' + 2y' + 2y = x^2 - 1,$$
$$y(0) = 1, y'(0) = 0, y''(0) = -1.$$

Exercise 2.6.2 (You can safely ignore this question; it won't be on the exam)

Use a Taylor series to provide a derivation for the formula

$$\frac{\partial^2 u}{\partial x^2}(t_i, x_j) \approx \frac{1}{H^2}\big(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}\big).$$

6. Exercises

Exercise 2.6.3

Suppose that a 3-stage Runge-Kutta method tableaux has the following entries:

$$\alpha_2 = \frac{1}{3},\ \alpha_3 = \frac{1}{9},\ b_1 = 4,\ b_2 = \frac{15}{4},\ \beta_{32} = -\frac{2}{27}.$$

(i) Assuming that the method is *consistent*, determine the value of b_3.

(ii) Consider the initial value problem:

$$y(0) = 1,\ y'(t) = \lambda y(t).$$

Using that the solution is $y(t) = e^{\lambda t}$, write out a Taylor series for $y(t_{i+1})$ about $y(t_i)$ up to terms of order h^4 (use that $h = t_{i+1} - t_i$). Using that your method should agree with the Taylor Series expansion up to terms of order h^3, determine β_{21} and β_{31}.

. . . .

Here are some entries for 3-stage Runge-Kutta method tableaux for Exercise 21.

Method 0: $\alpha_2 = 2/3$, $\alpha_3 = 0$, $b_1 = 1/12$, $b_2 = 3/4$, $\beta_{32} = 3/2$

6. Exercises

Method 1: $\alpha_2 = 1/4$, $\alpha_3 = 1$, $b_1 = -1/6$, $b_2 = 8/9$, $\beta_{32} = 12/5$

Method 2: $\alpha_2 = 1/4$, $\alpha_3 = 1/2$, $b_1 = 2/3$, $b_2 = -4/3$, $\beta_{32} = 2/5$

Method 3: $\alpha_2 = 1/4$, $\alpha_3 = 1/3$, $b_1 = 3/2$, $b_2 = -8$, $\beta_{32} = 4/45$

Method 4: $\alpha_2 = 1$, $\alpha_3 = 1/4$, $b_1 = -1/6$, $b_2 = 5/18$, $\beta_{32} = 3/16$

Method 5: $\alpha_2 = 1$, $\alpha_3 = 1/5$, $b_1 = -1/3$, $b_2 = 7/24$, $\beta_{32} = 4/25$

Method 6: $\alpha_2 = 1$, $\alpha_3 = 1/6$, $b_1 = -1/2$, $b_2 = 3/10$, $\beta_{32} = 5/36$

Method 7: $\alpha_2 = 1/2$, $\alpha_3 = 1/7$, $b_1 = 7/6$, $b_2 = 22/15$, $\beta_{32} = -10/49$

Method 8: $\alpha_2 = 1/2$, $\alpha_3 = 1/8$, $b_1 = 4/3$, $b_2 = 13/9$, $\beta_{32} = -3/16$

Method 9: $\alpha_2 = 1/3$, $\alpha_3 = 1/9$, $b_1 = 4$, $b_2 = 15/4$, $\beta_{32} = -2/27$

6. Exercises

Exercise 2.6.4 (Your own RK3 method $\star$)

Answer the following questions for Method K from the list above, where K is the last digit of your ID number. For example, if your ID number is 01234567, use Method 7.

(a) Assuming that the method is *consistent*, determine the value of b_3.

(b) Consider the initial value problem:

$$y(0) = 1, \; y'(t) = \lambda y(t).$$

Using that the solution is $y(t) = e^{\lambda t}$, write out a Taylor series for $y(t_{i+1})$ about $y(t_i)$ up to terms of order h^4 (use that $h = t_{i+1} - t_i$). Using that your method should agree with the Taylor Series expansion up to terms of order h^3, determine β_{21} and β_{31}.

6. Exercises

Exercise 2.6.5

(Attempt this exercises after completing Lab 3). Write a MATLAB program that implements your method from Exercise 21.
Use this program to check the order of convergence of the method. Have it compute the error for $n = 2$, $n = 4$, ..., $n = 1024$. Then produce a log-log plot of the errors as a function of n.