Homework 4

1.

EDU>> t = [4 8 12 16 20 24];

EDU>> c = [1590 1320 1000 900 650 560];

EDU>> logc = log(c)

logc =

7.3715 7.1854 6.9078 6.8024 6.4770 6.3279

EDU>> linearfit = [6 sum(t) sum(logc); sum(t) sum(t.^2) sum(t.\*logc)]

linearfit =

1.0e+003 \*

0.0060 0.0840 0.0411

0.0840 1.4560 0.5601

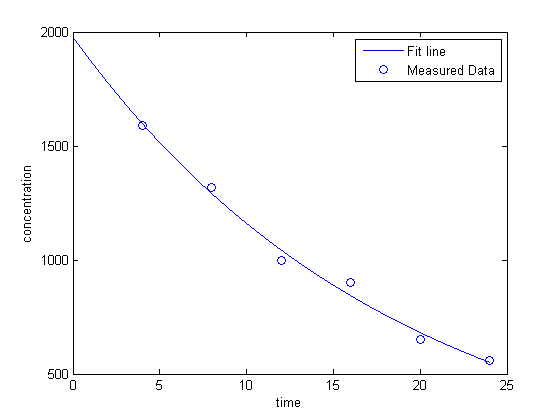
EDU>> linearcoeff = rref(linearfit)

linearcoeff =

1.0000 0 7.5902

0 1.0000 -0.0532

EDU>> a = linearcoeff(1,3)

a =

7.5902

EDU>> b = linearcoeff(2,3)

b =

-0.0532

EDU>> aN = exp(a);

EDU>> aN = exp(a)

aN =

1.9786e+003

EDU>> x = [0:0.0001:24];

EDU>> y = aN.\*exp(b.\*x);

2.

EDU>> w = [70 75 77 80 82 84 87 90];

a = [2.1 2.12 2.15 2.2 2.22 2.23 2.26 2.3];

logw = log10(w);

loga = log10(a);

bestfit = [8 sum(logw) sum(loga); sum(logw) sum(logw.^2) sum(logw.\*loga)]

fitcoeff = rref(bestfit)

b = fitcoeff(1,3); c = fitcoeff(2,3);

b = 10^b;

y = b.\*(x.^c);

bestfit =

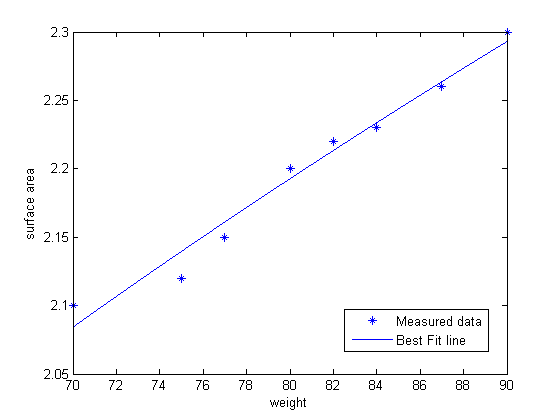
8.0000 15.2416 2.7339

15.2416 29.0472 5.2120

fitcoeff =

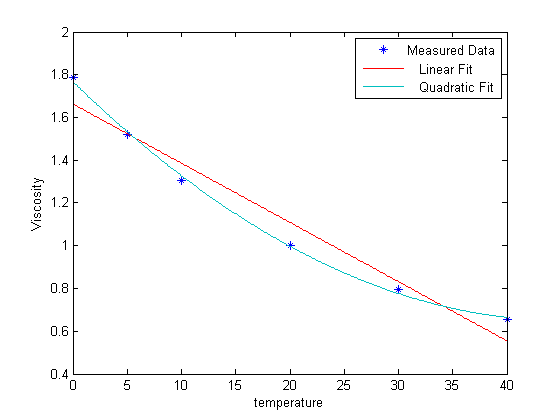
1.0000 0 -0.3821

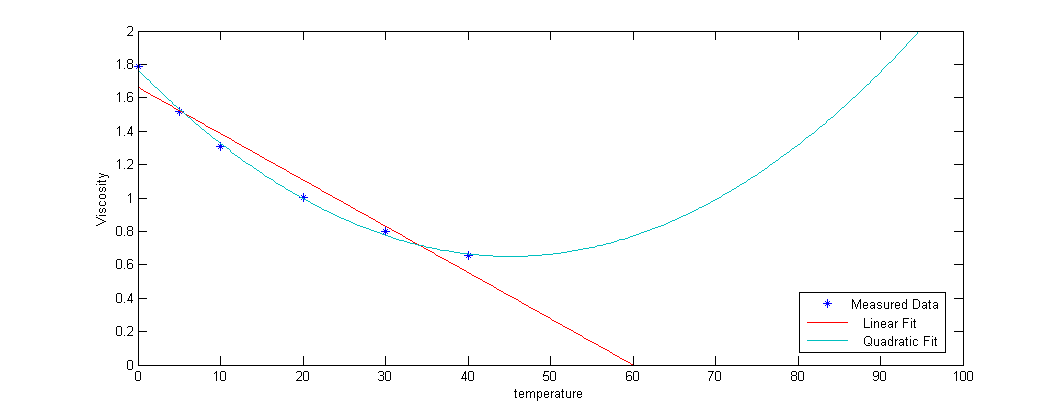
0 1.0000 0.3799

EDU>> x = [70:0.0001:90]; y = b.\*(x.^c);

3.

a) I would choose linear and quadratic to compare the extrapolated data with. I would choose these because they are the best fit for linearized data and someone could actually do something with data points like this, like predict a value based on values around it. It looks like the Quadratic approximation gives the best approximation, because it snaps onto the measured data points pretty well as can be seen.

