FMO6 — Web: https://tinyurl.com/ycaloqk6 Polls: https://pollev.com/johnarmstron561

Lecture 3

Dr John Armstrong

King's College London

August 22, 2020

- We have learned how to use Matlab
  - What is the difference between `*` and `.*`?
  - If `v = [8 -3 2 ]` what is `v>0`?
- We have learned to write code as small functions
- We have learned how to write unit tests
  - What is a unit test?
- We have learned how to integrate using the rectangle rule (a.k.a. midpoint rule).
  - What is the rectangle rule?

Integration using rectangle rule

Algorithm (Rectangle Rule)

Let $f : [a, b] \longrightarrow \mathbb{R}$ be a function we wish to integrate. Choose a number of points N and define:

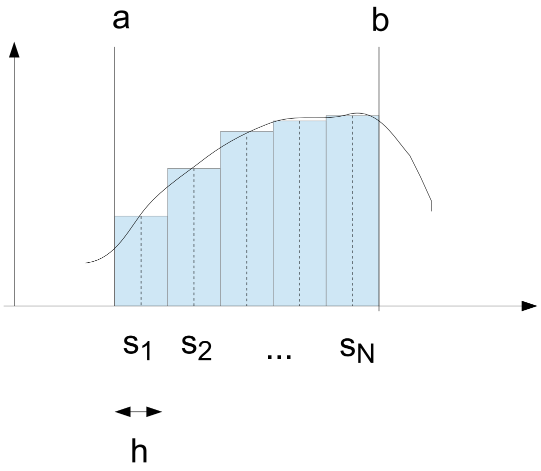
$$h = \frac{b-a}{N}$$

$$s_m = a + (m - \frac{1}{2})h$$

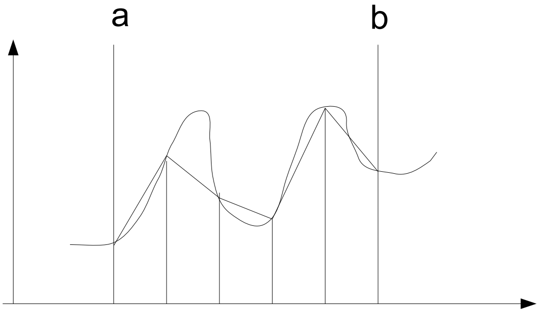
$$R = h \sum_{i=1}^{N} f(s_m)$$

Then R is the rectangle rule estimate for $\int_a^b f(t)\,\mathrm{d}t$.

Rectangle Rule



Trapezium Rule



Algorithm (Trapezium Rule)

Let $f : [a,b] \longrightarrow \mathbb{R}$ be a function we wish to integrate. Choose a number of panels n and define:

$$\begin{aligned} h &= \frac{b-a}{n} \\ s_m &= a + mh \\ T &= \frac{h}{2}(f(s_0) + 2f(s_1) + 2f(s_2) + \ldots + 2f(s_{n-1}) + f(s_n)) \end{aligned}$$

Then T is the trapezium rule estimate for $\int_a^b f(t)\,\mathrm{d}t$. The number of integration points is $N = n+1$.

- The next algorithm to consider is Simpson's rule where one approximates the function as being piecewise quadratic.
- Let f be a quadratic function then

$$\int_{-h}^{h} f(t)\mathrm{d}t = \frac{h}{3}(f(-h) + 4f(0) + f(h))$$

- To see this:
  - Observe that both sides are linear in f.
  - Observe that for constant f, the result is true since $2 = \frac{1}{3}(1 + 4 + 1)$
  - Observe that for odd functions, the formula will always hold. In particular it holds for $f(x) = x$. So by linearity it holds for all linear functions.
  - Observe that the result is true for $f(x) = x^2$. So by linearity it holds for all quadratic functions.
- More surprisingly the result is true if f is cubic. This follows from the fact that x^3 is an odd function and by linearity.

