Practical Session - Python

Working together is fine. There are plenty of questions here don't feel you have to all of them. If something seems too easy or too difficult, feel free to skip it (some questions do build on preceding ones, though). Note throughout the tutorials, you can pass $noisy\_fail()$ as the value of the `NagError` when calling any of the NAG routines. Be sure to check `fail.code` for Question 3.

For these questions, we will be using pylab for plotting. You can access it via

```
from pylab import plot,show
plot(x,y)
show()
```

1 Question 1 (Beginner)

Write a function `AmericanOption` which takes the following inputs

`S` the initial stock price	`K` the strike price	`T` the maturity
`r` the interest rate	`q` the dividend rate	`sigma` the volatility

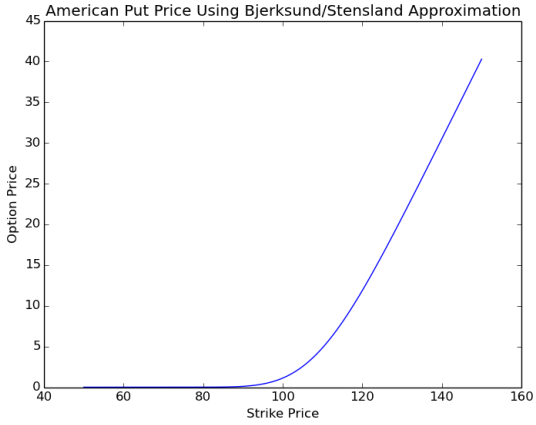
1. The function should call `s30qcc` to return the price of a **American Call** option. Use your function to price a call option with the following parameters

$$\begin{array}{lll} S = 110 & K = 100 & T = .25 \\ r = 0.08 & q = 0.12 & \sigma = 0.2 \end{array}$$

2. Now modify the above function so that it takes a numpy.array for `K` and `T`. Also add the option type (Call/Put) as an optional argument to your function. Use this new function and pylab to plot an **American Put** Price vs. Strike for `K` from 50 to 150. Use the same parameters above for `S, T, r, q` and σ .

1.1 Answer

1. You should get an answer of 10.33403003.
2. Your plot should look like



2 Question 2 (Intermediate)

Refer to the documentation for `c05awc`. We are now going to write a program to compute the implied volatility for a given call option price and set of parameters. In other words, given a call option price C and values of S, K, T, r, q, find the value of σ so that the option pricing formula from question 1 gives the price C. Modify the function `AmericanOption` from Question 1 to have the prototype

```
def BS_Callback(sigma, comm):
```

Use global variables to pass the option parameters to `BS_Callback`.

In `main`, call `c05awc` and pass it `BS_Callback` as the function of which the zero is to be computed. You will need to cast the callback via the function `NAG_C05AWC_FUN` available in `nag4py.c05`.

```
from nag4py.c05 import NAG_C05AWC_FUN
c_callback = NAG_C05AWC_FUN(BS_Callback)
```

Set `eps = eta = 1.0e-6` and set `nfmax = 1500`.

1. Use your program to compute the implied volatility for a (target) call option price of 12.35008695 and

$$S = 100 \qquad K = 90 \qquad T = 1.5 \qquad r = 0.03 \qquad q = 0.015$$

   Use an initial guess of $\sigma = 0.15$.

Note: `c05axc` is another NAG routine that solves this same problem using reverse communication.

2.1 Answer

1. You should get an answer of 0.09000096.

3 Question 3 (Advanced)

We now turn to the Markowitz model of portfolio optimization. Mainly we look to minimize the risk of a portfolio

$$\min_x x^T \Sigma x$$

for a given level of expected returns, $R^T x = \mu$. For our expected returns and covariance matrix, use

$$R = [\,-.068 \quad .103 \quad .087 \quad .135 \quad .07\,]$$

$$\texttt{Covariance} = \begin{pmatrix} 5.85 & -2.58 & 1.41 & -3.13 & 1.21 \\ -2.58 & 3.97 & -2.00 & 1.71 & -3.21 \\ 1.41 & -2.00 & 5.15 & 4.53 & 1.51 \\ -3.13 & 1.71 & 4.53 & 7.95 & 5.43 \\ 1.21 & -3.21 & 1.51 & 5.43 & 7.65 \end{pmatrix}$$

Create a python function to solve the above problem for a given level of return using `e04ncc`. The return value will be the minimum risk of the portfolio. The signature of your function should look like

```
def solveMarkowitz(mu):
```

You will need to import the following:

```
from nag4py.e04 import e04ncc,Nag_E04_Opt,nag_opt_init
from nag4py.util import quiet_fail,Nag_Comm,nag_int_type,Nag_QP1,Nag_NoPrint
from numpy import empty,array
```

Our quadratic programming problem will have 2 linear constraints

- The portfolio must sum to one $\sum_i x_i = 1$
- We look for the best allocation given an expected return $R^T x = \mu$

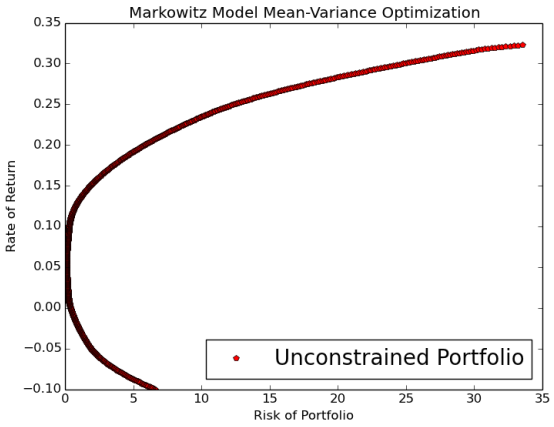
Now create a numpy array `bl` of lower bounds for x_i and give each element a value of `-1.0`. Create a similar array `bu` for the upper bounds and give upper bounds of 1.0. At the end of these arrays input the linear constraints lower and upper bounds. Also create the constraint matrix `A` and an initial guess `X`.

In your function, create a `NAG_E04_Opt` structure `options`, initialize it with `nag_opt_init` and set `options.prob = Nag_QP1`, `options.print_level = Nag_NoPrint`, and `options.list = 0`. This will turn off printing. We won't be using any of the other parameters, but they need to be initialized (`cvec = empty(0), b = empty(0), kx = empty(n,dtype = nag_int_type)`), and `comm = Nag_Comm()`). Now call `e04ncc` and, if there is no NAG error, return `2*objf` (the NAG function minimizes $\frac{1}{2}x^T \Sigma x$).

1. What is the minimum risk for $\mu = .1$?
2. Let $\mu \in [-.1, .4]$ and plot the efficient frontier.
3. Are you able to add a sector constraint $x_2 + x_3 \leq .5$ and plot both efficient frontiers on the same graph?

3.1 Answer

1. You should get an answer of 0.301877133.
2. Your graph should look like



3. Your graph should look like