Final Solution

Practical Lab Numerical Computing

Andrii Lischishin Lars Schleithoff Hendrik Kleikamp

July 25, 2017

Worksheet 1

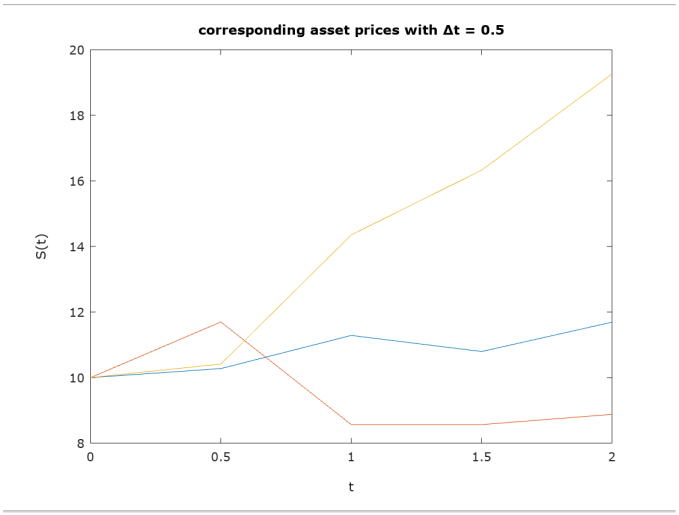
Task 1

The code generates two uniformly distributed random numbers on the interval $[0, 1]$. The first random number is generated by a standard $C++$ function called *rand*. The second random number is generated by a generator from the gsl library. The difference lies in the generator used for creating those random numbers. The second way uses a `gsl_rng` pointer which transports the information on the selected generator. One can use the same pointer for differently distributed random numbers, e.g. `double gsl_ran_gaussian (const gsl_rng * r, double sigma)`, where $\mu = 0$.

If you remove the expression (double), the result will be 0 due to rounding to the next smaller integer. The rand function creates an integer and `RAND_MAX` is a constant integer as well.

In fact, there won't be any different random numbers each time you run the code snippet because there is no seed declared.

Task 2



Task 3

The MATLAB script divides the interval $[0, 1]$ into 100 equidistant sections. The number of random values in each interval is then depicted by the density value of the points in the plot.

Fig.1 shows what happens if the interval bounds in step 1 of the rejection sampling algorithm are chosen to small, e.g. $[-2, 2]$.

Task 4

Let X be an uniformly distributed random variable on $(0, 1)$, and $\Phi(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} e^{-\frac{x^2}{2}} dx$.

$\Phi(t)$ is continuous and monotically increasing, therefore the solution of $\Phi(t) = y$, which is $\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} e^{-\frac{x^2}{2}} dx = y$, is unique. Now, the inverse function $f : (0, 1) \to \mathbb{R}$ takes in a value from the interval $(0, 1)$ and the output is in $\mathbb{R}$.

Specifically, $\Phi(t) = \frac{1}{\sqrt{2\pi}} \int_{a}^{b} e^{-\frac{x^2}{2}} dx = \Phi(b) - \Phi(a)$. This implies that $a \leq f(X) \leq b \Leftrightarrow \Phi(a) \leq X \leq \Phi(b)$, since $\Phi(f(X)) = X$.

For this reason, applying the inverse function of the standard normal distribution to an uniformly distributed random variable leads to a standard normal distributed random variable.

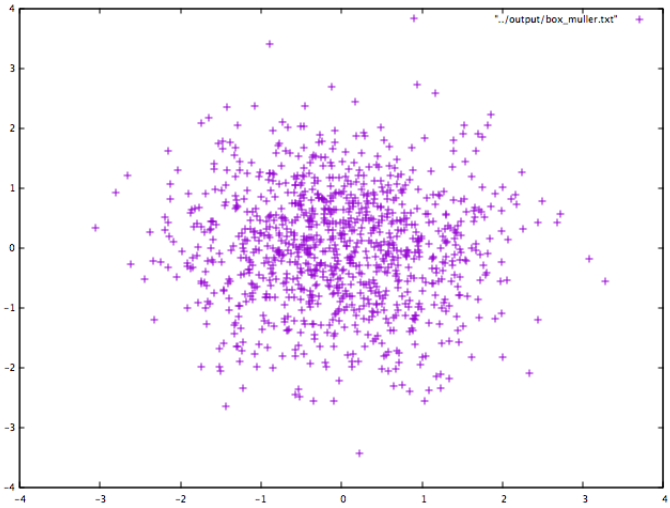
Task 5

The basic idea is to approximate the integral on different intervals by different functions. The first if-clause uses the symmetry of the standard normal distribution and the fact that

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{x^2}{2}} dx = 1.$$

For $x \geq 6$, the accuracy of 8 digits has already been reached, so the algorithm returns the value 1.

Task 6



Task 7

By transformation to polar coordinates, it follows that

$$z_1 = rcos(\phi), z_2 = rsin(\phi)$$

with $r = \sqrt{-2ln(u_1)}, \phi = 2\pi u_2$.

$\phi \in (0, 2\pi)$ is uniformly distributed on circles, and the square of the radius of these circles, $r^2 = Z_1^2 + Z_2^2$, has Chi-squared distribution.

The product of those two random variables ist normally distributed:

$$\frac{1}{2} e^{-\frac{1}{2}r^2} d(r^2) d\phi = \frac{1}{2\pi} e^{-\frac{1}{2}r^2} r dr d\phi = \frac{1}{2\pi} e^{-\frac{1}{2}(z_1^2 + z_2^2)} dz_1 dz_2$$

Therefore, z_1 and z_2 are standard normally distributed.

Task 8

The main advantage of the algorithm stated on the sheet is that $\hat{\mu}$ doesn't need to be calculated seperately, for this reason only one for-loop is needed. Furthermore, cancellation can be avoided and the algorithm is an online algorithm, which means that more values can be added easily at the end instead of going through the whole process again.

α, β and σ can be expressed in the following way (k is the number of the iteration of the forloop):

$$\alpha_k = \frac{\sum_{i=1}^{k} x_i}{k}$$

$$\beta_k = \beta_{k-1} + \frac{k}{k+1}\left(x_k - \left(\frac{x_1 + x_2 + ... + x_k}{k}\right)\right)^2$$

$$\sigma^2 = \frac{1}{N-1}\sum_{k=1}^{N} \beta_k$$

For $k \to \infty$, β_k converges to $\sum_{k=1}^{N}(x_k - \hat{\mu})^2$:

$$\underbrace{\frac{k}{k+1}}_{\to 1}\left(x_k - \left(\underbrace{\frac{x_1 + x_2 + ... + x_k}{k+1}}_{\to \hat{\mu}}\right)\right)^2 \xrightarrow{k \to \infty} (x_k - \hat{\mu})^2$$

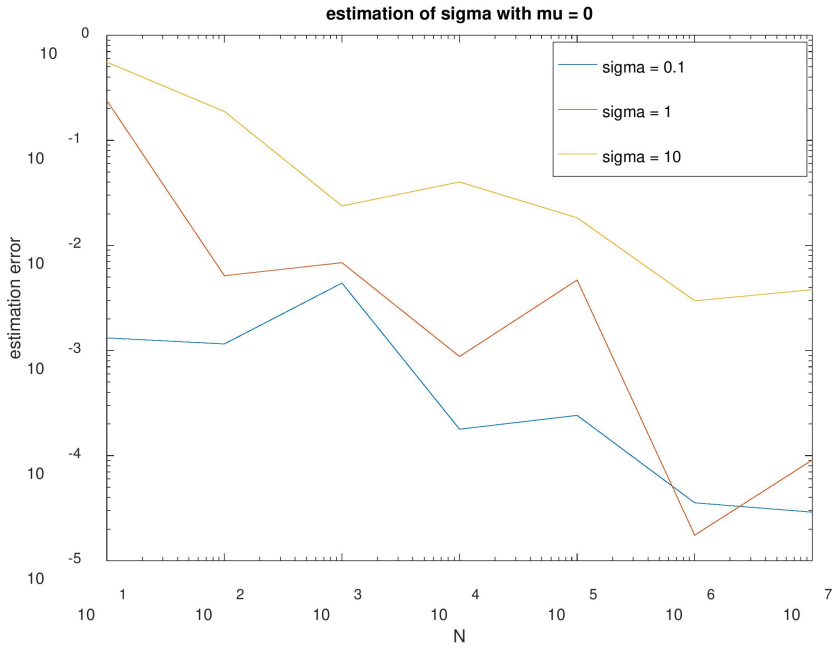
$$\beta = \sum_{k=1}^{N} \beta_k \to \sum_{k=1}^{N}(x_k - \hat{\mu})^2$$

$$\sigma^2 = \frac{1}{N-1}\sum_{k=1}^{N} \beta_k \to \frac{1}{N-1}\sum_{i=1}^{N}(x_k - \hat{\mu})^2$$

