**Normal Inverse Gaussian Option Valuation:**

**Inverse Gaussian Bridge**

**Versus Compound Poisson Process**

By

Kostyantyn Diordiyev

A research essay presented to the

University of Waterloo

In partial fulfillment of the requirements for the degree of

Master of Mathematics

in

Statistics/ Finance

Waterloo, Ontario, Canada, 2007

Table of Contents

[1 Introduction 3](#_Toc267568446)

[2 The Normal Inverse Gaussian Framework 4](#_Toc267568447)

[3 Option Pricing and Subordinator Monte-Carlo Method 7](#_Toc267568448)

[4 An Inverse Gaussian Bridge for the Normal Inverse Gaussian Process 8](#_Toc267568449)

[5 Compound Poisson Process Approximation to the NIG Process 12](#_Toc267568450)

[6 Options to Be Priced 15](#_Toc267568451)

[7 Numerical Results 16](#_Toc267568452)

[8 Conclusions 18](#_Toc267568453)

[9 References 20](#_Toc267568454)

[Appendix: Matlab Code 22](#_Toc267568455)

# 1 Introduction

The normal inverse Gaussian (NIG) model has been investigated and applied to the valuation of financial instruments by a number of authors. These include Barndorff-Nielsen [4], [5], Carr, Geman, Madan, and Yor [6], and Albrecher and Predota [7], for example. Analytical solutions are rarely available for both plain-vanilla and exotic options, so their valuation requires the use of numerical methods. The most common of these methods are Monte-Carlo simulation, Fourier transform methods, and PDE approaches.

The given essay investigates the use of Monte-Carlo simulation with the NIG underlying model. In particular, I compare and contrast the use of inverse Gaussian (IG) bridge and compound Poisson process to implement the model. Then I apply these methods to price arithmetic Asian, Russian, and barrier options and analyze the accuracy of the results and the time required for obtaining these results. The IG bridge methodology is based on the work by Ribeiro and Webber [1], and the compound Poisson process approximation is based on the papers by Rydberg [2] and Rasmus, Asmussen, and Wiktorsson [3].

Some other authors have investigated the standard Brownian bridge methodology. Beaglehole, Dybvig, and Zhou [8] used the knowledge of the distribution of extremes of a Brownian bridge to significantly speed up a Monte-Carlo method for pricing barrier options. In a related paper, Ribeiro and Webber [9] showed how to apply a bridge method to the variance-gamma underlying process.

The second section of this essay gives the details of the NIG process framework. In the third section, I talk about how Monte-Carlo methods may be used for option pricing by exploiting the subordinated Brownian motion (BM) representation of the NIG process. Next, I discuss the construction and application of the IG bridge method. In the fifth section, I present the details of the compound Poisson process approximation to NIG process. After that, I introduce the derivatives I have priced: the arithmetic Asian, barrier, and Russian options. The seventh section gives numerical results and their interpretation. The conclusions drawn are in section eight. The code used is attached in Appendix.

# 2 The Normal Inverse Gaussian Framework

The Normal Inverse Gaussian process is a particular kind of Levy processes; hence, we will start our discussion of NIG from the latter.

A Levy process is a stochastically continuous process with stationary and independent increments. Therefore, it has a characteristic exponent (κ(ξ)) that is defined by a so-called Levy triplet (α, σ, ν) as follows,

E[exp(iξXt)] = exp(tκ(iξ)) = exp{t(iαξ – σ2ξ2/2 + ∫R(eiξx – 1 – iξx**1**{|x|<1}) ν(dx))}

where α,ξ є R, *σ* є R+, and *ν* is a measure on R\{0} which satisfies ∫R(*x*2Λ1)*ν*(d*x*) < ∞.

The finite and infinite variation cases refer to the integral ∫|x|<1|x| *ν*(d*x*) being finite or infinite. More details are given in the books by Bertoin [10] and Sato [11].

In case a Levy process Lt is a BM or a Poisson process, the stock price model based on

St = S0\*exp(Lt)

is complete and the risk-neutral measure will, correspondingly, be unique. In all the other cases, however, the model is incomplete, and there are many risk-neutral martingale measures Q making Lt a Levy process and satisfying

exp(rt) = EQ[exp(Lt)] = exp{tκQ(1)}

where *κ*Q is the characteristic exponent of Lt under Q. If this Levy process is exponentially tilted, i.e.

κQ(x) = κ(x + θ) - κ(θ)

the equation κQ(1) = r has either a unique solution θ0 or none. In the former case, the tails are typically light and the corresponding κQ is called the Essher transform, and in the latter case the tails are typically heavy.

The NIG process Lt is a Levy process where increments in Lt are distributed according to the NIG distribution. The density of an NIG(*α*,*β*,*μ*,*δ*) distribution is

f(x) = αδ/π\*K1(α√{δ2 + (x – μ)2})/√{δ2 + (x – μ)2}\*exp{δγ + β (x – μ)}

where α, δ ≥ 0, |β| ≤ α, μ є R, γ2= α2- β 2, and K1 is the modified Bessel function of the second kind. The NIG Levy process is a Levy process such that Lt has an NIG(α,β,μt,δt) distribution, so we set δt = δt and μt = μt. The central moments of the distribution are μ1 = μt + δt βγ-1, μ2 = δtα2γ-3, μ3 = 3δtβα2γ-5, and μ4 = 3δtα2(α2+4β2)γ-7. The class of NIG distributions is closed under convolution. NIG is a pure jump process and its Levy measure has a density

ν(x;α,β,δ) = δα/(π|x|)\*exp{βx}\*K1(α|x|)

Hence, the characteristic exponent becomes

κ(ξ) = μξ + δ[(α2 – β2)1/2 – (α2 – (β + ξ)2)1/2]

and the Levy-Khintchine formula corresponding to this exponent is

κ(ξ) = ∫|x|≥1(1 – eiξx) ν(x;α,β,δ)dx + ∫|x|<1(1 – eiξx - iξx) ν(x;α,β,δ)dx + iξγ

where γ = 2*δα/*π \* ∫01 sinh(*βx*) K1(*αx*)dx . Under the Esscher measure, Lt is still a NIG Levy process with the same *α*,*μ,δ* but *β* replaced by

