Memo on Multi-Slit Emittance Measurement in the FAST/IOTA electron injector

Dean (Chip) Edstrom

September, 2015

A multi-slit method for measuring beam emittance was taken from a Physical Review Special Topics paper[1] and has been implemented in ACL and Python for a simulated case. The simulated case, shown below in *fig 1*, was derived from a series of Elegant simulations performed for beam passing through a multi-slit mask at x107 and seen on a screen at x110 ~0.5m downstream. The image was cropped to an 8 x 8 mm square area and inverted. Reading the image into a simple Java program (Appendix A), the histogram profile was found. Since the measurement is amplitude independent (only relative amplitudes are taken into account in the fitting covered later) the histogram was normalized to the highest peak. Operationally, the histograms for each plane will be available to us directly through the camera devices, N:DC*n*XH[0:*x*] and N:DC*n*YH[0:*y*] where *n* is the camera server being used and 0:*x* & 0:*y* are the respective pixel ranges in that dimension.

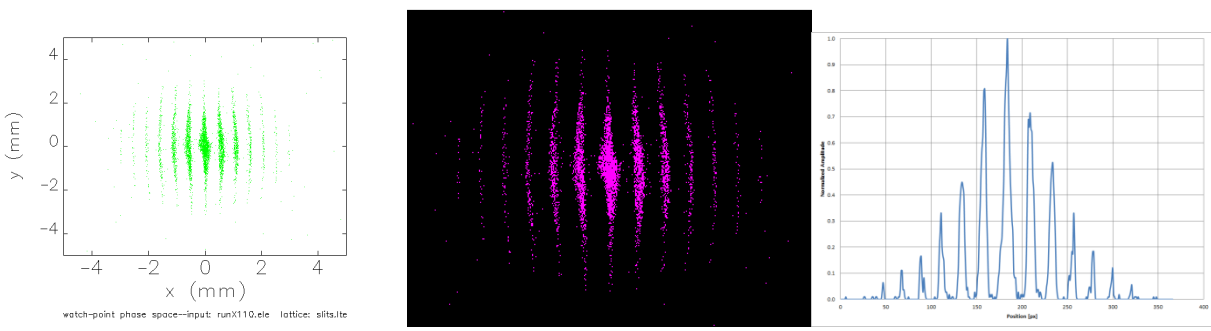

Fig 1 – The original simulated beam image (left, 3200_min_dE.X110.png), the cropped and inverted 8 x 8 mm image (middle), and the normalized horizontal histogram resulting from it (right)

An ACL (Appendix B) was written to locate the peaks and pass each one to Python to perform a Gaussian fit to it. Currently this process is not very well refined and realistically it will require some attention once we have an actual picture of the slit mask projection. The individual peaks are exported to a temporary file and fit using a Python Gaussian fitting routine (Appendix C) as seen in *Fig 2*. Once these fits are complete, enough information is available to calculate the emittance. The remaining necessary values are calculated:

- Beam divergences: $x'_{m,c} = \langle x_m - mw \rangle / L$
- RMS divergence spread: $\sigma'_m = \sqrt{\langle x_m^2 \rangle / L^2 - \left(x'_{m,c}\right)^2}$
- And finally the second moments: $\langle x^2 \rangle = \sum_{m=1}^{N} I_m x_{m,c}^2 / \sum_{m=1}^{N} I_m$

$$\langle x'^2 \rangle = \sum_{m=1}^{N} I_m \left(x'^2_{m,c} + \sigma'^2_m\right) \Big/ \sum_{m=1}^{N} I_m$$

$$\langle x x' \rangle = \sum_{m=1}^{N} I_m x_{m,c}\, x'_{m,c} \Big/ \sum_{m=1}^{N} I_m$$

[1] Space-charge effects in high brightness electron beam emittance measurements, Physical Review Special Topics – Accelerators and Beams, Volume 5, 014201 (2002) @ http://journals.aps.org/prstab/pdf/10.1103/PhysRevSTAB.5.014201

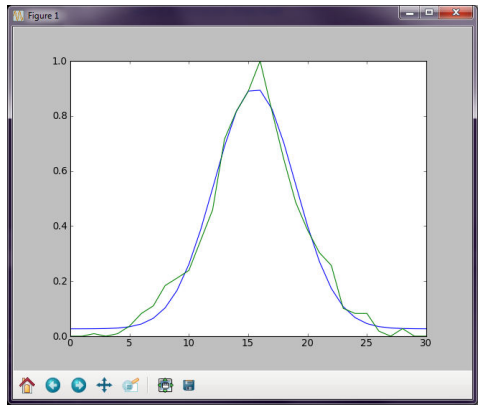

Fig 2 – The Gaussian fit to one of the individual slit beamlets. The Green trace is raw data, the blue is the fit. The x-axis is in units of px.