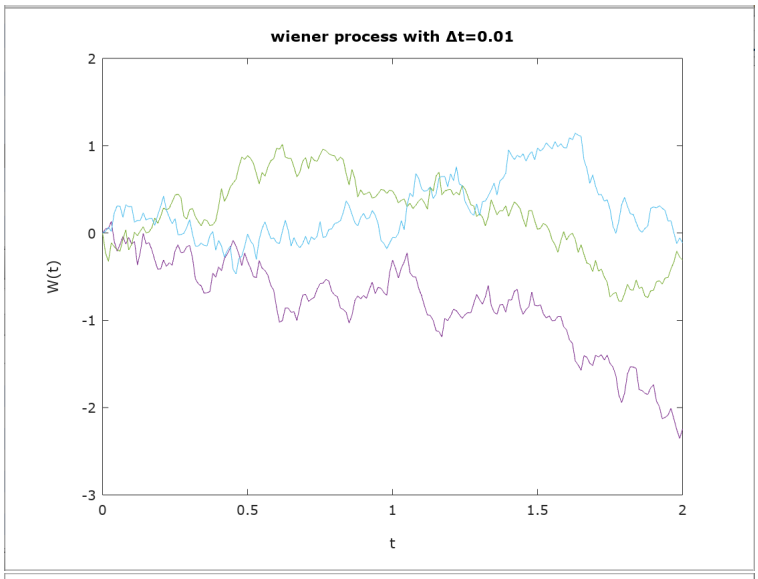
For this reason, the algorithm converges to the variance of $(x_i)_{i=1}^{N}$.

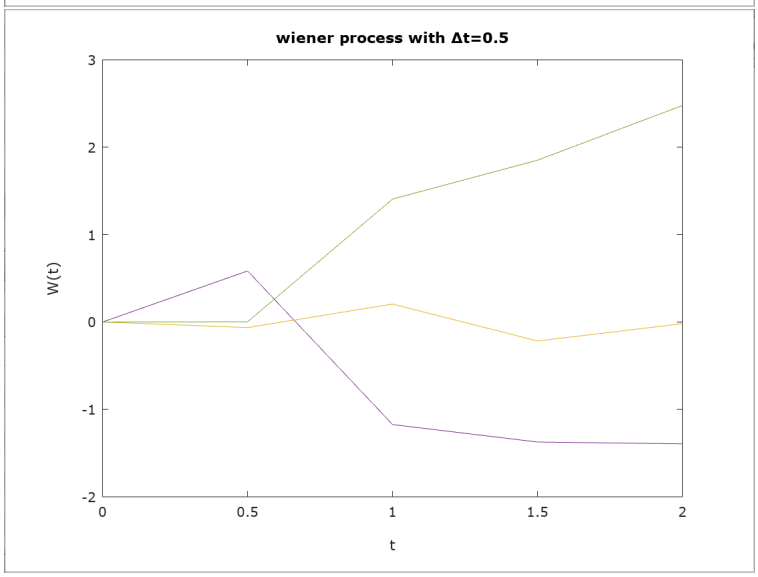
Task 9

The plot illustrates, how for small N, the deviation from the correct result is high and how it then converges to σ for higher N. This matches with the behaviour of the algorithm, which might have a great inaccuracy for small N, but gets more exact the bigger N gets.



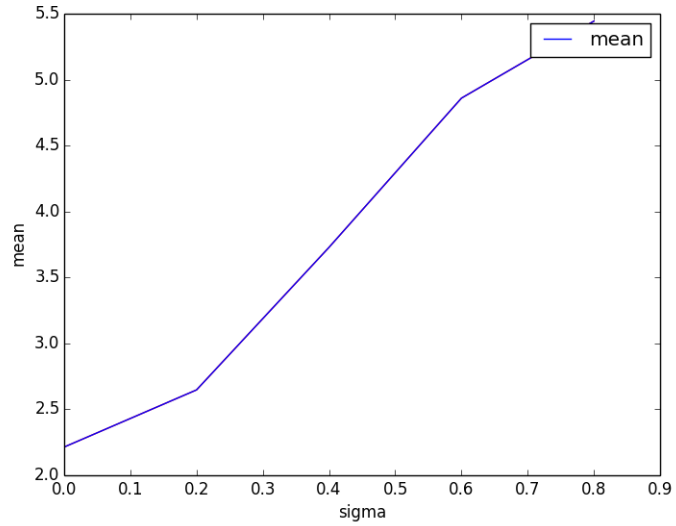
Task 10


wiener process with Δt=0.01
W(t)
t
2
1
0
-1
-2
-3
0
0.5
1
1.5
2


wiener process with Δt=0.5
W(t)
t
3
2
1
0
-1
-2
0
0.5
1
1.5
2

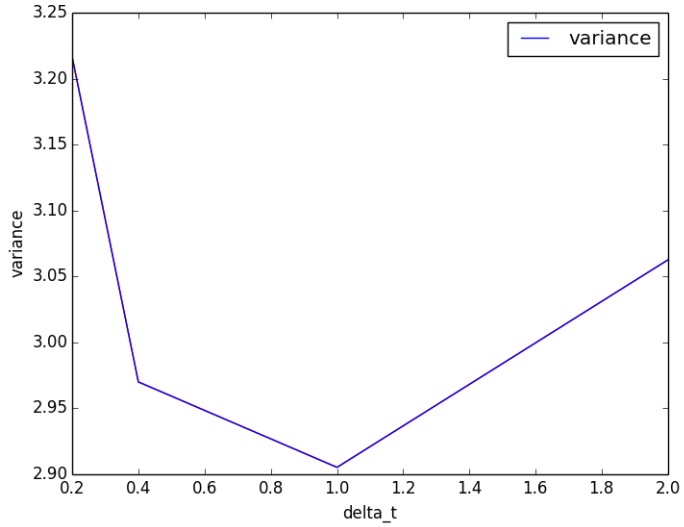
Worksheet 2

Task 1



Obviously, increasing σ leads to an increasing mean. This demonstrates, that with higher volatility, the fair price of the option increases as well (due to higher risk).

Task 2



The variance is not affected by Δt, as this figure demonstrates.

Task 3

To prove that

$$\mathbb{E}[V_{call}(S_T, 0)] = S(0)\exp(\mu T)\Phi(\sigma\sqrt{T} - \chi) - K\Phi(-\chi)$$

we use that by change-of-variable with $t := -t$ we get

$$\begin{aligned}
& \mathbb{E}[V_{call}(S_T, 0)] \\
= \; & \frac{1}{\sqrt{2\pi}} \int_{\chi}^{\infty} \left(S(0) \exp\left((\mu - \frac{1}{2}\sigma^2) \cdot T + \sigma\sqrt{T}s \right) - K \right) \exp\left(-\frac{s^2}{2} \right) ds \\
= \; & \frac{1}{2\pi} \int_{\chi}^{\infty} \left(S(0) \exp\left((\mu - \frac{1}{2}\sigma^2) \cdot T + \sigma\sqrt{T}s \right) \right) \exp\left(-\frac{s^2}{2} \right) ds - K \frac{1}{2\pi} \int_{\chi}^{\infty} \exp\left(-\frac{t^2}{2} \right) dt \\
= \; & \Psi - K \frac{1}{2\pi} \int_{-\infty}^{-\chi} \exp\left(-\frac{t^2}{2} \right) dt \\
= \; & \Psi - K\Phi(-\chi)
\end{aligned}$$

with

$$\Psi := \frac{1}{2\pi} \int_{\chi}^{\infty} \left(S(0) \exp\left((\mu - \frac{1}{2}\sigma^2) \cdot T + \sigma\sqrt{T}s \right) \right) \exp\left(-\frac{s^2}{2} \right) ds.$$