β + θ0 = -1/2 + √{(μ – r)2α2/((μ – r)2 + δ2) – (μ – r)2/4δ2}

The NIG process Lt can also be represented as a subordinated BM, Lt = μt + wh(t), where wt is a BM with drift β and variance 1, and h(t) is an inverse Gaussian process ht ~ IG(δt,γ). The density ftIG(x) of ht conditional on h(0) = 0 is

ftIG(x) = δt/√(2π)\*x-3/2exp(-1/2\* γ2/x\*(x- δt /γ)2)

The IG density can be alternatively parameterized as

ftIG(x) = √(λt/2π)\*x-3/2 exp(-λt/2\*(x- μt)2/xμt2)

where μt= δt/γ and λt = δt2.

The NIG process is an example of a set of generalized hyperbolic distributions. This set can be represented as a time-changed Brownian motion where the time change ht belongs to the generalized inverse Gaussian distributions indexed by t, ht ~ GIG(δt,λ,γ) with density ftGIG:

ftGIG(h;δt,λ,γ) = (γ/δt)λ \*1/2Kλ(δtγ)\*hλ-1exp(-1/2(δt2/h+ γ2h))

The GIG(δt,λ,γ) distributions are not closed under convolution. The NIG process is a special case with an inverse Gaussian time change, a GIG process with λ = -1/2. This subset is closed under convolution.

# 3 Option Pricing and Subordinator Monte-Carlo Method

Suppose that St is the price at time t of a non-dividend-paying stock. I will use the NIG process Lt to model returns of St. The state space Ω will be the path space of Lt with the filtration induced by Lt. The log-returns to the stock price process St under the pricing measure F are modeled as a Levy process. Following Madan, Carr, and Chang [12] and Eberlein and Raible [13],

St = S0exp(Lt + (r-ω)t)

where Lt is an NIG process, r is the constant short rate, and

ω = μ + δγ - δ√(α2 – (1+β)2)

is a compensator term defined by exp(ω) = E[exp(Lt)] to make sure that Ste-rt is a martingale under F.

From the martingale perspective, the value ct at time t<T of an option with payoff HT ≡ HT(ω) at time T is

ct = Et [HTe-r(T-t)]

with the money market account used as a numeraire and its associated risk-neutral measure used to price the option.

The equation above can be solved by Monte-Carlo integration. The idea is to construct a set {ŵm}m=1,…,M of discrete sample paths randomly selected under the discrete-time approximation to measure F. Then the approximation ĉt to ct can be given as

ĉt = e-r(T-t)1/M ∑Mm=1 HT(ŵm)

Discrete sample paths for a subordinated Brownian motion Lt = wh(t) can be constructed by first constructing discrete sample paths for the subordinator h(t) and then sampling the process wt at times determined by the paths for h(t).

First, discretize the time axis as 0 = t0 < t1 < … < tN = T and set Δtn = tn+1 - tn[[1]](#footnote-1). Then construct a discrete sample path {ĥn}n=0,…,N for h(t). Next, set ĥ0 = 0 and iteratively generate increments Δĥn = ĥn+1 - ĥn ~ IG(δΔtn,γ). Now set ŵ0 = 0 and iteratively generate increments Δŵn = ŵn+1 - ŵn ~ N(βΔĥn, Δĥn). The path ŵ = {ŵn + μtn}n=0,…,N will be a discrete approximation to a sample path w of Lt.

For the plain Monte-Carlo method, I construct M discrete sample paths {ŵm}m=1,…,M as above and compute HT(ŵm). The plain Monte-Carlo estimate is given by ĉt above.

There exists, however, a faster Monte-Carlo method for simulating NIG-distributed stock price and related stock option prices.

# 4 An Inverse Gaussian Bridge for the Normal Inverse Gaussian Process

A Brownian bridge is often used to simulate processes based on Wiener processes. In this section, though, I would like to describe the construction of an inverse Gaussian bridge and its application to the NIG process.

A bridge Monte-Carlo algorithm for the NIG process Lt proceeds in the following way. First, construct a sample {ĥNm}m=1,…,M from IG(δtN,γ). Second, construct an IG bridge ĥm = (ĥ0m,…,ĥNm), m=1,…M, starting from ĥ0m = 0. Third, for each m = 1,…,M, generate a sample point ŵNm~N(βĥNm,ĥNm) with mean rate β and variance 1. Fourth, construct a bridge ŵm = (ŵ0m,…, ŵNm) at times ĥ0m,…,ĥNm, m = 1,…,M with ŵ0m = 0. This is a standard Brownian bridge, as described in Ribeiro and Webber [14] among other papers. Finally, set Ljm = wjm + μtj for the NIG drift μ. This is a sample path for Lt.

Let’s examine in details now how the IG bridge is constructed and sampled. Suppose that X~FX, Y~FY, and Z~FZ are random variables with densities ƒX, ƒY, and ƒZ, where Z=X+Y. Of particular interest is the case where X, Y, and Z are increments in an IG process over the respective intervals [ti, tj], [tj, tk], and [ti, tk]. Given a sample z of Z, we want to sample X with the correct conditional distribution. The conditional density ƒX|Z of X|Z is

ƒX|Z(x) = ƒX,Y(x,z-x)/ƒZ(z) = ƒX(x)ƒY(z-x)/ƒZ(z)

where ƒX,Y is the joint density function of X and Y and the second equality follows if X and Y are independent, as they are in the given case.

