Ma3469 Practical Numerical Simulation

Notes by Chris Blair

Some notes on the theoretical part of Ma3469, as taught in 2009-10.

1 Ordinary differential equations

Some notation: we will develop our techniques for scalar equations in one dependent variable y; obviously everything we do is valid for vectors $\vec{y}$ too. We denote error terms using the definition that $z = O(h^n)$ means that $z \to ch^n$ for $h \to 0$ with $c \to c_0 \neq 0$. The length of the timestep used in computational methods will usually be denoted by h. The step error is the error at each step, while the total error is the step error multiplied by the number of steps. The latter equals the total runtime divided by h, so that if the step error is $O(h^n)$ the total error will be $O(h^{n-1})$.

1.1 Euler method

For an ordinary differential equation of the form

$$y'(t) = f(y,t)$$

We Taylor expand:

$$y(t+h) = y(t) + hy'(t) + O(h^2)$$

This gives the **Euler method**:

$$y(t+h) := y(t) + hf(y(t),t) \qquad \textbf{(Euler)}$$

Step error is $O(h^2)$ and total error is $O(h)$.

1.2 Runge-Kutta 2nd Order

For an ordinary differential equation of the form

$$y'(t) = f(y,t)$$

We Taylor expand:

$$\begin{aligned} y(t+h) &= y(t) + hy'(t) + \frac{1}{2}h^2 y''(t) + O(h^3) \\ &= y(t) + \frac{1}{2}h\left(y'(t) + y'(t) + hy''(t)\right) + O(h^3) \end{aligned}$$

We use $y'(t+h) = y'(t) + hy''(t) + O(h^2)$ to obtain the **Runge-Kutta 2nd order method**:

$$y(t+h) := y(t) + \frac{1}{2}h\Big(f(y(t),t) + f(y(t+h),t+h)\Big) \qquad \textbf{(RK2)}$$

Step error is $O(h^3)$ and total error is $O(h^2)$.

1.3 Leapfrog method and Euler-Cromer

For an ordinary differential equation of the form

$$y''(t) = f(y,t)$$

which can be written as a system of two coupled first-order equations:

$$\begin{aligned} y'(t) &= v \\ v'(t) &= f(y,t) \end{aligned}$$

We Taylor expand:

$$\begin{aligned} y(t+h) &= y(t) + hy'(t) + \frac{1}{2}h^2 y''(t) + O(h^3) \\ &= y(t) + h\left(y'(t) + \frac{1}{2}hy''(t)\right) + O(h^3) \end{aligned}$$

We use $y'(t+h/2) = y'(t) + \frac{1}{2}hy''(t) + O(h^2)$ and obtain:

$$y(t+h) = y(t) + hy'(t+h/2) + O(h^3)$$

Doing the same with $v(t+h)$ we obtain the **leapfrog method**: an initial step using the Euler method

$$v(t_0 + h/2) := v(t_0) + \frac{1}{2}hf(y(t_0), t_0) \qquad (\textbf{Leapfrog 1})$$

followed by

$$\begin{aligned} y(t+h) &:= y(t) + hv(t+h/2) \\ v(t+3h/2) &:= v(t+h/2) + hf(y(t+h), t+h) \end{aligned} \qquad (\textbf{Leapfrog 2})$$

Step error is $O(h^3)$ and total error is $O(h^2)$ (error in initial step is a once-off error of $O(h^2)$).

We obtain the **Euler-Cromer method** by defining $\widetilde{v}(t) = v(t+h/2)$:

$$\begin{aligned} y(t+h) &:= y(t) + h\widetilde{v}(t) \\ \widetilde{v}(t+h) &:= \widetilde{v}(t) + hf(y(t+h), t+h) \end{aligned} \qquad (\textbf{Euler-Cromer})$$

1.4 Symmetric methods

Consider a known value of the solution $y(t)$. Let $\overline{y}(t+H,h)$ denote the value of y at time $t+H$ obtained using a stepsize h, and let $\overline{y}(t+H,-h)$ denote the value of y at time $t+H$ from which we would obtain $y(t)$ by iterating our algorithm in reverse (i.e. with step $-h$). A method for solving ordinary differential equations is called **symmetric** if

$$\overline{y}(t+H,h) = \overline{y}(t+H,-h)$$

i.e. just if it is symmetric in time.