Now we prove that

$$\Psi = S(0)\exp(\mu T)\Phi(\sigma\sqrt{T} - \chi).$$

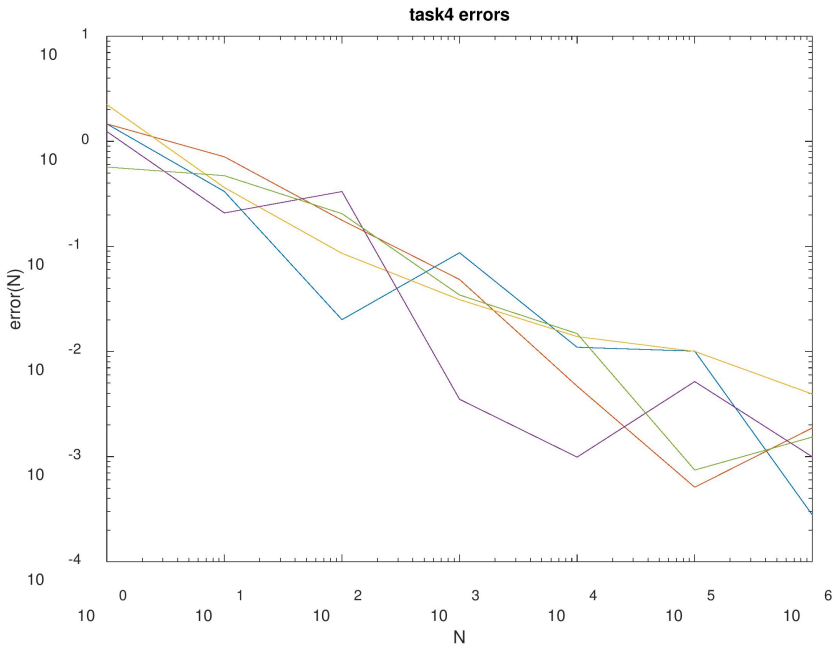
Again, we use a change-of-variable $z := t + \sigma\sqrt{T}$ and get

$$\begin{aligned}
& \frac{1}{\sqrt{2\pi}} \exp\left(\frac{\sigma^2}{2} T \right) \int_{-\infty}^{\sigma\sqrt{T}-\chi} \exp\left(-\frac{t^2}{2} \right) dt \\
= \; & \frac{1}{\sqrt{2\pi}} \exp\left(\frac{\sigma^2}{2} T \right) \int_{\chi-\sigma\sqrt{T}}^{\infty} \exp\left(-\frac{t^2}{2} \right) dt \\
= \; & \frac{1}{\sqrt{2\pi}} \exp\left(\frac{\sigma^2}{2} T \right) \int_{\chi}^{\infty} \exp\left(-\frac{(\sigma\sqrt{T} - z)^2}{2} \right) dz \\
= \; & \frac{1}{\sqrt{2\pi}} \int_{\chi}^{\infty} \exp\left(-\frac{z^2}{2} + z\sigma\sqrt{T} \right) dz.
\end{aligned}$$

We have

$$\begin{aligned}
\Psi \; = \; & \frac{1}{\sqrt{2\pi}} S(0) \exp\left((\mu - \frac{\sigma^2}{2}) \cdot T \right) \int_{\chi}^{\infty} \exp\left(-\frac{s^2}{2} + \sigma\sqrt{T}s \right) ds \\
= \; & S(0)\exp(\mu T) \exp\left(-\frac{\sigma^2}{2} T \right) \frac{1}{\sqrt{2\pi}} \int_{\chi}^{\infty} \exp\left(-\frac{s^2}{2} + \sigma\sqrt{T}s \right) ds \\
= \; & S(0)\exp(\mu T) \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\sigma^2}{2} T \right) \exp\left(\frac{\sigma^2}{2} T \right) \int_{-\infty}^{\sigma\sqrt{T}-\chi} \exp\left(-\frac{s^2}{2} \right) ds \\
= \; & S(0)\exp(\mu T)\Phi(\sigma\sqrt{T} - \chi).
\end{aligned}$$

Task 4



5 independent simulations of $\mathbb{E}[V_{call}(S_T, 0)]$. As expected, the rate of convergence is $O(N^{1/2})$.

Task 5

We have $\Phi^{-1}(0) = -\infty$ and $\Phi^{-1}(1) = \infty$ where $\Phi^{-1} : (0,1) \to (-\infty, \infty)$ is the inverse cumulative distribution function. As an integral of a positive continious function, Φ is a bijection, continious and differentiable. That means Φ is a diffeomorphism. Φ^{-1} is also differentiable, so we can use the transformation theorem with the change-of-variable $t = \Phi(s)$. We have $|\det(D\Phi(s))| = \frac{1}{\sqrt{2\pi}} \exp(-\frac{s^2}{2})$ and

$$\begin{aligned}
& \int_0^1 f(\Phi^{-1}(t))dt \\
= & \int_{\Phi^{-1}(0)}^{\Phi^{-1}(1)} f(\Phi^{-1}(\Phi(s))) \cdot |\det(D\Phi(s))| ds \\
= & \int_{-\infty}^{\infty} f(s) \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{s^2}{2}\right) ds \\
= & \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(s) \exp\left(-\frac{s^2}{2}\right) ds
\end{aligned}$$

what proves formula (7).

Task 6

In case of the trapezoidal-rule, the set of nodes of level l is a subset of the nodes of level $l+1$. Furthermore, the 2^l additional values lay exactly half way in between the nodes of level l. (Except for the first and the last node, which lie half way in between 0 and the first node of level l and in between the last node of level l and the end of the interval.)

Task 7

In case of the Gauß-Legendre Quadrature, the nodes of level l are not a subset of the nodes of level $l+1$. At the edges of the interval, the nodes are denser. According to Satz 1.18 from Einführung in die Grundlagen der Numerik, node $x_i^{(N_l)}$ from level l lays between the nodes $x_i^{(N_{l+1})}, x_{i+1}^{(N_{l+1})}$ from level $l+1$.

The nodes are the roots of a three-term-recurrence relation of the form $p_{n+1}(t) = (t-\alpha_n)p_n(t) - \beta_n^2 p_{n-1}(t),\ n \geq 0$. The roots are the eigenvalues of a tridiagonal $(N_l \times N_l)$-dimensional matrix of the form

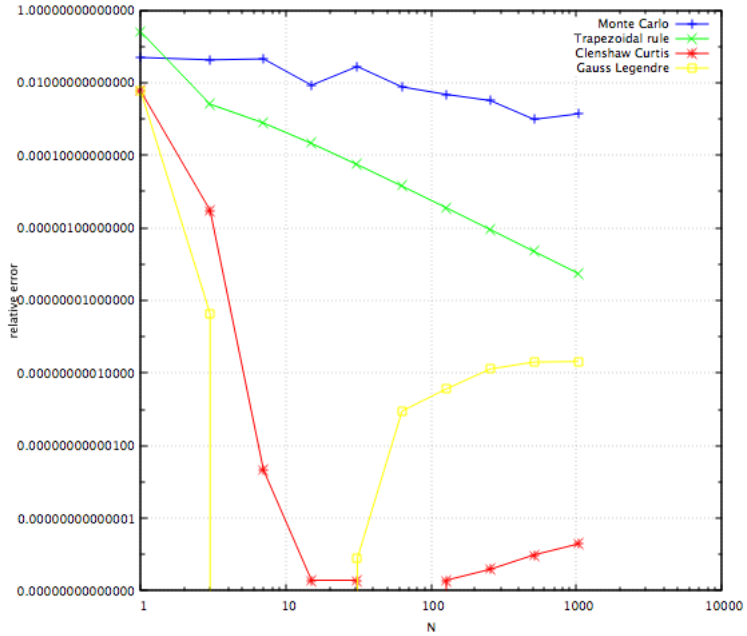
$$\begin{bmatrix} \alpha_0 & \beta_1 & 0 & ... & ... & ... \\ \beta_1 & \alpha_1 & \beta_2 & ... & ... & ... \\ 0 & \beta_2 & a_2 & \beta_3 & ... & ... \\ 0 & ... & ... & ... & ... & 0 \\ ... & ... & 0 & \beta_{N_l-2} & \alpha_{N_l-2} & \beta_{N_l-1} \\ ... & ... & ... & 0 & \beta_{N_l-1} & \alpha_{N_l-1} \end{bmatrix}$$

The weights are the first entry of the eigenvectors to the eigenvalues calculated for the nodes. Alternitavely, they can be recieved by taking $\Lambda_{n+1}(x_i)$ of the Christoffel-function, with $x_i, i = 1, ..., n$ nodes.

Task 8

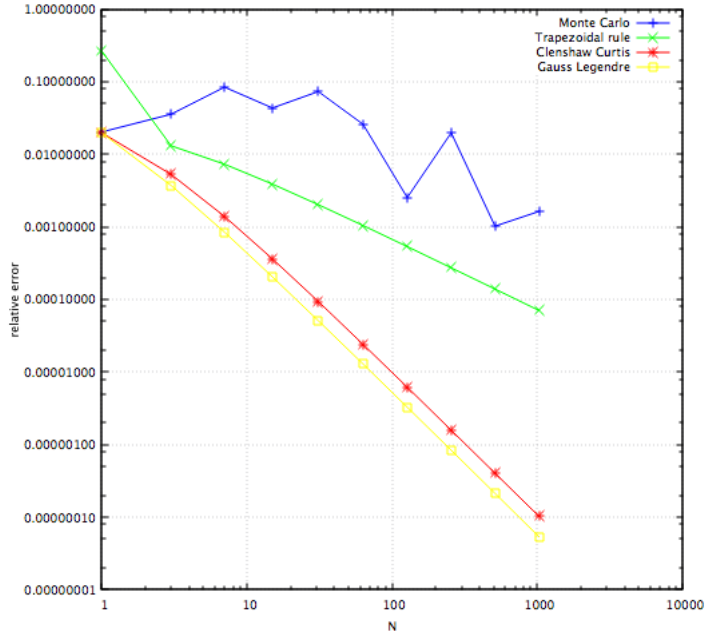
The Clenshaw-Curtis Quadrature rule uses nested nodes, the nodes of each level are a subset of the nodes of higher levels. Similarly to Gauß-Legendre quadrature, the number of nodes is higher at the edges of the interval.