Simpson's rule

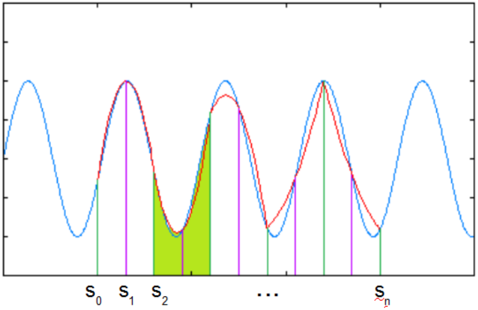


Figure: Simpson's Rule

Algorithm (Simpson's Rule)

Let $f : [a, b] \longrightarrow \mathbb{R}$ be a function we wish to integrate. Choose an even *number of panels n and define:*

$$\begin{aligned} h &= \frac{b-a}{n} \\ s_m &= a + mh \\ S &= \frac{h}{3} \sum_{i=0}^{\frac{n}{2}-1} (f(s_{2i}) + 4f(s_{2i+1}) + f(s_{2i+2})) \\ &= \frac{h}{3}(f(s_0) + 4f(s_1) + 2f(s_2) + 4f(s_3) + 2f(s_4) + \dots \\ &\quad + 2f(s_{n-2}) + 4f(s_{n-1}) + f(s_n)) \end{aligned}$$

Then S is the Simpson's rule estimate for $\int_a^b f(t)\,\mathrm{d}t$. The number of integration points is $N = n + 1$.

Theorem

If f is four times differentiable with $|f^{(4)}| < K$ for some K, then the error in the Simpson's rule estimate can be bounded above by

$$\frac{c}{N^4}$$

where c is a constant depending upon a, b and K.

Lemma

If f is four times differentiable with $|f^{(4)}| < K$ for some K, then the error in the Simpson's rule estimate over an interval of length $2h$ with two steps can be bounded above by $c_1 h^5$ for some constant c_1 depending on K.

The theorem follows from the lemma because we're adding up $\frac{n}{2}$ copies of the 2-step Simpson's rule. Cumulative error is:

$$\frac{n}{2} c_1 h^5 = \frac{n}{2} c_1 h^5 = \frac{n}{2} c_1 \left(\frac{b-a}{n} \right)^5 \leq c \frac{1}{N^4}$$

Proof of Lemma

- By Taylor's theorem:

$$\begin{aligned} f(x) &= f(x_0) + (x-x_0)f'(x_0) + \frac{1}{2}(x-x_0)^2 f^{(2)}(x_0) + \ldots \\ &+ \frac{1}{3!}(x-x_0)^3 f^{(3)}(x_0) + \frac{1}{4!}(x-x_0)^4 f^{(4)}(\xi) \end{aligned}$$

  for some $\xi(x) \in [x_0, x]$.
- By construction Simpson's rule is exact for quadratic functions.
- Notice also that it is exact for cubic functions: this is because the integral of x^3 over $[-h, h]$ is zero as is the value given by Simpson's rule. By adding appropriate quadratic terms, any given cubic can be transformed to this special case.
- Therefore the error from Simpson's rule over an interval $[x_0, x_0 + 2h]$ is equal to the integral of the non-cubic term in the above expansion

$$\text{error} = \left| \int_{x_0}^{x_0+h} \frac{1}{4!}(x-x_0)^4 f^{(4)}(\xi(x))\mathrm{d}x \right| \leq \int_{x_0}^{x_0+h} \frac{1}{4!}(x-x_0)^4 K \mathrm{d}x = ch^5$$

Big O notation

Definition

Given a function $f : \mathbb{N} \to \mathbb{R}$ we write that a sequence $s_n = O(f(n))$ if $s_n \leq C|f(n)|$ for some constant.

Theorem

If f is four times differentiable with $|f^{(4)}| < K$ for some K, then the error in the Simpson's rule estimate is $O(N^{-4})$.

