Introduction

Mathematical models for financial derivatives

In the simplest setting, consider a contingent claim on a risky asset S, whose value S_T at time $t = T$ is unknow. The holder of this derivative recieves a payoff $g(S_T)$ at expiry T (*European option*).

Generally, valuation of such a contract requires modelling assumptions on the underlying asset.

The class of models we will study in the first instance is one where the option price is the *expected value* of the stock under a process of the form

$$\mathrm{d}S = rS\,\mathrm{d}t + a(S,t)\,\mathrm{d}W, \tag{1.1}$$

where rS is the return on a risk free investment and $\mathrm{d}W$ a Brownian increment. Future cashflows have to be discounted back to today, with the risk-free interest rate, that is

$$u(S,t) = \mathbb{E}\left(\mathrm{e}^{-r(T-t)}g(S_T)|S_t = S\right).$$

The partial derivatives, which are important trading parameters,

$$\frac{\partial u}{\partial S}, \frac{\partial^2 u}{\partial S^2}, \frac{\partial u}{\partial t}$$

are related by the *Feynman-Kac PDE*

$$\frac{\partial u}{\partial t} + \frac{1}{2}a^2(S,t)\frac{\partial^2 u}{\partial S^2} + rS\frac{\partial u}{\partial S} - ru = 0.$$

This PDE is complemented with the terminal condition

$$u(S,T) = g(S),$$

which states that the payoff at $t = T$ is g.

Workflow in computational finance

1. Identify a suitable model for the underlying asset.
2. Find fast and accurate approximations to simple derivative contracts.
3. Invert these relations to imply model parameters from quoted market prices.
4. Develop a numerical scheme suitable for pricing more complex contracts.
5. Test the numerical scheme for the simple products where the solution is known.
6. Stress-test the model and pricing method under different market scenarios.
7. Compute hedge parameters.
8. Use the above setup to compute the (unknown!) prices of complex derivatives.

Types of calculation

Solution methods often fall into one of the following main categories:

- semi-analytic, if distributions are known analytically or semi-analytically, e.g. via their characteristic function;
- estimation of the expected pay-off by simulation;
- numerical approximation of a corresonding PDE or deterministic integral.

If the terminal probability density is known, the (discounted) expectation can be evaluated as an integral

$$u(S,0) = \int_0^\infty \mathrm{e}^{-rT} g(S') p(S,0;S',T)\,\mathrm{d}S', \tag{1.2}$$

where $p(S,0;S',T)$ is the transition probability to go from S at $t=0$ to S' at $t=T$.

In some cases, like the Black-Scholes problem, integration can be carried out analytically, which results in closed-form solutions. If this is not the case, approximation of (1.2) via a *quadrature rule*, e.g. the trapezoidal rule, gives

$$u_N(S,0) = \frac{1}{N}\sum_{k=0}^{N} \alpha_k \mathrm{e}^{-rT} g(s_k) p(S,0;s_k,T)$$

with integration points s_k, $k=0,\ldots,N$, and weights α_k. Note that if the pay-off and density have infinite support this also means truncating the integration interval.

In a *Monte Carlo* method, the expectation is approximated by the average over a (finite) random sample $S^{(i)}$, $i=1,\ldots,N$, drawn independently from the distribution with density $p(S,0;S',T)$,

$$u_N(S,0) = \frac{1}{N}\mathrm{e}^{-rT}\sum_{i=1}^{N} g(S^{(i)}).$$

If the terminal distribution is not known, but given as the solution of a stochastic differential equation (1.1), one can still approximate the paths via a Monte Carlo method, e. g. the *Euler scheme*

$$S_{k+1}^{(i)} - S_k^{(i)} = rS_k^{(i)}\tau + a(S_k^{(i)}, t)\sqrt{\tau}X_k^{(i)}$$

where $X_k^{(i)} \sim N(0,1)$ is a standard normal, τ the timestep.

If we choose to solve the equivalent PDE analytically or numerically, we are faced with a final value problem: the value of the option at expiry is known to be $g(S)$, the PDE is solved backwards from expiry.

The main methods compared

- The solution to the PDE is the value function for all possible values of the underlying. This makes it straightforward to calculate sensitivities with respect to the underlying.
- The downside is that we need to calculate the solution for a large number of values for the underlying. This raises severe complexity issues in higher dimensions.
- Monte Carlo path simulation with the current spot price as starting point yields the option price for a single underlying value. Accurate and stable sensitivity calculation is more involved.
- Monte Carlo is based on forward simulation and allows us to 'look back' on the path. This is useful when the option depends on the whole path of the stock, e. g. *Asian options* on the average of the price over the lifetime of the option.
- In contrast to this, PDE methods are based on backward induction. This is advantageous when the option contract allows early exercise (*American options*), as we can 'look forward' and take the optimal decision over all different scenarios.

From random walks to finite differences and back

This introductory chapter is concerned with random walks which move between a discrete set of points, it studies their transition probabilities and quantities derived from those, for instance the expected value of a function of their final state. We will see that if we shrink the distance between points and compensate by moving at shorter time intervals, the process approaches *Brownian motion*, a stochastic process in "continuous time". In this limit, its transition density is governed by the *heat equation*, a partial differential equation (PDE).

We then turn the objective around and pose questions of the kind: given this PDE, can we use the discrete model to devise a computationally tractable method to approximate the solution to the PDE; what determines the accuracy of this approximation; which properties of the continuous model are preserved by its discrete approximation. These questions form the backbone of numerical analysis and will be a central topic of the first part of this book. This perspective is relevant in financial engineering because models are usually elegantly formulated in continuous time, without having first looked at a discrete version. It should be added that even in cases where there is a finacially meaningful discrete process in the backgound, there are often more accurate discretisation techniques which give faster solutions to the PDE than the discrete model which may have motivated the PDE in the first place. A probabilistic interpretation of numerical methods can non-the-less give valuable insights in the properties of numerical schemes.