The leapfrog method is manifestly symmetric and hence so too is Euler-Cromer.

1.5 Error lemma

The **"error lemma"** for an ordinary differential equation $y'(t) = f(y,t)$, where for $t \in [a,b]$, f has bounded partial derivatives up to and including the $N+2$th such derivative, and we denote by $\overline{y}(t;h)$ the solution obtained via a pth order one-step method, then

$$\overline{y}(t;h) = y(t) + h^p e_p(t) + h^{p+1}e_{p+1}(t) + \cdots + h_N e_N(t) + h^{N+1}E_{N+1}(t;h) \qquad (\textbf{Error lemma})$$

where the functions e_k are independent of h and E_{N+1} is bounded on $[a,b]$.

We define the error by

$$\epsilon(t;h) = \overline{y}(t;h) - y(t)$$

1.6 Two-point Richardson extrapolation

We will demonstrate the usefulness of the error lemma by using it to estimate the error in a single step of our method. We can write

$$\overline{y}(t+h;h) = y(t+h) + h^p e_p(t+h) + O(h^{p+1})$$

$$\overline{y}(t+h;h/2) = y(t+h) + \frac{h^p}{2^p} e_p(t+h) + O(h^{p+1})$$

We can view these as two simultaneous equations in $e_p(t+h)$ and $y(t+h)$. Up to the error terms we can solve for $y(t+h)$ and so obtain an improved estimate for the solution at time $t+h$. Multipying the second equation by 2^p and subtracting the first from it we get

$$y^*(t+h) = y(t+h) + O(h^{p+1}) = \frac{2^p \overline{y}(t+h;h/2) - \overline{y}(t+h;h)}{2^p - 1}$$

We can also obtain an estimate for the error. The error for the smaller stepsize is

$$\epsilon(t+h;h/2) = \overline{y}(t+h;h/2) - y(t+h) = \frac{h^p}{2^p} e_p(t+h) + O(h^{p+1})$$

We also obviously have

$$\overline{y}(t+h;h) - \overline{y}(t+h;h/2) = h^p e_p(t+h) - \frac{h^p}{2^p} e_p(t+h) + O(h^{p+1}) = (2^p - 1)\frac{h^p}{2^p} e_p(t+h)$$

so that

$$\epsilon(t+h;h/2) = \frac{\overline{y}(t+h;h) - \overline{y}(t+h;h/2)}{2^p - 1} + O(h^{p+1})$$

Actually, we can magically improve the order of accuracy by one by noting that if we are interested in the error in the step since the time t, we can suppose that at time t our solution is the exact one. This means that $e_k(t) = 0$, and so $e_k(t+h) = he'_k(t) + O(h^2)$. This changes $O(h^{p+1})$ to $O(h^{p+2})$ in the above.

1.7 Full extrapolation

A full Richardson extrapolation involves calculating y at time $t+H$ for $N+1$ stepsizes $h_i = H/n_i$ (where the n_i are positive integers). We can write

$$\overline{y}(t+H;h_i) = y(t+H) + \sum_{k=p}^{p+N-1} h_i^k e_k(t+H) + h_i^{p+N} E_{p+N}(t+H;h_i)$$

Note that if the method is symmetric $\overline{y}(t+H;h_i) = \overline{y}(t+H;-h_i)$, so all the odd powers of k vanish, and the error lemma becomes

$$\overline{y}(t+H;h_i) = y(t+H) + \sum_{k=p}^{p+N-1} h_i^{2k} e_k(t+H) + h_i^{2(p+N)} E_{p+N}(t+H;h_i)$$

We can then use these N solutions to extrapolate an improved value for y at time $t+H$. The essential idea is to consider the values $\overline{y}(t+H;h_i)$ as giving $N+1$ simultaneous equations for the $N+1$ quantities $y(t+H)$ and $e_p, \ldots, e_{p+N-1}$: we solve these for the improved solution $y^*(t+H) = y(t+H) + O(H^{2(p+N-1)})$.