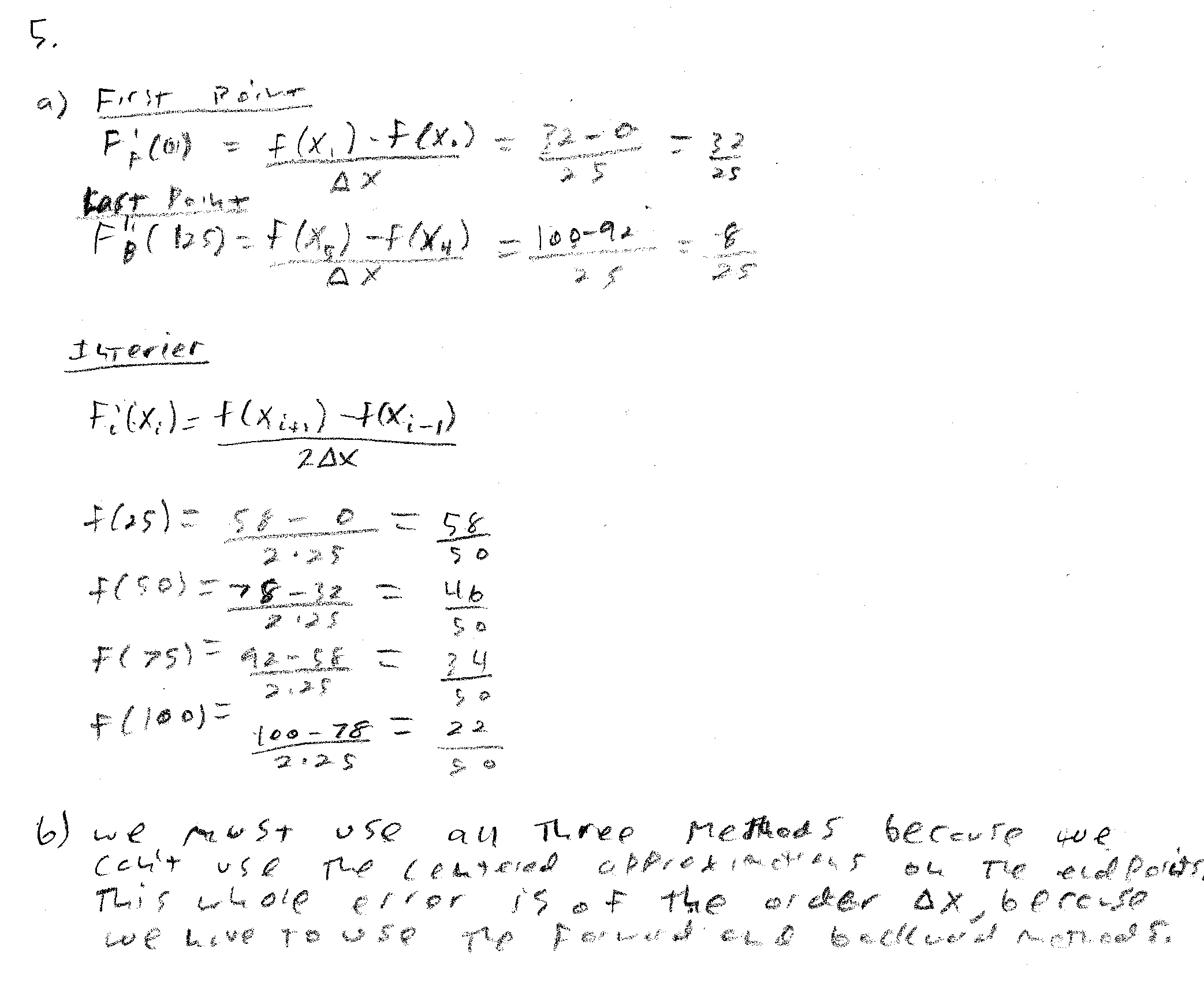


b) As for predicting higher values past temperatures of 50 degrees, the linear might be the better route to take. This data more than likely doesn’t take the quadratic route that is in the below graph.