Random walks and the heat equation

A symmetric random walk

Consider a process (a "marker") which performs a random walk on the real line, and whose position X_t at time t evolves as follows. At time $t_0 = 0$, it starts off at $X_0 = 0$. At equally spaced points in time t_m, taken out of $\{m\Delta t : m \in \mathbb{N}\}$ with intervals $\Delta t > 0$, the marker moves left or right an amount $\Delta x > 0$ with equal probability $0 \leq p \leq 1/2$, or stays put with probability $1 - 2p$. We assume that each move is independent of previous moves. So if we

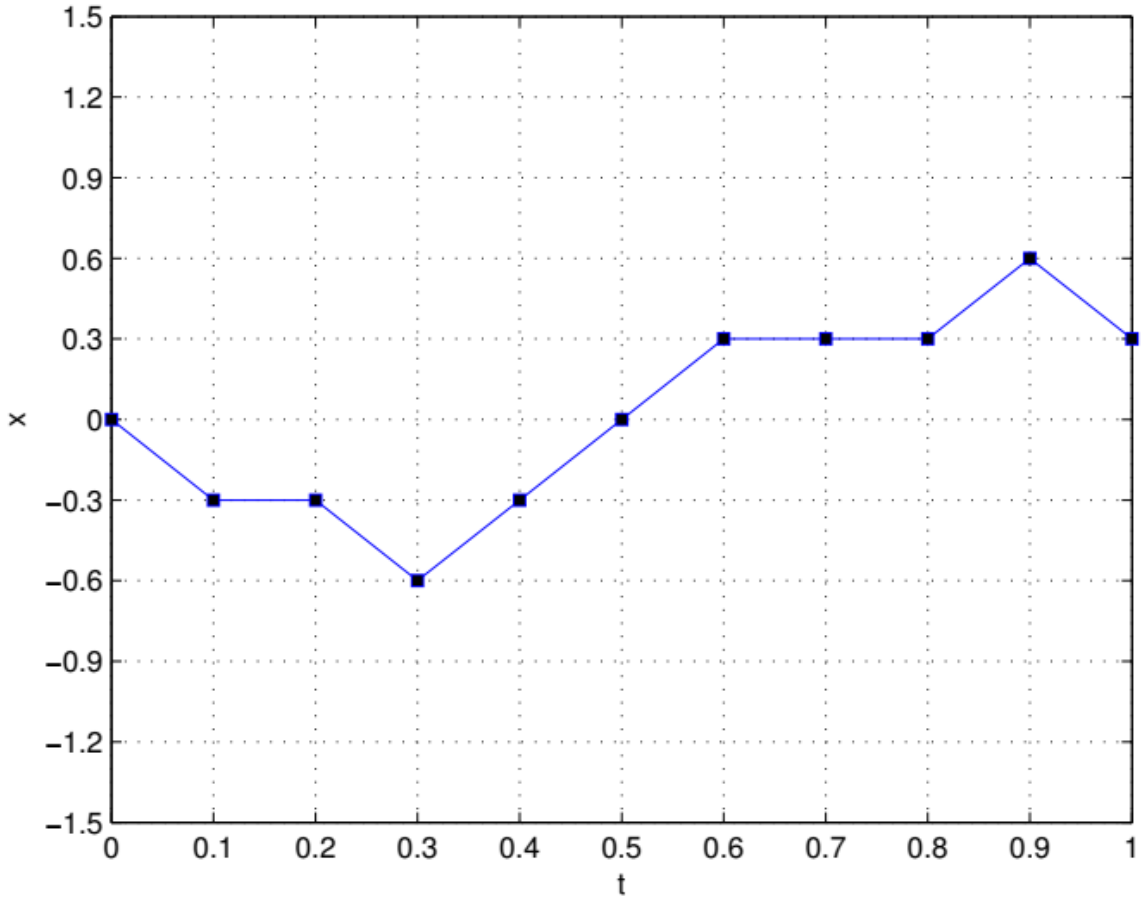

Figure 2.1: A realisation of X_t for $t \in [0,1]$, $\Delta x = 0.3$, $\Delta t = 0.1$, $p = 1/3$.

define

$$\xi_m = \begin{cases} 1 & \text{with probability } p \\ -1 & \text{with probability } p \\ 0 & \text{with probability } 1-2p \end{cases} \tag{2.1}$$

the independent directions of travel, the posistion at t_m is

$$X_{t_m+\Delta t} = X_{t_m} + \xi_m \Delta x$$

and lies on a grid $\{n\Delta x : n \in \mathbb{Z}\}$. In fact, the range of possible positions is restricted by

$$X_{m\Delta t} = \sum_{k=0}^{m-1} \xi_k \Delta x \in \{-m\Delta x, \ldots, 0, \ldots, m\Delta x\}. \tag{2.2}$$

Between these times, we assume the marker moves uniformly to its new position,

$$X_t = X_{t_m} + \frac{t - t_m}{\Delta t}(X_{t_m+\Delta t} - X_{t_m}), \quad t \in [t_m, t_{m+1}].$$

Fig. 2.1 shows a possible realisation of such a path.

