QF607 2024 Exam Cheat Sheet

Fixed Point Representation

To define a fixed point type, we need two parameters:

- width of the number representation
- binary point position
- We use `fixed<w, b>` to denote a fixed point type with `w` width and binary point position at `b` counting from 0 (the least significant bit).
- For example, `fixed<8, 3>` denotes a 8 bit fixed point number, of which 3 right most bits are fractional. Therefore, the number `00010110` represents:

$$00010.110_2 = 1 \times 2^1 + 1 \times 2^{-1} + 1 \times 2^{-2} = 2.75 \tag{1}$$

- The same bit patter `00010110` represents a different number if it is of a different type, say `fixed<8, 5>`:

$$000.10110_2 = 1 \times 2^{-1} + 1 \times 2^{-3} + 1 \times 2^{-4} = 0.6875 \tag{2}$$

- Two's-complement: the value x of a `fixed<w, b>` number $b_{w-1}b_{w-2}\dots b_0$ is given by:

$$x = -b_{w-1}2^{w-1-b} + \sum_{i=0}^{w-2} b_i 2^{i-b} \tag{3}$$

Jarrow-Rudd Risk Neutral Model (JRRN)

JRRN binomial tree model:

$$\begin{cases} p &= \frac{e^{r\Delta t} - d}{u-d} \\ u &= e^{(r-\frac{\sigma^2}{2})\Delta t + \sigma\sqrt{\Delta t}} \\ d &= e^{(r-\frac{\sigma^2}{2})\Delta t - \sigma\sqrt{\Delta t}} \end{cases} \tag{4}$$

Bisection Method

Start with $[a, b]$ that brackets the root, cut it by half, then see which half contains the root.

Algorithm 1 root = rfbisect(f, a, b, tol)

Require: $f(a)f(b) < 0$, f is continuous, , tolerance tol

while $(b-a)/2 > tol$ **do**
  $c \leftarrow \frac{a+b}{2}$
  if $f(c) == 0$ **then**
    return c;
  else if $f(a)f(c) < 0$ **then**
    $b \leftarrow c$
  else
    $a \leftarrow c$
  end if
end while

Secant Method

Start with x_1 and x_2, connect $(x_1, f(x_1))$ and $(x_2, f(x_2))$, find the intersection with x axis at x_3, then continue with x_2 and x_3

Algorithm 2 root = rfsecant(f, x_1, x_2, tol)

Require: f is continuous, tolerance tol

while $abs(x_2 - x_1) > tol$ **do**
  $x_3 \leftarrow \frac{x_1 f(x_2) - x_2 f(x_1)}{f(x_2) - f(x_1)}$
  if $f(x_3) == 0$ **then**
    return x_3;
  else
    $x_1 \leftarrow x_2$
    $x_2 \leftarrow x_3$
  end if
end while

False Position Method (Regula Falsi)

Start with $[a, b]$ that brackets the root, connect $(a, f(a))$ and $(b, f(b))$, find the intersection with x axis at c, then continue with a or b, depending on who brackets the root with c

Algorithm 3 root = rffalsi(f, a, b, tol)

Require: $f(a)f(b) < 0$, f is continuous

while $b - a > tol$ **do**
  $c \leftarrow \frac{af(b) - bf(a)}{f(b) - f(a)}$
  if $f(c) == 0$ **then**
    return c;
  else if $f(a)f(c) < 0$ **then**
    $b \leftarrow c$
  else
    $a \leftarrow c$
  end if
end while

Monte Carlo As Integrator

- An integral $\int_0^1 f(x)dx$ is an expectation $E[f(x)]$ with uniformly distributed from 0 to 1 ($\mathcal{U}(0,1)$):

$$\int_0^1 f(x)dx = \int_0^1 f(x)p(x)dx = E[f(x)] \tag{5}$$

  because $p(x) = 1$ for $\mathcal{U}(0,1)$

- To integrate the interval $[a, b]$

$$E[f(x)] = \int_a^b f(x)p(x)dx = \int_a^b f(x)\frac{1}{b-a}dx = \frac{1}{b-a}\int_a^b f(x)dx \tag{6}$$

  So $\int_a^b f(x)dx = (b-a)E[f(x)]$ for $x \sim \mathcal{U}(a, b)$

Control Variates

- We are trying to estimate the expectation of a function $h(\vec{X})$
- If we know the expectation of a function $g(\vec{X})$ analytically, we can use the estimator:

$$E[h(\vec{X})] \approx \frac{1}{n}\sum_{i=1}^{n}\Big(h(\vec{X}_i) + \beta(g^* - g(\vec{X}_i))\Big) \tag{7}$$

  where g^* is the known expectation of $g(\vec{X})$ and β is a parameter.
- The variance of the samples is

$$Var[h] + \beta^2 Var[g] - 2\beta Cov[h, g] \tag{8}$$

- Taking the first derivative w.r.t β, the variance is minimized for $\beta = \dfrac{Cov[h,g]}{Var[g]}$ (can be estimated using simulation samples)
- The minimized variance is

$$Var[h] - \frac{Cov[h,g]^2}{Var[g]} = Var[h](1-\rho^2) \tag{9}$$

- The higher correlation g and h is, the more effective the technique is.