4.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Error f’(1)-F’(1) | F’(1) Forward Diff | F’(1) Backward Diff | F’(1) Centered Diff | F’’(1) Central Diff |
| x = .2 | 3.009 | 2.464 | 2.7365 | 2.725 |
| x = .1 | 2.859 | 2.587 | 2.726 | 2.72 |

The O( ) estimates for the first two mean that the error of the function is on an order of the step size. The second half say the error is on the order of the step size squared, which since the step size should be small when squared makes it even smaller. So the squared step size has a much smaller error than the single step size.



c) EDU>> t = [ 0 25 50 75 100 125]

t =

0 25 50 75 100 125

EDU>> y = [0 32 58 78 92 100];

EDU>> t = [ 0 25 50 75 100 125];

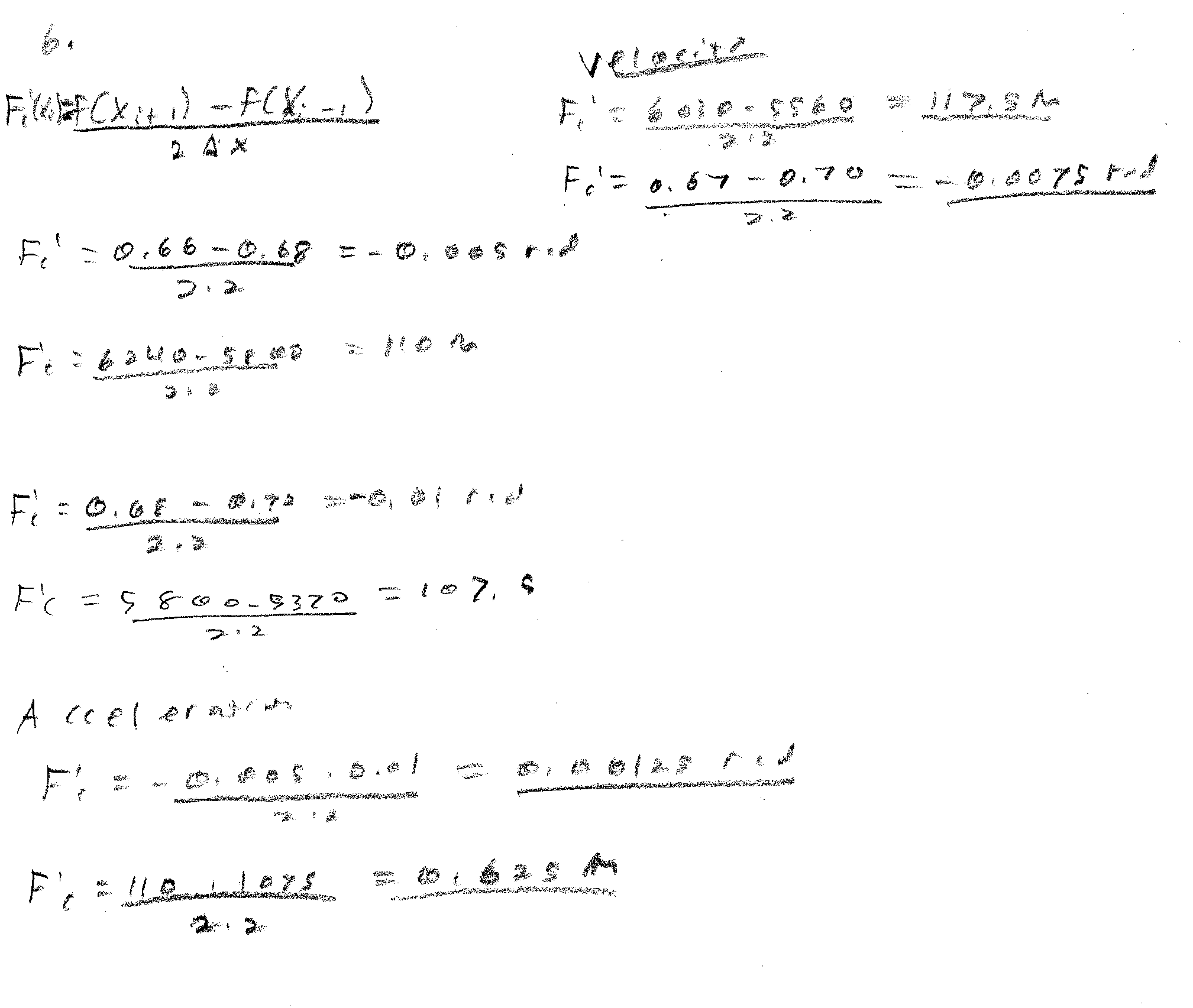
EDU>> f = polyfit(t,y,2)

f =

-0.0048 1.4000 0.0000

EDU>> dy = gradient(y,25)

dy = 1.2800 1.1600 0.9200 0.6800 0.4400 0.3200



7.

EDU>> t = [0 0.52 1.04 1.75 2.37 3.25 3.83];

EDU>> x = [153 185 208 249 261 271 273];

EDU>> polyfit(t,x,2)

ans =

-10.0731 70.1201 151.4961

EDU>> dy = gradient(x,.52)

dy =

61.5385 52.8846 61.5385 50.9615 21.1538 11.5385 3.8462

8.

EDU>> v1 = [.4 .7 .77 .88];

EDU>> v2 = [.88 1.05 1.17 1.35];

EDU>> gradient(v1,10)

ans =

0.0300 0.0185 0.0090 0.0110

EDU>> gradient(v2,15)

ans =

0.0113 0.0097 0.0100 0.0120

Q is the last two results appended with respect to the appropriate T’s.