As $\mathbb{E}(\xi_m) = 0$, from (2.2) follows

$$\mathbb{E}(X_m) = 0.$$

As the ξ_m are independent and identically distributed (i.i.d.),

$$Var(X_{m\Delta t}) = m\Delta x^2 Var(\xi_0) = m\,\Delta x^2\,2p. \tag{2.3}$$

The variance of the process at a time t is intuitive determined by the number of steps m, the size of a step Δx and the probability $2p$ of a non-zero move. We "standardize" the process to have unit variance per unit time,

$$Var(X_t) = t, \quad t/\Delta t \in \mathbb{N}.$$

This is the case exactly if $Var(X_{\Delta t}) = \Delta t$, which together with (2.3) requires the relation

$$p = \frac{1}{2}\frac{\Delta t}{\Delta x^2}. \tag{2.4}$$

We need $p \leq 1/2$ for (2.1) to make sense, which restricts the range of allowable Δt and Δx^2, specifically $\Delta t \leq \Delta x^2$. This is reflection of the fact that if the number of timesteps is too small relative to the (square of) the grid size, we cannot achieve a prescribed variance. If we do have (more than) enough timesteps, we can adjust the probability of moves to ensure the variance is exactly matched.

Transition probabilities

We now move on to find the probability that the marker is found at a specific point at a given time.

The process starts at $X_0 = 0$, such that

$$\mathbb{P}(X_0 = 0) = 1, \tag{2.5}$$

while at time t, from (2.2),

$$\mathbb{P}(X_{m\Delta t} < -m\Delta x) = \mathbb{P}(X_{m\Delta t} > m\Delta x) = 0. \tag{2.6}$$

Denote by U_n^m the probability that the marker is at position $x_n = n\Delta x$ at time $t_m = m\Delta t$,

$$U_n^m = \mathbb{P}(X_{t_m} = x_n).$$

This discrete probability density can be computed directly by considering all paths leading to a certain point, and the probability of following each of these paths. This is left as an exercise (Exercise 1) and we follow a different tack here.

By partitioning with respect to the position at the current time t_m,

$$\mathbb{P}(X_{t_{m+1}} = x_n | X_{t_m} = x_k) = \begin{cases} p & n = k \pm 1, \\ 1-2p & n = k, \end{cases}$$

and therefore

$$\begin{aligned} U_n^{m+1} &= \sum_{k=-\infty}^{\infty} \mathbb{P}(X_{t_{m+1}} = x_n | X_{t_m} = x_k)\, \mathbb{P}(X_{t_m} = x_k) \\ &= p \cdot U_{n+1}^m + (1-2p) \cdot U_k^m + p \cdot U_{n-1}^m. \end{aligned} \tag{2.7}$$

This defines an induction by which all values U_n^{m+1}, $n \in \mathbb{Z}$, are defined by – and can be explicitly calculated from – the values of U_n^m. The initial condition for this induction is seen from (2.5) as

$$U_n^0 = \delta_{n0} = \begin{cases} 1 & n = 0, \\ 0 & n \neq 0. \end{cases} \tag{2.8}$$

Because of (2.6), only finitely many U_n^m are non-zero for fixed m and need to be computed.

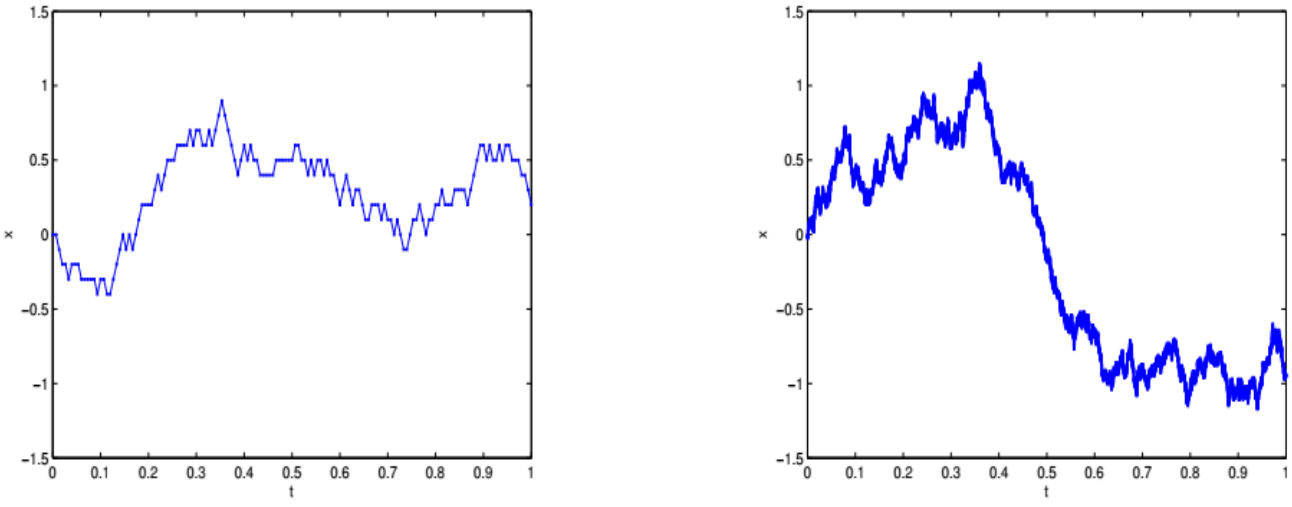

Figure 2.2: Realisation of X_t for $t \in [0,1]$, $\Delta x = 0.1$ and $\Delta x = 0.01$ respectively, $p = 1/3$ and $\Delta t = 2p\Delta x^2$ ensuring unit variance.

The continuous-time limit

Now fix t and let $m \to \infty$ with $\Delta t = t/m$, $0 < p \leq 1$ fixed and $\Delta x = \sqrt{\Delta t/2p}$ as in (2.4). Examples of paths are plotted in Fig. 2.2. We make the dependence on Δt explicit by writing $X_t = X_t^{\Delta t}$. By the assumption that ξ_k are i.i.d. and such that $Var(X_t^{\Delta t}) = t$, it follows from (2.2) by virtue of the Central Limit Theorem (see eg [Grimmet and Stirzaker, 2001]) that

$$X_t^{\Delta t} \to W_t \sim N(0,t).$$

Here $N(0,t)$ denotes the normal distribution with mean 0 and variance t. The density of W_t is therefore given by the normal probability density function

$$u(x,t) = \frac{1}{\sqrt{2\pi t}} e^{-x^2/2t}. \tag{2.9}$$

Convergence is in distribution, [Grimmet and Stirzaker, 2001].

