wb1443 Homework Set #4 Answers

4.1. fzerogui('x^3-2*x-5',[0,3])
Easy problem. Converges to x = 2.09455148154233 in 7 steps.
fzerogui('sin(x)',[1,4])
Easy problem. Converges to x = pi in 7 steps, all secant.
fzerogui('x^3-.001',[-1,1])
Moderately diffcult. There is only one real root, but there are two nearby complex roots. Requires 15 steps to converge to x = 1/10.
fzerogui('log(x+2/3)',[0,1])
Easy problem. Converges to x = 1/3 in 6 steps.
fzerogui('sign(x-2)*sqrt(abs(x-2))',[1,4])
This is the “perverse" example where Newton's method fails. $f'(x)$ is unbounded. fzero uses secant for all its steps. Slow convergence, only about half a decimal digit per step. Converges to x = 2 in 32 steps.
fzerogui('atan(x)-pi/3',[0,5])
Easy problem. Converges to x = sqrt(3) in 8 steps.
fzerogui('1/(x-pi)',[0,5])
Sign change is a pole, not a zero. Take over 50 steps towards x = pi. Eventually divides by zero and generates an error in the plot scaling.

4.2.

(a)
```
>> syms x
>> f = x^3 - 2*x - 5;
>> z = solve(f)
<messy symbolic expressions>
>> z(1)
ans =
1/6*(540+12*1929^(1/2))^(1/3)+4/(540+12*1929^(1/2))^(1/3)
>> length(char(z))
ans =
340
>> double(z)
2.09455148154233
-1.04727574077116 + 1.13593988908893i
-1.04727574077116 - 1.13593988908893i
```
(b)
```
>> p = [1 0 -2 -5]
p =
1 0 -2 -5
>> roots(p)
ans =
2.09455148154233
-1.04727574077116 + 1.13593988908893i
-1.04727574077116 - 1.13593988908893i
```
(c)
```
>> F = inline(char(f));
>> fzerotx(F,[2,3])
ans =
2.09455148154233
```
(d)
```
>> Fp = inline(char(diff(f)));
Fp =
Inline function:
Fp(x) = 3.*x.^2-2
>> x = 1i;
>> x = x - F(x)/Fp(x)
x =
-1.00000000000000 + 0.40000000000000i
>> x = x - F(x)/Fp(x)
x =
-0.56274873971876 + 1.77192889360573i
Use uparrow to iterate .....
>> x = x - F(x)/Fp(x)
x =
-1.04727574077116 + 1.13593988908893i
```
(e) No. There is no notion of sign change or positive/negative for complex numbers.

4.3. $p(x) = 816*x^3 - 3835*x^2 + 6000*x - 3125$

(a) What are the exact roots of p?
```
>> p = poly2sym([816 -3835 6000 -3125])
p = 816*x^3-3835*x^2+6000*x-3125
>> factor(p)
ans = (16*x-25)*(17*x-25)*(3*x-5)
>> z = solve(p)
z =
[ 25/15]
[ 25/16]
[ 25/17]
```
(b)
```
>> p = inline(char(p));
>> ezplot(p,1.43,1.71)
>> hold on, plot(double(z),zeros(3,1),'o')
```
(c)
```
>> x = 1.5
>> x = x - (816*x^3-3835*x^2+6000*x-3125)/(2448*x^2-7670*x+6000)
```
Use up arrow to iterate. Converges easily to the nearest root, x = 1.47058823529416 = 25/17

(d) Starting with $x0$ = 1 and $x1$ = 2, the secant method converges to 1.6666666666666666 = 25/15.

(e) The first step reduces the interval to [1,1.5], which contains only one root. Consequently, converges to x = 1.47… = 25/17.

(f) The initial secant step happens to be to x = 1.69…, which is near the root at 25/15. fzerotx then takes 10 steps, 7 with IQI, to converge to 25/15. The interval [a,b] always includes all three roots. (Note that none of these methods found the "middle" root, 25/16.)

