PHY307F/407F - Computational Physics
Background Material for Expt. 3 - Heat Equation

David Harrison

INTRODUCTION

In the Pendulum Experiment, we studied the Runge-Kutta algorithm for solving ordinary differential equations (ODEs). Here, we study techniques for solving partial differential equations (PDEs).

The general problem of solving PDEs is *huge*; Press et al. in *Numerical Recipes* claim it requires an entire book.[1] However, there are some simple cases.

Separation of Variables: Consider the Schrödinger equation:

$$-\hbar^2/2m \;\frac{\partial^2\Psi(x,t)}{\partial x^2} + V(x)\Psi(x,t) = i\hbar\frac{\partial\Psi(x,t)}{\partial t} \qquad (1)$$

If we assume that the wavefunction can be written as a function of position times a function of time:

$$\Psi(x,t) = \psi(x)\phi(t)$$

then:

$$\frac{1}{\psi(x)}\left[-\hbar^2/2m\frac{d^2\psi(x)}{dx^2} + V(x)\psi(x)\right] = \frac{1}{\phi(t)}i\hbar\frac{d\phi}{dt} = c \qquad (2)$$

where c is a constant. Thus we have turned the problem into two ordinary differential equations.

First order PDEs: If the equations are linear, then a closed solution can easily be found. If the equations are non-linear, a family of parametric solutions can be found. The standard *Mathematica* package `Calculus‘PDSolve1‘` handles this latter case.

Second order PDEs: There are two general classes of these:

1. Poisson, sometimes called "elliptic."

1. § 17.0.

2. Cauchy, sometimes called "initial value."

The Poisson equation is, of course:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \rho(x,t) \tag{3}$$

This type of equation turns out to be very easy to solve using a technique called a *relaxation method*.

The Cauchy class of second order PDEs is subdivided further:

i. The wave equation (a hyperbolic form):

$$\frac{\partial^2 u}{\partial t^2} = v^2 \frac{\partial^2 u}{\partial x^2} \tag{4}$$

ii. The heat equation (a parabolic form):

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(c \frac{\partial u}{\partial x} \right) \tag{5}$$

For both of these, you will discover that stability is a problem: many reasonable looking algorithms just don't work unless care is taken.

In this Experiment, we will concentrate on the heat equation. Usually, u is the temperature. We will assume that we are solving the equation for a one dimensional slab of width L. We will usually assume that c is a constant so the heat equation becomes:

$$\frac{\partial u(x,t)}{\partial t} = c \frac{\partial^2 u(x,t)}{\partial x^2}$$

We will adopt units where $x/L \to x$ and $tc/L^2 \to t$, so the heat equation is now:

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial^2 u(x,t)}{\partial x^2} \tag{6}$$
$$0 \le x \le 1$$

These notes are organised as follows:

I. Explicit algorithm.

II. Implicit algorithm.

III. Crank-Nicholson algorithm.

IV. Schrödinger's equation.

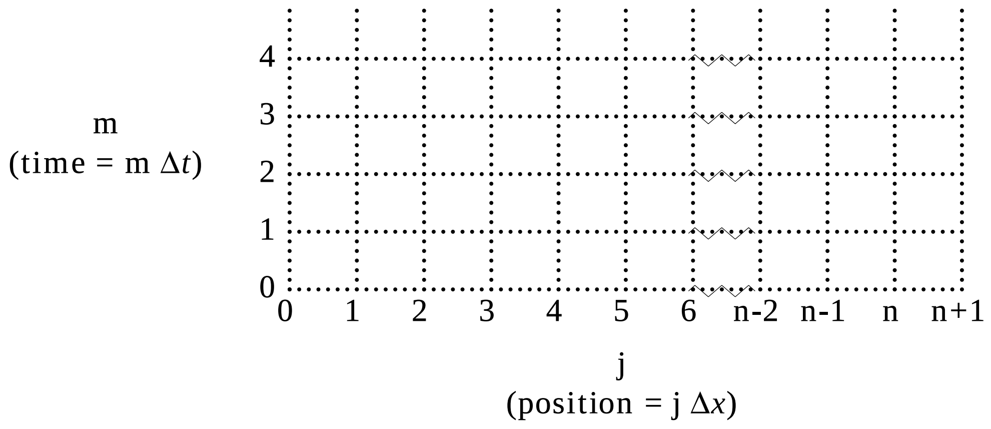
V. References.

