Problem 1: QR iteration with Shifts

Check problem1.py

Result for A_1:

Computed eigenvalues: [10.56208918 -3.00003534 -1.56205384]

Actual eigenvalues: [11. -2. -3.]

Result for A_2:

Computed eigenvalues: [7.28798259 0.6316235   2.08039391]

Actual eigenvalues: [7.28799214 2.13307448 0.57893339]

The eigenvalues are almost the same.

Problem 2: Lanczos Iteration and Convergence of Ritz Values

a) $H=Q^TAQ$ where A is symmetric and real-valued. Q is orthogonal

Thus, $Q^T = Q^{-1}$

Thus, $H = Q^{-1}AQ$, which is a similarity transformation

Therefore A is symmetric ⇒ H is symmetric

H is upper Hessenberg and symmetric

⇒ H is tridiagonal

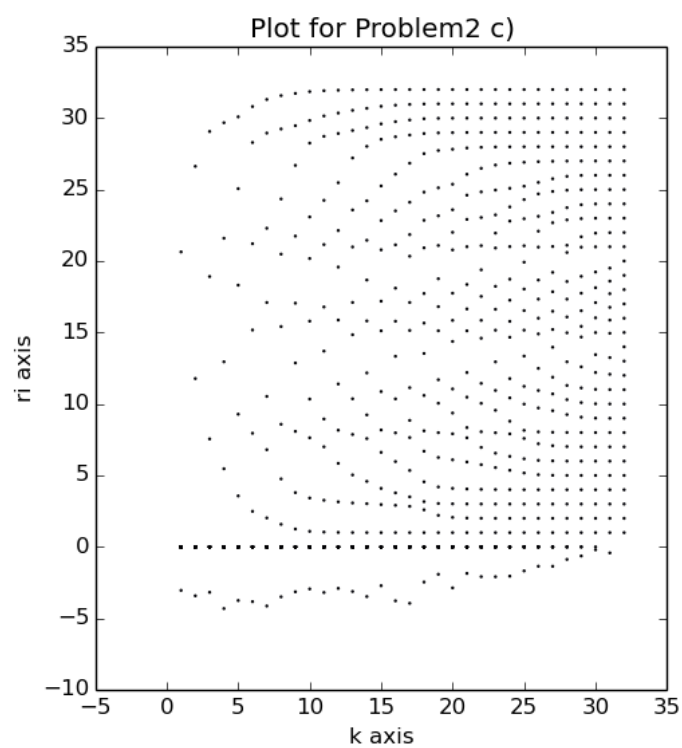
⇒ If A is symmetric and real valued, Arnoldi iteration reduce to lanczos iteration and the Hessenberg matrix H generated by Arnoldi iteration become symmetric and tridiagonal

b) Check problem2_b.py and problem2_b_test.py

$|| QQ^T - I || = 4.06188221315e-14$

$|| Q^TAQ - H || / ||A||= 1.99126150471e-14$

c) Check problem2_c.py



Problem 3: Reduction to Hessenberg form

a) In order to annihilate rows 3, …, n of the first column of A. The Household reflector H has the form,

$$H = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots \end{bmatrix}$$

which has n rows and n columns and where "..." are all non-zero entries

Then

$$HA = \begin{bmatrix} x & x & \dots & x \\ x & x & \dots & x \\ 0 & x & \dots & x \\ \dots & \dots & \dots & \dots \\ 0 & x & \dots & x \end{bmatrix}$$

