DR BAUER
SUM 2011

HMWK 2 (falling jumper)
due FRI, JUNE 17

Consider the function $v(t) = \left(\frac{gx}{c}\right) * \left(1 - e^{-(c/x)t}\right)$.

This can be rearranged $f(x) = \left(\frac{gx}{c}\right) * \left(1 - e^{-(c/x)t}\right) - v(t)$

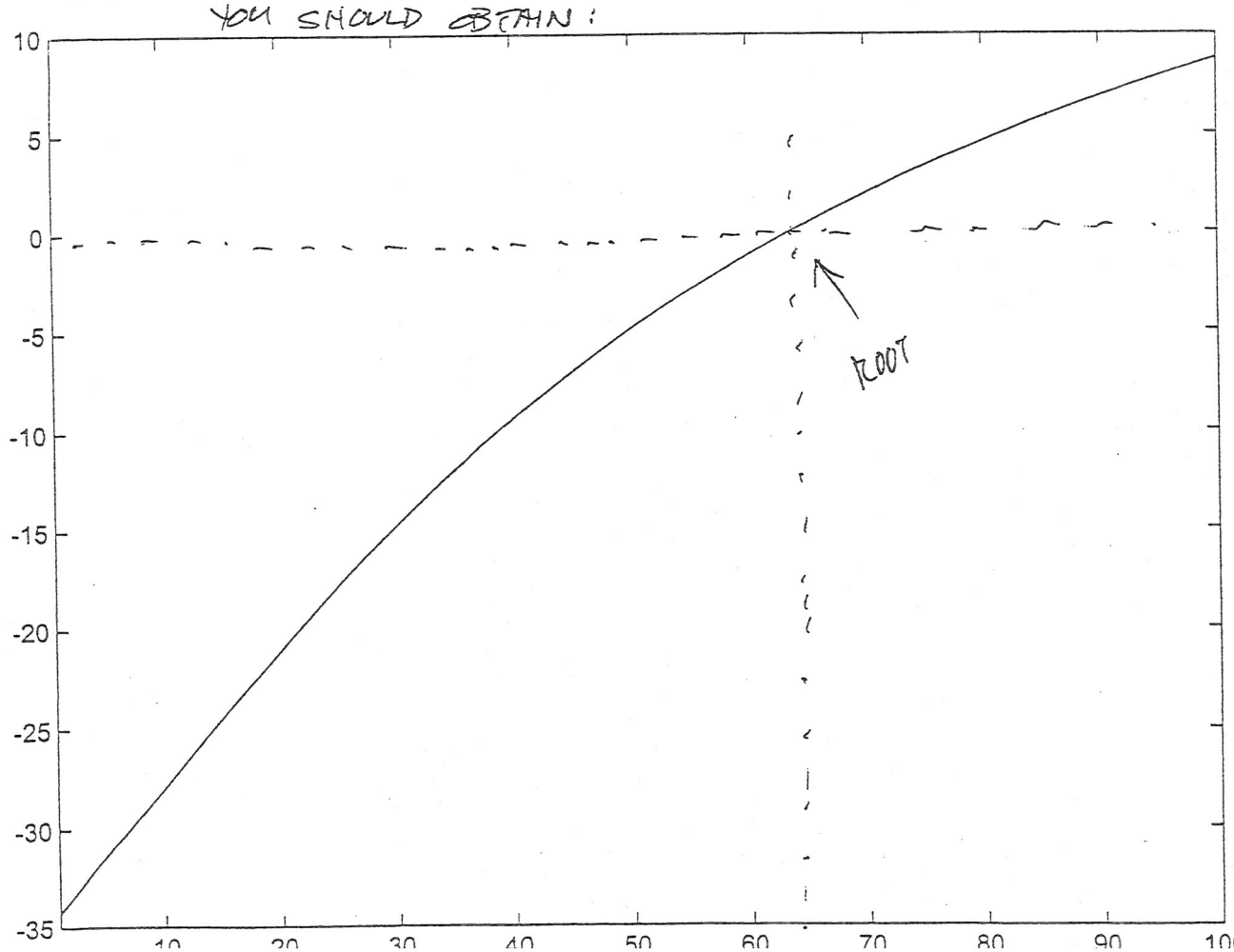
1) Create an "m-file" definition of this function:
(x is the mass of the jumper; it is the root we want to find).

```
function y=f(x)
g=9.8;
c=14.;
v=35.;
t=7.;
y=(g*x/c)*(1-exp(-(c/x)*t))-v;
```