Task 9

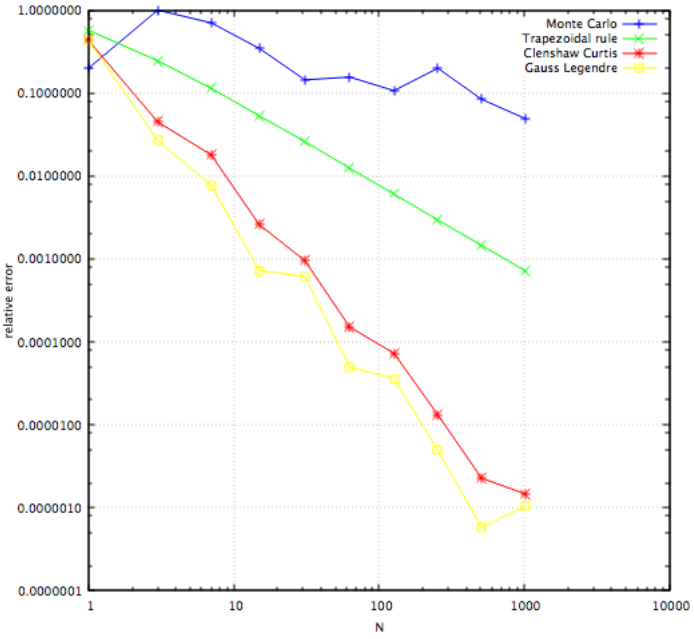


Task 10

For $K = 0$:

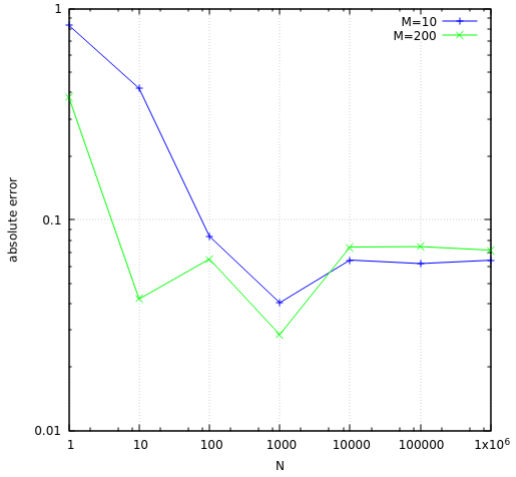


For $K = 10$:



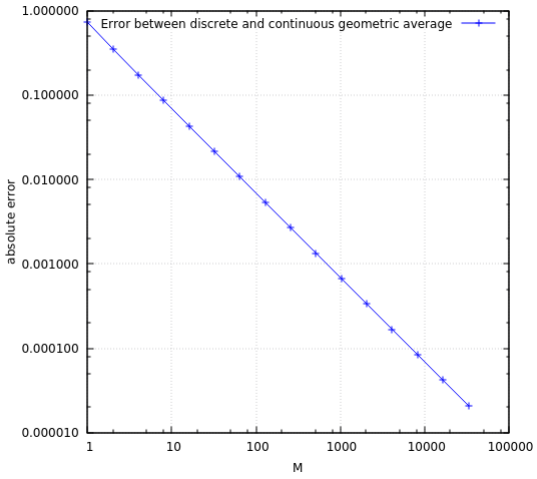
Worksheet 3

Task 3



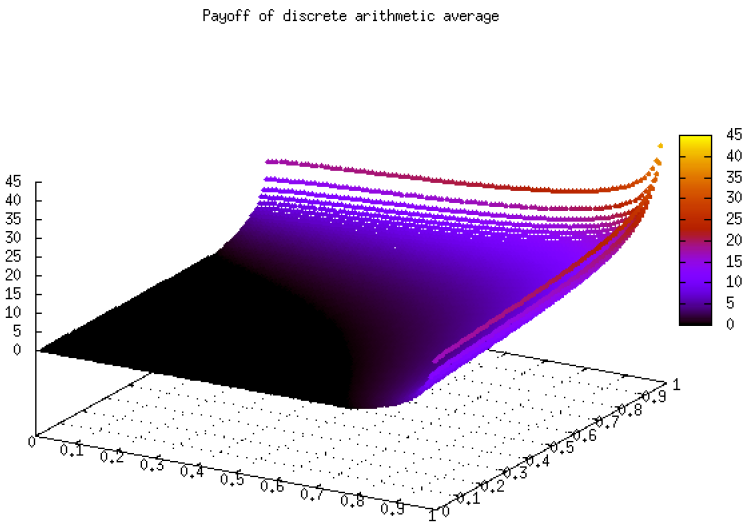
The number of timesteps is not really effecting the convergence.

Task 4



The discrete geometric average converges linearly to the continuous geometric average.

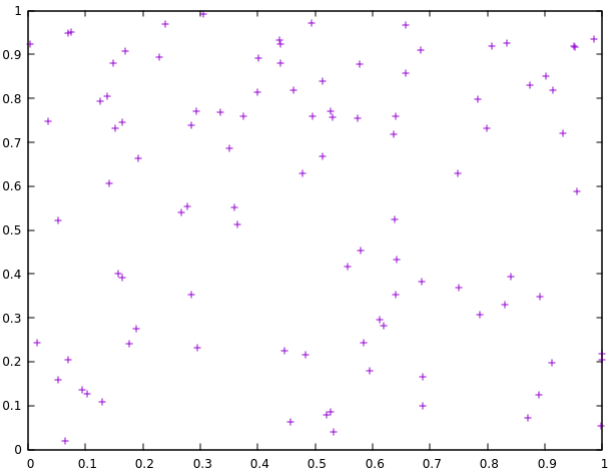
Task 5



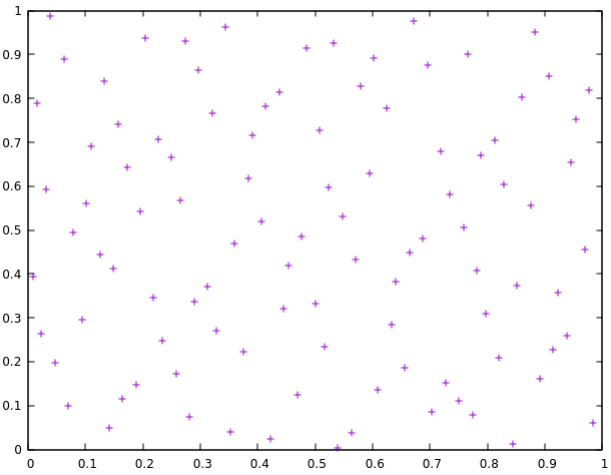
The discrete geometric payoff function is evaluated on 10.000 points in $[0,1]^2$.

Task 7

Uniform random numbers in $(0,1)^2$:



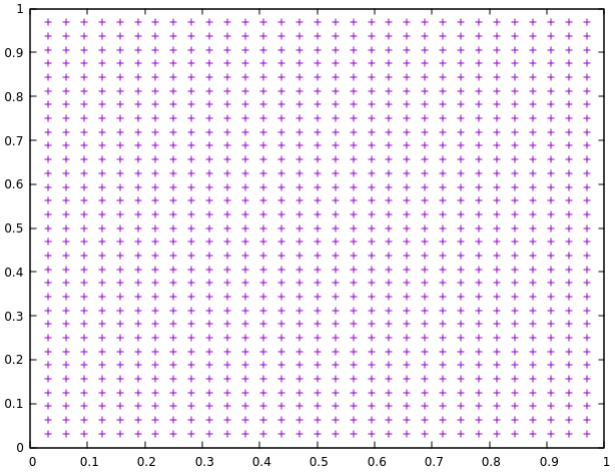
Halton sequence:



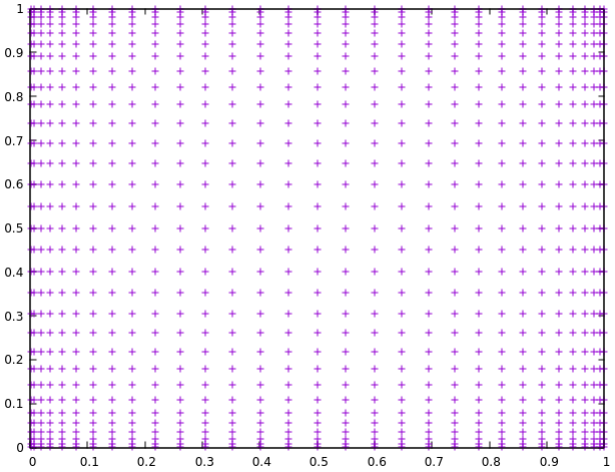
One can see that there are some regions in $(0,1)^2$ where no uniform random numbers are set. This is not the case in the point set calculated by the Halton sequence. The Halton sequence gives a very uniform spread set without any holes.

