Scientific Computing Homework 6

Michael Nelson

Problem 1

Exercise 1. Consider the Householder transformation

$$P = \mathrm{I} - 2vv^\top$$

for a vector v in $\mathbb{R}^m$ with $\|v\|_2 = 1$.

1. Show that P is orthogonal and symmetric.
2. Argue why this is also true for the transformation extended by the identity from an $m \times m$ to and $n \times n$ matrix as
$$\widehat{P} = \begin{pmatrix} \mathrm{I}_{m-n} & 0 \\ 0 & P \end{pmatrix}.$$

Solution 1. 1. Suppose $v = (a_1, \dots, a_n)^\top$. Let $1 \leq i < j \leq n$. Then (i,j)th entry of P is $-2a_ia_j$ and the (j,i)th entry of P is $-2a_ja_i$. Since $-2a_ia_j = -2a_ja_i$, we see that P is symmetric. To see that it is orthogonal, let w_1 and w_2 be vectors in $\mathbb{R}^m$. Then

$$\begin{aligned}
\langle Pw_1, Pw_2 \rangle &= (Pw_1)^\top Pw_2 \\
&= (w_1 - 2vv^\top w_1)^\top (w_2 - 2vv^\top w_2) \\
&= (w_1^\top - 2w_1^\top vv^\top)(w_2 - 2vv^\top w_2) \\
&= w_1^\top w_2 - 2w_1^\top vv^\top w_2 - 2w_1^\top vv^\top w_2 + 4w_1^\top vv^\top vv^\top w_2 \\
&= w_1^\top w_2 - 2w_1^\top vv^\top w_2 - 2w_1^\top vv^\top w_2 + 4w_1^\top vv^\top w_2 \\
&= w_1^\top w_2 \\
&= \langle w_1, w_2 \rangle.
\end{aligned}$$

It follows that P is orthogonal.

2. Clearly $\widehat{P}$ is symmetric since the identity matrix is symmetric and since P is symmetric. To see why $\widehat{P}$ is orthogonal, note that

$$\begin{aligned}
\widehat{P}\widehat{P}^\top &= \widehat{P}\widehat{P} \\
&= \begin{pmatrix} \mathrm{I}_{m-n} & 0 \\ 0 & P \end{pmatrix} \begin{pmatrix} \mathrm{I}_{m-n} & 0 \\ 0 & P \end{pmatrix} \\
&= \begin{pmatrix} \mathrm{I}_{m-n}\mathrm{I}_{m-n} & 0 \\ 0 & PP \end{pmatrix} \\
&= \begin{pmatrix} \mathrm{I}_{m-n} & 0 \\ 0 & \mathrm{I}_m \end{pmatrix} \\
&= \mathrm{I}_n.
\end{aligned}$$

A similar computation shows $\widehat{P}^\top \widehat{P} = \mathrm{I}_n$. Thus $\widehat{P}$ is also orthogonal.

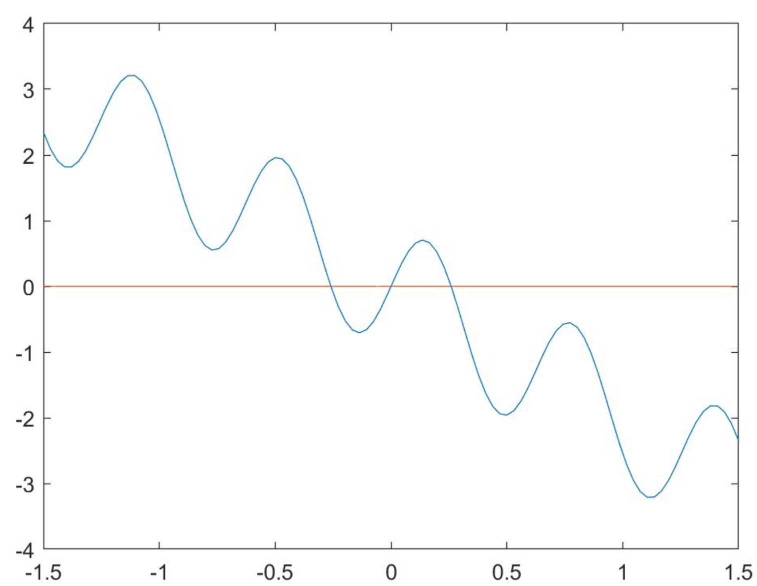
Problem 2

Exercise 2. Consider the function

$$f(x) = \sin(10x) - 2x.$$

1. How many roots does the function f have? Consider generating a plot in MATLAB to help.
2. With this knowledge, find bounds for the bisection method with a sign change and determine all roots with 8 digits of accuracy this way.

Solution 2. 1. Using MATLAB, we plot $f(x) = \sin(10x) - 2x$ together with the zero function below.



It appears that f has three roots.