1.8 Neville algorithm

The Neville algorithm constructs a polynomial of degree at most N from $N+1$ data points (x_i, y_i). We start with the trivial polynomials

$$P_0(x) = y_0 \qquad P_1(x) = y_1 \quad \ldots \quad P_N(x) = y_N$$

and then recursively define

$$P_{j-k,\ldots,j}(x) = \frac{(x_j - x)P_{j-k,\ldots,j-1}(x) + (x - x_{j-k})P_{j-k+1,\ldots,j}(x)}{x_j - x_{j-k}} \tag{\textbf{Neville}}$$

which passes through the points y_{j-k}, y_{j-k+1}, $\ldots$, y_j. It's easy to see this: clearly $P_{j-k,\ldots,j}(x_j) = P_{j-k+1,\ldots,j}(x_j) = y_j$, and $P_{j-k,\ldots,j}(x_{j-k}) = P_{j-k,\ldots,j}(x_{j-k}) = y_{j-k}$. For x_m, $j-k < m < j$, we have $P_{j-k,\ldots,j}(x_m) = (x_j - x_m + x_m - x_{j-k})y_m/(x_j - x_{j-k}) = y_m$.

1.9 Bulirsch-Stoer algorithm

The Bulirsh-Stoer algorithm uses a symmetric method (the modified midpoint method) to compute values $\overline{y}(t+H; h_i)$ for y at time $t+H$, using $N+1$ stepsizes $h_i = H/n_i$. This gives a system of $N+1$ equations:

$$\overline{y}_i \equiv \overline{y}(t+H; h_i) = \overline{y}(t+H) + \sum_{k=1}^{N} h_i^{2k} e_p(t+H) \equiv P_N(h_i^2)$$

which we view as particular values of a polynomial $P_N(h^2)$. Here $\overline{y}(t+H)$ is our extrapolated result for y, and is equal to $P_N(0)$.

To generate the polynomial $P_N(h^2)$ we use the Neville algorithm:

$$P_{j-k,\ldots,j}(h^2) = \frac{(h_j^2 - h^2)P_{j-k,\ldots,j-1}(h^2) + (h^2 - h_{j-k}^2)P_{j-k+1,\ldots,j}(h^2)}{h_j^2 - h_{j-k}^2}$$

and

$$T_{j,k} = P_{j-k,\ldots,j}(0)$$

Note that k indexes which iteration of the Neville algorithm we are at: so initially we have $N+1$ polynomials P_i which combine pairwise to produce N polynomials $P_{i-1,i}$, corresponding to $T_{i,1}$. These then combine pairwise to produce $N-1$ polynomials $P_{i-2,i-1,i}$, corresponding to $T_{i,2}$. Note we have N $T_{i,1}$ and $N-1$ $T_{i,2}$, and wish to calculate $T_{N,N}$. We have

$$T_{j,k} = \frac{T_{j-1,k-1}h_j^2 - T_{j,k-1}h_{j-k}^2}{h_j^2 - h_{j-k}^2}$$

or

$$\begin{aligned}
T_{j,k} &= \frac{T_{j-1,k-1}\frac{1}{n_j^2} - T_{j,k-1}\frac{1}{n_{j-k}^2}}{\frac{1}{n_j^2} - \frac{1}{n_{j-k}^2}} \\
&= \frac{T_{j-1,k-1} - T_{j,k-1}\left(\frac{n_j^2}{n_{j-k}^2} - 1\right) - T_{j,k-1}}{1 - \frac{n_j^2}{n_{j-k}^2}} \\
&= T_{j,k-1} + \frac{T_{j,k-1} - T_{j-1,k-1}}{\frac{n_j^2}{n_{j-k}^2} - 1}
\end{aligned}$$

Starting with $T_{i,0} = \overline{y}_i$ we can then use this recursion relation to calculate $T_{N,N}$.