Note that the "usual" code listing is not included in this document. That is because you will be modifying some of the code to produce a new procedure, and cutting and pasting the code within the notebook is the simplest way to do this. Thus, the code listing appears in the notebook instead of these supplementary notes for the Experiment.

I. EXPLICIT ALGORITHM

We imagine that we divide our one dimensional slab into n *interior* points, and will call the distance between each point Δx. We will step through timesteps Δt. We will be using the notation that:

$$u_j^m \equiv u(x_j = j\Delta x, t_m = m\Delta t) \qquad \text{(I.1)}$$
$$0 \le j \le n+1$$
$$(n+1)\Delta x = 1$$
$$m = 0, 1, 2, \cdots$$



We will write the initial condition as:

$$u_j^0 = g(x) \qquad \text{(I.2)}$$

and the boundary conditions as:

$$u_0^m = \alpha \qquad \text{(I.3a)}$$
$$u_{n+1}^m = \beta \qquad \text{(I.3b)}$$

It is reasonable to write the left hand side of the heat equation, Equation (6), as:

$$\frac{\partial u}{\partial t} = \frac{u_j^{m+1} - u_j^m}{\Delta t} \tag{I.4}$$

We write the right hand side of Equation (6) as:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_j} = \left(\left.\frac{\partial u}{\partial x}\right|_{x_j + \frac{1}{2}} - \left.\frac{\partial u}{\partial x}\right|_{x_j - \frac{1}{2}} \right) / \Delta x \tag{I.5}$$

Note that we write $\partial u / \partial x$ evaluated at two points that are not on our grid. This doesn't matter because:

$$\left.\frac{\partial u}{\partial x}\right|_{x_j + \frac{1}{2}} = \frac{u_{j+1}^m - u_j^m}{\Delta x}$$

$$\left.\frac{\partial u}{\partial x}\right|_{x_j - \frac{1}{2}} = \frac{u_j^m - u_{j-1}^m}{\Delta x}$$

Thus, Equation (I.5) becomes:

$$\left.\frac{\partial^2 u(x,t)}{\partial x^2}\right|_{x_j} = \frac{1}{\Delta x^2}\left(u_{j+1}^m - 2u_j^m + u_{j-1}^m \right) \tag{I.6}$$

The above way of solving the second-order partial derivative is called the method of *finite differences*.

The heat equation states that Equation (I.4) equals Equation (I.6):

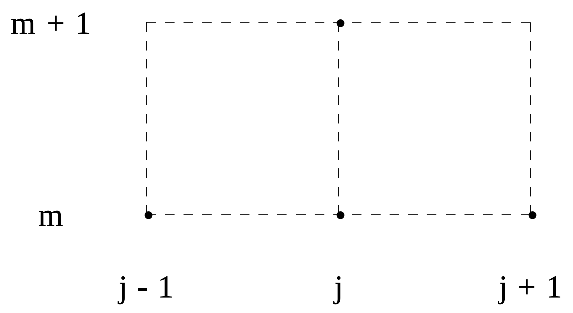
$$\frac{u_j^{m+1} - u_j^m}{\Delta t} = \frac{1}{\Delta x^2}\left(u_{j+1}^m - 2u_j^m + u_{j-1}^m \right) \tag{I.7}$$

so:

$$u_j^{m+1} = u_j^m + \frac{\Delta t}{\Delta x^2}\left(u_{j+1}^m - 2u_j^m + u_{j-1}^m \right) \tag{I.8}$$

Note that the right hand side of Equation (I.8) contains values of u at time $t_m = m\Delta t$, and these terms are combined to give the value of u at the later time $t_{m+1} = (m + 1)\Delta t$. Once the values of u at time $t_{m+1} = (m + 1)\Delta t$ are known we can then find the values at $t_{m+2} = (m + 2)\Delta t$, and so on. Thus we can solve the heat equation for all values of t using this **explicit algorithm**.