In this case, the average position of each beamlet projection, $\langle x_m \rangle$ was simply the centroid position from the Gaussian fit. The integral of each beamlet, I_m, was also found in fit.py. The highest peak was picked to be the central peak and all other beamlets were arranged around it such that the highest peak was m = 1 and the furthest out was m = 5 with a peak detection threshold of 0.05. L, the drift length was picked to be 0.445 m, close to the actual distance between x107 and x110, and the slit width was picked to be 40 µm. This gives us an emittance of:

$$\varepsilon \equiv \sqrt{\langle x^2 \rangle \langle x'^2 \rangle - \langle xx' \rangle^2} \; = \; 3.976 \, mmmr$$

A few other things were tried, including factoring in the amplitude of the full pulse amplitude, as seen in Fig 3, but this didn't have any impact on the final emittance calculation because any factor change gets taken out by dividing by the same factor in calculating each of the second moments. The Twiss parameters were also calculated and the phase ellipse was plotted using another Python routine (Appendix D), also seen in Fig 3. The Twiss parameters are calculated as follows:

$$\alpha = -\frac{\langle xx' \rangle}{\varepsilon}, \qquad \beta = \frac{\langle x^2 \rangle}{\varepsilon}, \qquad \gamma = \frac{1 + \alpha^2}{\beta}$$

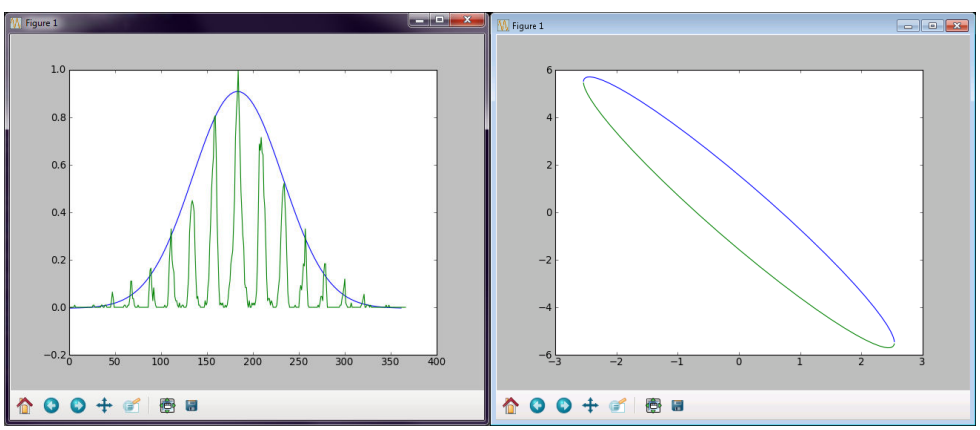

Fig 3 – A fit to the envelope (left) did not change the emittance when factored into the emittance calculation in normalizing each of the separate beamlet integrals. The phase ellipse from the Twiss parameters (right) has axis units of position (x [mm] along the x-axis) and divergence (x' [mrad] along the y-axis)

Appendix A – MyHistogram.java

```
import java.awt.Toolkit;
import java.awt.datatransfer.Clipboard;
import java.awt.datatransfer.StringSelection;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;

import javax.imageio.ImageIO;

public class MyHistogram {


    public static void main(String[] args) {
        BufferedImage img = null;
        try { img = ImageIO.read(new File("C:\\Temp\\3200_min_dE.X110.png"));
                } catch (IOException e) { e.printStackTrace(); }

        int myWidth; int myHeight;
        myWidth = img.getWidth();
        myHeight = img.getHeight();
        float mySum;
        int[][] myImg = new int[myWidth][myHeight];
        float maxamp = 0;
        int i=0, j=0;
        String myOut = "";

        while (i < myWidth) {
                j=0;
                while ( j < myHeight ) {
                        myImg[i][j] = img.getRGB(i, j);
                        j++;
                }
                i++;
        }

        i=0;
        while ( i <  myWidth){
                j = 0;
                mySum = 0;
                while (j < myHeight) {
                        mySum += myImg[i][j];
                        j++;
                }
                mySum = mySum;
                myOut += mySum + "\n";
                i++;
        }
        StringSelection selection = new StringSelection(myOut);
        Toolkit defaultToolkit = Toolkit.getDefaultToolkit();
        Clipboard clp = defaultToolkit.getSystemClipboard();
        clp.setContents(selection, selection);
    }
}
```