It can indeed be shown that the limiting stochastic process W_t has the following intuitive properties:

- $W_0 = 0$;
- for any finite set of times $0 \leq t_1 < t_2 \cdots < t_m$,

  $$W_{t_2} - W_{t_1},\ W_{t_3} - W_{t_2},\ \ldots, W_{t_m} - W_{t_{m-1}}$$

  are independent;
- for any $0 \leq s \leq t$,

  $$W_t - W_s \sim N(0, t-s);$$

- W_t is continuous in t with probability 1.

These are the defining properties of *standard Brownian motion.* See any of [Steele, 2001, Shreve, 2004].

We now move on to investigate whether there is a continuous-time limit to (2.7) which the continuous probability density (2.9) must satisfy. Inserting (2.4) in (2.7), and rearranging,

$$\frac{U_n^{m+1} - U_n^m}{\Delta t} = \frac{1}{2}\frac{U_{n+1}^m - 2U_n^m + U_{n-1}^m}{\Delta x^2}. \tag{2.10}$$

Motivated by (2.9), we assume that for small Δt the density can be approximated by the smooth function u. We take into account the appropriate scaling with Δx and set

$$u_n^m = \frac{1}{\Delta x} U_n^m,$$

such that u_n^m becomes interpretable as

$$u_n^m \, \Delta x \approx \int_{x_n - \Delta x/2}^{x_n + \Delta x/2} u(x, t_m) \, \mathrm{d}x.$$

Note that u_n^m also satisfies (2.10) because of linearity. By Taylor expansion, for smooth enough u,

$$u(x, t + \Delta t) = u(x,t) + \Delta t u_t(x,t) + o(\Delta t), \tag{2.11}$$

$$u(x \pm \Delta x, t) = u(x,t) \pm \Delta x u_x(x,t) + \frac{1}{2}\Delta x^2 u_{xx}(x,t) + o(\Delta x^2). \tag{2.12}$$

We use order notation for a small parameter h,

$$f(h) = O(g(h)) \quad :\Leftrightarrow \quad \limsup_{h \to 0} \frac{f(h)}{g(h)} < \infty$$

(f goes to 0 "at least as fast as" g), and

$$f(h) = o(g(h)) \quad :\Leftrightarrow \quad \limsup_{h \to 0} \frac{f(h)}{g(h)} = 0$$

(f goes to 0 "faster than" g), where $\limsup_{h\to 0} g(h) = \lim_{h \to 0} \sup_{0 < k \leq h} g(k)$.

From (2.11) and (2.12), by insertion,

$$\delta_t^+ u = \frac{u(x, t + \Delta t) - u(x,t)}{\Delta t} \to \frac{\partial u}{\partial t} \qquad \text{as } \Delta t \to 0,$$

$$\delta_x^2 u = \frac{u(x + \Delta x, t) - 2u(x,t) + u(x - \Delta x, t)}{\Delta x^2} \to \frac{\partial^2 u}{\partial x^2} \qquad \text{as } \Delta x \to 0.$$

We conjecture that u_n^m can be approximated by $u(x_n, t_m)$, then identifying terms with (2.10), letting again $\Delta x, \Delta t \to 0$, one approaches formally

$$\frac{\partial u}{\partial t} = \frac{1}{2}\frac{\partial^2 u}{\partial x^2}. \tag{2.13}$$

So in some sense the discrete inductive formula approaches the heat equation. We will make this notion more precise in the next section. For now, one verifies easily by insertion that (2.9) indeed solves (2.13). Taking $t \to 0$ in (2.9),

$$u(x, 0) = \delta(x), \tag{2.14}$$

the *Dirac delta distribution*. The marker starts from 0 with probability 1. One also checks easily that

$$u_0(x) = \left\{ \begin{array}{ll} \frac{1}{\Delta x} & x \in [-\Delta x/2, \Delta x/2] \\ 0 & \text{else} \end{array} \right\} \longrightarrow \delta(x) \qquad \text{for } \Delta x \to 0,$$

confirming $u_n^0 = U_n^0/\Delta x = \delta_{n0}/\Delta x$ as a reasonable approximation. We summarize this as

$$\frac{u_n^{m+1} - u_n^m}{\Delta t} = \frac{1}{2}\frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{\Delta x^2}, \quad m \geq 0 \tag{2.15}$$

$$u_n^0 = \frac{1}{\Delta x}\delta_{n0} = \left\{ \begin{array}{ll} \Delta x^{-1} & n = 0, \\ 0 & \text{otherwise.} \end{array} \right. \tag{2.16}$$

This scheme can be used as an implementable numerical method to approxiamte the solution to the heaat equation, as detailed in the next section.

The explicit (forward) Euler scheme

Definition of the scheme

We saw in 2.1.3 that in the "continuous-time limit" the symmetric random walk approaches Brownian motion, whose transition probability, the normal distribution, solves the heat equation. Conversely, retracing the steps, the solution to the heat equation approximately satisfies the discrete inductive scheme (2.15). More precisely, if u solves the heat equation, one sees

$$\frac{u(x, t+\Delta t) - u(x,t)}{\Delta t} = \frac{1}{2}\frac{u(x+\Delta x, t) - 2u(x,t) + u(x-\Delta x, t)}{\Delta x^2} + o(1) \tag{2.17}$$

where $o(1)$ is a term that goes to zero if Δt and Δx do. In fact, taking the Taylor expansions further – by one order in (2.11), and two in (2.12) – the term can be seen to be of order $O(\Delta t, \Delta x^2)$ if u is sufficiently smooth to allow this analysis. This opens up the opportunity of using (2.15) as a numerical scheme to approximate the solution to the heat equation.