The determination of u_j^{m+1} depends on knowing the values of u at three positions at the earlier time:



Note that this scheme does not allow one to determine u_0^{m+1} because we have no values for u_{-1}^m. But that does not matter: u_0^{m+1} is the value of u at the left side of the slab, and is given by the boundary condition Equation (I.3a). Similarly the boundary condition Equation (I.3b) gives the value for u_{n+1}^{m+1}. So Equation (I.8) is used only to evaluate the interior values of u^{m+1}.

The above way of solving the heat equation is pretty simple. Of the three algorithms you will investigate to solve the heat equation, this one is also the fastest and also can give the most accurate result. However, the result will be accurate only if you choose timesteps Δt and a space grid size Δx such that:

$$\frac{\Delta t}{(\Delta x)^2} \equiv \mu \leq \frac{1}{2} \tag{I.9}$$

Solutions produced in violation of Equation (I.9) will be unstable, often producing ridiculous results. Note that this means that for a given timestep Δt the space grid size Δx can not be too small. For a given space grid size, the timestep can not be too big.

We will discuss later in this document the fact that even if the timestep and space grid satisfy Equation (I.9), the explicit algorithm can still produce wrong results for some physical systems.

The reasons for the instability of the explicit algorithm is somewhat beyond the level of this course; consult the references for further information.[2] However, we can justify Equation (I.9).

2. Golub and Ortega, § 8.2.

The heat equation also governs the *diffusion* of, say, a small quantity of perfume in the air. You probably already know that diffusion is a form of *random walk* so after a time t we expect the perfume has diffused a distance $x \propto \sqrt{t}$. One solution to the heat equation gives the density of the gas as a function of position and time:

$$u(x,t) \equiv \rho(x,t) = \frac{e^{-\frac{x^2}{2\sigma^2}}}{\sigma} \qquad \text{(I.10)}$$

where:

$\sigma = \sqrt{2ct}$

and c is the heat constant defined in Equation (5), for which we usually have been choosing units so that it equals 1. In a diffusion context c is often called the *diffusion constant*. Equation (I.10) shows that the density of the diffusing gas is a Gaussian, and that the standard deviation describing the width of the Gaussian increases as the square root of t. This means that in a time Δt, the molecules travel a distance Δx on the order of:

$$\Delta x \approx \sqrt{2c\Delta t} = \sqrt{2\Delta t}$$

Thus, if we are solving the equations using an explicit algorithm, for a given Δt:

$$\Delta x \geq \sqrt{2\Delta t} \qquad \text{(I.11)}$$

since otherwise the distribution is not spreading as fast as we know it should. A trivial rearrangement of Equation (I.11) gives:

$$\frac{\Delta t}{(\Delta x)^2} \leq \frac{1}{2}$$

which is just Equation (I.9).

II. IMPLICIT ALGORITHM

Consider the following equation and compare it to Equation (I.7) from the previous section:

$$\frac{u_m^{m+1} - u_j^m}{\Delta t} = \frac{1}{(\Delta x)^2}(u_{j+1}^{m+1} - 2u_j^{m+1} + u_{j-1}^{m+1}) \qquad \text{(II.1)}$$

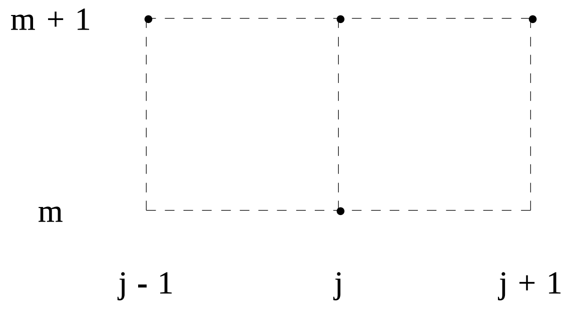
Here we are evaluating $\partial^2 u / \partial x^2$ using finite differences at time t^{m+1} instead of t^m. A moment's reflection should convince you that Equation (II.1) is just as reasonable as Equation (I.7). We rearrange Equation (II.1):

$$(1+2\mu)u_j^{m+1} - \mu(u_{j+1}^{m+1} + u_{j-1}^{m+1}) = u_j^m \tag{II.2}$$

where recall that:

$$\mu = \frac{\Delta t}{(\Delta x)^2}$$

Thus we are trying to find u at times $m+1$ at three different positions $j\text{-}1$, j and $j+1$ from a single value of u at the earlier time m.