Appendix B – fast_emit_measurement.acl

```
################################################################################
#+description
# Slitmask emittance measurement
#
#-description
################################################################################
#+author
# Chip Edstrom (13417N)
#-author
################################################################################

declare myLockName string = "nml_emit_measurement"
declare myErrMax integer = 10
declare myErrCount integer
declare myErrLast integer
on_error herr
enable settings
aclOption default_clock_type=nml
ftd 1_shot

lock myLockName

# Declarations
declare myThresh float = 0.1      # Threshold for determining a gaussian

declare myW float = 40e-6         # Slit Width [m]
declare myL float = 0.445         # Drift length between slit face and screen [m]
declare mpp float = 2.18579235e-5 # m / px conversion (8mm / 366px for sample)

declare myHist[1000] float
declare minHist float
declare maxHist float
declare szHist int
declare nextbump boolean
declare upBump[20] int
declare dnBump[20] int

declare nBumps int = 0
declare myBump[20] int

declare myBumpMax[20] float
declare myBumpMaxx[20] int

declare tstPts int = 0
declare myP[5] float
declare myP0[20] float            # Fit Amplitudes for each myBump
declare myP1[20] float            # Fit Centroids for each myBump
declare myP2[20] float            # Fit Sigmas for each myBump
declare myP3[20] float            # Fit Linear Offset for each myBump

declare myInt[20] float           # Fit Integral
declare myIntSum float = 0        # Sum of all fit integrals
declare myIntSumMW float = 0      # Sum of all offset fit integrals

declare myP0Cent float            # Amplitude of central gaussian
declare myP1Cent float            # Position of central gaussian's centroid
declare myNCent float             # Bump number of the central gaussian

declare myDiv[20] float           # Divergences
declare myDSpr[20] float  # Divergence Spreads
declare myTrx[20] float           # Trace space centroids
declare SecMom[4] float           # Second Moments: [0] = <x^2> , [1] = <x`^2>, [2] = <xx`>

declare myEmit float              # Measured Emittance
declare myEmitMmmr float  # Emittance for output (scaled to mmmr)
declare myAlpha float             # Measured Twiss Parameters
declare myBeta float
declare myGamma float

declare myOut string

##################################################### Read histogram from file
read_file/num_lines=szHist/output=variable:myHist "sample.dat"

##################################################### Find Breaks Between Gaussians
i=-1
while (i++ < szHist)      # Find the min and max points
```

```
        if (i = 0)
                minHist = myHist[i]
                maxHist = myHist[i]
        else
                if ( myHist[i] < minHist )
                        minHist = myHist[i]
                endif
                if ( myHist[i] > maxHist )
                        maxHist = myHist[i]
                endif
        endif
endwhile
# Convert threshold from fraction to an actual amplitude
myThresh = minHist + (myThresh * (maxHist-minHist))
i=-1
j=0
nextbump = false
while (i++ < szHist)      # Find bumps
        if (( myHist[i] ge myThresh ) && ( not nextbump ))
                nextbump = true
                j=nBumps
                nBumps++
                upBump[j] = i
        endif
        if (( myHist[i] lt myThresh ) && ( nextbump ))
                nextbump = false
                dnBump[j] = i
        endif
endwhile
i=0
while (i++ < nBumps)
        # start with index 1 and average the dnBump[i-1] and upBump[i] ending with (i < nBump)
        j = i-1
        myBump[i] = ( dnBump[j] + ((upBump[i]-dnBump[j])/2) )
endwhile
# Take care of the ends...
myBump[0] = upBump[0] - (myBump[1] - dnBump[0])
j = nBumps - 1
myBump[nBumps] = dnBump[j] + (upBump[j] - myBump[j])


##################################################### Fit to each gaussian & Get envelope
i=-1
while(i++ < nBumps)
        output "single.dat"
        j = -1
        k = i + 1
        tstPts = myBump[k] - myBump[i]
        myBumpMax[i] = 0
        while(j++ < tstPts)
                l = j + myBump[i]
                if (myHist[l] > myBumpMax[i])
                        myBumpMaxx[i] = l
                        myBumpMax[i] = myHist[l]
                endif
                print l "        " myHist[l]
                # print " Bump " i " [" j "] : x=" tstGx[j] ", y=" tstGy[j]
        endwhile
        output/close
        j = i + 1
        print " -- Fitting [" j "/" nBumps "]"
        run "/usr/bin/python fit.py"
        read_file/output=variable:myP "lastfit.dat"
        myP0[i]=myp[0]
        myP1[i]=(myp[1] + myBump[i]) # put centroid offset back in
        myP2[i]=myp[2]
        myP3[i]=myp[3]
        myInt[i]=myp[4]
endwhile

##################################################### Prep for finding emittance
i=-1
while(i++ < nBumps)                                # Find the central peak and sum of the integrals
        if (myP0[i] > myP0Cent)
                myP0Cent = myP0[i]
                myP1Cent = myP1[i]
                myNCent = i
        endif
        myInt[i] = myInt[i] * mpp
        myIntSum = myIntSum + myInt[i]
endwhile
```

```
i=-1
while(i++ < nBumps)                                 # Center all the gaussians around 0
         myP1[i] = myP1[i] - myP1Cent
endwhile
i=-1
while(i++ < nBumps)                                 # Convert all remaining relevant values from px to m
         myP1[i] = myP1[i]*mpp
         myP2[i] = myP2[i]*mpp
         m = (i - myNCent)
         myIntSumMW = myIntSumMW + (myInt[i] * m * myW)      # Find the offset integral sum while we're at it
endwhile
i=-1
while(i++ < nBumps)                                          # Find Divergence / Div Spread / Trace Space Dist
         m = (i - myNCent)                                   # Center these around the Central peak
         myDiv[i] = (myP1[i] - (m * myW)) / myL              # Calculate Divergence
         myDSpr[i] = ((myP1[i]/myL)^2 - myDiv[i]^2)^0.5      # Calculate Divergence Spread
         myTrx[i] = (m * myW) - (myIntSumMW / myIntSum)      # Calculate Trace Space Distribution
endwhile

##################################################### Find Emittance & Twiss Parameters
i=-1
while(i++ < nBumps)
         secMom[0] = secMom[0] + (myInt[i] * (myP1[i]^2))
         secMom[1] = secMom[1] + (myInt[i] * (myDiv[i]^2 + myDSpr[i]^2))
         secMom[2] = secMom[2] + (myInt[i] * myP1[i] * myDiv[i])
endwhile
secMom[0] = (secMom[0] / myIntSum)
secMom[1] = (secMom[1] / myIntSum)
secMom[2] = (secMom[2] / myIntSum)

myEmit = (secMom[0]*secMom[1] - (secMom[2]^2))^0.5

myAlpha = -1 * (secMom[2]/myEmit)
myBeta = secMom[0]/myEmit
myGamma = (1 + (myAlpha^2))/myBeta

##################################################### Calculate Emittance and Output
print
print "---- Summary ----"
print "Histogram Size: " szHist " px"
myOut = sprintf("Threshold for Gaussian Detection: %.3f (x Normalized Amplitude)", myThresh)
print myOut
print "Total Number of Gaussians Found: " nBumps
print "Gaussians:         Amp [Norm]        Cent [m]         Sig [m]  LinOff          Integral [m]      x`
[Rad]    Sig` [Rad]"
myOut = ""
i=-1
while(i++ < nBumps)
         j = i+1
         myOut = sprintf("%s   [%2i] %3i-%3ipx\t%.3e\t%.3e\t%.3e\t%.3e\t%.3e\t%.3e\t%.3e\n", myOut, j,
myBump[i], myBump[j], myP0[i],myP1[i],myP2[i],myP3[i],myInt[i],myDiv[i],myDSpr[i])
endwhile
print myOut
print
print "Second Moments:"
myOut = sprintf("   <x^2>  : %.3e m^2", secMom[0])
print myOut
myOut = sprintf("   <x`^2> : %.3e rad^2", secMom[1])
print myOut
myOut = sprintf("   <xx`>  : %.3e m*rad", secMom[2])
print myOut
print
myOut = sprintf("Distance from Screen (L): %.3e m", myL)
print myOut
myOut = sprintf("Slit Width (w): %.3e m", myW)
print myOut
myEmitMmmr = myEmit * 1000 * 1000                           # x(mm/m) & x(mrad/rad)
myOut = sprintf("Emittance: %.3f mmmr", myEmitMmmr)
print myOut
print
print "Twiss Parameters:"
myOut = sprintf("   Alpha : %.3f", myAlpha)
print myOut
myOut = sprintf("   Beta  : %.3f", myBeta)
print myOut
myOut = sprintf("   Gamma : %.3f", myGamma)
print myOut

output "twiss.dat"
print myAlpha
print myBeta
```

```
print myGamma
print myEmit
output/close

print "Plotting Phase Elipse..."
run "/usr/bin/python plot_twiss.py"


exit

# ERROR HANDLING
herr:
        if ($_error_line eq myErrLast)
                myErrCount++
        else
                myErrLast = $_error_line
                myErrCount = 0
        endif
        e = sprintf("%s on line %s (%s)",$_error_status, $_error_line, $_error_source_text)
        print/stdout e
        if ((myErrCount > myErrMax) and (myErrMax > 0))
                print "Last Error: " e
                exit/error_status
        endif
        wait/milliseconds 1000
        retry
exit

always:
unlock myLockName
```