2) Save this on the C-drive or your floppy disk or dongle (saves retyping - VCC erases disk files daily)

3) Use a statement like "`>> fplot('f',[1,100])`" (CR)

YOU SHOULD OBTAIN:



```
%bauer cut at bisection function root finder...
function dummy= bisect(xl,xu,es,imax,xr,iter,ea)
                    in=[xl,xu,es,imax,xr,iter,ea];
disp(in)
iter=0;
while iter<imax
    xrold=xr;
    xr=(xl+xu)/2;
    iter=iter+1;
    if xr~=0
        ea=abs((xr-xrold)/xr)*100;
    end
    test=f(xl)*f(xr);
%    disp(test)
    if test<0
        xu=xr;
    elseif test>0
        xl=xr;
    end
    out=[xl,f(xl),xr,f(xr),xu,f(xu),ea,iter];
    disp(out)
    bisect=xr;
end
```

Not needed

NOTE THIS VERSION DOES NOT HAVE EA CUTOFF CHECK AGAINST ES

ALSO, THIS VERSION DOES NOT CHECK (X_L, X_U) INPUTS FOR VALID ROOT INTERVAL

```
>> bisect(1,100,.1,100,0,0,1000)
   1.0e+003 *
```

CR

ECHO OF INPUT

0.0010	0.1000	0.0001	0.1000	0	0	1.0000	
x_L	$f(x_L)$	x_R	$f(x_R)$	x_U	$f(x_U)$	ε_a	ITER
50.5000	-4.7269	50.5000	-4.7269	100.0000	8.7282	100.0000	1.0000
50.5000	-4.7269	75.2500	3.3527	75.2500	3.3527	32.8904	2.0000
62.8750	-0.2486	62.8750	-0.2486	75.2500	3.3527	19.6819	3.0000
62.8750	-0.2486	69.0625	1.6467	69.0625	1.6467	8.9593	4.0000
62.8750	-0.2486	65.9688	0.7244	65.9688	0.7244	4.6897	5.0000
62.8750	-0.2486	64.4219	0.2445	64.4219	0.2445	2.4012	6.0000
63.6484	-0.0004	63.6484	-0.0004	64.4219	0.2445	1.2152	7.0000
63.6484	-0.0004	64.0352	0.1225	64.0352	0.1225	0.6039	8.0000
63.6484	-0.0004	63.8418	0.0611	63.8418	0.0611	0.3029	9.0000
63.6484	-0.0004	63.7451	0.0304	63.7451	0.0304	0.1517	10.0000
63.6484	-0.0004	63.6968	0.0150	63.6968	0.0150	0.0759	11.0000
63.6484	-0.0004	63.6726	0.0073	63.6726	0.0073	0.0380	12.0000
63.6484	-0.0004	63.6605	0.0035	63.6605	0.0035	0.0190	13.0000
63.6484	-0.0004	63.6545	0.0015	63.6545	0.0015	0.0095	14.0000
63.6484	-0.0004	63.6515	0.0006	63.6515	0.0006	0.0047	15.0000
63.6484	-0.0004	63.6499	0.0001	63.6499	0.0001	0.0024	16.0000
63.6492	-0.0002	63.6492	-0.0002	63.6499	0.0001	0.0012	17.0000
63.6496	-0.0000	63.6496	-0.0000	63.6499	0.0001	0.0006	18.0000
63.6496	-0.0000	63.6498	0.0000	63.6498	0.0000	0.0003	19.0000
63.6497	-0.0000	63.6497	-0.0000	63.6498	0.0000	0.0001	20.0000
63.6497	-0.0000	63.6497	0.0000	63.6497	0.0000	0.0001	21.0000
63.6497	-0.0000	63.6497	-0.0000	63.6497	0.0000	0.0000	22.0000
63.6497	-0.0000	63.6497	0.0000	63.6497	0.0000	0.0000	23.0000
63.6497	-0.0000	63.6497	0.0000	63.6497	0.0000	0.0000	24.0000
63.6497	-0.0000	63.6497	-0.0000	63.6497	0.0000	0.0000	25.0000
63.6497	-0.0000	63.6497	0.0000	63.6497	0.0000	0.0000	26.0000