Task 9

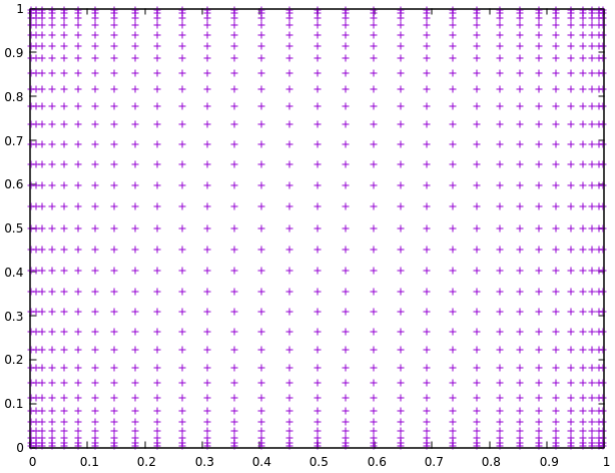
Quadrature nodes of the two-dimensional product rule for trapezoidal rule:



Quadrature nodes of the two-dimensional product rule for Gauss-Legendre:



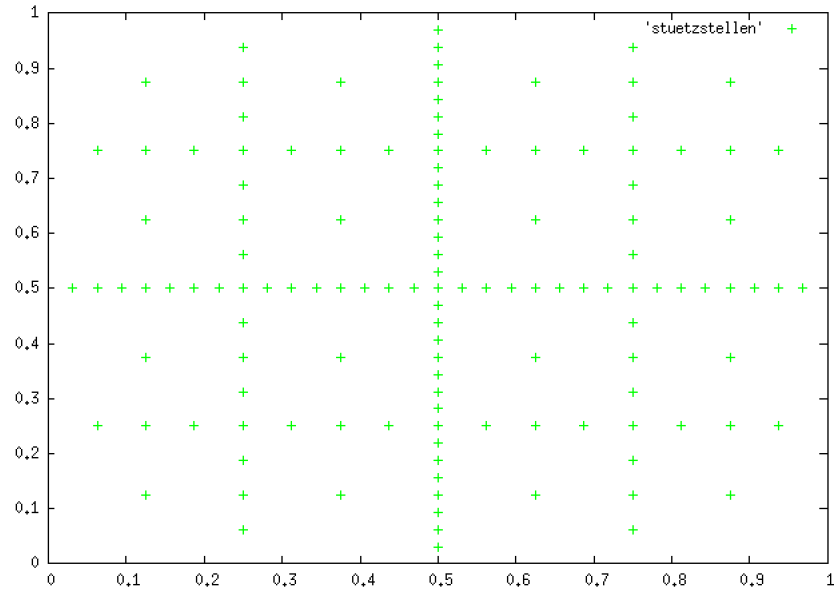
Quadrature nodes of the two-dimensional product rule for Clenshaw Curtis:



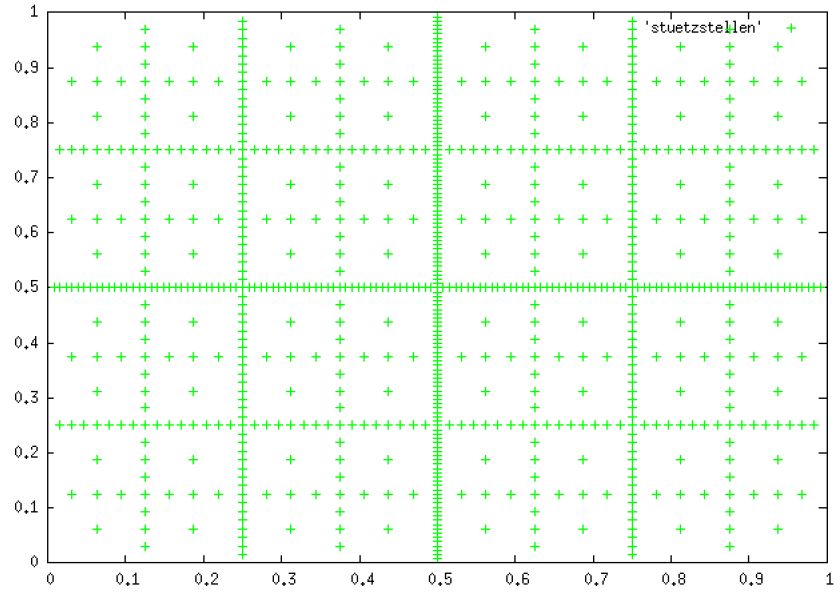
Both Gauß-Quadrature and Clenshaw-Curtis-Quadrature have an increased amount of nodes at the edges of the integration region. The trapeziodal rule distributes the nodes equidistantly over the integration region.

Task 11

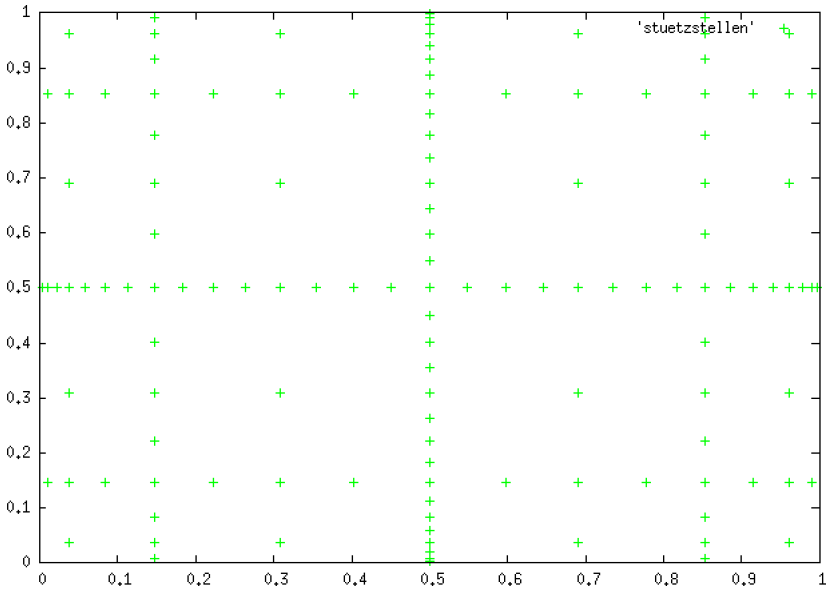
Two-dimensional Sparse Grid using Trapeziodal Rule for $l = 5$:



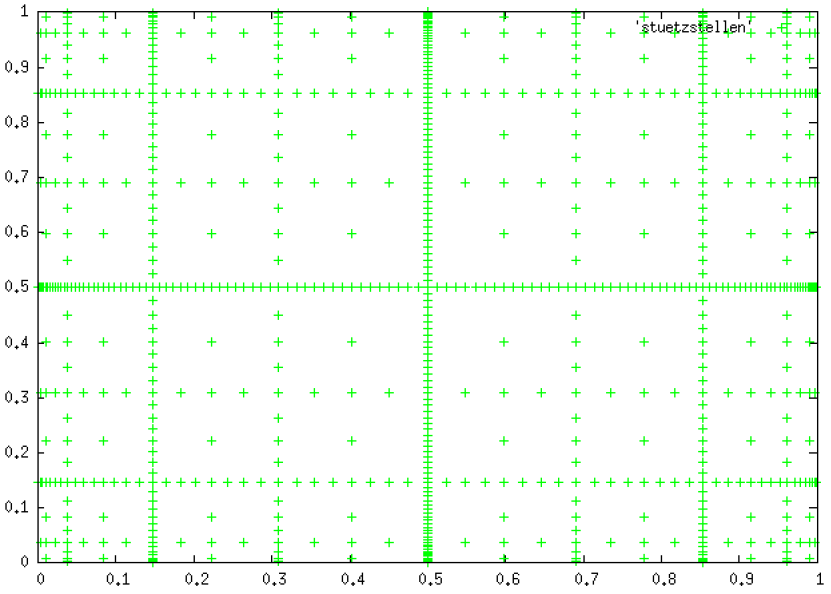
Two-dimensional Sparse Grid using Trapezoidal Rule for $l = 7$:



Two-dimensional Sparse Grid using Clenshaw-Curtis Quadrature Rule for $l = 5$:



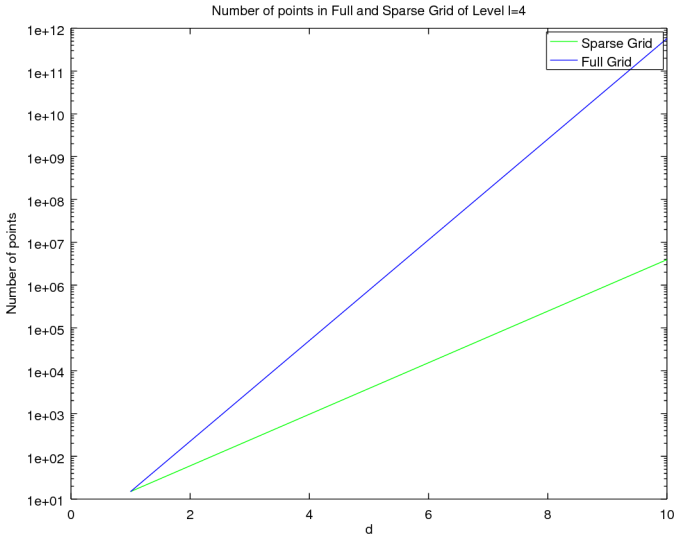
Two-dimensional Sparse Grid using Clenshaw-Curtis Quadrature Rule for $l = 7$:



For Clenshaw-Curtis Quadrature, with increasing level, more nodes are situated at the edges of the integration region. For Trapezoidal Rule, this is not the case.

Task 12

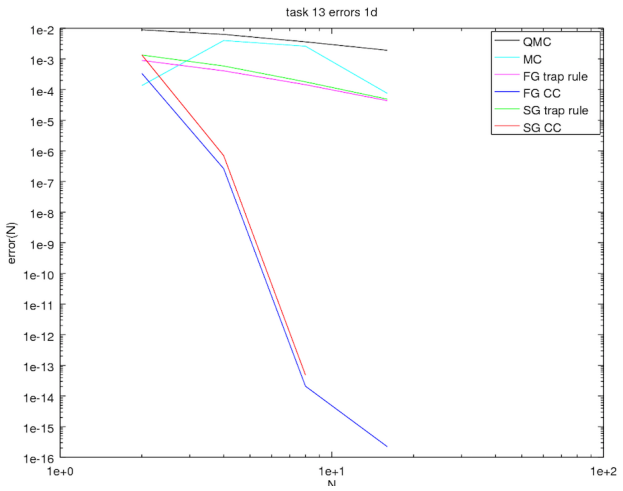
Number of points in Sparse Grid and Full Grid in Level $l = 4$:

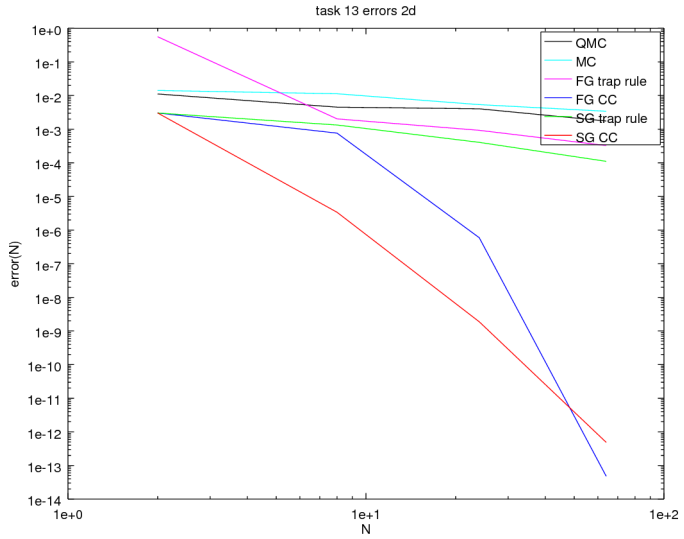


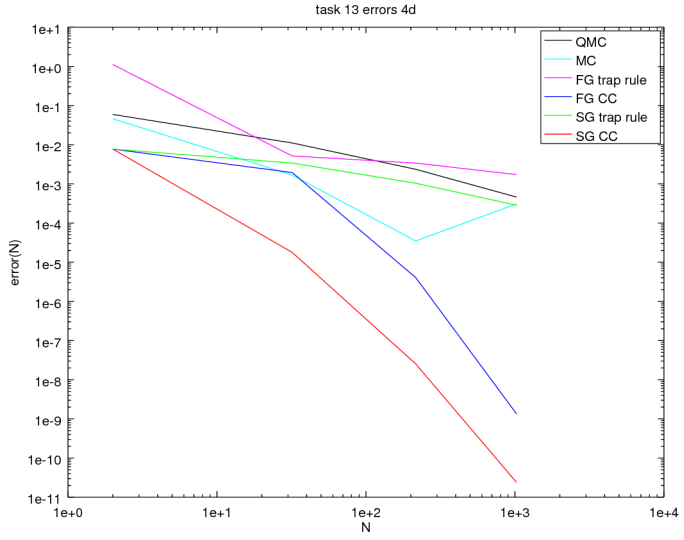
One can see that the number of points increases much faster in the Full Grid than in the Sparse Grid.

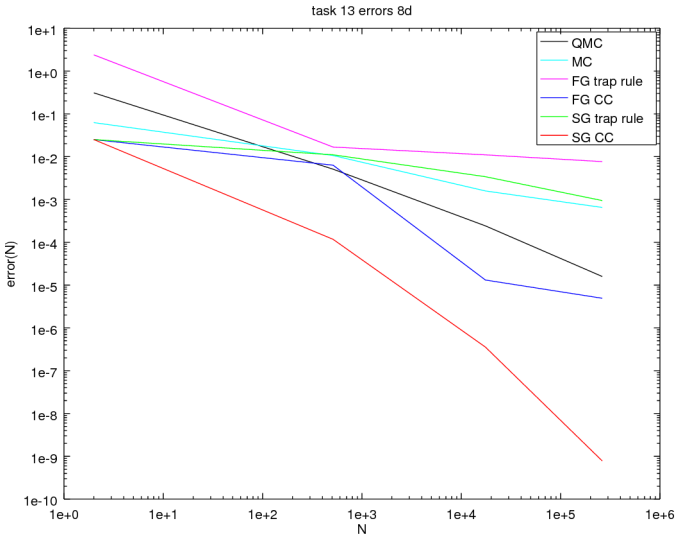
Task 13

Convergence plots for the testfunction f_γ in different dimensions:



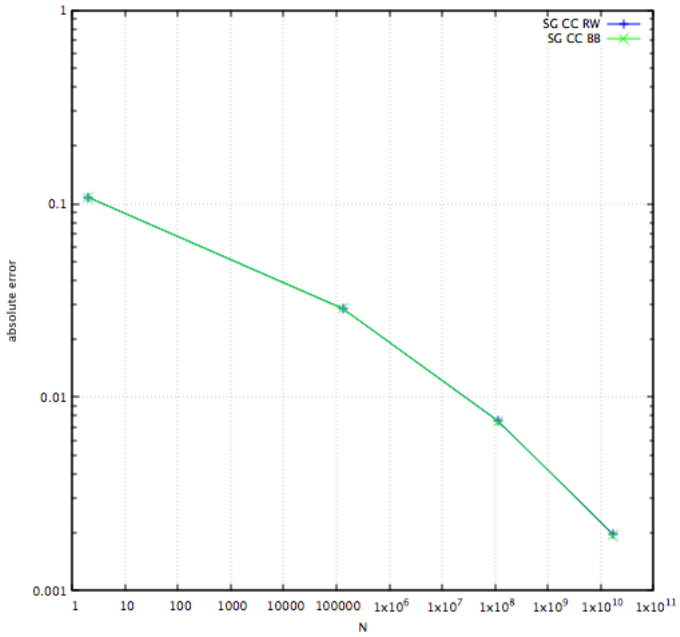

task 13 errors 2d
error(N)
N
QMC
MC
FG trap rule
FG CC
SG trap rule
SG CC
1e+0
1e-1
1e-2
1e-3
1e-4
1e-5
1e-6
1e-7
1e-8
1e-9
1e-10
1e-11
1e-12
1e-13
1e-14
1e+0
1e+1
1e+2


task 13 errors 4d
error(N)
N
QMC
MC
FG trap rule
FG CC
SG trap rule
SG CC
1e+1
1e+0
1e-1
1e-2
1e-3
1e-4
1e-5
1e-6
1e-7
1e-8
1e-9
1e-10
1e-11
1e+0
1e+1
1e+2
1e+3
1e+4



