Lanczos

May 18, 2018

1 Lanczos

by Iliya Sabzevari

The Lanczos algorithm is an efficient way to decompose a large, Hermitian/symmetric matrix into a tridiagonal matrix that can then be solved easily. I leave the nuts and bolts of the algorithm to wikipedia which does a great job of outlining the details. The program can be seen below.

```
In [1]: import numpy as np
        import scipy.linalg

        def lanczos(H_func,dimension):
                guess = np.random.rand(dimension)
                q0 = np.zeros(dimension)
                q1 = guess/np.linalg.norm(guess)
                a = []
                b = []
                beta = 0.
                for i in range(dimension):
                        r = H_func(q1) - beta*q0

                        alpha = np.dot(q1.T,r)
                        r = r - alpha*q1
                        beta = np.linalg.norm(r)

                        q0 = q1
                        q1 = r/beta

                        a.append(alpha)
                        if i < dimension-1:
                                b.append(beta)
                w,v = scipy.linalg.eigh_tridiagonal(a,b)
                w = np.sort(w)
                return w[0]
```

I tested the program on a large, diagonally dominant matrix (the same test I perfromed on the davidson algorithm)

```
In [2]: import math
        import numpy as np
        import lanczos as L
        import time

        #build diagonally dominant Hamiltonian and Direct method function of Hamiltonian
        n = 1500
        H = np.zeros((n,n))


        r = range(n)        #take diagonal elements to be increasing integer values
        for i in r:
                H[i,i] = i+1        #take off diagonal elements to be decreasing in order of mag
                for j in r[(i+1):]:
                        H[i,j] = (10**(i-j+1))
        H = (H.T + H)/2

        def A(v):               #make direct method function
                return np.dot(H,v)

        #Lanczos
        start_Lanczos = time.time()

        E = L.lanczos(A,n)

        end_Lanczos = time.time()

        start_numpy = time.time()

        print("Lanczos = ", E,":",end_Lanczos-start_Lanczos, "seconds")

        #Numpy
        start_numpy = time.time()

        E,V = np.linalg.eig(H)
        E = np.sort(E)

        end_numpy = time.time()

        print("Numpy = ", E[0],":",end_numpy-start_numpy,"seconds")

Lanczos =  0.7825165335918781 : 4.232024669647217 seconds
Numpy =  0.7825165335930493 : 14.23384690284729 seconds
```

Pretty good, but not as good as the Davidson algorithm.