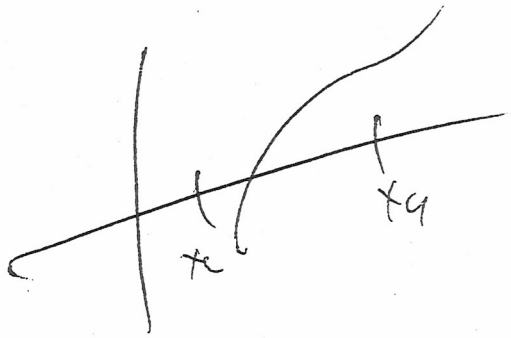
Continues for 100 iterations

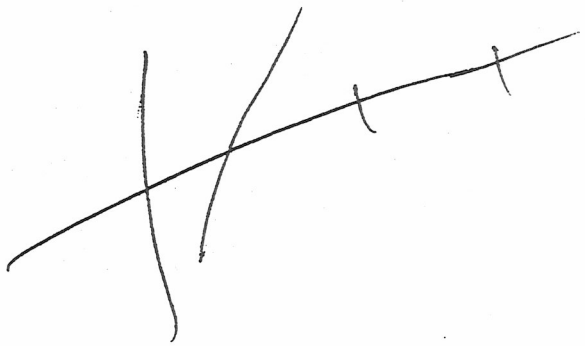
```
%bauer cut at bisection function root finder...
function dummy= bisect(xl,xu,es,imax,xr,iter,ea)
                    in=[xl,xu,es,imax,xr,iter,ea];
disp(in)
iter=0;
while (iter<imax)  &  (ea>es)
    xrold=xr;
    xr=(xl+xu)/2;
    iter=iter+1;
    if xr~=0
        ea=abs((xr-xrold)/xr)*100;
    end
    test=f(xl)*f(xr);
%    disp(test)
    if test<0
        xu=xr;
    elseif test>0
        xl=xr;
    end
    out=[xl,f(xl),xr,f(xr),xu,f(xu),ea,iter];
    disp(out)
    bisect=xr;
end
```

Adds

ε_a vs ε_s

test





```
>> bisect(1,100,.1,100,0,0,1000)
  1.0e+003 *

    0.0010    0.1000    0.0001    0.1000         0         0    1.0000

   50.5000   -4.7269   50.5000   -4.7269  100.0000    8.7282  100.0000    1.0000

   50.5000   -4.7269   75.2500    3.3527   75.2500    3.3527   32.8904    2.0000

   62.8750   -0.2486   62.8750   -0.2486   75.2500    3.3527   19.6819    3.0000

   62.8750   -0.2486   69.0625    1.6467   69.0625    1.6467    8.9593    4.0000

   62.8750   -0.2486   65.9688    0.7244   65.9688    0.7244    4.6897    5.0000

   62.8750   -0.2486   64.4219    0.2445   64.4219    0.2445    2.4012    6.0000

   63.6484   -0.0004   63.6484   -0.0004   64.4219    0.2445    1.2152    7.0000

   63.6484   -0.0004   64.0352    0.1225   64.0352    0.1225    0.6039    8.0000

   63.6484   -0.0004   63.8418    0.0611   63.8418    0.0611    0.3029    9.0000

   63.6484   -0.0004   63.7451    0.0304   63.7451    0.0304    0.1517   10.0000

   63.6484   -0.0004   63.6968    0.0150   63.6968    0.0150    0.0759   11.0000
```

REQUIRED WORK —

(a) MODIFY THIS PROGRAM TO INCLUDE AN INITIAL CHECK TO SEE IF YOUR SUPPLIED (XL, XU) VALUES DEFINE AN INTERVAL CONTAINING A ROOT.

(b) RUN AN EXAMPLE WHERE THE VALUES DO NOT CONTAIN A ROOT AND DEMONSTRATE THAT THE CODE DETECTS THIS CASE.

(c) Run an example with "good" I.C's to find the root