Let u_n^m denote the numerical approximation to the solution at x_n, t_m, where $x_n = n\Delta x$, $n \in \mathbb{Z}$, and $t_m = m\Delta t$, $m = 0, \ldots, M$ with $M\Delta t = T$. Motivated by (2.17), one defines the *explicit Euler scheme* as

$$\frac{u_n^{m+1} - u_n^m}{\Delta t} = \frac{1}{2}\frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{\Delta x^2}, \tag{2.18}$$

which we write shorthand as

$$\delta_t^+ u_n^m = \delta_x^2 u_n^m.$$

We call this a *finite difference scheme*, because the (partial) derivatives of the heat equation have been replaced by "finite differences" in the coordinate directions. This step allows the computation of approximations to the PDE solution by computation of a finite number of values, which, even if the number of equations can be large if a fine grid is used, is implemetable on a computer.

Rewriting (2.18) as

$$u_n^{m+1} = pu_{n+1}^m + (1-2p)u_n^m + pu_{n-1}^m \tag{2.19}$$

with $2p = \Delta t/\Delta x^2$, we see why the scheme is called *explicit*: the value at a desired time, say t_{m+1}, can be explicitly computed from values at time t_m using (2.18), and hence inductively from initial data at t_0.

To define the scheme fully, we have to specify inital conditions for u_n^0, most naturally as in (2.16).

2.2.2 Key properties

If $2p \in [0,1]$, ie

$$\Delta t \leq \Delta x^2,$$

u_n^{m+1} defined by (2.19) is a weighted average of u_{n+1}^m, u_n^m, u_{n-1}^m, the solution in an immediate neighbourhood at the previous timestep. It follows

$$\min_{k\in\mathbb{Z}} u_k^m \leq \min\{u_{n-1}^m, u_n^m, u_{n+1}^m\} \leq \underbrace{pu_{n+1}^m + (1-2p)u_n^m + pu_{n-1}^m}_{u_n^{m+1}} \leq \max\{u_{n-1}^m, u_n^m, u_{n+1}^m\} \leq \max_{k\in\mathbb{Z}} u_k^m.$$

Inductively, for all m

$$\min_{k\in\mathbb{Z}} u_k^0 \leq u_n^m \leq \max_{k\in\mathbb{Z}} u_k^0$$

We can say the numerical solution satisfies a *discrete maximum principle*. This is the discrete analogue of the maximum principle obeyed by the heat equation and other parabolic initial-value problems. The solution remains bounded from below and above, which is a measure of stability of the scheme as a numerical method.

Moreover, the solution is *monotone*, i.e. if $u^m = (\ldots, u_{-1}^m, u_0^m, u_1^m \ldots)$ and v^m are finite difference solutions to the initial conditions u^0 and v^0 with $u^0 \geq v^0$, then $u^m \geq v^m$ for all m. In particular, $u^m \geq 0$ if $u^0 \geq 0$. Lastly,

$$\sum_{n\in\mathbb{Z}} u_n^{m+1} = \sum_{n\in\mathbb{Z}} pu_{n+1}^m + (1-2p)u_n^m + pu_{n-1}^m = \sum_{n\in\mathbb{Z}} u_n^m,$$

so if u^0 has the properties of a discrete probability density – non-negativity and correct scaling – then so too has u^m for all m.

2.2.3 Nodes and trees

Note from the construction that $u_n^m = 0$ for $n < -m$ and $n > m$: the underlying random walk stays in a range $-m\Delta t \leq x \leq m\Delta t$ with probability one. Therefore, it is sufficient to compute u_n^m in the range $n \in \{-m, \ldots, m\} \subset \{-M, \ldots, M\}$. This motivates the name *trinomial tree*: every parent *node* (x_n, t_m) has three children nodes $(x_{n-1}, t_{m+1}), (x_n, t_{m+1}), (x_{n+1}, t_{m+1})$, the states the walk can move to in the next timestep. The *root* of the tree is $(0,0)$. For $p = 1/2$, the probability for a move from (x_n, t_m) to (x_n, t_{m-1}), i.e. not moving, is zero. The tree collapses to a *binomial tree* with only two possible children $(x_{n-1}, t_{m+1}), (x_{n+1}, t_{m+1})$.

The *support* of u^m, i.e. the range of gridpoints x_n, where u^m has non-zero value, is $[-m\Delta x, m\Delta x]$ where $m\Delta t = t$ and

$$m\Delta x = m\sqrt{\Delta t/2p} = \sqrt{m/2p}\sqrt{t}. \tag{2.20}$$

This is in contrast to the continuous problem where the marker can move to any position in infinitesimally small time, albeit with low probability if that position is far away. The

continuous model therefore has infinite *speed of propagation*, whereas the discrete model has finite speed of propagation. The underlying Brownian motion is normally distributed with variance t and standard deviation $\sqrt{t}$. The relevant range of points which the continuous process is likely to visit should therefore increase proportional to $\sqrt{t}$. This is also seen from the scaling invariance of the solution to the heat equation in (2.9), $u(x,t) = 1/\sqrt{t}\, u(x/\sqrt{t}, 1)$.

Either way, the range of x with non-negligible $u(x,t)$ for a given t depends on $\sqrt{t}$ alone, certainly not on any parameters of the discretisation. For small timesteps Δt, the range imposed by (2.20) is unnecessarily large if an approximate solution is the goal. From a practical perspective, we want to keep the number of nodes to a minimum for computational efficiency.