Suppose that X, Y, and Z are generalized inverse Gaussian (GIG) random variables. Let ht ~ GIG(δt,λ,γ), and let τx = [ti,tj], τy = [tj,tk], and τz = [ti,tk]. Further suppose that X = h(τx) ~ GIG(δτx,λ,γ), Y = h(τy) ~ GIG(δτy,λ,γ), and Z = h(τz) ~ GIG(δτz,λ,γ). Then the conditional distribution above becomes

ƒX|Z(x) = (γ/δ)λ(τxτy/τz)-λ Kλ(δτzγ)/(2Kλ(δτxγ)Kλ(δτyγ))\*(xy/z)λ-1exp{-1/2\*δ2 (τx2/x + τy2/y - τz2/z)}

where y = z-x. Since the set of GIG distributions is not closed under convolution, h(τz) is not distributed as h(τx)+ h(τy). However, in two special cases it is true that h(τz) ~ h(τx)+ h(τy). One is δ=0 with λ = t/v, which is the gamma bridge case. The second is when λ = -1/2 for the inverse Gaussian process with the inverse Gaussian bridge density ƒIGX|Z(x)

ƒIGX|Z(x) = δ/√(2π)\* τxτy/τz\*(xy/z)-3/2exp{-1/2\*δ2(τx2/x + τy2/y - τz2/z)}

where y = z-x. Quite remarkably, the latter expression does not depend on γ.

To sample X from this distribution, it is useful to apply Tweedie’s theorem and a result of Michael, Schucany, and Haas [14] (MSH).

One version of Tweedie’s theorem is given in Seshadri [15]. Suppose X ~ IG(δτx,δ-1), Y ~ IG(δτy,δ-1), and Z ~ IG(δτz,δ-1) are inverse Gaussian with Z = X + Y, then

Q = δ2(τx2/X + τy2/Y - τz2/Z)

is chi-squared with one degree of freedom, Q~χ2(1).

In our case, X, Y, and Z do not have these particular distributions, but since γ is not a part of the equation for ƒIGX|Z, it is still possible to apply the proof given in Seshadri to cover this situation. Hence, when X, Y, and Z are increments to an inverse Gaussian process, as in our case, the variable Q in the equation above is χ2(1). It actually appears that this is precisely the most general case to which the Seshadri’s proof applies.

Suppose that q = δ2(τx2/x + τy2/y - τz2/z) is the exponent in the equation for ƒIGX|Z. Set s = y/x, λ = δ2τy2/z and μ = τy/τx. Then

q = λ(s - μ)2/(sμ2) ≡ g(s).

For any q, there are exactly two solutions, s1 and s2, to this equation. These are:

s1 = μ + μ2q/(2λ) – μ/(2λ)\*√(4μλq + μ2q2)

s2 = μ2/s1.

Changing variable in the equation for ƒIGX|Z to S = Y/X | Z, the density of S is

ƒS(s) = √(λ/(2π))/(1+ μ)\*s-3/2(1+s)exp{- λ/2\*(s-μ)2/(sμ2)}

This is a well-defined distribution since μ is the mean of the IG density ƒ(s) = √(λ/(2π))\* \*s-3/2 exp{- λ/2\*(s-μ)2/(sμ2)}.

This is the place where the MSH result can be used. Suppose Q = g(S) where the first derivative g’ of g exists, is continuous, and is non-zero except on a closed set with probability zero. Suppose S has density ƒ(s) and, for a fixed q, suppose q = g(si) for i = 1,…,N. MSH show that a sample for S can be drawn by first making a draw q for Q and then choosing the ith root si with probability pi(q) where

pi(q) = [1 + N∑j=1,j≠i|g’(si)/g’(sj)|\*f(sj)/f(si)]-1

Combining the equations above and applying the MSH result, we get

g’(s1)/g’(s2) = -(μ/s1)2

f(s2)/f(s1) = s12/μ3\*(μ2+s1)/(1+ s1)

Therefore, the smaller root, s1, should be chosen with probability

p1(q) = μ(1+ s1)/((1+ μ)(μ+s1))

The algorithm for sampling values from an IG bridge proceeds as follows. The value h(tj) of an IG process at an intermediate time tj, given h(ti) and h(tk), where ti < tk, is generated as follows:

1. Generate q~ χ2(1) and compute the roots s1 and s2.
2. Set s(tj) = s1 with probability μ(1+ s1)/((1+ μ)(μ+s1)), else set s(tj) = s2.
3. Set h(tj) = h(ti) + (h(tk)- h(ti))/(1+ s(tj)).

It is necessary to make two draws to sample using either the MSH algorithm or using our algorithm for bridge distribution, the first from the χ2(1) distribution and the second a regular Bernoulli draw. Suppose that ŵm = (ûm,ŷm), m = 1,…,M is a sample from the unit square. For each m, ûm is used to construct the χ2(1) variable y as the square of a normal variable by inverse transform. ŷm is then used to determine which of the two roots we select. If ŷm ≤ μ(1+s1)/((1+ +μ)(μ+s1)), choose s1, else choose s2.

# 5 Compound Poisson Process Approximation to the NIG Process

The material in this section is based on the Rydberg [2] article. It should be noted that the methods described below can be applied to any Levy process with characteristic triplet (α,σ,ν). In his 1990 book, Protter [16] gives the representation of an NIG process in terms of a Poisson process:

Lt = γt + ∫|y|<1 y(Nt(dy) – tν(dy)) + ∫|y|≥1 yNt(dy)

where ν(dy) = f(y;α,β,δ)dy and γ = 2*δα/*π \* ∫01 sinh(*βx*) K1(*αx*)dx. If ν is a finite discrete

measure, this becomes a compound Poisson process. Furthermore, notice that ν(Λ) = ∞ if

\_

0 є Λ. From Protter [16], we also get that NtΛ = ∫Λ Nt(dy) is a Poisson process with

\_

intensity ν(Λ) if Λ є R\{0}. Finally, NtΛ1 and NtΛ2 are independent if Λ1 and Λ2 are disjoint subsets of R. These facts can be used to approximate Lt.

