Stochastic Calculus, Courant Institute, Fall 2019

http://www.math.nyu.edu/faculty/goodman/teaching/StochCalc2019/StochCalc.html

Always check the `classes` message board before doing any work on the assignment.

Assignment 7, due November 4

Corrections: [none yet]

1. Consider the pair of differential equations

$$\frac{dy}{dt} = v$$
$$\frac{dv}{dt} = -\gamma v - \omega^2 y \, .$$

The first equation says that y moves with speed v. The second equation is related to force (the mass is left out for simplicity) The first term, $-\gamma v$.represents friction, which is a force proportional to the speed, but in the opposite direction to slow it down. The second term, $-\omega^2 y$ represents a "spring", which is a force proportional to the displacement of y from 0. The coefficient is written ω^2 partly to emphasize that it's positive.

   (a) Define $x_t \in \mathbb{R}^2$ to represent both components: $x_t = \begin{pmatrix} y_t \\ v_t \end{pmatrix}$. Write the differential equation system in the form $\frac{d}{dt} x_t = A x_t$, and identify A.

   (b) Find the eigenvalues and right eigenvectors of A. Show that the eigenvalues and right are real if $\gamma > 2|\omega|$ and complex otherwise. This may not work on the borderline case $\gamma = 2|\omega|$.

   (c) Form the right eivenvector matrix $U = \begin{pmatrix} | & | \\ u_1 & u_2 \\ | & | \end{pmatrix}$. Form $V = U^{-1}$ and check that the rows of V are left eigenvectors of A with the same eigenvalues.

2. Suppose A is a "data matrix" with m rows and n columns. For example, you can think of the columns of A as time series, each series having m entries. This exercise goes through the singular value decomposition (which many students have seen already). By tradition, the singular values are ordered from largest to smallest rather than from smallest to largest as for eigenvalues of symmetric matrices. In this exercise, $\|x\|$ means the 2 norm, which is $(x^t x)^{\frac{1}{2}}$.

   (a) Consider the quotient

$$R(x) = \frac{\|Ax\|^2}{\|x\|^2} \, .$$

Show that the maximum of R is attained. Let $v_1 \in \mathbb{R}^n$ and σ_1 be defined by

$$\|v_1\| = 1 \ , \ \ \sigma_1 = \max_{x \neq 0} R(x) \ , \ \ \sigma_1 = R(v_1) \ .$$

v_1 is the first *right singular vector* and σ_1 is the first *singular value*. Define $u_1 \in \mathbb{R}^m$ by $Av_1 = \sigma_1 u_1$, so that $\|u_1\| = 1$. u_1 is the first *left singular vector*.

(b) (This is the main point.) Show that if $v_1^t x = 0$, then $u_1^t A x = 0$. *Hint:* The week 6 class notes do this for symmetric matrices. The argument is similar.

(c) Define σ_2 and v_2 and u_2 by

$$\sigma_2 = \max_{v_1^t x = 0, x \neq 0} R(x) \ , \ \ \sigma_2 = R(v_2) \ , \ \ v_1^t v_2 = 0 \ , \ \ \|v_1\| = 1 \ , \ \ Av_2 = \sigma_2 u_2 \ .$$

Show that $\sigma_2 \leq \sigma_1$ and $\sigma_2 \geq 0$.

(d) Suppose A is a "tall thin" matrix, which is a matrix with $m \geq n$. Show that there is an $m \times m$ orthogonal matrix V whose columns are the right singular vectors v_k so that $AV = \widetilde{U}\widetilde{\Sigma}$. There $\widetilde{U}$ is an $m \times n$ matrix with ortho-normal columns and $\widetilde{\Sigma}$ is diagonal $m \times m$ matrix with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m$ on the diagonal. It is common to write the matrix formula in the equivalent form $A = \widetilde{U}\widetilde{\Sigma}V^t$. It is common to *complete* $\widetilde{U}$ to have n columns by adding $n-m$ additional orthonormal columns, all of them orthogonal to the columns of $\widetilde{U}$. This $n \times n$ orthogonal matrix is called U. It is common to expand $\widetilde{\Sigma}$ to have the same shape as A by adding $n - m$ rows of zeros below the diagonal part. This matrix is called Σ. Show that the expanded matrices also satisfy the formula $A = U\Sigma V^t$. This formula, and the matrices U, Σ, and V are the *singular value decomposition* of A.

(e) Show that

$$\sum_{j=1}^{m} \sigma_j^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2 = \mathrm{Tr}(A^t A) = \mathrm{Tr}(AA^t) \ .$$

(f) Let $C = A^t A$. Show that the eigenvalues of C are the squares of the singular values of A. With our orderings, that is $\lambda_{m-j+1} = \sigma_j^2$. Show that v_j are the corresponding eigenvectors. *Hint*: Relate $R(x)$ to the Rayleigh quotient of C.

(g) Suppose you want to predict the columns of A using $k < n$ vectors. Let a_i be column i of A. The predictor is

$$\widehat{a_i} = \sum_{j=1}^{k} m_{ij} w_j \ .$$

The criterion is to minimize the sum of squares prediction error

$$S^2 = \sum_{i=1}^{n} \|a_i - \widehat{a}_i\|^2 \ .$$

Show that S^2 is minimized using the first k left singular vectors $u_1, \ldots, u_k$. Find a formula for S^2 in terms of singular values.