Monte Carlo Algorithm

Algorithm 4 $\hat{\mu}_n = \text{MC}\ (h)$

```
s = 0
for i = 1 to n do
    Generate X_i
    h_i = h(X_i)
    s += h_i
end for
μ̂_n = s / n
return μ̂_n
```

Euler Discretization Scheme

- Using time step Δt, the Euler discretization scheme approximates

$$\int_t^{t+\Delta t} a(X(u))du = a(X(t))\Delta t \tag{10}$$

$$\int_t^{t+\Delta t} b(X(u))dW_u = b(X(t))\Delta W = b(X(t))\sqrt{\Delta t}Z \tag{11}$$

  where $Z \sim N(0,1)$
- The stepwise induction is

$$X(t+\Delta t) = X(t) + a(X(t))\Delta t + b(X(t))\sqrt{\Delta t}Z \tag{12}$$

- The truncation error of the drift term is $O(\Delta t)$
- The truncation error of the diffusion term is $O(\Delta W)$, i.e., $O(\sqrt{\Delta t})$
- The overall convergence order of Euler scheme is therefore $O(\sqrt{\Delta t})$

Cholesky Decomposition

- If a matrix is symmetric and positive definite, a special LU decomposition — Cholesky decomposition is faster than the other LU decomposition and satisfies the property that:

$$\vec{C} = \vec{L}\vec{L}^\top \tag{13}$$

  and $\vec{L}$ is a lower triangular matrix.
- The requirement that the Cholesky decomposition exists is the matrix $\vec{C}$ is symmetric positive definite. This is true if $\rho \neq \pm 1$:

$$\vec{x}^\top C \vec{x} = \frac{1}{n}\underbrace{\vec{x}^\top \begin{bmatrix}\vec{\xi}_1\\ \vec{\xi}_2\end{bmatrix}}_{(\vec{y}^\top)_{1\times n}} \times \underbrace{\begin{bmatrix}\vec{\xi}_1^\top, & \vec{\xi}_2^\top\end{bmatrix}\vec{x}}_{\vec{y}_{n\times 1}} = \frac{\|\ \vec{y}\ \|}{n} \geq 0 \tag{14}$$

  So (15) exists by definition.
- Now to generate two correlated Brownian motions we can
  1. Generate two independent standard normal samples $\vec{\zeta}_1$ and $\vec{\zeta}_2$
  2. Decompose the correlation matrix using Cholesky: $\vec{C} = \vec{L}\vec{L}^\top$
  3. Generate the correlated normal variates by: $\begin{bmatrix}\vec{\xi}_1\\ \vec{\xi}_2\end{bmatrix} = \vec{L}\begin{bmatrix}\vec{\zeta}_1\\ \vec{\zeta}_2\end{bmatrix}$

  And this routine works in general, for m random variates.
- The Cholesky decomposition algorithm is straight-forward:

$$\vec{C} = \vec{L}\vec{L}^\top = \begin{pmatrix} L_{11} & 0 & 0\\ L_{21} & L_{22} & 0\\ L_{31} & L_{23} & L_{33}\end{pmatrix}\begin{pmatrix} L_{11} & L_{21} & L_{31}\\ 0 & L_{22} & L_{23}\\ 0 & 0 & L_{33}\end{pmatrix} \tag{15}$$

$$= \begin{pmatrix} L_{11}^2 & & \text{(symmetric)}\\ L_{21}L_{11} & L_{21}^2 + L_{22}^2 & \\ L_{31}L_{11} & L_{31}L_{21} + L_{32}L_{22} & \sum_{i=1}^3 L_{3i}^2\end{pmatrix} \tag{16}$$

  Just need to start from the top-left corner and progressively solve for each L_{ij}

Euler Scheme

To discretize the time derivatives, the simplest way is to use Euler scheme. But there are two ways to do that:

- Left difference (Explicit)

$$\frac{\partial V_{i,j}}{\partial t} \approx \frac{V_{i,j} - V_{i,j-1}}{\Delta t} \tag{17}$$

- Right difference (Implicit):

$$\frac{\partial V_{i,j}}{\partial t} \approx \frac{V_{i,j+1} - V_{i,j}}{\Delta t} \tag{18}$$

Substituting the left difference to the system of ODE we obtained by discretizing the spot dimension, we get the **explicit** Euler scheme:

$$\frac{1}{\Delta t}\begin{pmatrix} \Delta t \cdot b_{(0,j)}\\ V_{(1,j)} - V_{(1,j-1)}\\ V_{(2,j)} - V_{(2,j-1)}\\ \vdots\\ V_{(N_S-2,j)} - V_{(N_S-2,j-1)}\\ \Delta t \cdot b_{(N_S-1,j)}\end{pmatrix} = \vec{M}\begin{pmatrix} V_{(0,j)}\\ V_{(1,j)}\\ V_{(2,j)}\\ \vdots\\ V_{(N_S-2,j)}\\ V_{(N_S-1,j)}\end{pmatrix} \tag{19}$$

The reason it is explicit is that we know our V from the back. The linear system is an explicit function from $\vec{V}_j$ to $\vec{V}_{j-1}$.