Theorem

If f is twice differentiable with $|f^{(2)}| < K$ for some K, then the error in the Trapezium rule estimate is $O(N^{-2})$. The same is true for the error in the Rectangle rule.

Adaptive methods

- Why distribute the points evenly? Maybe we'll get better convergence if we concentrate the points in areas where f is rapidly changing.
- At stage n suppose we have grid G_n. We subdivide to get G_{n+1}. We can check to find which splits were the most effective and use this to guide the next subdivision.
- (Adaptive methods are not examinable)

Gaussian integration

- All our formulae have been of the form
$$\int_a^b f \approx \sum_{i=1}^{N} w_i f(x_i)$$
for some weights $w_i \in \mathbb{R}$ and points $x_i \in [a, b]$.
- Gauss gave a recipe to find an integration rule which gives a perfect answer for polynomials of degree $2d - 1$ with only d points.
- Gauss found: (when $a = 0$, $b = 1$) choose the x_i to be the roots of the Legendre polynomials and the weights can be calculated in terms of the derivatives of the Legendre polynomials.
- (Gaussian quadrature is not examinable, sadly)

Practical methods

- Matlab has a function `integral` which you can use in practice.
- It uses an adaptive Gaussian quadrature strategy.
- Similar functions can be found in other scientific libraries e.g. the GNU scientific library for `C`.
- When using a general purpose integral routine, you can normally specify some key points on your function to help understand the shape and scale of your function. That's because if your function behaves in a very different way over some intervals than others, it is important that there are samples taken from all these areas.
- If there are singularities in your integrand, it can help to rewrite your function f as $f = f_1 + f_2$ where f_1 is non singular and f_2 is singular but analytically integrable.

The curse of dimensionality

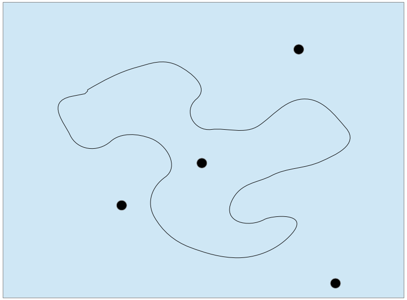
- By induction we can estimate d-dimensional integrals by applying one of the rules above to an $d-1$-dimensional integral

$$\int\int f(x_1, x_2, \ldots x_d)\,\mathrm{d}x_1\,\mathrm{d}x_2, \ldots \mathrm{d}x_d$$
$$= \int \left(\int f(x_2, \ldots x_d)\mathrm{d}x_2, \ldots \mathrm{d}x_d \right) \mathrm{d}x_1$$

- Suppose we use N grid points for each one dimensional integration.
- Let n_d be the number of function evaluations required to compute a d-dimensional integral in this way.
- $n_1 = N$, $n_d = N * n_{d-1}$. Therefore $n_d = N^d$.
- e.g. for a 100-dimensional problem with 10 grid points we need 1 googol function evaluations (1 googol=10^{100}).

Throwing darts

If the area of the rectangle is $1m^2$
I estimate the area of the blob is $0.25m^2$



Throwing darts

To find the area of a given shape Σ

- Bound it by a rectangle of known area A.
- Throw n darts randomly (i.e. uniformly distribution) into the rectangle
- Count the number of darts, n_Σ, that land in the shape Σ.
- The area is approximately

$$\frac{n_\Sigma}{n} A$$

Monte Carlo integration

Algorithm (Monte Carlo Integration)

Let $f : [a,b] \longrightarrow \mathbb{R}$. Estimate the integral of f by generating N random numbers x_m which are uniformly distributed in the interval $[a,b]$. The Monte Carlo estimate for the integral is:

$$\frac{1}{N}(b-a)\sum_{i=1}^{N} f(x_m)$$

Theorem

By the central limit theorem, under reasonable conditions on f, the estimate converges at a rate of $c\frac{1}{\sqrt{N}}$.