Based on the Levy-Khintchine formula for the characteristic exponent of NIG and the formula above, an approximation of Lt could be

Lt(n) = γt + ∑ n i=-n,i≠0 ci(Nt(i) –tλi)

where R\(-εn;εn) is split into 2n disjoint intervals. Let ν(n) be the appropriate discretization of ν corresponding to the λi‘s and ci a jump distributed according to ν(n). This way, the ci‘s are either deterministic or random depending on how ν(n) is chosen. Nt(i) is a Poisson process with intensity λi = ν((ai; ai+1)), such that ∑ n i=-n,i≠0 Nt(i) is a Poisson process with intensity ∑ n i=-n,i≠0 λi = ν(Λn), Λn = (-∞;a-n]U…U(a-1;-εn]U[εn;a1)U…U[an; ∞) equivalent to Λn = R\(-εn;εn). Hence, using Jacod and Shiryaev [17], Chapter VII, Corollary 3.6, we can see that

Lt(n) ~→ L for n → ∞ if ν(n) → ν for n → ∞

since convergence of the characteristic triplet implies weak convergence of the corresponding processes.

However, the approximation above has a flaw since it can be shown that the small jumps are dominating the behaviour of L (see Barndorff-Nielsen [4]). This follows from the fact that for y↓0, f(y;α,β,δ)~ δ/(xy2). Therefore, the approximation Lt(n) of Lt should be corrected by a Wiener term representing the small jumps thrown away in the interval (-εn;εn) around 0. It should be noted that the interval (-εn;εn) ought to be chosen in such a way that it converges to 0 for n → ∞. This leads to the following representation:

Lt(n) = γt + ∑ n i=-n,i≠0 ci(Nt(i) –tλi) + σnWt

where Wt is a standard Wiener process and σn2 = єn∫-εn y2ν(dy). Since σn2 → 0 for n → ∞, we can see that

(γ,σn2,ν(n)) → (γ,0,ν) for n → ∞

from which it follows that

Lt(n) ~→ L for n → ∞

The advantage of this kind of approximation is that the discretization can be made in such a way that ν(n) is proportional to a Dirac measure on intervals (ai; ai+1]. This benefit is particularly appealing from the mathematical finance point of view. The representation above will then resemble other models studied in the literature if the number of jumps is limited enough. Furthermore, the jump sizes of the Poisson process will then be deterministic. This will result in a complete market if there are enough assets to hedge, i.e. as many assets as Poisson processes plus the Wiener process.

When we simulate the approximate NIG process above, we have to decide how to discretize the Levy measure ν. First, we choose ν(n) to be proportional to a Dirac measure on (ai; ai+1]. Second, considerations about the contributions of the variance from the different intervals should be made since the variance is very important in describing the behaviour of the process. Once the discretization has been chosen, we have to find the appropriate jump size on each interval, which is where the point mass of ν(n) will be concentrated. The idea is to solve

ai∫-ai y2ν(dy) = ci2λi for y є (ai; ai+1]

for ci and then to make it the jump size corresponding to y є (ai; ai+1]. The integral represents the variance of a Poisson process with intensity λi.

# 6 Options to Be Priced

I have priced 3 exotic options to examine how the simulation algorithms above work. For all 3 options, there exists a closed-form pricing formula from the Black-Scholes case.

**Barrier Option.** A Barrier option is a contract where the option to exercise depends on the underlying crossing or reaching a given barrier level. I examine the up-and-in barrier option with barrier H on the European call option with strike price K. Its corresponding payoff function is

ΦB(T) = (ST - K)**1**{ ST > K, (MS)T > H}

where (MS)T is the running maximum of Ss up to time t.

**Russian Option**. A Russian option is a contract that gives the holder the right to exercise at any a.s. finite stopping time τ yielding the payoff

ΦR(T) = ε-ητ max(M0, sup0≤u≤τ Su)

where St is the underlying asset, M0 ≥ S0 is the starting maximum, and η > 0 is the punish factor. The computation is also using the first passage function,

vk(y) = Ey[exp{-ητk + R(τk)}]

which is maximized over k, Rt, and τk.

**Arithmetic Asian Option**. An arithmetic Asian option is a contract whose payoff depends on the price of the underlying asset at each of the preset averaging times (usually, equally spaced). The payoff of an arithmetic Asian call option is given as

Φ*A*(T) = ∑nk=1 S(tk)/n – K = 1/n[∑nk=1S(tk) – nK]

The usual averaging frequency is a monthly or weekly frequency; however, any desired frequency less than or equal to the lifetime of the option is theoretically plausible.

# 7 Numerical Results

I price the options described in section 6 using the 3 methods discussed above: plain Monte-Carlo, IG bridge, and compound Poisson process. I generate 100,000 simulations for each method and compare the accuracy of price estimate, the standard deviation, and the runtime. The annualized parameter values used are α=75.49, β=-4.089, δ=3, and μ=0. These parameters correspond to Rydberg’s [2] estimates from the daily returns for Deutsche Bank. It should be noted that any value of μ is compensated away by the compensator term ω. Parameter values implied from option prices would differ from these prices, but since the emphasis in this essay is on numerical algorithms, I proceed with them.

The initial asset value is S0 = $100 and the riskless rate r = 0.1, in agreement with Ribeiro and Webber [1]. The time to maturity for all options is 1 year. The barrier level for the eponymous option is set at H = $105.50, as in Rasmus, Asmussen, and Wiktorsson [3]. Also according to this paper, the punish level for the Russian option is fixed at η = 0.003, which implies τk ≈ 1. The exercise price for the barrier and Asian options is K = $100. For the Asian option, I investigate reset times from quarterly to daily (252). The number of time steps is equal to the number of reset times. In the compound Poisson process simulation, I approximate the NIG with 10 Poisson jumps, as advised in Rydberg [2], and set ε = 0.0001.

All the implementation is done in Matlab 7.2.0 (2006). The program is run on the Intel Pentium 4 processor with 3.00 GHz CPU and 1.00 GB of RAM.