Sample Output:

```
---- Summary ----
Histogram Size: 366 px
Threshold for Gaussian Detection: 0.100 (x Normalized Amplitude)
Total Number of Gaussians Found: 11
Gaussians:              Amp [Norm]      Cent [m]        Sig [m]         LinOff          Integral [m]    x` [Rad]        Sig` [Rad]
   [ 1]  57- 77px       1.181e-01       -4.962e-03      2.176e-05       4.528e-03       1.979e-06       -1.070e-02      3.134e-03
   [ 2]  77- 98px       1.675e-01       -4.065e-03      2.591e-05       6.871e-03       3.154e-06       -8.774e-03      2.538e-03
   [ 3]  98-121px       2.903e-01       -3.115e-03      4.305e-05       3.907e-03       1.964e-06       -6.731e-03      1.924e-03
   [ 4] 121-144px       4.850e-01       -2.112e-03      5.715e-05       -3.796e-03      1.908e-06       -4.567e-03      1.294e-03
   [ 5] 144-168px       7.766e-01       -1.088e-03      6.245e-05       1.082e-03       5.677e-07       -2.354e-03      6.567e-04
   [ 6] 168-197px       8.713e-01       0.000e+00       7.386e-05       3.213e-02       2.037e-05       0.000e+00       0.000e+00
   [ 7] 197-221px       7.440e-01       1.197e-03       6.593e-05       -1.196e-03      6.275e-07       2.600e-03       6.896e-04
   [ 8] 221-245px       5.352e-01       2.246e-03       5.801e-05       -5.003e-03      2.625e-06       4.867e-03       1.335e-03
   [ 9] 245-267px       2.601e-01       3.279e-03       3.984e-05       8.114e-03       3.902e-06       7.099e-03       1.975e-03
   [10] 267-289px       1.977e-01       4.234e-03       2.743e-05       5.104e-03       2.454e-06       9.155e-03       2.591e-03
   [11] 289-310px       1.041e-01       5.177e-03       2.514e-05       2.755e-03       1.264e-06       1.118e-02       3.202e-03


Second Moments:
   <x^2>  : 6.445e-06 m^2
   <x`^2> : 3.255e-05 rad^2
   <xx`>  : 1.393e-05 m*rad

Distance from Screen (L): 4.450e-01 m
Slit Width (w): 4.000e-05 m
Emittance: 3.976 mmmr

Twiss Parameters:
   Alpha : -3.503
   Beta  : 1.621
   Gamma : 8.186
```