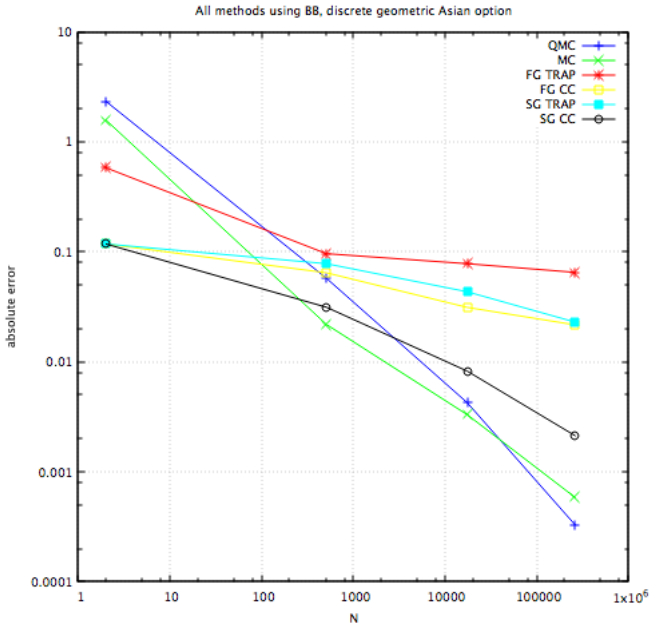
Obviously, the Sparse Grid Method has the best convergence and the Full Grid method has the second best convergence, while MC and QMC can't match these convergence rates. Since the quadrature nodes are better distributed on the integration region, QMC has better convergence than MC.

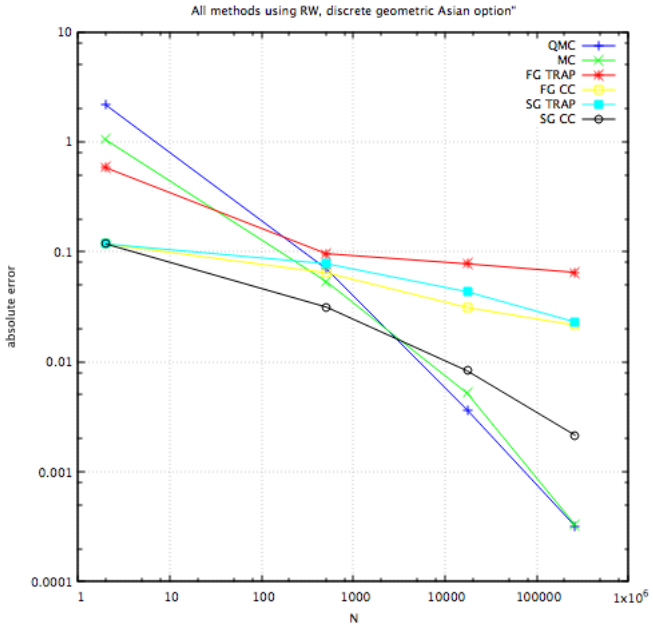
Task 15



Here, since the payoff function isn't sufficiently smooth for Clenshaw-Curtis to work properly, the Brownian Bridge Method shows no advantage in convergence over Random Walk.

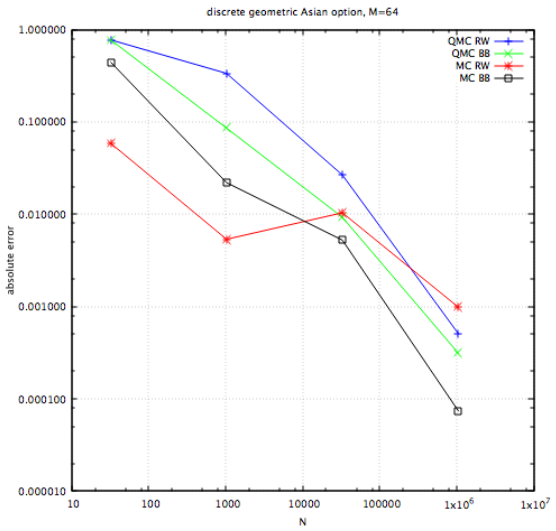
Task 16





In this case, due to lack of smoothness, Monte Carlo and Quasi Monte Carlo perform better than more sophisticated quadrature rules.

Task 17



This illustrates the superiority of Brownian Bridge Interpolation over Random Walk for Monte Carlo and Quasi Monte Carlo integration.

Task 18

The difference between MC and QMC is, that MC uses actual pseudo random numbers, while QMC uses deterministic sequences. For this reason, QMC Quadrature Nodes are more evenly distributed over the integration region, which enhances the convergence rate of QMC (which is $\mathcal{O}(N)$) compared to the convergence of $\mathcal{O}(N^{1/2})$ of Monte Carlo Method. However, there is a higher computational effort in creating quasi-random-number sequences, which increases the cost of QMC especially in high dimensions compared to Monte Carlo.

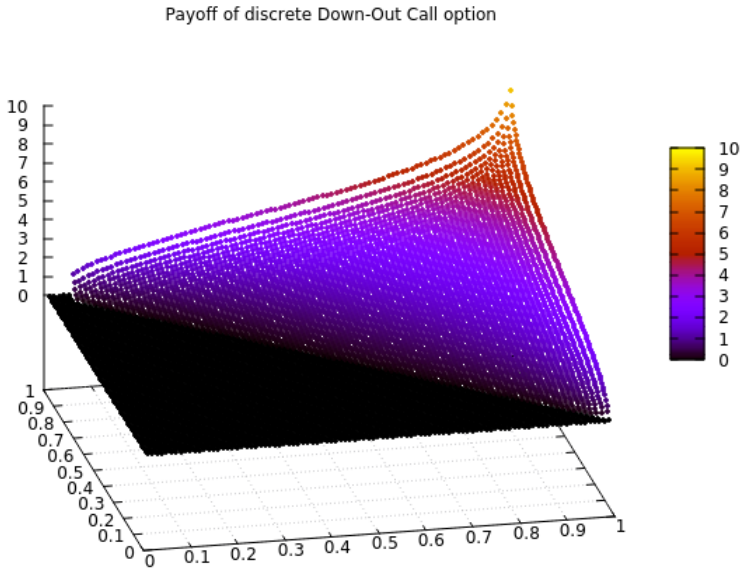
Both, QMC and MC, can integrate a wide range of functions, regardless of how smooth these functions are.

Sparse Grids and Full Grids make use of nodes and weights which result from quadrature methods like Clenshaw-Curtis or Trapezoidal rule. These methods can only be applied to somewhat smoother functions, but they have much better convergence rates. Full Grids are heavily affected by the dimension, with the number of nodes increasing exponentially with increasing dimension. For sparse grids, dependence on dimension is considerably lower.

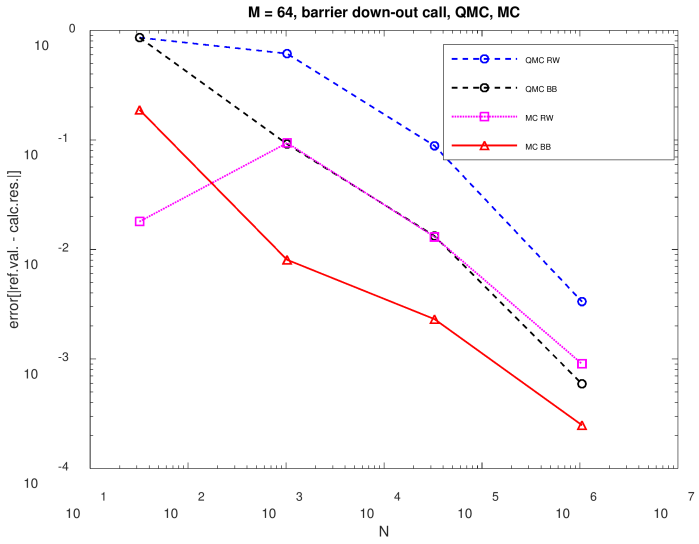
Therefore, if it is known that the integrand will be relatively smooth, using Sparse Grid methods is the best way to compute integrals in high dimensions. However, if the integrand is unknown or if the integrand is a function that is not differentiable in many points, MC and QMC integration are the preferred methods, espacially in high dimensions.

Worksheet 4

Task 1



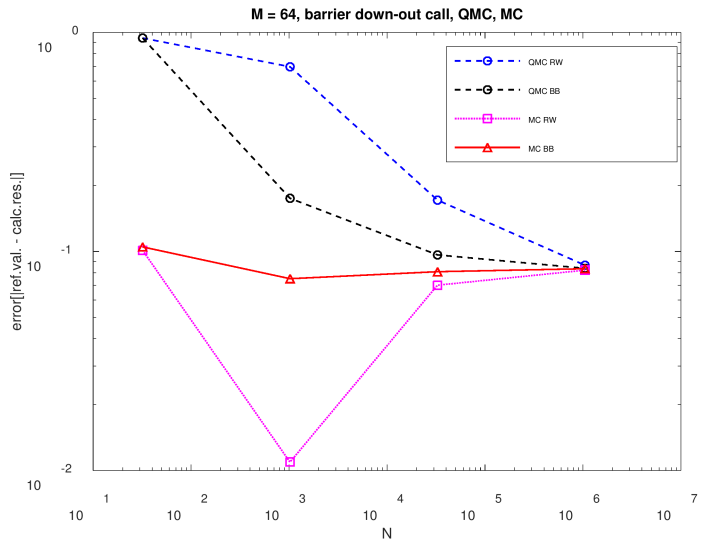
Task 2



- On this figure the absolute error: $|ref.val. - calc.val.|$ using different methods(QMC,MC) for Barrier Down-Out Call Option is plotted in loglog-scala

against number of points.