The results obtained are summarized in Tables 1, 2, and 3 on p. 18. The run times for all options are the same because I was pricing them together for every code run. The runtime is given in seconds.

We can see that all 3 methods produce quite accurate results (which can be checked against the numbers in Ribeiro and Webber [1]), and the corresponding standard deviations are approximately the same for them. The biggest difference among these methods is the runtime. IG bridge MC has consistently proved to be the fastest algorithm. Plain MC is somewhat slower, with its performance against IG bridge MC deteriorating with the increase in the number of reset times. The compound Poisson process approximation, however, is extremely slow compared to the other 2 methods and is, therefore, quite inefficient to use. It should also be noted that it is the longest to program, with plain MC and IG bridge MC taking approximately the same amount of time. Its strong part, though, is that it can be used with any Levy process having a standard characteristic triplet (with slight modifications). To further increase efficiency, a number of variance reduction techniques can be employed, e.g. stratified sampling, antithetic random numbers, control variates, conditioning, etc.

# 8 Conclusions

We recommend using the IG bridge methodology introduced in this essay. It is the fastest of the 3 NIG implementation methods studied and gives the same accuracy of estimation and standard deviations as the other two methods. The compound Poisson process is by far the slowest and takes the longest time to program. It has an advantage of being applicable with slight changes to any Levy process having a standard characteristic triplet, though. For each of the three methods, efficiency can be further increased with the help of common variance reduction techniques.

**Table 1. Barrier Option Prices.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **4 resets** | **8 resets** | **12 resets** | **52 resets** | **252 resets** |
| **Plain MC** | 13.1306  (0.0509)  [59] | 13.1141  (0.0512)  [121] | 13.1831  (0.0512)  [166] | 13.1704  (0.0509)  [700] | 13.1118  (0.0509)  [3523] |
| **IG Bridge MC** | 13.1012  (0.0506)  [56] | 13.1143  (0.0509)  [117] | 13.1423  (0.0515)  [156] | 13.1722  (0.0506)  [654] | 13.1579  (0.0509)  [3248] |
| **Compound Poisson Process** | 13.1132  (0.0506)  [23271] | 13.0973  (0.0509)  [29990] | 13.1703  (0.0512)  [30984] | 13.1895  (0.0509)  [36323] | 13.1666  (0.0512)  [56075] |

**Table 2. Russian Option Prices.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **4 resets** | **8 resets** | **12 resets** | **52 resets** | **252 resets** |
| **Plain MC** | 106.4958  (0.0496)  [59] | 107.7504  (0.0500)  [121] | 108.4878  (0.0503)  [166] | 110.0929  (0.0506)  [700] | 110.6000  (0.0512)  [3523] |
| **IG Bridge MC** | 106.4293  (0.0496)  [56] | 107.7160  (0.0500)  [117] | 108.4965  (0.0509)  [156] | 110.0346  (0.0496)  [654] | 110.4024  (0.0506)  [3248] |
| **Compound Poisson Process** | 106.5101  (0.0490)  [23271] | 107.6747  (0.0496)  [29990] | 108.5533  (0.0506)  [30984] | 110.1318  (0.0506)  [36323] | 110.5615  (0.0512)  [56075] |

**Table 3. Arithmetic Asian Option Prices.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **4 resets** | **8 resets** | **12 resets** | **52 resets** | **252 resets** |
| **Plain MC** | 8.5519  (0.0326)  [59] | 7.7913  (0.0297)  [121] | 7.5909  (0.0294)  [166] | 7.1586  (0.0272)  [700] | 7.0154  (0.0272)  [3523] |
| **IG Bridge MC** | 8.5744  (0.0326)  [56] | 7.7754  (0.0297)  [117] | 7.6267  (0.0288)  [156] | 7.1287  (0.0272)  [654] | 7.0490  (0.0272)  [3248] |
| **Compound Poisson Process** | 8.5854  (0.0323)  [23271] | 7.7930  (0.0297)  [29990] | 7.6688  (0.0291)  [30984] | 7.1765  (0.0272)  [36323] | 7.0840  (0.0269)  [56075] |

# 9 References

[1] Ribeiro, C. & Webber, N. (2003). A Monte Carlo Method for the Normal Inverse Gaussian Option Valuation Model Using an Inverse Gaussian Bridge. *Computing in Economics and Finance, no. 5.* Society for Computational Economics.

[2] Rydberg, T.H. (1997). The Normal Inverse Gaussian Levy Process: Simulation and Approximation. *Communications in Statistics: Stochastic Models*, 13(4): 887-910.

[3] Rasmus, S., Asmussen, S., & Wiktorsson, M. (2004). Pricing of Some Exotic Options with NIG-Levy Input. *Computational Science – ICCS 2004.* Springer Berlin/ Heidelberg.

[4] Barndorff-Nielsen, O.E. (1995). Normal Inverse Gaussian Processes and the Modelling of Stock Returns. *Research Report 300*. University of Aarhus.

[5] Barndorff-Nielsen, O.E. (1997). Processes of Normal Inverse Gaussian Type. *Finance and Statistics*, 2(1): 41-68.

[6] Carr, P., Geman, H., Madan, D.B., & Yor, M. (2001). Stochastic Volatility for Levy Processes.

[7] Albrecher, H. & Predota, M. (2004). On Asian Option Pricing for NIG Levy Processes. *Journal of Comp. and Appl. Math*.

[8] Beaglehole, D., Dybvig, P., & Zhou, G. (1997). Going to Extreme: Correcting Simulation Bias in Exotic Option Valuation. *Financial Analysts Journal*, January/February: 62-68.

[9] Ribeiro, C. & Webber, N. (2004). Valuing Path Dependent Options in the Variance-Gamma Model by Monte Carlo with a Gamma Bridge. *Journal of Computational Finance,* 7 (2): 81-100.

[10] Bertoin, J. (1996). *Levy Processes*. Cambridge University Press.

[11] Sato, K-I. (1999). *Levy Processes and Infinitely Divisible Distributions*. Cambridge University Press.

