

Computational Methods in Economics and Finance

Quiz 1

Please provide your answer in the form of a .ZIP file that includes all of the MATLAB code (scripts and functions) that you write and a document that describes your procedures, your code and your interpretations of the results. You will be evaluated on the efficiency of your code, on the quality of the documentation and on the thoroughness of replies.

1.1. Consider the process

$$dS = \alpha(\mu - S)dt + \sigma SdW$$

This process can be simulated using Euler's method by the iterative algorithm

$$S_{t+\Delta} = S_t + \alpha(\mu - S_t)\Delta + \sigma S_t\sqrt{\Delta}e_t$$

where $e_t \sim i.i.d. N(0, 1)$.

Use simulated paths to approximate the expected time it takes to first move from $S = S_0$ to $S = S_1$ (this is known as the expected first passage time).

Do this using the following parameter values: $\alpha = 0.25$, $\mu = 1$, $\sigma = 0.2$, $S_0 = 2$ and $S_1 = 1.5$. Also use 10000 paths and $\Delta = 0.01$ (hint: when debugging your code, use fewer paths and/or a larger Δ . Once you are sure the code is running correctly use the given parameter values.)

Consider the closely related process

$$dS = \alpha(\mu - S)dt + \sigma dW$$

Suppose that you knew, by other means, that the expected time for this process, starting at $S_0 = 2$ to reach $S_1 = 1.5$ is 2.4452. Use the control variate technique to improve the approximation for the original process.

Discuss (but only implement if you want to) how you would determine if the control variate method improves on the original approximation you obtained. Incidentally the correct answer for the original process is 2.0366.

1.2. It is well known that

$$\exp(x) = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

This fact can be used to compute the exponential function using only arithmetic operations (addition, subtraction, multiplication and division). Of course, one cannot compute the infinite sum but we can compute a sum over i until the sum stops

changing because the i th term is so small that, in computer arithmetic, it is like adding 0 to the sum.

There are numerous ways to actually implement an algorithm that uses this expression. Write an algorithm that works for all values of x , including negative ones and ones that are large in absolute value (the largest number for which $\exp(x)$ is less than infinity is about 709.7827 in MATLAB; this can be obtained using `log(realmax)`). Be sure your code executes efficiently and discuss why you have made the choices you did. Your answer should be implemented as a function that takes the value x and returns $\exp(x)$. You should also include a script file that demonstrates that the function performs well.

- 1.3. Solving a linear equation system $Ax = b$, where a is $n \times n$, can generally be done using direct methods when n is small. As n grows in size, the use of sparse matrices and iterative methods may be more advantageous.

Consider two types of matrices. The first is tridiagonal and is formed using the code

```
A=speye(n)+spdiags(rand(n,3)/2,[-1 0 1],n,n);
```

The second has a non-zero diagonal and randomly spaced non-zero non-diagonal elements and is formed using

```
A=speye(n)+sprand(n,n,5/n);A=speye(n)+diagmult(1./sum(A,2),A);
```

Evaluate the use of the “\” operator (which uses the LU decomposition), the MATLAB `bicg` function and the Gauss-Seidel algorithm discussed in section 2.6 of the text (and implemented as `gseidel` in the CompEcon Toolbox, available on my website). For the latter two approaches, set the number of iterations high enough so convergence is assured and set the convergence tolerances around 10^{-15} and 10^{-14} , respectively.

Compare these three algorithms for values on n equal to 10, 100, 1000 and 5000. Draw conclusions and discuss under what circumstances (if any) one use would each of these algorithms. Note that I have defined both A matrices in a special way that ensures that they are diagonally dominant, meaning that in each row the diagonal elements are larger in absolute value than the sum of the absolute values of the non-diagonal elements. This is a sufficient condition to ensure that the Gauss-Seidel algorithm converges.

Note that if you use the toolbox function `gseidel` you can set the maximum iterations and the convergence tolerance using

```
optset('gseidel','maxit',1000)
```

and

```
optset('gseidel','tol',1e-14)
```

Also you can time algorithms using MATLAB's `tic` and `toc` functions.