For now, choose a value x_{max} large enough, as measured in units of standard deviations, to ensure that the approximation $u(-x_{max}, t) = u(x_{max}, t) = 0$ is justified for all $t \leq T$ where T is the time horizon of interest.

For the heat equation, $u(-x_{max}, 1) = u(x_{max}, 1) = \phi(x_{max})$ where ϕ is the standard normal density, e.g. $\phi(5) = 1.4867e-06$. Hence set in the following $x_{max} = 5\sqrt{T}$, and

$$u^m_{-N} = u^m_N = 0 \tag{2.21}$$

as boundary condition. Then (2.19) is well-defined for all m and $-N+1 \leq n \leq N-1$.

2.2.4 Implementation and tests

A practically extremely useful feature of the explicit scheme is that values for u^{m+1}_n are explicitly computable from the range of u^m_n's by (2.19) and (2.21). Starting from the initial condition (2.16), the solution for all m is therefore found inductively. This is sketched as pseudocode in Algorithm 1.

A few comments:

- When implementing, if we are only interested in the solution at $t = T$, i.e. $m = M$, there is no need to store the solution for all m and therefore we omit the superscript. At a given timestep, only two vectors are needed, denoted u for u^m and w for u^{m+1} in the algorithm.

- Most programming languages start indexing at 0 or 1. Therefore, we now shift indices by $N+1$ and use as solution vector, omitting m, $u = (u_1, u_2, \ldots, u_{2N}, u_{2N+1})$, corresponding to nodes $(-x_{max}, -x_{max} + \Delta x, \ldots, -\Delta x, 0, \Delta x, \ldots, x_{max} - \Delta x, x_{max})$, where $\Delta x = x_{max}/N$. That is u_n is thought to approximate $u((n-N-1)\Delta x, m\Delta t)$ rather than $u(n\Delta x, m\Delta t)$ as earlier.

- In lines 5-8, the initial condition is set according to (2.16) with the indices shifted as discussed above. Node $N+1$ is halfway between indices 1 and $2N+1$ and therefore $x_{N+1} = 0$.

- The main body is in lines 9-15, where the outer loop is over all timesteps, the inner loop implements (2.19) for all inner grid points, and line 10 incorporates (2.21).

In principle, Δt and Δx can now be chosen independently, however only for $p = \Delta t/2\Delta x^2 \leq 1/2$ the result is meaningful. (In the limiting case $p = 1/2$, the trinomial tree reduces to a binomial tree.) We therefore focus on the case that $\Delta t, \Delta x \to 0$ with $2p = \Delta t/\Delta x^2$ fixed and less than 1.

Algorithm 1 Explicit Euler scheme

$T \leftarrow 1$, $x_{max} \leftarrow 5\sqrt{T}$
$N \leftarrow 16$, $M \leftarrow 16$
$\Delta t \leftarrow T/M$, $\Delta x \leftarrow x_{max}/N$
$p = \Delta t/2\Delta x^2$
for $n \leftarrow 1,\ 2N+1$ **do**
    $u_n \leftarrow 0$
end for
$u_{N+1} \leftarrow 1/\Delta x$
for $m \leftarrow 1,\ M$ **do**
    $w_1 \leftarrow 0$, $w_{2N+1} \leftarrow 0$
    for $n \leftarrow 2,\ 2N$ **do**
        $w_n \leftarrow p\,u_{n-1} + (1-2p)\,u_n + p\,u_{n+1}$
    end for
    $u = w$
end for

2.3 The implicit (backward) Euler scheme

A major drawback of the explicit scheme is the restriction on the timestep to obtain an *accurate* and *stable* solution. In practice, this means that a very large number of timesteps is needed, resulting in large computing times. We look at accuracy and stability in turn.

To develop an intuition for the stability of the scheme, rewrite (2.19) as

$$u_n^{m+1} = u_n^m + \frac{\Delta t}{\Delta x^2}\left(\frac{u_{n+1}^m + u_{n-1}^m}{2} - u_n^m\right).$$

If the solution at t_m is locally *convex* at x_n, such that the value u_n^m is smaller than the average of the neighbouring values, the increment $u_n^{m+1} - u_n^m$ is positive and the solution pulled up towards the average. Conversely, if the solution at t_m is locally *concave* at x_n, such that the value u_n^m is larger than the average of the neighbouring values, it is pulled down towards the average. Note this is basically the mechanism underlying the maximum/minimum principle of parabolic PDEs. However, if the timestep is chosen too large, the value overshoots the average and, in the extreme, $u_n^{m+1} \to \pm\infty$ as $\Delta t \to \infty$. This happens because the instantaneous time change is extapolated too far into the future, neglecting the two-way interplay of the change at x_n with the change of neighbouring values. To allow larger timesteps, we need to "couple" the equation for u_n^{m+1} somehow with those for u_{n-1}^{m+1}, u_{n+1}^{m+1} etc.

As concerns accuracy, we have seen that (2.17) defines an approximation to the heat equation which is accurate up to order $O(\Delta t, \Delta x^2)$. It is more accurate in the x-direction than in t. A simple high-level argument why the finite x difference is of second order accurate is the symmetry in $x \pm \Delta x$. There is no such symmetry in t in (2.17). This gives the idea for the following scheme.

2.3.1 Definition of the scheme

The mirror image in time of the scheme (2.18) is

$$\delta_t^- u_n^m = \frac{u_n^m - u_n^{m-1}}{\Delta t} = \frac{1}{2}\frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{\Delta x^2} = \delta_x^2 u_n^m, \tag{2.22}$$

where the right-hand side is evaluated at new time t_m instead of t_{m-1}.

Taylor expansion shows again that

$$\frac{u(x,t)-u(x,t-\Delta t)}{\Delta t}=\frac{1}{2}\frac{u(x+\Delta x,t)-2u(x,t)+u(x-\Delta x,t)}{\Delta x^2}+O(\Delta t,\Delta x^2),$$

in analogy to the explicit scheme.