Multi-dimensional Monte Carlo integration

Algorithm (Monte Carlo Integration)

Let $f : [0,1]^d \longrightarrow \mathbb{R}$. Estimate the integral of f by generating N random numbers x_m which are uniformly distributed in the hypercube $[0,1]^d$. The Monte Carlo estimate for the integral is:

$$\frac{1}{N}\sum_{i=1}^{N} f(x_m)$$

Theorem

By the central limit theorem, under reasonable conditions on f, the estimate converges at a rate of $O(\frac{1}{\sqrt{N}})$.

Monte Carlo method

- Experience suggests that dimension 3 or 4 is the approximate dimension where Monte Carlo integration begins to out-perform methods based on 1 dimensional integration rules.
- Since you cannot generate true random numbers on a computer you have to make do with pseudo-random numbers. We will assume in this course that the pseudo-random number generator in MATLAB is up to the job.
- In practice you should check how many random numbers you will need and make sure that you are using a generator that guarantees to provide high quality pseudo-randomness for that many numbers.
- Monte Carlo doesn't completely get rid of curse of dimensionality. The error is less than $cN^{-\frac{1}{2}}$ but c can grow to be enormous for high dimensions.

Indefinite integrals

- By performing a substitution one can transform an infinite integral to an integral over an open interval.
- Our integration rules (other than the rectangle rule) use the end points of the integral, so these limits had better exist after the substitution.
- The choice of substitution makes a big difference to the performance of the method. This doesn't just apply to infinite integrals.
- When integrating singular functions, it is a good practice to subtract the singularity from the integral. For example if the singularity can be approximated by the term $1/x^{1/2}$, subtract this off and integrate it separately analytically.

The rectangle rule

```
function ret = integrateByRectangleRule( f, a, b, N )
h = (b-a)/N;
s = a + h*((1:N) - 0.5);
total = 0;
for x=s
    total = total+f(x);
end
ret = h*total;
end
```

The rectangle rule

```matlab
function ret = integrateByTrapeziumRule( f, a, b, N )
n = N-1; % N = number of grid points, n=number of panels
h = (b-a)/n;
s = a + h*(1:n-1);
total = f(a)+f(b);
for x=s
    total = total+2*f(x);
end
ret = 0.5*h*total;
end
```

Simpson's rule

Exercise

Indefinite integration

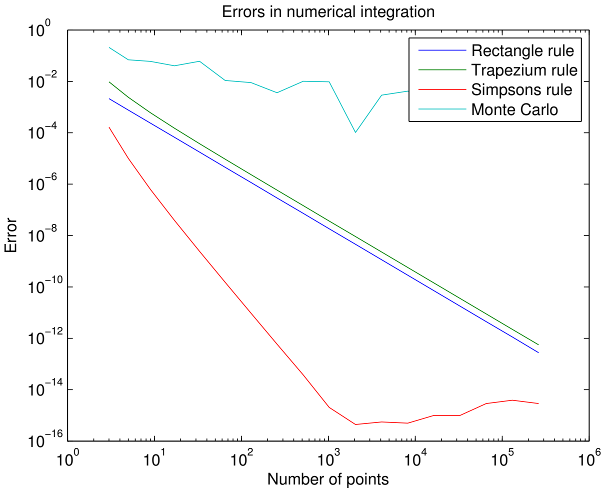
```
function ret = integrateToInfinity( f, x, N, integrationRule)
% Performs a substitution to change
% the infinite integral to a finite integral.
if nargin<4
   integrationRule=@integrateByRectangleRule;
end

function r = transformedFunction( s )
    r = s^(-2) * f( x - 1 + 1/s );
    if (isnan(r))
        r = 0.0;
    end
end

ret=integrationRule( @transformedFunction, 0, 1, N );
end
```

- You can write a function that provides default values for arguments using `nargin`. This contains the number of parameters the user has actually passed in. So if it is lower than you expect, supply default values.
- Use the function `isnan` to see if a calculation has evaluated to "Not a number".
- Our `integrateToInfinity` uses the rectangle rule by default and assumes that the limit of the transformed integral is 0.