Appendix C – fit.py

```
import numpy
from numpy import arange,exp, abs,concatenate,newaxis,delete,array, where
from numpy.random import randn
from scipy.optimize import leastsq
from scipy.integrate import quad as singleIntegral
from math import pi

myIn,x,xfit,y = [],[],[],[]

myFile = open("single.dat","r")
#myFile = open("sample.dat","r")
myIn = myFile.read().split('\n')           # Input from File
myFile.close()

n_pts = len(myIn) - 1
myIn = delete(myIn,n_pts)                  # Remove last, empty line

for l in myIn:
        row = l.split()
        x.append(row[0])
        y.append(row[1])

y = array(y)                                       # Convert from list to numpy array
y = y.astype(numpy.float)                  # Convert from string to float
x = array(x)
x = x.astype(numpy.float)
xmin = numpy.min(x)
xmax = numpy.max(x)

# My Fit Equation
def peval(x, p):
        return (p[0]*exp(-0.5*(((x-p[1])/p[2])**2))) + p[3]

def residuals(p, y, x):
        err = y - peval(x, p)
        return err

#print "Fitting Gaussian..."

# Initial Guesses for p[0] = Gauss Amp || p[1] = Centroid || p[2] = Sigma || p[3] = Linear Offset
p = [0.5, (xmin + ((xmax - xmin)/2)), 2, 0]
#print "Initial Fit Parameters: ", p

plsq = leastsq(residuals, p, args=(y, x))  # Do a least-squares fit
p = plsq[0]                                        # Pull out the fit parameters
#print "Final Fit Parameters: ", p

Ic = singleIntegral(peval, 0, n_pts, args=(p))     # Calculate the integral
I=(abs(Ic))[0]                                     # Pull out the real part
#print "Integral: ", I

myOut = '{0:-e}\n{1:-e}\n{2:-e}\n{3:-e}\n{4:-e}'.format(p[0],p[1],p[2],p[3],I)
myFile = open("lastfit.dat","w")
myFile.write(myOut)
myFile.close()

# These should be un-commented to view each fit
# import matplotlib.pyplot as plt
# plt.figure()
# xfit = arange(xmin,xmax,((xmax - xmin)/100))
# plt.plot(xfit,peval(xfit,p),x,y)
# plt.show()
```