Computing exercise

This is an exercise in simulation and the auto-covariance function of a stochastic process. Let Y_k be a sequence of random variables. These may be "observations" of a continuous time stochastic process at times $t_k = k\Delta t$. The lag j auto-covariance function is

$$C_j = \mathrm{cov}_p(Y_k, Y_{k+j}) \ .$$

The subscript p means that Y_k is in the steady state distribution so C_j does not depend on k. If we have a simulated or data time series of length m, we can estimate the auto-covariance function using

$$\widehat{C}_j = \frac{1}{m-b-j} \sum_{k=b+1}^{m-j} (Y_k - \overline{Y})(Y_{k+j} - \overline{Y}) \ .$$

Here b is the "burn-in" time, which is values of k so small that Y_k is not yet in its steady state distribution, and $\overline{Y}$ is the average of the time series. The denominator in front is the number of terms in the sum. It is common to use $m-b-j-1$, there should be enough data that subtracting 1 makes a negligible difference.

1. Consider the Ornstein Uhlenbeck process with $\gamma = 1$ and $\sigma = 1$. This is $dX_t = -X_t dt + dW_t$. Choose a small Δt and simulate the process using the Euler method as on an earlier assignment. Start with $X_0 = 0$ and simulate up to a large time T. Make T and Δt parameters in the code that you can "play with". Make sure Δt is printed on the plot and t is marked on the horizontal axis. Make a plot of the observed sequence $Y_k = X_{t_k}$. For small Δt, this should look like a mess, but you will see that it settles into a steady state mess after not too much time has passed.

2. Estimate and plot the auto-covariance function of this process as a function of t on the horizontal axis. Calculate (analytically) and plot the exact auto-covariance function from theory. Do several experiments that show that estimated auto-covariance function is more accurate when T is larger. Choose a b so that for $k > b$, Y_k seems to be more or less in the steady state. You can find this from the plot of part (1).

   (a) Computing the sums in "scalar" Python (one number at a time as: `for in in range ... for j in range ... sum += Y[k]*Y[k+j]`)

can be slow. Try to learn vector Python, which is one instruction telling it to do all the products and sums. It is something like: `np.sum(Y[b:m-j]*Y[b+j:m]` (this is not quite right).

(b) Be careful with ranges of indices. Python uses "half open" intervals that include the left endpoint but not the right endpoint, so `Y[0:m]` is the sequence running from $k = 0$ to $k = m-1$. We want sequences that run from $k = 0$ to $k = m$, which is `Y[0:(m+1)]`. The parentheses are necessary. Leave them out (`Y[0:m+1]`) to see why.

(c) You may not want Δt to be too small, but there will be noticeable Δt errors if unless Δt is small enough. Do not start with $\Delta t = .0001$. Experiment to what size you need.

(d) It can take a long time to compute the whole auto-covariance function. You will see that after a certain point the computed function is all noise. It is probably not necessary to compute it beyond $t = t_f = 3$ or so. You can experiment to find a good t_f.

3. A *reflecting Brownian motion* is a process $dX_t = dW_t + dL_t - dM_t$. The functions L_t and M_t are both monotone increasing, so $L_{t_2} \geq L_{t_1}$ if $t_2 > t_1$, and similarly for M_t. They are *boundary controls* that keep X_t in the interval $[a, b]$ and are active only at the endpoints. The force dL_t is "active" only when $X_t = a$ and prevents X from going below a. L_t being monotone means that dL pushes only to the right. Similarly, M_t is active only when $X_t = b$ and $-dM_t$ pushes X_t to the left. We say L (or M) is active if it is not constant. So $L_{t_2} = L_{t_1}$ if $X_t > a$ for all $t \in [t_1, t_2]$. Since L_t is monotone, this implies that L is constant on any interval of time in which X_t does not touch a. As long as $a < X_t < b$, this behaves exactly like Brownian motion because $dL = 0$ and $dM = 0$.

   Boundary control has applications in finance. For example, a country might want to let its currency fluctuate within a certain range but not let is go below a chosen minimum or above a chosen maximum. Such a country would trade in its currency only to prevent it from going outside the range. Boundary controls happen in engineering applications too.

   Here is one way to simulate a reflecting Brownian motion. You have Δt and $t_k = k\Delta t$. Suppose $X_{t_k} \geq a$, and L_{t_k} has been computed. Define ΔL_k to be the smallest non-negative number so that $X_{t_{k+1}} = X_{t_k} + \Delta W_k + \Delta L_k \geq 0$. Usually $\Delta L_k = 0$, but sometimes it is necessary to prevent $X_{t_{k+1}} < a$.

   (a) Simulate and plot some reflecting Brownian motion paths up to a time when they hit both boundaries at least a little. Simulate up to some time T that is not too large and with a Δt that you cannot see easily in plots. Make plots of L_t and M_t to see that they are constant except where X_t is at (near, in simulation) one of the boundaries.

   (b) Simulate for a longer time until the process reaches steady state, then compute the auto-covariance function. This may take quite a

bit of computing, so you should be willing to let the computer work overnight.

(c) Plot the auto-covariance function (over a range where it does not seem to be noise) in a semi-log plot (log on the vertical axis, linear on the horizontal t axis. This will be a straight line if $C(t)$ is a single exponential as it is for part 2. Is that true here?