Question

- What does a log-log plot of $y = cx^n$ look like?
- How can you tell if $f(n) = O(x^n)$ from a log-log plot of f against n?

Calculating $\int_0^1 \sin(x)\,\mathrm{d}x$



Interpretation

- The gradients of the log-log plot show the rate of the convergence.
- Simpson's rule has gradient -4, Trapezium rule has gradient -2, Rectangle rule has gradient -2 as expected.
- Monte Carlo is not inconsistent with the gradient $-\frac{1}{2}$ prediction. We could perform repeated runs to find the average Monte Carlo error.
- After a certain point the error of Simpson's rule increases. This is due to accumulating rounding errors. These grow with slope $1/2$ consistent with the central limit theorem: we're adding independent errors.
- Simpson's rule approaches machine precision accuracy.

The code

```
function plotErrors()
% Plots the error of computing integral of sin from 0 to 1
points=1:18;
NValues = 2.^points + 1; % Exercise: explain this line
answer = -cos(2)+cos(0);

errorR = zeros( 1, length(NValues));
errorT = zeros( 1, length(NValues));
errorS = zeros( 1, length(NValues));
errorM = zeros( 1, length(NValues));
for i=1:length(NValues)
    N = NValues(i);
    fprintf('Running calculation with %d points\n', N);
    errorR(i) = abs(integrateByRectangleRule(@sin,0,2,N) - answer);
    errorT(i) = abs(integrateByTrapeziumRule(@sin,0,2,N) - answer);
    errorS(i) = abs(integrateBySimpsonsRule(@sin,0,2,N) - answer);
    errorM(i) = abs(integrateByMonteCarlo(@sin,0,2,N) - answer);
end

figure();
loglog( NValues, errorR, NValues, errorT, NValues, errorS, NValues, errorM );
title('Errors in numerical integration');
```

Matlab functions for plotting

- `figure`. Start a new drawing window.
- `plot(x1,y1,x2,y2,...,xn,yn)`. Plot a line chart of the vector y_i against the vector x_i. So this will contain n lines.
- `loglog`. Plot a log-log plot. Otherwise used just the same as `plot`.
- `xlabel`, `ylabel`, `title`. Self explanatory.
- `legend`. Provide names for each series in the plot. For example:

```
legend('Series1','Series2','Series3');
```

Question

★ How accurate do we need to be when pricing derivatives?

How much accuracy?

- There is a bid-ask spread on the stock price.
- The model parameters used in a calculation is found by finding the best fit to a the smile or the best fit to historical data.
- No need for machine precision typically.
- Calculations of unlikely events or across entire portfolios with much hedging are sensitive to rounding errors.

Risk neutral pricing

- The risk neutral price of a derivative is its discounted expected value under a given risk neutral measure $\mathbb{Q}$.
- For the Black Scholes model you are given the price process in the physical measure $\mathbb{P}$ and can deduce the process in $\mathbb{Q}$ (by Girsanov's theorem)
- For more complex models, one often simply chooses a $\mathbb{Q}$ measure process (by calibrating to the market) and ignores the $\mathbb{P}$ measure entirely.

The Big Fight

Example

Wladimir Klitschko is due to fight Dr John Armstrong at the O_2 Arena. A bookmaker is offering odds of 1000 to 1 on Dr Armstrong winning. What can you say about the bookmaker's odds on Klitschko? What does this tell us about the probability of Dr Armstrong winning?