At first glance, this is impossible. However, consider an $(n+2)\times(n+2)$ identity matrix Id:

$$Id = \begin{pmatrix} 1 & 0 & 0 & . & 0 \\ 0 & 1 & 0 & . & 0 \\ 0 & 0 & 1 & . & 0 \\ . & . & . & . & . \\ 0 & 0 & 0 & . & 1 \end{pmatrix}$$

an $(n+2)\times(n+2)$ *(2,-1) triangular matrix* A:

$$A = \begin{pmatrix} 2 & -1 & 0 & . & 0 & 0 \\ -1 & 2 & -1 & . & 0 & 0 \\ 0 & -1 & 2 & . & 0 & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & . & 2 & -1 \\ 0 & 0 & 0 & . & -1 & 2 \end{pmatrix}$$

and an $n+2$ dimensional column matrix $\vec{b}$:

$$\vec{b} = \begin{pmatrix} \mu\alpha \\ 0 \\ 0 \\ . \\ \mu\beta \end{pmatrix}$$

Then the entire set of equations for all values of j given by Equation (II.2) can be written as:

$$(Id + \mu A)\vec{u}^{m+1} = \vec{u}^m + \vec{b} \qquad \text{(II.3)}$$

where:

$$\vec{u} = \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ . \\ u_n \\ u_{n+1} \end{pmatrix}$$

If we know $\vec{u}^m$, the values of u at time m, Equation (II.3) may be solved for $\vec{u}^{m+1}$ using, for example, the LU decomposition that you studied in the Exercise.

The above technique for solving the heat equation is called an **implicit algorithm**. You will discover in the Experiment that this algorithm is stable for all values of μ. That is the good news. There is also some bad news: the implicit technique is much slower than the explicit one and is also much less accurate for a given value of timestep Δt and space grid size Δx.

III. CRANK-NICHOLSON ALGORITHM

The explicit algorithm begins with:

$$\frac{u_m^{m+1} - u_j^m}{\Delta t} = \frac{1}{(\Delta x)^2}(u_{j+1}^m - 2u_j^m + u_{j-1}^m)$$

while the implicit one begins with:

$$\frac{u_m^{m+1} - u_j^m}{\Delta t} = \frac{1}{(\Delta x)^2}(u_{j+1}^{m+1} - 2u_j^{m+1} + u_{j-1}^{m+1})$$

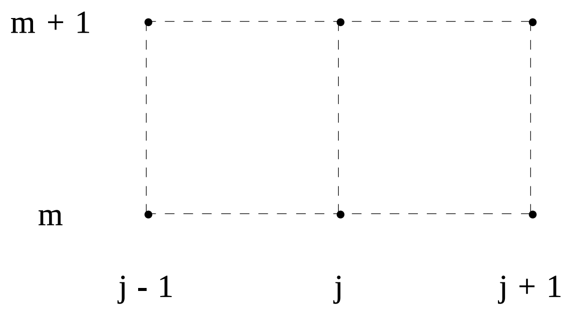
Average these two equations:

$$\frac{u_m^{m+1} - u_j^m}{\Delta t} = \frac{1}{2}\frac{1}{(\Delta x)^2}(u_{j+1}^m - 2u_j^m + u_{j-1}^m + u_{j+1}^{m+1} - 2u_j^{m+1} + u_{j-1}^{m+1})) \qquad \text{(III.1)}$$

In terms of the matrices defined in the previous section, we can write Equation (III.1) as:

$$(Id + \frac{\mu}{2}A)\vec{u}^{m+1} = (Id - \frac{\mu}{2}A)\vec{u}^m + \vec{b} \qquad \text{(III.2)}$$

This equation is, of course, solvable using LU decomposition. We are finding the values of u_{j-1}^{m+1}, u_j^{m+1} and u_{j+1}^{m+1} from the values of u_{j-1}^m, u_j^m and u_{j+1}^m.