Now write (2.22) as

$$-pu_{n+1}^m+(1+2p)u_n^m-pu_{n-1}^m=u_n^{m-1} \tag{2.23}$$

with $2p=\Delta t/\Delta x^2$ as before. We understand (2.23) again to define $u^m=(u_n^m)_{-\infty<n<\infty}$ in terms of u^{m-1}, such that any u^m is determined by forward induction from an initial condition u^0. In contrast to the explicit Euler scheme, however, a single u_n^m can no longer be explicitly computed from u^{m-1}, but is implicitly defined via a set of equations. Hence the name *implicit* scheme.

Going back to the large timestep behaviour, which was the crux of the explicit Euler scheme, if we let $\Delta t\to\infty$, i.e. $p\to\infty$ in (2.23), one formally gets $u_n^m\to(u_{n+1}^m+u_{n-1}^m)/2$ as desired. Note that p here no longer has any interpretation of a transition probability of a random walk, but is merely a parameter of the discretisation, so it is not unnatural to let it increase. There is no overshoot and the scheme is expected to be stable. We analyse this more carefully now.

2.3.2 Key properties

As for any numerical method, a crucial question is whether the numerical solution preserves properties of the solution to the original equation. In particular, we can ask here whether the solution can be seen as a discrete probability density. Recall this was the case for the explicit scheme, in fact it was the original motivation in its derivation.

Assuming boundedness, the solution u^1 after one step of the implicit Euler scheme is found (see Exercise 2) to be

$$u_n^1=c\,z^{|n|}, \tag{2.24}$$

where $z=\alpha-\sqrt{\alpha^2-1}\in(0,1)$, $\alpha=(2p+1)/(2p)\in(1,\infty)$, $c\,\Delta x=(1-2p)/(1+2p)$. (**Warning**: This is obviously wrong as c would be negative for $p>1/2$ – need to re-compute.) One verifies that

$$\Delta x\sum_{n\in\mathbb{Z}}u_n^1=1,$$

and clearly $u_n^1>0$. This makes $u_n^1\Delta x$ interpretable as transition probability from $x_0=0$ at $t_0=0$ to x_n at time t_1.

So if we define an i.i.d. sequence (ξ_m) such that

$$\xi_m=n \text{ with probability } u_n^1\Delta x, \tag{2.25}$$

instead of (2.1), a similar construction to Section 2.1.1 is possible. Note that u_n^1 is strictly positive for all n, such that there is a positive probability for the random walk to move to any point within a single timestep. This is characteristic for implicit schemes, and contrasts the behaviour of the explicit scheme. We can use this argument to construct u_n^m inductively and conclude that it indeed has the properties of a probability density.

There is another quick way to see that the scheme preserves non-negativity. Assume the opposite and that m is the first time index where the solution goes below zero somewhere. Then it follows from $u_n^m \to 0$ for $n \to \pm\infty$ that a negative minimum over all points is attained at a certain index k, $u_k^m = \min_{n\in\mathbb{Z}} u_n^m$. At this point,

$$u_k^m \leq \frac{1}{2}(u_{k+1}^m + u_{k-1}^m)$$

and hence

$$u_k^{m-1} = -pu_{k+1}^m + (1+2p)u_k^m - pu_{k-1}^m \leq u_k^m \tag{2.26}$$

and from $u_n^{m-1} \geq 0$ follows a contradiction, hence the solution must be non-negative.

Moreover, we get by similar reasoning a discrete minimum/maximum principle, i.e. for all $m \geq 0$, $n \in \mathbb{Z}$,

$$\min_{k\in\mathbb{Z}} u_n^0 \leq u_n^m \leq \max_{k\in\mathbb{Z}} u_n^0.$$

This implies further that there is a unique solution for all m. We can also deduce monotonicity in the initial data as in the explicit case, particularly $u^m \geq v^m$ for two solutions u^m and v^m with $u^0 \geq v^0$. By summing (2.23) over n, one gets again

$$\sum_{n\in\mathbb{Z}} u_n^m = \sum_{n\in\mathbb{Z}} u_n^{m-1}.$$

In particular, the properties of a discrete probability density are preserved.

2.3.3 Boundary conditions

In contrast to the explicit method, the grid value at any point in the implicit scheme has an influence on all other values in the next timestep. Equation (2.23) constitutes an infinite system which has to be solved simultaneously. Unless a closed-form solution to (2.23) is available, one has to restrict the sytem to finitely many unknowns to make it computationally feasible, which is done again by setting up approximate boundary conditions. We argued earlier that $u(x_{max}, t) = 0$ for large enough x_{max}, and similar for $-x_{max}$, such that we can restrict the system of equations (2.23), at least approximately, to a range $-N+1 \leq n \leq N-1$ by setting

$$u_{-N}^m = u_N^m = 0. \tag{2.27}$$

Equations (2.23) and (2.27) give a linear system for $u^m = (u_{-N}^m, \ldots, u_0^m, \ldots u_N^m)$,

$$Ku^m = u^{m-1}, \tag{2.28}$$

where

$$K = \begin{pmatrix} 1 & 0 & 0 & & & 0 & 0 \\ -p & 1+2p & -p & & \ldots & 0 & 0 \\ 0 & -p & 1+2p & -p & \ldots & 0 & 0 \\ 0 & & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & & \ldots & -p & 1+2p & -p & 0 \\ 0 & & \ldots & 0 & -p & 1+2p & -p \\ 0 & 0 & & & 0 & 0 & 1 \end{pmatrix} \in \mathbb{R}^{(2N+1)\times(2N+1)}.$$

The first and last line determine the boundary values. The linear system (2.28) has to be solved in each timestep, replacing the explicit calculations of the previous scheme.