[12] Madan, D.B., Carr, P.P., & Chang, E.C. (1998). The Variance Gamma Process and Option Pricing. *European Finance Review*, 2:79-105.

[13] Eberlein, E. & Raible, S. (1999). Term Structure Models Driven by General Levy Processes. *Mathematical Finance*, 9:31-35.

[14] Michael, J., Schucany, W., & Haas, R. (1976). Generating Random Variates Using Transformations with Multiple Roots. *The American Statistician*, 30:88-89.

[15] Seshadri, V. (1993). *The Inverse Gaussian Distribution*. Clarendon Press.

[16] Protter, P. (1990). Stochastic Integration and Differential Equations: a New Approach. *Applications of Mathematics*, Vol. 21. Springer-Verlag, Berlin.

[17] Jacod, J. & Shiryaev, A.N. (1987). Limit Theorems for Stochastic Processes. *Grundlehren der Mathematischen Wissenschaften*, Vol. 288. Springer-Verlag, Berlin.

# Appendix: Matlab Code

plain\_mc.m

randn('state', sum(100\*clock));

% NIG parameters

alpha = 75.49;

beta = -4.089;

delta = 3;

mu = 0;

gamma = sqrt(alpha^2-beta^2); % gamma

omega = mu+delta\*gamma-delta\*sqrt(alpha^2-(1+beta)^2); % compensator term

%simulation parameters

M = 100000; % sample paths

N = 4; % time steps = reset times

% stock and option parameters

S0 = 100; %initial stock price

K = 100; % strike price

r = 0.1; % interest rate

T = 1; % time to maturity in years

bar = 105.5; % barrier for the barrier option

eta = 0.003; % punish factor for the Russian option

tau = 1; % approximate stopping time for the Russian option

m0 = 100; % starting maximum for the Russian option

% algorithm initialization

L = zeros(M,N+1); % approximation to the NIG process

S = zeros(M,N+1); % approximation to the stock price process

S(:,1) = S0;

payoff\_bar = zeros(M,1); %payoffs for the barrier option

payoff\_as = zeros(M,1); %payoffs for the Asian option

payoff\_rus = zeros(M,1); %payoffs for the Russian option

% path generation

for i=1:M

maxS = zeros(1,N+1); % max stock price for the given path

maxS(1) = S0;

for j=1:N

deltah = igrnd(delta\*T/N,gamma); %igrnd2(1,delta\*T/N,gamma); %increment in the subordinator process

deltaw = normrnd(beta\*deltah,sqrt(deltah)); % increment in the subordinated Brownian motion

L(i,j+1) = L(i,j)+ deltaw + mu\*T/N; % increment in the NIG process

S(i,j+1) = S0\*exp(r\*j\*T/N+L(i,j+1)-omega\*j\*T/N); % increment in the stock price process

maxS(j+1)=max(S(i,1:(j+1))); % increment in the max stock price vector

end

% Barrier option payoff for the given path

if (maxS(N+1)>bar) && (S(i,N+1)>K)

payoff\_bar(i) = S(i,N+1) - K;

else payoff\_bar(i) = 0;

end

% Asian option payoff for the given path

payoff\_as(i) = max(mean(S(i,2:(N+1)))-K,0);

% Russian option payoff for the given path

payoff\_rus(i) = exp(-eta\*tau)\*max(m0,maxS(N+1));

end

% averaging the option payoffs over all paths

fin\_payoff\_bar = mean(payoff\_bar);

fin\_payoff\_rus = mean(payoff\_rus);

fin\_payoff\_as = mean(payoff\_as);

% calculating the option prices at time 0

price\_bar = fin\_payoff\_bar\*exp(-r\*T);

price\_rus = fin\_payoff\_rus\*exp(-r\*T);

price\_as = fin\_payoff\_as\*exp(-r\*T);

igbridge\_mc.m

randn('state', sum(100\*clock));

% NIG parameters

alpha = 75.49;

beta = -4.089;

delta = 3;

mu = 0;

gamma = sqrt(alpha^2-beta^2); % gamma

omega = mu+delta\*gamma-delta\*sqrt(alpha^2-(1+beta)^2); % compensator term

%simulation parameters

M = 100000; % sample paths

N = 4; % time steps = reset times

% stock and option parameters

S0 = 100; %initial stock price

K = 100; % strike price

r = 0.1; % interest rate

T = 1; % time to maturity in years

bar = 105.5; % barrier for the barrier option

eta = 0.003; % punish factor for the Russian option

tau = 1; % approximate stopping time for the Russian option

m0 = 100; % starting maximum for the Russian option

% algorithm initialization

h = zeros(M,N+1); % subordinator process

w = zeros(M,N+1); % subordinated Brownian motion

L = zeros(M,N+1); % NIG process

S = zeros(M,N+1); % stock price process

S(:,1) = S0;

payoff\_bar = zeros(M,1); %payoffs for the barrier option

payoff\_as = zeros(M,1); %payoffs for the Asian option

payoff\_rus = zeros(M,1); %payoffs for the Russian option

% path generation

for i=1:M

h(i,N+1) = igrnd(delta\*T,gamma); %igrnd2(1,delta\*T,gamma); % terminal value in the subordinator process

w(i,N+1) = normrnd(beta\*h(i,N+1),sqrt(h(i,N+1))); % terminal value in the subordinated BM

L(i,N+1) = w(i,N+1) + mu\*T; % terminal value in the NIG process

S(i,N+1) = S0\*exp(r\*T+L(i,N+1)-omega\*T); % terminal value in the stock price process

maxS = zeros(1,N+1); % maximum stock price for the given path

maxS(1) = S0;

for j=1:(N-1)

h(i,j+1) = igbridge(h(i,j),h(i,N+1),T/N,(N-j)\*T/N,delta); % next value for the subordinator process using the IG bridge

w(i,j+1) = brbridge(w(i,j),w(i,N+1),h(i,j),h(i,j+1),h(i,N+1)); % next value for the subordinated BM using the Brownian bridge