2 Partial differential equations

We will consider solving the following parabolic partial differential equation for an unknown function $f(x,t)$:

$$\frac{\partial f}{\partial t}(x,t) = \alpha \frac{\partial^2 f}{\partial x^2}(x,t) + g(x,t)f(x,t)$$

For instance, we obtain the Schrödinger equation (in dimensionless units) by setting $\alpha = i/2$, and choosing $g(x,t) = -iU(x,t)$ to represent some potential. One could also add in an inhomogeneous part without adding any great difficulties to the methods we will discuss below.

We will solve the equation on $[x_0, x_1] \times [t_0, t_1]$, subdivided into a grid with meshsizes Δx and Δt, and with Dirichlet boundary conditions: $f(x_0, t) = f(x_1, t) = 0$. Note that more complicated boundary conditions could conceivable be used (and that we also have to specify an initial f at $t = t_0$ which is consist with these boundary conditions).

Generalising to higher spatial dimensions should be straightforward.

2.1 Discretising derivatives

We will need the following forms for time and space derivatives:

$$\left.\frac{\partial f}{\partial t}\right|_{FT} = \frac{f(t+\Delta t) - f(t)}{\Delta t} + O(\Delta t) \qquad \text{forwards time}$$

$$\left.\frac{\partial f}{\partial t}\right|_{BT} = \frac{f(t) - f(t-\Delta t)}{\Delta t} + O(\Delta t) \qquad \text{backwards time}$$

$$\left.\frac{\partial f}{\partial t}\right|_{CT} = \frac{f(t+\Delta t) - f(t-\Delta t)}{2\Delta t} + O(\Delta t^2) \qquad \text{centred time}$$

$$\left.\frac{\partial^2 f}{\partial x^2}\right|_{CS} = \frac{f(x+\Delta x) + f(x-\Delta x) - 2f(x)}{(\Delta x)^2} + O(\Delta x^2) \qquad \text{centered space}$$

The validity of these expressions, and their error terms, can be easily checked by Taylor expanding.

2.2 FTCS

The **forward-time centred-space scheme** is:

$$f(x,t+\Delta t) := f(x,t) + \alpha\frac{\Delta t}{(\Delta x)^2}\Big(f(x+\Delta x,t) + f(x-\Delta x,t) - 2f(x,t)\Big) + \Delta t g(x,t)f(x,t) + O(\Delta t^2) \quad (\textbf{FTCS})$$

Introducing the discretisation $f_j \equiv f(x_j)$ where $x_j = x_0 + j\Delta x$, and letting $\vec{f} = (f_1, \ldots, f_N)$ this becomes a matrix equation

$$\vec{f}(t+\Delta t) := \left(1 + \alpha\frac{\Delta t}{(\Delta x)^2}D_2 + \Delta t G(t)\right)\vec{f}(t)$$

where $G(t) = \mathrm{diag}(g_1(t), \ldots, g_N(t))$ and

$$D_2 = \begin{pmatrix} -2 & 1 & 0 & & 0 \\ 1 & -2 & 1 & & \\ 0 & 1 & -2 & & \\ \vdots & & & \ddots & 1 \\ 0 & & & & -2 \end{pmatrix}$$

2.3 BTCS

The **backward-time centred-space scheme** is:

$$f(t+\Delta t) := f(t)+\alpha\frac{\Delta t}{(\Delta x)^2}\Big(f(x+\Delta x,t+\Delta t)+f(x-\Delta x,t+\Delta t)-2f(x)\Big)+\Delta t g(x,t+\Delta t)f(x,t+\Delta t)+O(\Delta t^2) \tag{\textbf{BTCS}}$$

or

$$\left(1-\alpha\frac{\Delta t}{(\Delta x)^2}D_2-\Delta t G(t+\Delta t)\right)\vec{f}(t+\Delta t) := \vec{f}(t)$$