The Big Fight

Solution

- *The odds on Klitschko will be* 1 *to* 1000 *or less otherwise you can arbitrage against the bookmaker.*
- *This tells us nothing about the probability of Armstrong winning, only about market prices for the bet. People may be willing to bet on Armstrong at* 1000 *to* 1 *even though the probability of him winning is probably somewhat less than this. After all you might win* \$1000 *and it only costs* \$1*.*
- *The risk neutral probability of Armstrong winning is* $1/1001$ *and the risk neutral probability of Armstrong losing is* $1000/1001$*.*
- $\mathbb{Q}$ *is determined by matching trades.*
- *The bookmaker knows* $\mathbb{Q}$ *but has no opinion about* $\mathbb{P}$*.*

Risk neutral pricing and integration

- In risk neutral pricing the price is the discounted $\mathbb{Q}$-expected payoff of the derivative.
- For a derivative with payoff determined entirely by the stock price S at time T we have, by definition of expectation:

$$E_{\mathbb{Q}}(\text{payoff}) = \int_{\mathbb{R}} \text{payoff}(S_T) \times \mathbb{Q}\text{-probability-density}(S)\,\mathrm{d}S$$

- Write q for the probability density function of S at time T. Write $P(S)$ for the payoff given the final stock price.

$$\begin{aligned} \text{price} &= e^{-rT} E_{\mathbb{Q}}(\text{payoff}) \\ &= e^{-rT} \int_{\mathbb{R}} P(S) q(S)\,\mathrm{d}S \end{aligned}$$

- So we can price such a derivative by integration once we know the p.d.f. q.

The Black Scholes Model, $\mathbb{P}$ measure

Measure	$\mathbb{P}$
Price process S_t:	$\mathrm{d}S_t = S_t(\mu\,\mathrm{d}t + \sigma\,\mathrm{d}W_t)$
Ito's Lemma $\Downarrow$	
Log price process z_t:	$\mathrm{d}z_t = (\mu - \frac{1}{2}\sigma^2)\,\mathrm{d}t + \sigma\,\mathrm{d}W_t$
Definition of s.d.e.	
and W_t $\Downarrow$	
Distribution of z_T:	$\mathcal{N}(z_0 + (\mu - \frac{1}{2}\sigma^2)T, \sigma\sqrt{T})$
Definition of $\ln\mathcal{N}$ $\Downarrow$	
Distribution of S_T:	$\ln\mathcal{N}(z_0 + (\mu - \frac{1}{2}\sigma^2)T, \sigma\sqrt{T})$
Substitute $z = \ln S$	
into $\int$ of normal p.d.f. $\Downarrow$	
p.d.f. of S_T:	$\frac{1}{S\sigma\sqrt{2\pi T}} \exp\left(-\frac{(\ln(S/S_0)-(\mu-\frac{1}{2}\sigma^2)T)^2}{2T\sigma^2}\right)$
Integrate $\Downarrow$	
Mean of S_T:	$\exp(z_0 + \mu T) = S_0\exp(\mu T)$

Log normal p.d.f.

To derive the p.d.f. of the log normal distribution, suppose that $x \sim \mathcal{N}(\alpha, \beta)$. So

$$\begin{aligned} P(x \leq t) &= \int_{-\infty}^{t} \frac{1}{\beta\sqrt{2\pi}} \exp\left(-\frac{(x-\alpha)^2}{2\beta^2}\right) \mathrm{d}x \\ &= \int_{0}^{e^t} \frac{1}{X\beta\sqrt{2\pi}} \exp\left(-\frac{(\ln X-\alpha)^2}{2\beta^2}\right) \mathrm{d}X \end{aligned}$$

Here we've made the substitution $x = \ln(X)$. We also have:

$$P(x \leq t) = P(\ln(X) \leq t) = P(X \leq e^t)$$

So the p.d.f. of the log normal distribution is:

$$\frac{1}{X\beta\sqrt{2\pi}} \exp\left(-\frac{(\ln X-\alpha)^2}{2\beta^2}\right)$$

The risk neutral measure

- By Girsanov's theorem, if we can change the drift term in our price process by changing to an equivalent measure.
- From our expression $E_{\mathbb{P}}(S_T) = S_0 \exp(\mu T)$ we see that the process with $\mu = r$ will be risk neutral. That is to say, if $\mu = r$, the expected stock price grows at the same rate as the zero coupon bond.
- Therefore by taking $\mu = r$ we obtain the process in the risk neutral measure $\mathbb{Q}$.