4.4 The convergence test in fzerotx is

```
m = 0.5*(a - b);
tol = 2.0*eps*max(abs(b),1.0);
if (abs(m) <= tol) | (fb == 0.0)
   break
end
```

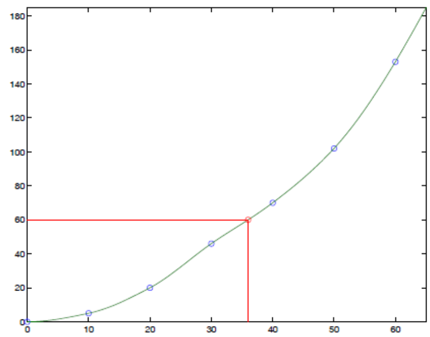
This says that we have luckily found a b for which f(b) is exactly zero, or the length of the interval, abs(b-a) , is roundoff error in b or 1 . Note this is a relative error test if b is larger than 1, but an absolute error test if b is less than 1.

4.9. First ten solutions of: x = tan x.

```
for k = 1:10
z(k) = fzerotx('tan(x)-x',[k k+1/2-k*eps]*pi);
end
z
= 4.4934 7.7253 10.9041 14.0662 ... 29.8116 32.9564
z/pi
= 1.4303 2.4590 3.4709 4.4774 ... 9.4893 10.4903
```

4.14. What is the speed limit for this vehicle?

```
x1 =
  35.83333333333334
x2 =
  36.00066760428985
x3 =
  35.98756534518393
x4 =
  35.86433220451173
x5 =
  36.00342638805324
```



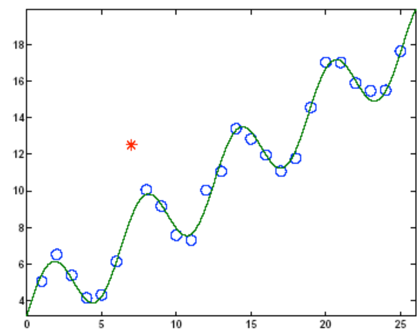
4.16. Freezing water mains.

```
function T = pipetemp(x)
% Temperature of water main at depth of x meters after 60 days.
Ti = 20;
Ts = -15;
alpha = 0.138e-6;
t = 60*24*60*60; % 60 days * (24*60*60) secs/day
c = 2*sqrt(alpha*t);
T = Ts + (Ti - Ts)*erf(x/c);
ezplot(@pipetemp,[0 2])
fzerotx(@pipetemp,[0 2])
ans =
0.6770
```

5.8 (a) beta = 4.01269200000000
0.53264276923077
t(7),y(7) is an outliner

(c) beta = 3.15359848998641
0.58690874100321
1.97332105560748

(d)

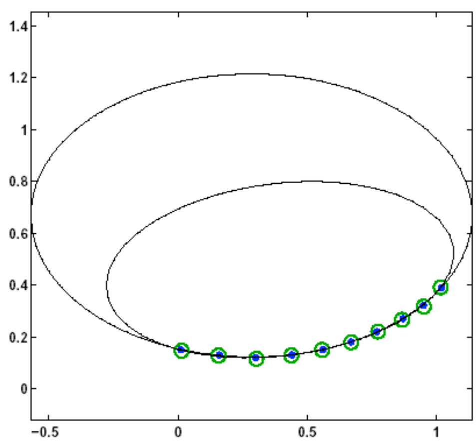


5.12 Two orbits, one from original data, one after small random perturbation.

(a) c = -2.2537948175
-0.0063247132
-5.5221834331
1.2898102053
7.3773544034

(b) c = -2.4423051434
1.7739414419
-9.5532799239
1.1162584602
8.0629704039

Plot: use axis equal!



5.xx See textbook section 5.8 Separable Least Squares.

- FIN -