PAG Test

All code produced to answer questions should be written with simplicity in mind. Well structured and commented code is necessary. Any Python code should follow the PEP8 standard. Make sure your Python code at a minimum produces no errors when parsing with flake8.

Question 1

To answer this question you need to manipulate a sparse matrix in a combination of Python and either C, C++ or Fortran 90. Note, parts **a**., **b**. and **c**. are not independent questions, so you should write **a**. in a way that most easily facilitates an answer to **b**.

a. Using C, C++ or Fortran 90 write a procedure that computes a sparse matrix vector multplication, where the sparse matrix is in compressed sparse row (CSR) format. Refer to http://www-users.cs.umn.edu/~saad/books.html "Iterative methods for sparse linear systems (2nd edition), if you are unfamiliar with this format. The vector should be assumed as dense. You should write your own algorithm and not use a pre-existing library.

b. The following Python code produces a sparse matrix in CSR format. This format is described by the dimensions of the sparse matrix and the three vectors (referred to as data, indices and ptr, in the code below). Ensure you can call the procedure written in **a**. using Python, for example you could use Cython or F2py to produce a shared library. You should then pass the the necessary information from Python to the procedure you created in **a.** The result should be accessable from Python. Write a test procedure in Python to test your code. You should use scipy's sparse matrix algorithms here.

c. Provide clear instructions on how to compile and test your code.

```
"""Python code to setup sparse matrix vector computation."""

import numpy as np
from scipy import sparse

# Set random seed
np.random.seed(12345)
```

```
# Randomly Generate a sparse matrix in compressed sparse
# row (CSR) format
sp_mat = sparse.random(50, 30, format='csr')

# The information required to describe the CSR matrix is
# contained in three vectors
# Non-zero real valued elements
data = sp_mat.data

# Column indices
indices = sp_mat.indices

# Row pointers
ptr = sp_mat.indptr

# Random vector used in sparse matrix vector calculation
xvec = np.random.randn(30)

# Sparse matrix vector computation
bvec = sp_mat * xvec

# Call your own compiled function and test your solution against above.
```

Question 2

Now using C, C++ or Fortran 90 write your own procedure to add two sparse matrices, in CSR format, together. The matrix that is the solution to this problem is also to be in a CSR format. As before, compile your code to be called from Python. Create 2 sparse matrices in CSR format in Python and call your new procedure to add them together. The result of this procedure should be accessable from Python. Write a test function in Python to check the your function produces the correct result.

Question 3

In this section, write code in C, C++ or Fortran 90. You should write a driver routine that calls your function. Please include instructions to compile your code. Remember to focus on simplicity, rather than speed. Note that, if $A \in \mathcal{R}^{m \times n}$ then A is an $(m \times n)$ matrix of reals. The arguments of your function are vectors and matrices; use double precision. Let y_t be the t^{th} element of $y \in \mathcal{R}^n$, $Z^{1 \times m}, H \in \mathcal{R}, T \in \mathcal{R}^{m \times m}, G \in \mathcal{R}^{m \times q}$ and $Q \in R^{r \times r}$ (where Q is a symmetric positive definite matrix). Given $a_1 \in \mathcal{R}^m$ and $P_1^{m \times m}$, where P_1 is symmetric positive definite, write a function to compute,

for $t = 1, 2, \ldots, n$,

$$\begin{aligned}
\nu_t &= y_t - Z a_t \\
f_t &= Z P_t Z^T + h \\
K_t &= T P_t Z / f_t \\
L_t &= T - K_t Z \\
a_{t+1} &= T a_t + K_t \nu_t \\
P_{t+1} &= T P_t L_t + G Q G^T
\end{aligned}$$

Have your function return the calculation

$$logL = -0.5 \times \left(\sum_{t=1}^{n} \left(\log(2\pi) + \log(f_t) + \frac{\nu_t^2}{f_t} \right) \right).$$

Feel free to use established linear algebra libraries to aid in these calculations. For your test code let $Z = \begin{bmatrix} 1 & 0 \end{bmatrix}, h = 1, T = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, G = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, Q = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.1 \end{bmatrix}, a_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 1 \times 10^7 & 0 \\ 0 & 1 \times 10^7 \end{bmatrix}$ and let $y = (1120, 1160, 963, 1210, 1160, 813, 1230, 1370)$.