The pricing kernel

Theorem

The $\mathbb{Q}$-p.d.f. of S_T is:

$$q_T(S) = \frac{1}{S\sigma\sqrt{2\pi T}} \exp\left(-\frac{(\ln(S/S_0) - (r - \frac{1}{2}\sigma^2)T)^2}{2T\sigma^2} \right)$$

Given a European derivative that pays off $P(S)$ if the stock price at time T is S, the risk neutral price of this derivative is:

$$\text{price} = e^{-rT} \int_0^\infty P(S) q_T(S) \, \mathrm{d}S.$$

Note that the derivative is assumed to be European and not path dependent. The function $q_T(S)$ is often called the pricing kernel.

Pricing by numerical integration

Example

A dividend free stock follows the Black Scholes process with unknown drift and 10% volatility. The current stock price is \$100. You should assume that the risk free rate is 5% APR. Use numerical integration to find the risk neutral price of a 3 month European call option on the stock with strike \$105?

Solution - Units

- In finance time is measured in years, so $T = 0.25\,\text{years}$.
- Volatility is quoted as a percentage change over a year, so $\sigma = 0.1\,\text{years}^{-\frac{1}{2}}$.
- The risk free rate, r, used in the Black Scholes Formula is a continuously compounded rate. So we have $e^r = 1.05$ if the annual percentage rate (APR) is 5%. Thus $r = \ln(1.05)$.
- I'd expect most exam questions to say more simply $\sigma = 0.1$ and $r = \ln(1.05) \approx 0.05$, but you need to be more careful when using real market data.

Solution - The callPayoff function

The payoff of a European call option can be computed using the code below.

```
function p=callPayoff(K, S)
inMoney = S > K;
p = inMoney.*(S-K);
end
```

You could rewrite this using the maximum function if you wished.

Solution - The pricingKernel function

We have computed the pricing kernel mathematically and found that it is:

$$q_T(S) = \frac{1}{S\sigma\sqrt{2\pi T}} \exp\left(-\frac{(\ln(S/S_0) - (r - \frac{1}{2}\sigma^2)T)^2}{2T\sigma^2} \right)$$

we can translate this into MATLAB:

```
function q=pricingKernel( S, T, S0, r, sigma )
coefficient = 1./(S * sigma * sqrt( 2 * pi * T));
numerator=-(log(S/S0)-(r-0.5*sigma^2)*T).^2;
denominator = 2*T*sigma^2 ;
q =  coefficient .* exp( numerator/denominator);
end
```

Solution - The `priceCallByIntegration` function

We can now use the theory of risk neutral pricing to compute the payoff.

$$\text{price} = e^{-rT} \int_0^\infty P(S) q_T(S) \, \mathrm{d}S.$$

```
function p=priceCallByIntegration( K, T, S0, r, sigma )
function ret=integrand( S )
   ret = callPayoff( K, S ) .*...
           pricingKernel( S, T, S0, r, sigma );
end
p = exp(-r*T) ...
    * integrateToInfinity( @integrand, 0, 10001);
end
```

How about some tests?

```
function testPriceCallByIntegration()
% A call option with strike 0 is the same thing
% as buying the stock. Check we get the correct answer.
% This is effectively a test that our pricing
% kernel is risk neutral
r = log(1.05);
S0 = 100;
sigma = 0.1;
T = 0.25;
actual = priceCallByIntegration(0,T, S0, r, sigma );
assertApproxEqual( actual, S0, 0.001 );
end
```