2.4 CTCS

The **centred-time centred-space scheme** is:

$$f(x,t+\Delta t) := f(x,t-\Delta t)+2\alpha\frac{\Delta t}{(\Delta x)^2}\Big(f(x+\Delta x,t)+f(x-\Delta x,t)-2f(x,t)\Big)+2\Delta t g(x,t)f(x,t)+O(\Delta t^3) \tag{\textbf{CTCS}}$$

or

$$\vec{f}(t+\Delta t) := \vec{f}(t-\Delta t)+2\left(\alpha\frac{\Delta t}{(\Delta x)^2}D_2+G(t)\right)\vec{f}(t)$$

2.5 CN

The **Crank-Nicholson scheme** involves combining one FTCS step at stepsize $\Delta t/2$ with one BTCS step to obtain the following implicit expression for $\vec{f}(t+\Delta t)$:

$$\left(1-\alpha\frac{\Delta t}{2(\Delta x)^2}D_2-\frac{\Delta t}{2}G(t+\Delta t)\right)\vec{f}(t+\Delta t) := \left(1+\alpha\frac{\Delta t}{2(\Delta x)^2}D_2+\frac{\Delta t}{2}G(t)\right)\vec{f}(t) \tag{\textbf{CN}}$$

2.6 Von-Neumann stability analysis

We can estimate stability requirements for our schemes using **Von Neumann stability analysis**. This essentially involves Fourier transforming: we write our solution as $f_j(t) = A(t)e^{-ikx_j}$ and see how the amplitude $A(t)$ evolves. In doing so we will ignore boundary conditions and the function $g(x,t)$.

For the **FTCS scheme**, we have

$$A(t+\Delta t)e^{-ikx_j} = \left(1+\alpha\frac{\Delta t}{(\Delta x)^2}\left(e^{ik\Delta x}+e^{-ik\Delta x}-2\right)\right)A(t)e^{-ikx_j}$$

Now, $e^{ik\Delta x}+e^{-ik\Delta x}-2 = \left(e^{ik\Delta x/2}-e^{-ik\Delta x/2}\right)^2 = -4\sin^2 k\Delta x/2$, hence we find

$$A(t+\Delta t) = \left(1-4\alpha\frac{\Delta t}{(\Delta x)^2}\sin^2 k\Delta x/2\right)A(t)$$

hence we have stability if

$$\left|1-4\alpha\frac{\Delta t}{(\Delta x)^2}\sin^2 k\Delta x/2\right| < 1$$

So for instance if α is real and positive this means we must have

$$4\alpha\frac{\Delta t}{(\Delta x)^2} < 2$$

while if α is real and negative we will always be unstable.

If α is imaginary, we have

$$\left|1 + 16|\alpha|^2 \frac{\Delta t^2}{(\Delta x)^4} \sin^2 k\Delta x/2\right| < 1$$

and hence the scheme is unconditionally unstable.

For the **BTCS scheme** we have

$$A(t + \Delta t)\left(1 + 4\alpha \frac{\Delta t}{(\Delta x)^2} \sin^2 k\Delta x/2\right) = A(t)$$

and so require that

$$\left|\frac{1}{1 + 4\alpha \frac{\Delta t}{(\Delta x)^2} \sin^2 k\Delta x/2}\right| < 1$$

which is unconditionally stable if α is real and positive, and unstable if α is real and negative. If α is imaginary the scheme is unconditionally stable.