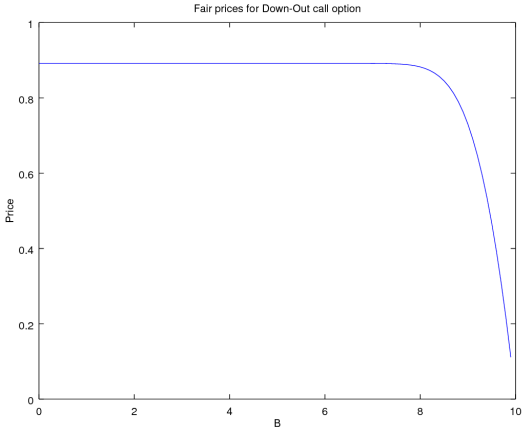
- From the plots, one can observe that there is no really difference between using Brownian-Bridge or Random-Walk constructions.



- On this figure the absolute error: $|ref.val. - calc.val.|$ using different methods(QMC,MC) for Barrier Down-Out Call Option is plotted in loglog-scala against number of points. However now, reference value was computed using far more lower precision, where-from there is no convergence.

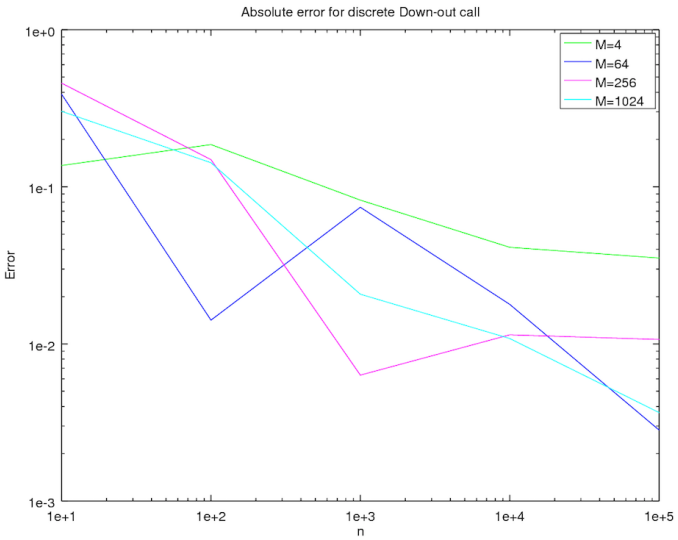
Task 3

The picture below shows the fair price for a Down-Out call option with barrier B. One can see that the fair price is going down if the barrier is larger than about $B = 8$. This makes sense, because when the barrier is high, the payoff is 0 in many cases (because for this value of B it happens more often that the price of the underlying is below the barrier).

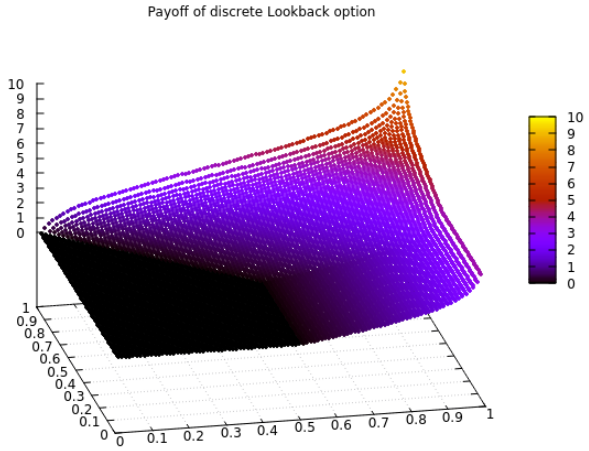


Task 4

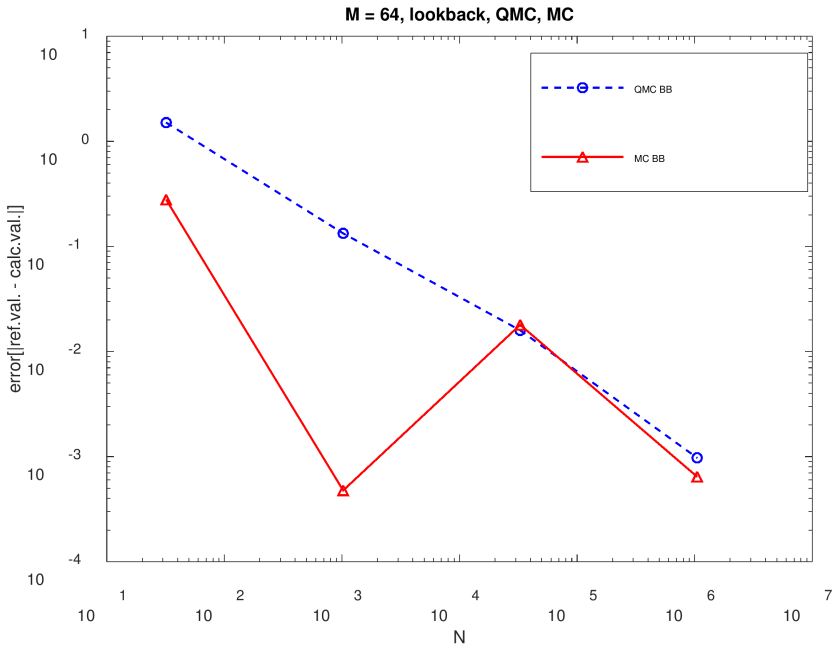
The next plot shows the absolute error for the discrete Down-Out call option for different values of M. One can see that for $M \geq 64$ the convergence is much better than for $M = 4$.



Task 5

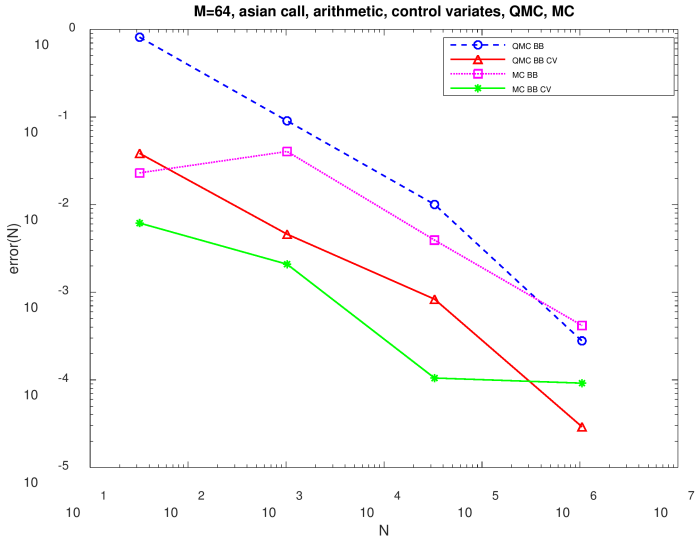


Task 6



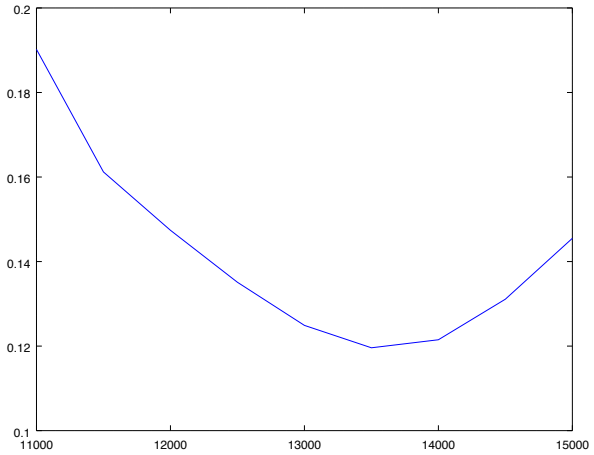
- On this figure the absolute error: $|ref.val. - calc.val.|$ using different methods(QMC,MC with Brownian-Bridge) for Lookback Call Option is plotted in loglog-scala against number of points.

Task 7



- On this figure, results of the **control variate** method are presented.

Task 9



Volatility of Call-Options for DAX, expiring in December, 2017. In this case, the volatility smile is clearly visible. The current value of the DAX is at about 12450 points.