2. It is easy to see that one of the roots is $x = 0$, so we just need to find the other roots. Furthermore, since

$$\begin{aligned} f(-x) &= \sin(10 \cdot (-x)) - 2 \cdot (-x) \\ &= \sin(-10x) + 2x \\ &= -\sin(10x) + 2x \\ &= -f(x), \end{aligned}$$

it suffices to find the positive root of f since the negative root is just negative of the postive root. To find the positive root, we use the bound $[0.2, 0.5]$ in the bisect.m function:

```
format long
a = 0.2;
b = 0.5;
f = @(x) sin(10*x)-2*x;
t = 0.0000000001;
bisect(a,b,f,t)

ans =

        0.259573907998856
```

Thus the roots are (within 8 digits of accuracy) given in the set below:

$$\{-0.259573908, 0, 0.259573908\}$$

Problem 3

Exercise 3. For the equation

$$f(x) = x^2 - 3x + 2 = 0$$

consider the fixed point problems

$$\begin{aligned} g_1(x) &= (x^2+2)/3 \\ g_2(x) &= \sqrt{3x-2} \\ g_3(x) &= 3 - \frac{2}{x} \\ g_4(x) &= (x^2-2)/(2x-3). \end{aligned}$$

1. Analyze the convergence properties for each $g_i(x)$ iteration for the root $x = 2$.
2. Confirm this by implementing the fixed-point iteration for each $g_i(x)$ and check the convergence and approximate convergence rate.

Solution 3. First we consider g_1. We calculate

$$\begin{aligned} |g_1'(2)| &= \left|\frac{2 \cdot 2}{3}\right| \\ &= \left|\frac{4}{3}\right| \\ &> 1. \end{aligned}$$

Thus the iterative scheme with respect to g_1 is divergent.

Next we consider g_2. We calculate

$$\begin{aligned} |g_2'(2)| &= \left|\frac{3}{2\sqrt{3 \cdot 2 - 2}}\right| \\ &= \left|\frac{3}{4}\right| \\ &< 1. \end{aligned}$$

Thus the iterative scheme with respect to g_2 is locally convergent. It converges linearly with constant $C = 3/4$.

Next we consider g_3. We calculate

$$\begin{aligned} |g_3'(2)| &= \left|\frac{2}{2^2}\right| \\ &= \left|\frac{1}{2}\right| \\ &< 1. \end{aligned}$$

Thus the iterative scheme with respect to g_3 is locally convergent. It converges linearly with constant $C = 1/2$.

Finally we consider g_4. We calculate

$$\begin{aligned} |g_4'(2)| &= \left|\frac{2(2^2 - 3 \cdot 2 + 2)}{(2 \cdot 2 - 3)^2}\right| \\ &= |0| \\ &= 0. \end{aligned}$$

Thus the iterative scheme with respect to g_3 is locally convergent. It converges quadratically with constant $C = g''(2)/2 = 2$.

2. To do this, we first write the following function in MATLAB and save it as iterationserrors.m:

```
function [iterations, errors] = iterationserrors(g,a);
format long;
x = g(a);
e = x - 2;
iterations = [x];
errors = [e];
for i=1:4
        x = g(x);
        e = x-2;
        iterations = [iterations x];
        errors = [errors e];
end
for i=1:5
        disp([i iterations(i) errors(i)]);
end
```

With this code in hand, we can look at the convergence tables for each g_i:

```
g1 = @(x) (x^2 +2)/3
g2 = @(x) (3*x - 2)^(1/2)
g3 = @(x) 3 - 2/x
g4 = @(x) (x^2 - 2)/(2*x - 3)
a = 1.9

iterationserrors(g1,a);

  1.000000000000000    1.870000000000000   -0.130000000000000
    2.000000000000000    1.832300000000000   -0.167700000000000
    3.000000000000000    1.785774430000000   -0.214225570000000
    4.000000000000000    1.729663438280608   -0.270336561719392

% diverges

iterationserrors(g2,a);

   1.000000000000000    1.923538406167134   -0.076461593832866
    2.000000000000000    1.941807204256232   -0.058192795743768
    3.000000000000000    1.955868506001540   -0.044131493998460
    4.000000000000000    1.966622871321449   -0.033377128678551

% converges linearly

iterationserrors(g3,a);

   1.000000000000000    1.947368421052632   -0.052631578947368
    2.000000000000000    1.972972972972973   -0.027027027027027
    3.000000000000000    1.986301369863014   -0.013698630136986
    4.000000000000000    1.993103448275862   -0.006896551724138

% converges linearly

iterationserrors(g4,a);

   1.000000000000000    2.012500000000000    0.012500000000000
    2.000000000000000    2.000152439024390    0.000152439024390
    3.000000000000000    2.000000023230574    0.000000023230574
    4.000000000000000    2.000000000000001    0.000000000000001
```

% converges quadratically