For the **Crank-Nicholson scheme** we have

$$\left(1 + 2\alpha \frac{\Delta t}{(\Delta x)^2} \sin^2 k\Delta x/2\right) A(t + \Delta t) = \left(1 - 2\alpha \frac{\Delta t}{(\Delta x)^2} \sin^2 k\Delta x/2\right) A(t)$$

and hence stability if

$$\left|\frac{1 - 2\alpha \frac{\Delta t}{(\Delta x)^2} \sin^2 k\Delta x/2}{1 + 2\alpha \frac{\Delta t}{(\Delta x)^2} \sin^2 k\Delta x/2}\right| < 1$$

This is unconditionally stable for α real and positive and unstable for α real and negative. If α is imaginary the left-hand side is equal to 1 and we cannot draw a definite conclusion about stability.

2.7 Special case: unitarity of methods for the Schrödinger equation

Sometimes the precise details of the equation we are studying impose their own specialised stability conditions. As an example we will consider the Schrödinger equation:

$$i\frac{\partial f}{\partial t} = Hf$$

where $H = -\frac{1}{2}\frac{\partial^2}{\partial x^2} + U(x)$ is hermitian. The norm $f^\dagger f$ is conserved as is easily checked:

$$\frac{\partial}{\partial t}(f^\dagger f) = if^\dagger Hf - if^\dagger Hf = 0$$

Denoting $f \equiv f(t)$ and $f' \equiv f(t + \Delta t)$ for slight convenience, we can write the **FTCS** implementation of this equation as $f' = (1 - i\Delta tH)f$. We observe that in this scheme

$$f'^\dagger f' = f^\dagger(1 + i\Delta tH)(1 - i\Delta H)f = f^\dagger f + f^\dagger \Delta t^2 H^\dagger Hf$$

The latter term is always positive and so the norm always increases, violating unitarity. Note that when discretised H contains the term $\frac{1}{\Delta x^2}D_2$, so that $\Delta t^2 H^2$ can be expected to not be negligible.

For the **BTCS** implementation we have $f' = (1 + i\Delta tH)^{-1}f$. We have

$$f'^\dagger f' = f^\dagger(1 - i\Delta tH)^{-1}(1 - i\Delta H)^{-1}f = f^\dagger(1 + \Delta t^2 H^\dagger H)^{-1}f$$

We now have to note that f can be expanded in terms of (orthonormal) eigenfunctions ψ_k of H: $f = \sum_k c_k\psi_k$ where $H\psi_k = E_k\psi_k$. Now $(1+\Delta t^2H^2)f = \sum_k(1+\Delta t^2E_k^2)c_k\psi_k$ so $(1+\Delta t^2H^2)^{-1}f = \left(\sum_k(1 + \Delta t^2E_k^2)\right)^{-1}c_k\psi_k$ and

$$f^\dagger(1 + \Delta t^2 H^\dagger H)^{-1}f = \sum_k \frac{1}{1 + \Delta t^2 E_k^2} c_k^* c_k$$

and this is less than $\sum_k c_k^* c_k = f^\dagger f$ and so the norm in fact decreases.

For the **Crank-Nicholson** scheme we have $f' = (1 + i\frac{1}{2}\Delta t H)^{-1}(1 - i\frac{1}{2}\Delta t H)f$ and so

$$\begin{aligned}
f'^\dagger f' &= f^\dagger (1 + i\frac{1}{2}\Delta t H)^{-1}(1 - i\frac{1}{2}\Delta H)^{-1}(1 + i\frac{1}{2}\Delta H)^{-1}(1 - i\frac{1}{2}\Delta t H)f \\
&= f^\dagger (1 + i\frac{1}{2}\Delta t H)^{-1}(1 + \frac{1}{4}\Delta t^2 H)^{-1}(1 - i\frac{1}{2}\Delta t H)f \\
&= f^\dagger (1 + \frac{1}{4}\Delta t^2 H)^{-1}(1 + \frac{1}{4}\Delta t^2 H)f = f^\dagger f
\end{aligned}$$

so the Crank-Nicholson scheme preserves the norm.