L(i,j+1) = w(i,j+1) + mu\*j\*T/N; % next value for NIG process

S(i,j+1) = S0\*exp(r\*j\*T/N+L(i,j+1)-omega\*j\*T/N); % next value for stock price process

maxS(j+1)=max(S(i,1:(j+1))); % next value for the max stock price

end

% Barrier option payoff for the given path

maxS(N+1) = max(maxS(N),S(i,N+1));

if (maxS(N+1)>bar) && (S(i,N+1)>K)

payoff\_bar(i) = S(i,N+1) - K;

else payoff\_bar(i) = 0;

end

% Asian option payoff for the given path

payoff\_as(i) = max(mean(S(i,2:(N+1)))-K,0);

% Russian option payoff for the given path

payoff\_rus(i) = exp(-eta\*tau)\*max(m0,maxS(N+1));

end

% averaging the option payoffs over all paths

fin\_payoff\_bar = mean(payoff\_bar);

fin\_payoff\_rus = mean(payoff\_rus);

fin\_payoff\_as = mean(payoff\_as);

% calculating the option prices at time 0

price\_bar = fin\_payoff\_bar\*exp(-r\*T);

price\_rus = fin\_payoff\_rus\*exp(-r\*T);

price\_as = fin\_payoff\_as\*exp(-r\*T);

igbridge.m

function hmid = igbridge(hbeg,hend,taux,tauy,delta)

% generates an intermediate value for the Inverse Gaussian bridge

% conditioning on the initial and final values

% hbeg = initial value of the IG process

% hend = final value of the IG process

% taux = amount of time between the initial and intermediate values

% tauy = amount of time between the intermediate and final values

% delta = delta parameter of the IG process

lam = delta^2\*tauy^2/hend; % lambda

mu = tauy/taux; % mu

q = chi2rnd(1);

s1 = mu + mu^2\*q/(2\*lam) - mu/(2\*lam)\*sqrt(4\*mu\*lam\*q+mu^2\*q^2); % 1 of 2 possible solutions for the equation for q

p1 = mu\*(1+s1)/((1+mu)\*(mu+s1)); %mu/(mu+s1); % probability of s1

U = rand;

H = (U<=p1);

s = s1\*H + mu^2/s1\*(1-H); % 2nd solution is mu^2/s1

hmid = hbeg + (hend-hbeg)/(1+s);

brbridge.m

function wmid = brbridge(wbeg,wend,tbeg,tmid,tend)

% generates an intermediate value for the Brownian bridge conditioning on

% the initial and final values

% wbeg = initial value for the BM

% wend = final value for the BM

% tbeg = time at which the initial BM value has been estimated

% tmid = time at which the intermediate BM value is estimated

% tend = time at which the final BM value has been estimated

wmid = (tend-tmid)/(tend-tbeg)\*wbeg + (tmid-tbeg)/(tend-tbeg)\*wend + sqrt((tmid-tbeg)\*(tend-tmid)/(tend-tbeg))\*randn;

igrnd.m

function s = igrnd(delta,gamma)

% Inverse Gaussian generator based on the Michael, Schucany, and Haas

% (1976) algorithm

% delta = delta from NIG

% gamma = gamma from NIG

mu = delta/gamma; % reparameterization

lam = delta^2; % reparameterization

q = chi2rnd(1);

s1 = mu + mu^2\*q/(2\*lam) - mu/(2\*lam)\*sqrt(4\*mu\*lam\*q+mu^2\*q^2); % 1 of 2 possible solutions for q

p1 = mu/(mu+s1); % probability of s1

U = rand;

H = (U<=p1);

s = s1\*H + mu^2/s1\*(1-H); % 2nd solution is mu^2/s1

poisson\_mc.m

randn('state', sum(100\*clock));

% NIG parameters

alpha = 75.49;

beta = -4.089;

delta = 3;

mu = 0;

%simulation parameters

poissonNum = 10; % number of Poisson jumps

epsJump = 0.0001; % epsilon

infJump = 0.1+epsJump; % jump at infinity

M = 100000; % sample paths

N = 4; % time steps

% stock and option parameters

K = 100; % exercise price

r = 0.1; % riskless rate

bar = 105.50; % barrier for the Barrier option

eta = 0.003; % punish factor for the Russian option

m0 = 100; % starting maximum for the Russian option

T = 1; % lifetime of all 3 options

tau = 1; % optimal exercise time for the Russian option

% algorithm initialization

payoff\_bar = zeros(M,1);

payoff\_as = zeros(M,1);

paroff\_rus = zeros(M,1);

% path generation

for i=1:M

S = GenerateStockNIG (100, r, poissonNum, epsJump, infJump, N, T, alpha, beta, delta, mu);

% maximum stock price

maxS = zeros(1,N);

for j=1:(N+1)

maxS(j)=max(S(1:j));

end

% barrier option payoff

if (maxS(N+1)>bar) && (S(N+1)>K)

payoff\_bar(i) = S(N+1) - K;

else payoff\_bar(i) = 0;

end

% arithmetic Asian Option payoff

payoff\_as(i) = max(mean(S(2:(N+1)))-K,0);

% Russian Option payoff

payoff\_rus(i) = exp(-eta\*tau)\*max(m0,maxS(N+1));

end

% averaging the option payoffs over all paths

fin\_payoff\_bar = mean(payoff\_bar);

fin\_payoff\_rus = mean(payoff\_rus);

fin\_payoff\_as = mean(payoff\_as);

% calculating the option prices at time 0

price\_bar = fin\_payoff\_bar\*exp(-r\*T);

price\_rus = fin\_payoff\_rus\*exp(-r\*T);

price\_as = fin\_payoff\_as\*exp(-r\*T);

CalcGamma.m

function gamma = CalcGamma(alpha, beta, delta, mu)

