Assignment 3, due September 30

Corrections: September 27:
The first formula in Exercise (1c) should end with $O(\Delta t^2)$, not $O(\Delta t^4)$.
Exercise (2a) should not have $\frac{1}{2}$ on the right. It should say $\mathrm{E}[(\Delta Y)^2] = \Delta t + \cdots$.

1. (*Finite differences*) This exercise explains the finite difference formulas used in the computational exercise. The description of the computational exercise has some background you will need here.

   (a) Suppose $f(x,t)$ is a sufficiently smooth function and Δx and Δt are small. Show that

   $$\begin{aligned}\partial_x f(x,t) &= \frac{f(x+\Delta x,t) - f(x-\Delta x,t)}{2\Delta x} + O(\Delta x^2)\\ \partial_x^2 f(x,t) &= \frac{f(x+\Delta x,t) - 2f(x,t) + f(x-\Delta x,t)}{\Delta x^2} + O(\Delta x^2)\\ \partial_t f(x,t) &= \frac{f(x,t) - f(x,t-\Delta t)}{\Delta t} + O(\Delta t)\ .\end{aligned}$$

   These are *finite difference formulas*. There are other finite difference formulas to approximate the same derivatives, but these lead to the overall finite difference method used in the Computing Exercise.

   (b) Suppose x_l and T are fixed. Define $x_j = x_l + j\Delta x$ and $t_k = T - k\Delta t$. Suppose that f satisfies the backward equation $\partial_t f + a\partial_x f + \frac{1}{2}\partial_x^2 f = 0$. Suppose that $\lambda = \frac{\Delta t}{2\Delta x^2}$ is fixed as $\Delta x \to 0$. Show that

   $$\begin{aligned}f(x_j,t_{k+1}) = f(x_j,t_k)&\\ &+ \frac{a\Delta t}{2\Delta x}\left[f(x_{j+1},t_k) - f(x_{j-1},t_k)\right]\\ &+ \frac{\Delta t}{2\Delta x^2}\left[f(x_{j+1},t_k) - 2f(x_j,t_k) + f(x_{j-1},t_k)\right]\\ &+ O(\Delta x^4)\ .\end{aligned}$$

   (c) In the code, this is written in the form

   $$f(x_j,t_{k+1}) = p_- f(x_{j-1},t_k) + p_0 f(x_j,t_k) + p_+ f(x_{j+1},t_k) + O(\Delta t^2)\ .$$

   Show that $p_- + p_0 + p_+ = 1$. Show that if $\lambda < \frac{1}{2}$ and Δx is small enough, then $p_- > 0$, $p_0 > 0$ and $p_+ > 0$. The condition $\lambda < \frac{1}{2}$ is called the *CFL* condition, after the people who discovered it. Those are Courant (the founder of the Courant Institute), Friedrichs (one of the founding professors), and Lewy (a colleague of theirs who became a professor at Berkeley).

(d) We will use the approximate formula from part (c) that applies to the exact solution as motivation to declare an exact formula for an approximation solution. The numbers f_{kj} are supposed to approximate $f(x_j, t_k)$ and the formula is

$$f_{k+1,j} = p_- f_{k,j-1} + p_0 f_{kj} + p_+ f_{k,j+1} \ .$$

This will be defined for $k > 0$, starting with $f_{0,j} = v(x_j)$, which is an exact formula for $f(x_j, T)$. The boundary conditions are $f_{k,0} = 0$ and $f_{k,n+1} = 0$, corresponding to $f(x_l, t) = 0$ and $f(x_r, t) = 0$. Show that the operator $f_k \rightsquigarrow f_{k+1}$ is *stable* in the sense that, if p_1, p_0 and p_+ are positive and sum to 1, then

$$\sum_{j=1}^{n} |f_{k+1,j}| \leq \sum_{j=1}^{n} |f_{k,j}| \ .$$

This implies that the numbers f_{kj} do not "blow up" for large k.

2. Consider the discrete random walk Y_k that has the stochastic evolution

$$Y_{i+1} = \begin{cases} Y_i - \Delta x & \text{with probability } p_- \\ Y_i & \text{with probability } p_0 \\ Y_i + \Delta x & \text{with probability } p_+ \end{cases}$$

Define $t_i = i\Delta t$ [warning, not as in Exercise 1]. Suppose p_-, p_0, and p_+ are defined as in exercise 1.

(a) Show that Y has "infinitesimal" mean a and "infinitesimal variance" $\frac{1}{2}$ in the sense that $\mathrm{E}[\Delta Y] = a\Delta t$ and $\mathrm{E}[(\Delta Y)^2] = \Delta t + O(\Delta t^2)$. Here $\Delta Y = Y_{i+1} - Y_i$. (not really infinitesimal because Δt doesn't go to zero for a given process.)