2.8 Tridiagonal matrix inversion

We will need to solve equations of the form

$$\begin{pmatrix}
\beta_1 & \gamma_1 & 0 & \dots & \dots & 0 \\
\alpha_1 & \beta_2 & \gamma_2 & 0 & & \\
0 & \alpha_2 & \beta_3 & 0 & & \\
 & & & \ddots & & \\
 & & & & \ddots & \gamma_{N-1} \\
 & & & & \alpha_{N-1} & \beta_N
\end{pmatrix}
\begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ \\ \vdots \\ f_N \end{pmatrix}
=
\begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ \\ \vdots \\ g_N \end{pmatrix}$$

This can be achieved using the following algorithm:

1. Define $\beta_1' = \beta_1$, $\gamma_1' = \gamma_1$, $g_1' = g_1$.

2. For $j > 1$, we subtract $(\alpha_{j-1}/\beta_{j-1}')$ times row $(j-1)$ from row j, which amounts to

$$\alpha_{j-1}' = \alpha_{j-1} - \frac{\alpha_{j-1}}{\beta_{j-1}'}\beta_{j-1}' = 0$$

$$\beta_j' = \beta_j - \frac{\alpha_{j-1}}{\beta_{j-1}'}\gamma_{j-1}'$$

$$\gamma_j' = \gamma_j$$

$$g_j' = g_j - \frac{\alpha_{j-1}}{\beta_{j-1}'}g_{j-1}'$$

   The effect of this forward substitution is to make all α_j zero.

3. We then solve using

$$f_N = \frac{g_N'}{\beta_N'}$$

   and for $j < N$ we have

$$\beta_j' f_j + \gamma_j' f_{j+1} = g_j'$$

   so

$$f_j = \frac{g_j' - \gamma_j' f_{j+1}}{\beta_j'}$$

3 Random numbers

3.1 Random numbers

The probability of a uniformly distributed random number ξ^u lying in an interval $d\xi^u$ is just $Pr(\xi^u \in \{d\xi^u\}) = d\xi^u$. Computers can generate uniformly distributed pseudo-random numbers quite easily.

We can use uniformly distributed random numbers to generate random numbers f distributed according to some probability distribution $P(f)$. The idea is to find a transform G such that $\xi^u = G(f)$. This implies $d\xi^u = G'(f)df$, and we suppose that $Pr(f \in \{df\}) = Pr(G(f) \in \{G(f)\}$. Hence we have the equality

$$Pr(\xi^u \in \{d\xi^u\}) = d\xi^u = G'(f)df = Pr(f \in \{df\}) = P(f)df$$

so $P(f) = G'(f)$ and

$$G(f) = \int_{-\infty}^{f} P(f')df'$$

3.2 Box-Muller algorithm

The Box-Muller algorithm is a mechanism for generating pairs of normally distributed random numbers. The normal distribution is

$$P(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

Hence the probability of finding a pair (x, y) in a particular infinitesimal region of area $dxdy$ is

$$Pr(x \in dx, y \in dy) = \frac{1}{2\pi} dxdy e^{-(x^2+y^2)/2}$$

Changing to polar coordinates

$$\frac{1}{2\pi} dxdy e^{-(x^2+y^2)/2} = \frac{1}{2\pi} d\varphi r dr e^{-r^2/2}$$

Letting $t = \frac{1}{2}r^2$ and integrating out over φ (note that $\varphi/2\pi$ is uniformly distributed) we have

$$Pr(t \in dt) = e^{-t}dt$$

From the previous section we see that we can find a transformation $G(t)$ which takes the random numbers t into uniformly distributed random numbers ξ^u:

$$G(t) = \int_0^t e^{-t'}dt' = 1 - e^{-t} = \xi^u$$

Hence given a pair (ξ_1^u, ξ_2^u) of uniformly distributed random numbers in $(0, 1]$ we can obtain a pair of normally distributed random numbers via:

$$\xi_1^n = \sqrt{-2\ln\xi_1^u}\sin 2\pi\xi_2^u \qquad \xi_2^n = \sqrt{-2\ln\xi_1^u}\cos 2\pi\xi_2^u \tag{\textbf{Box-Muller}}$$