% calculates the gamma part of the NIG approximation

CalcSinhBesselY1(0, alpha, beta);

gamma = quad(@CalcSinhBesselY1, 0,1);

gamma = 2\*delta\*alpha/pi\*gamma + mu;

CalcJumpSize.m

function res = CalcJumpSize (poissPoint, jumpDelta, lambdaI)

% calculates the jump size for the compund Poisson simulation

res = quad(@CalcNIGDensityXX, poissPoint, poissPoint+jumpDelta);

res = sign(poissPoint)\*sqrt(res/lambdaI);

CalcLambda.m

function lambda = CalcLambda(poissPoint, jumpDelta)

% calculates lambda for the Poisson NIG approximation

lambda = quad(@CalcNIGDensity, poissPoint, poissPoint+jumpDelta);

CalcNIGDensity.m

function niu = CalcNIGDensity(x, alpha, beta, delta)

% evaluates the NIG density at a particular point

persistent a

persistent b

persistent d

if nargin == 4

a = alpha;

b = beta;

d = delta;

else

N = size(x, 2);

niu = zeros(1, N);

for i =1:N

niu(i) = (d\*a)/(pi\*abs(x(i)))\*exp(b\*x(i))\*besselk(1, a\*abs(x(i)));

end

end

CalcNIGDensityXX.m

function niu = CalcNIGDensityXX(x, alpha, beta, delta)

% evaluates the NIG density at a particular point times the square value of

% that point

persistent a

persistent b

persistent d

if nargin == 4

a = alpha;

b = beta;

d = delta;

else

N = size(x, 2);

niu = zeros(1, N);

for i =1:N

niu(i) = (d\*a)/(pi\*abs(x(i)))\*exp(b\*x(i))\*besselk(1, a\*abs(x(i)))\*x(i)\*x(i);

end

end

CalcOmega.m

function omega = CalcOmega(alpha, beta, delta, mu)

% calculates omega for the NIG

gamma = sqrt (power(alpha,2) - power(beta ,2));

omega = mu + delta\*(gamma - sqrt (power(alpha,2) - power((beta +1),2)));

CalcSigma.m

function sigma = CalcSigma (eps)

% calculates sigma for the NIG

sigma = quad(@CalcNIGDensityXX, -eps, -eps/10000)+quad(@CalcNIGDensityXX, eps/10000, eps);

CalcSinhBesselY1.m

function niu = CalcSinhBesselY1(x, alpha, beta)

% calculates the hyperbolic sine times the value of the Bessel function;

% used in Poisson approximation

persistent a

persistent b

if nargin == 3

a = alpha;

b = beta;

else

N = size(x, 2);

niu = zeros(1, N);

for i =1:N

niu(i) = sinh(b\*x(i))\* besselk (1,a\*x(i));

end

end

GenerateNIG.m

function NIG = GenerateNIG(poissonNum, epsJump, infJump, timeSlicesN, T, alpha, beta, delta, mu)

% generates the NIG-distributed sample path

% poissonNum = number of Poisson jumps to approximate the NIG distribution

% epsJump = epsilon

% infJump = jump at infinity

% timeSlicesN = number of time steps

% T = time length of the path

% alpha, beta, delta, mu = NIG parameters

poissStep = (infJump - epsJump)/poissonNum;

posA = epsJump + poissStep\*[0:poissonNum];

negA = -posA;

CalcNIGDensity(0, alpha, beta, delta);

CalcNIGDensityXX(0, alpha, beta, delta);

% lambdas and jump sizes for the compound Poisson part

for i=1:poissonNum

posLambdas(i) = CalcLambda(posA(i), poissStep);

negLambdas(i) = CalcLambda(negA(i+1), poissStep);

posJumps(i) = CalcJumpSize(posA(i), poissStep, posLambdas(i));

negJumps(i) = CalcJumpSize(negA(i+1), poissStep, negLambdas(i));

end

% input for the gamma part

deltaT = T / timeSlicesN;

gamma = CalcGamma(alpha, beta, delta, mu);

% input for the BM part of the approximation

sigma = CalcSigma(epsJump);

normalVals = randn(1,timeSlicesN+1);

normalVals(1) = 0;

% Compound Poisson element generation

posPoissons(1) = 0;

negPoissons(1) = 0;

for i = 1:timeSlicesN

posJumpVal = dot(posJumps,(poissrnd(posLambdas\*deltaT) - deltaT\*posLambdas));

negJumpVal = dot(negJumps, (poissrnd(negLambdas\*deltaT) - deltaT\*negLambdas));

posPoissons(i+1) = posPoissons(i)+posJumpVal;

negPoissons(i+1) = negPoissons(i)+negJumpVal;

end

gammaProc = gamma\*deltaT\*[0:timeSlicesN]; % gamma part of the approximation

BMProc = cumsum(normalVals\*sqrt(deltaT)\*sigma); % BM part of the approximation

CompPoissonProc = posPoissons+negPoissons; % compound Poisson part of the approximation

NIG = gammaProc+BMProc+CompPoissonProc; % approximated NIG path

GenerateStockNIG.m

function res = GenerateStockNIG(S0, r, poissonNum, epsJump, infJump, timeSlicesN, T, alpha, beta, delta, mu)

% calculates the stock price path based on the NIG process

% S0 = initial stock price

% r = riskless rate

% poissonNum = number of Poisson jumps to approximate the NIG distribution

% epsJump = epsilon

% infJump = jump at infinity

% timeSlicesN = number of time steps

% T = time length of the path

% alpha, beta, delta, mu = NIG parameters

nig = GenerateNIG(poissonNum, epsJump, infJump, timeSlicesN, T, alpha, beta, delta, mu);

w = CalcOmega(alpha, beta, delta, mu);

deltaT = T / timeSlicesN;

res = S0\*exp((r - w)\*[0:timeSlicesN]\*deltaT + nig);

1. Later we may assume that Δt = T/N is a constant. This assumption can easily be relaxed. [↑](#footnote-ref-1)