Appendix D – plot_twiss.py

```
# Plot a phase ellipse based on twiss parameters: Alpha, Beta, Gamma, Epsilon
# (Epsilon = emittance)
# x` = (B +/- sqrt(B^2 - 4AC)) / 2A
#       A = Beta
#       B = 2 * Alpha * x
#       C = Gamma * x^2 - Epsilon

myIn,p = [],[]
myFile = open("twiss.dat","r")
myIn = myFile.read().split('\n')

Alpha = float(myIn[0])
Beta = float(myIn[1])
Gamma = float(myIn[2])
Epsilon = (float(myIn[3]) * 1000 * 1000)

p = [Alpha, Beta, Gamma, Epsilon]

def pevalpos(x, p):       # Positive half of the sqrt
        #      ( (     B      + (    B^2        -  4 *  A *           C              )^0.5 )/(2 *  A  ))
        return ( ( (2*p[0]*x) + ( (2*p[0]*x)**2 - (4 * p[1]* ((p[2]*(x*x)) - p[3]) ) )**0.5)/(2 * p[1]))

def pevalneg(x, p):       # Negative half of the sqrt
        #      ( (     B      - (    B^2        -  4 *  A *           C              )^0.5 )/(2 *  A  ))
        return ( ( (2*p[0]*x) - ( (2*p[0]*x)**2 - (4 * p[1]* ((p[2]*(x*x)) - p[3]) ) )**0.5)/(2 * p[1]))

import matplotlib.pyplot as plt
from numpy import arange
plt.figure()
x = arange(-3,3,0.001)
plt.plot(x,pevalpos(x,p),x,pevalneg(x,p))
plt.show()
```

Fun little derivation:

$$ax^2 + bx + c = 0$$

$$x^2 + \frac{bx}{a} = -\frac{c}{a}$$

$$x^2 + \frac{bx}{a} + \left(\frac{b}{2a}\right)^2 = \left(\frac{b}{2a}\right)^2 - \frac{c}{a}$$

$$\left(x + \frac{b}{2a}\right)^2 = \frac{b^2 - 4ac}{4a^2}$$

$$x + \frac{b}{2a} = \frac{\pm\sqrt{b^2 - 4ac}}{2a}$$

$$\therefore\ x = \frac{b \pm \sqrt{b^2 - 4ac}}{2a}$$

Appendix E – Full Resolution Image of 3200_min_dE.X110.png