2.3.4 Implementation and tests

The implicit Euler algorithm follows similar lines to the explicit one, with the modification that a linear system has to be set up, and solved in every timestep. We explain this first, then sketch the overall algorithm.

Solving the linear system

The system (2.28) is of the general form

$$\begin{pmatrix} b_1 & c_1 & 0 & \dots & 0 \\ a_2 & b_2 & c_2 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & \dots & 0 & a_n & b_n \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix}, \tag{2.29}$$

where $n = 2N + 1$ for simplicity. For future reference, denote the system matrix by

$$K = \text{tridiag}(a, b, c, n), \tag{2.30}$$

n being the dimension of the square matrix $K \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$ the diagonal, and a and c left and right of the diagonal respectively. Note that a_1 and c_n are not needed but we define $a, c \in \mathbb{R}^n$ for ease of notation.

The system matrix K is *sparse*, meaning that per line there is a given number – in this case three – of non-zero elements. This is independent of the size of the matrix and does not increase if the number N of grid points increases. Loosely speaking, this comes from the fact that differentiation is a "local" operation, so a discretisation of a differential equation should not have to take into account grid values some distance away.

More specifically, K is a *tridiagonal* matrix: it has non-zero elements only in the main diagonal and for neighbouring entries. This can be taken advantage of when solving the system. An important consequence is that the number of calculations, and hence the computational time required for the solution of the system is proportional to the number of unknowns. This is referred to as an $O(n)$ algorithm. The overall computational time of the implicit scheme is therefore not notably larger than for the simpler explicit scheme.

One easy way to solve such a linear system is by an adaptation of Gaußian elimination, known as the Thomas algorithm. It exploits the tridiagonal structure of the matrix to solve the system in $O(n)$ operations.

Although it is sometimes useful to write $u^m = K^{-1}u^{m-1}$, there are almost no circumstances under which it is advisable to invert these matrices directly in practice. To see why,

consider the discretisation matrix K and its inverse K^{-1}, for $x_{max} = 3$, $N = 3$, $M = 1$:

$$K = \begin{pmatrix} 2 & -0.5 & 0 & 0 & 0 & 0 & 0 \\ -0.5 & 2 & -0.5 & 0 & 0 & 0 & 0 \\ 0 & -0.5 & 2 & -0.5 & 0 & 0 & 0 \\ 0 & 0 & -0.5 & 2 & -0.5 & 0 & 0 \\ 0 & 0 & 0 & -0.5 & 2 & -0.5 & 0 \\ 0 & 0 & 0 & 0 & -0.5 & 2 & -0.5 \\ 0 & 0 & 0 & 0 & 0 & -0.5 & 2 \end{pmatrix} \tag{2.31}$$

$$K^{-1} = \begin{pmatrix} 0.5359 & 0.1436 & 0.0385 & 0.0103 & 0.0028 & 0.0007 & 0.0002 \\ 0.1436 & 0.5744 & 0.1539 & 0.0412 & 0.0110 & 0.0029 & 0.0007 \\ 0.0385 & 0.1539 & 0.5771 & 0.1546 & 0.0414 & 0.0110 & 0.0028 \\ 0.0103 & 0.0412 & 0.1546 & 0.5773 & 0.1546 & 0.0412 & 0.0103 \\ 0.0028 & 0.0110 & 0.0414 & 0.1546 & 0.5771 & 0.1539 & 0.0385 \\ 0.0007 & 0.0029 & 0.0110 & 0.0412 & 0.1539 & 0.5744 & 0.1436 \\ 0.0002 & 0.0007 & 0.0028 & 0.0103 & 0.0385 & 0.1436 & 0.5359 \end{pmatrix} \tag{2.32}$$

The inverse of a sparse (tridiagonal) matrix is normally non-sparse, with $O(n^2)$ non-zero elements. In each timestep of the implicit Euler scheme, we would multiply this matrix with a vector. The complexity of this, i.e. the number of operations necessary, is also $O(n^2)$. The complexity of the inversion itself is $O(n^3)$, but has to be performed once in the set-up phase only. This compares to $O(n)$ per timestep when solving the system.

The implicit Euler algorithm

The full algorithm of the implicit Euler scheme is sketched in Algorithm 2. Lines 5-13 define the initial condition and system matrix, where we stick to the format from (2.28), (2.30). Line 16 replaces lines 11 to 13 of Algorithm 1.

Algorithm 2 Implicit Euler scheme

1: $T \leftarrow 1$, $x_{max} \leftarrow 5\sqrt{T}$
2: $N \leftarrow 16$, $M \leftarrow 16$
3: $\Delta t \leftarrow T/M$, $\Delta x \leftarrow x_{max}/N$
4: $\lambda = \Delta t/\Delta x^2$
5: $u_1 \leftarrow 0$
6: $a_1 \leftarrow 0$, $b_1 \leftarrow 1$, $c_1 \leftarrow 0$
7: **for** $n \leftarrow 2$, $2N-1$ **do**
8: $\quad u_n \leftarrow 0$
9: $\quad a_n \leftarrow -\lambda/2$, $b_n \leftarrow 1+\lambda$, $c_n \leftarrow -\lambda/2$
10: **end for**
11: $u_{2N+1} \leftarrow 0$
12: $a_{2N+1} \leftarrow 0$, $b_{2N+1} \leftarrow 1$, $c_{2N+1} \leftarrow 0$
13: $u_{N+1} \leftarrow 1/\Delta x$
14: $K = \text{tridiag}(a, b, c, 2N+1)$
15: **for** $m \leftarrow 1$, M **do**
16: $\quad$ solve $Kv = u$ for v, set $u \leftarrow v$
17: **end for**