Comparison with the Black Scholes Formula

```
function testBlackScholesCallPrice()
%Compares the black scholes price against
%that obtained by integration
r = log(1.05);
S0 = 100;
sigma = 0.1;
T = 0.25;
K = 110;
actual = priceCallByIntegration(K,T, S0, r, sigma );
expected = blackScholesCallPrice(...
           K,T, S0, r, sigma );
assertApproxEqual( actual, expected, 0.001 );
end
```

Now we're confident, let's answer the question

```
function answerQuestion()
%Let's answer the question finally
T = 0.25;
r = log(1.05);
S0 = 100;
K = 105;
vol = 0.1;
answer = priceCallByIntegration(K,T,S0,r,vol);
fprintf('The answer is %f dollars\n', answer );
fprintf('The answer to two d.p. is %.2f dollars\n', answer );
end
```

This demonstrates the MATLAB code required to print things more attractively.

What have we gained?

- We can easily adapt the code to puts and digital options.
- We can price derivatives with arbitrary payoffs.
- Given a pricing kernel $q_T(S)$ we can price derivatives. For example, you could take a pricing kernel with fat tails.

A standard substitution

What substitution should we use to change our indefinite integral to a definite integral? We have the following formula for the price:

$$e^{-rT} \int_0^\infty \text{payoff}(S) q(S) \, \mathrm{d}S$$

where $q(S)$ is the $\mathbb{Q}$-p.d.f. Let $Q(S)$ be the $\mathbb{Q}$ cumulative density function and make the substitution $R = Q(S)$.

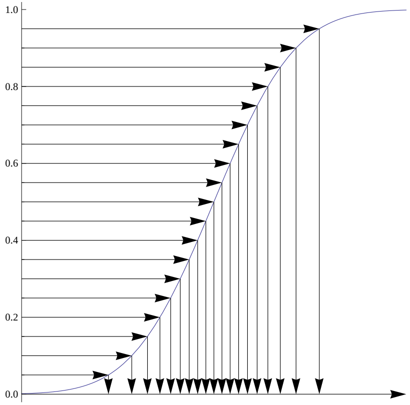
$$\mathrm{d}R = q(S) \, \mathrm{d}S$$

by definition of the probability density function as the derivative of the c.d.f.. So the price is given by:

$$e^{-rT} \int_0^1 \text{payoff}(Q^{-1}(R)) \mathrm{d}R$$

Inverse CDF

If you apply the inverse CDF to uniformly distributed numbers you get numbers with the desired distribution. So you can use the inverse CDF to simulate a distribution.



Monte Carlo pricing

The price is given by:

$$\exp^{-rT} \int_0^1 \text{payoff}(Q^{-1}(U))\mathrm{d}U$$

So the monte carlo price is given by:

- Generate random numbers U between 0 and 1.
- Apply Q^{-1} to get a sequence of random numbers with the same distribution as the $\mathbb{Q}$-distribution of stock prices.
- Compute the average discounted payoff.

i.e. compute the expected value using a $\mathbb{Q}$-measure simulation.

The Black Scholes case

Under the measure $\mathbb{Q}$

$$S \sim \log \mathcal{N}\left(z_0 + \left(r - \frac{\sigma^2}{2}\right) T, \sigma\sqrt{T}\right)$$

So by definition of the log normal distribution

$$Q(S) = N\left(\frac{1}{\sigma\sqrt{T}}\left(\ln(S/S_0) - \left(r - \frac{\sigma^2}{2}\right) T\right)\right)$$

Solve $Q(S) = U$ for S to get:

$$Q^{-1}(U) = S_0 \exp\left(\left(r - \frac{\sigma^2}{2}\right) T + \sigma\sqrt{T} N^{-1}(U)\right)$$

We can compute N^{-1} using MATLAB's `norminv` function.

Homework

Worksheet 3