(b) Define a discrete value function

$$f_{ij} = \mathrm{E}[\, v(Y_{n_t}) \mid Y_i = j\Delta x] \ .$$

Assume that $Y_0 = j\Delta x$ for some j. Show that the numbers f_{ij} satisfy a discrete backward equation that is identical (up to changes in notation) to the finite difference update formula from Exercise (1d). The convergence of random walk to Brownian motion implies a relation between the discrete process and the partial differential equation that is the backward equation for the diffusion process.

3. Suppose $Z \sim \mathcal{N}(\mu, \sigma^2)$. Calculate $A = \mathrm{E}\big[\, e^{\alpha Z}\big]$. Hint, write A as in terms of an integral like $\int e^{-bx - cx^2} dx$ and complete the square to make $bx + cx^2 = d + c(x - x_0)^2$, then use a change of variables $y = x - x_0$ to calculate the integral.

4. Consider Brownian motion with drift $X_t = W_t + at$ as in Assignment 2. Assume that $X_t = x$ and show that $X_T = x + Z$ where Z is normal with a certain mean and variance. Calculate the value function with payout $v(x) = e^{ax}$. Use the formula from Exercise (3). Then check that your answer satisfies the backward equation.

Computing exercise

This exercise involves the backward equation for standard Brownian motion or Brownian motion with constant drift with absorbing boundaries at $x = x_l$ and $x = x_r$. The code computes numbers $f_{kj} \approx f(x_j, t_k)$. The x points are $x_j = x_j + j\Delta x$, for $j = 1, \ldots, n$. [Be careful, in the code j runs from 0 to $n-1$ because it's Python.] The distance between points is always the same, so $x_r - x_l = (n+1)\Delta x$. In the code, you give n and it computes Δx. The times in the code are $t_k = T - k\Delta t$, so $t_0 = T$, $t_1 = T - \Delta t$, etc. These go backwards because that's the right way to go with the backward equation. The CFL ratio is $\lambda = \frac{\Delta t}{2\Delta x}$. In the code, you specify λ and it computes Δt. The number of time steps is $n_t = T/\Delta t$. The problem with this is that n_t is likely not to be an integer. Therefore the code makes Δt a little smaller in order to round n_t up to the nearest integer.

The finite difference calculation uses the transition probabilities p_+, p_0, and p_- described in Exercise 1. Suppose f_k is the n component vector with components f_{kj}. The "inner loop" of the code computes f_{k+1} from f_k. The endpoint calculations, which are $j = 1$ next to x_l and $j = n$ next to $x = x_r$ are special. The formula for them assume the absorbing boundary conditions $f(x_l, t) = 0$ and $f(x_r, t) = 0$. The "interior" points ($j = 2, \ldots, n-1$) use the full three point update formula.

Most finite difference calculations like this one save storage by saving only two vectors rather than the whole solution. You need two vectors f_k and f_{k+1} to take a time step. The code computes f_{k+1} from f_k,. These vectors are called `fk` and `fkp1` in the code. Then it copies the newly computed values `fkp1` into the array `fk` to get ready for the next time step. Real codes that solve real PDEs in three or more dimensions would not have enough storage for all the f_k.

To keep the code simple, the code makes a movie frame every time step. A real code probably would make a new frame every so many time steps, to make a smaller movie file.

Task 1. Download the code `BackwardEquationDemo.py`, run it. It should create a movie called `BackwardEquationMovie.mp4`. Check that this is the same as as `BackwardEquationMovieDownload.mp4` that is posted on the web site. Check that the movie looks about the same if you change the *resolution* of the computation, which is $\Delta x = (x_r - x_l)/(n+1)$. The resolution is determined by the number of x points, which is n. This will not work if n is too small. It will take a long time to run if n is too large. Try it.

Task 2. The code "out of the box" has a payout function equal to $v(x) = x^2$. Try other payout functions. Examples you might try are $v(x) = 1$ (you get 1 if you survive to time T) or $v(x) = 1$ only for $|x| < r$ and $v(x) = 0$ otherwise. Notice properties these all have in common – how they behave near the endpoints when t is close to T, and how they behave when t is closer to 0.

Task 3. Modify the code so that it solves the backward equation for Brownian motion with constant drift velocity a. Note whether the solution in the movie moves with velocity a or $-a$ and explain the direction. Note whether the solutions decay (become small) faster when a is large or small and explain this in terms of hitting time to the boundary when there is drift.

Task 4. The finite difference method is *unstable* if $\lambda > \frac{1}{2}$. Modify λ in the code (maybe $\lambda = .6$ instead of .4) and describe what happens when you "violate the CFL condition".