Since H is symmetric, $H = H^T$

$B = HAH^T = HAH$

$$= \begin{bmatrix} x & x & \dots & x \\ x & x & \dots & x \\ 0 & x & \dots & x \\ \dots & \dots & \dots & \dots \\ 0 & x & \dots & x \end{bmatrix} \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & x & \dots & x \\ 0 & x & \dots & x \\ \dots & \dots & \dots & \dots \\ 0 & x & \dots & x \end{bmatrix} = \begin{bmatrix} x & x & \dots & x \\ x & x & \dots & x \\ 0 & x & \dots & x \\ \dots & \dots & \dots & \dots \\ 0 & x & \dots & x \end{bmatrix}$$

b) Check problem3_b.py
c) Check problem3_c.py

relative error: 3.60274e-16

d) Check problem3_d.py

relative error: 4.62765e-16

and the Hessenberg matrix U in this case is a tridiagonal matrix

Problem 4: Newton's method in 1D

a)
   i) $g(x^*) = x^* - f(x^*)/d$. For convergence we need to find derivative $g'(x^*)=1-f'(x^*)/d$. For local convergence the condition is

$$| 1 - f'(x^*)/d | < 1$$

$$\Rightarrow -1 < 1 - f'(x^*)/d < 1$$

$$\Rightarrow 0 < f'(x^*)/d < -2$$

   ii) In general, the convergence rate will be linear with the constant $C=|1-f'(x^*)/d|$
   iii) For the quadratic convergence we need $1-f'(x^*)/d=0$ which implies $d = f'(x^*)$

b) Check problem4_b.py
   i) The result for $f(x) = x^2 - 1$:

      computed root: 1.0

      rate of convergence: 2.0004601301

   ii) The result for $(x) = (x - 1)^4$:

      computed root: 1.0

      rate of convergence: 1.0

   iii) The result for $f(x) = x - \cos(x)$:

      computed root: 0.739085133215

      rate of convergence: 1.99803192973

Problem 5: Newton's method for a system

a) Check problem5_a.py
b) Check problem5_b.py
   Set x_0 be [1,1,1]
c) Check problem5_c.py

random x, y, z = [1.76405235 0.40015721 0.97873798]
r, theta, phi = [2.05668046 1.07482931 0.22306486]
Final relative residual: 1.23687e-16
Final relative error: 0
random x, y, z = [1.62434536 -0.61175641 -0.52817175]
r, theta, phi = [1.8143068   23.26655289 53.04688737]
Final relative residual: 2.17901e-15
Final relative error: 21.8967
random x, y, z = [-0.41675785 -0.05626683 -2.1361961]
r, theta, phi = [-2.17719701   6.47756318 18.9837553]
Final relative residual: 3.88562e-16
Final relative error: 4.78752
random x, y, z = [1.78862847 0.43650985 0.09649747]
r, theta, phi = [1.84364976 1.51843194 0.23936827]
Final relative residual: 5.26914e-17
Final relative error: 1.15628e-17
random x, y, z = [0.05056171 0.49995133 -0.99590893]
r, theta, phi = [1.11550097   2.6742991   58.01867353]
Final relative residual: 1.05022e-15
Final relative error: 17.404
random x, y, z = [0.44122749 -0.33087015 2.43077119]
r, theta, phi = [2.49254996 0.22310732 -0.64342791]
Final relative residual: 0
Final relative error: 3.22251e-17
random x, y, z = [-0.31178367 0.72900392 0.21782079]
r, theta, phi = [0.82225402 1.30268892 1.97493856]
Final relative residual: 1.50959e-16
Final relative error: 1.32977e-16
random x, y, z = [1.6905257 -0.46593737 0.03282016]
r, theta, phi = [1.75386771 4.73110309 2.87265305]
Final relative residual: 3.92855e-16
Final relative error: 1.89591
random x, y, z = [0.09120472 1.09128273 -1.94697031]
r, theta, phi = [-2.23381058   6.79554221 -33.07010478]
Final relative residual: 1.20108e-15
Final relative error: 9.34073
random x, y, z = [1.10855471e-03 -2.89544069e-01 -1.11606630e+00]
r, theta, phi = [1.15301387 3.39543102 -4.70856038]
Final relative residual: 1.4592e-16
Final relative error: 0.913957

When residual are always small, error could be considerable large.
The start guessing can influence the approximation at each iteration. And when the initial guessing is too far away from the true value, the method could result in a reasonable but not the expected θ and φ, which lead to a significant error.