The above is the **Crank-Nicholson algorithm**. You will discover that it maintains the absolute stability of the implicit algorithm while recovering some of the accuracy of the explicit one.

IV. SCHRÖDINGER'S EQUATION

Assuming $\hbar = 1$ and $m = 1/2$, Schrödinger's equation is:

$$i\frac{\partial\Psi(x,t)}{\partial t} = H\Psi(x,t) \qquad \text{(IV.1)}$$

$$H = -\frac{\partial^2}{\partial x^2} + V(x)$$

In the late 1960's, Goldberg, Schey and Schwartz had a good idea for a lecture demonstration in quantum mechanics.[3] They decided to solve this equation using then-new computer technology to produce a movie. The initial conditions for Equation (IV.1) are

3. Goldberg, Schey and Schwartz, Amer. Jour. Phys. **35**, (1967) 177.

$\Psi(x, 0) = g(x)$

and the boundary conditions are:

$\Psi(\infty, t) = \Psi(-\infty, t) = 0$

The solution to Equation (IV.1) can be written as:

$$\Psi(x,t) = e^{-iHt}\Psi(x, 0) \tag{IV.2}$$

Expand Equation (IV.2) and throw out higher order terms:

$$\Psi(x,t + \Delta t) \approx (1 - iHt)\Psi(x,t) \tag{IV.3}$$

Thus if we know $\Psi(x,t)$ we can produce a time series of values for Ψ at times $t + \Delta t$, $t + 2\Delta t, \cdots$. Using finite differences to evaluate the $\partial^2/\partial x^2$ term in the Hamiltonian H, the right hand side of Equation (IV.3) will give a term involving $(\Psi_{j-1}^m - 2\Psi_j^m + \Psi_{j+1}^m)$. This should be recognisable to you as an *explicit* algorithm.

We can multiply Equation (IV.2) by e^{iHt} and expand to get:

$$(1 + iHT)\Psi(x,t + \Delta t) \approx \Psi(x,t) \tag{IV.4}$$

Now evaluating the Hamiltonian using finite differences will give a term involving $(\Psi_{j-1}^{m+1} - 2\Psi_j^{m+1} + \Psi_{j+1}^{m+1})$ on the left hand side. This is, of course, just an *implicit* algorithm.

Everything we have said about explicit and implicit algorithms applies to these cases too. However, the *physics* of Quantum Mechanics puts an additional constraint on Ψ:

$$\int_{x=-\infty}^{\infty} \Psi^*(x,t)\Psi(x,t)dx = 1 \tag{IV.5}$$

for all times t. This is called *unitarity* and physically means that the probability that the object being described by the wave function is somewhere between $x = -\infty$ and $x = \infty$ is one.

It has been known since long before the invention of the computer that Equations (IV.3) and (IV.4) violate unitarity. This means that neither the explicit or implicit algorithm can be used to solve Schrödinger's equation because they both violate a physical principle.

A long-known equation that does not violate unitarity is called *Cayley's form*. Multiply Equation (IV.2) by $e^{iHt/2}$ and expand both sides:

$$(1 + \frac{i}{2} H t) \Psi(x, t + \Delta t) = (1 - \frac{i}{2} H t) \Psi(x, t) \qquad \text{(IV.6)}$$

Using finite differences to evaluate the $\partial^2 / \partial x^2$ terms in the Hamiltonian on both sides of the equation will give us a *Crank-Nicholson algorithm*.

The lesson to be learned here is that just knowing the numerical methods is sometimes not sufficient. Just as for the pendulum, where the physics pointed us to the symplectic algorithm, here the physics points us to a Crank-Nicholson algorithm when solving Schrödinger's equation.

V. REFERENCES

- Gene H. Golub and James M. Ortega, **Scientific Computing and Differential Equations** (Academic Press, 1992), Chapter 7.
- William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, **Numerical Recipes: The Art of Scientific Computing or Numerical Recipes in C: The Art of Scientific Computing** (Cambridge Univ. Press), Chapter 17.

Examples of solving Poisson's equation using relaxation methods, and a visualisation of Schrödinger's equation solved using a Crank-Nicholson algorithm may be found at:

`http://emerald.math.buffalo.edu/˜ringland/instruction/pde/`