ECE 270: Computer Methods in ECE


# Assignment #4

Sampling and Plotting

Hussein El-Souri

April 22, 2017

# 1 Statement of the Problem

The purpose of this assignment is to create a function called the damped sinu-soid, which is given by:

$$y = eaxcos(wx) \tag{1}$$

Also, we will make a plot of a real world function based on underlying function rather than a mathematical formula.

# 2 Description of Solution

To make a plot of the damped sinusoid function, we need to rst create a set of x-values and calculate how many x-samples there are. The rst function takes in the following arguments,xmin, xmax, steps, and an array.
xmin:the minimum x-value xmax:the maximum x-value Step: the amount of space between samples The sampling process will contain the following sequence of values:

$$x : x_{min}, x_{min} + step, x_{min} + 2 * step, ..., x_{max} \tag{2}$$

The total number of n-samples will be:

$$n = \frac{x_{max} - x_{min}}{2} \tag{3}$$

The $i^{th}$ sample is given by:

$$x : x_{min} + i * step, \ for\ i = 0, 1, 2, ....n \tag{4}$$

Next, you will use the damped sinusoid function (1) to get y-values from x-values. The function will take in the following arguments: a value for alpha, a value for w, an x-array, a y-array, and a n-sample.
It will then input these x-values into the damped sinusoid function to get y-values.
in: the value you want to map
$in_{Min}$: the minimum value of your in
$in_{Max}$:the maximum value of your in
$out_{Min}$:the minimum value of your out
$out_{Max}$: the maximum value of your out
The rst thing this function does is nd the slope:

$$m = \frac{outMax - outMin}{inMax - inMin} \tag{5}$$

Next, it finds the intercept:

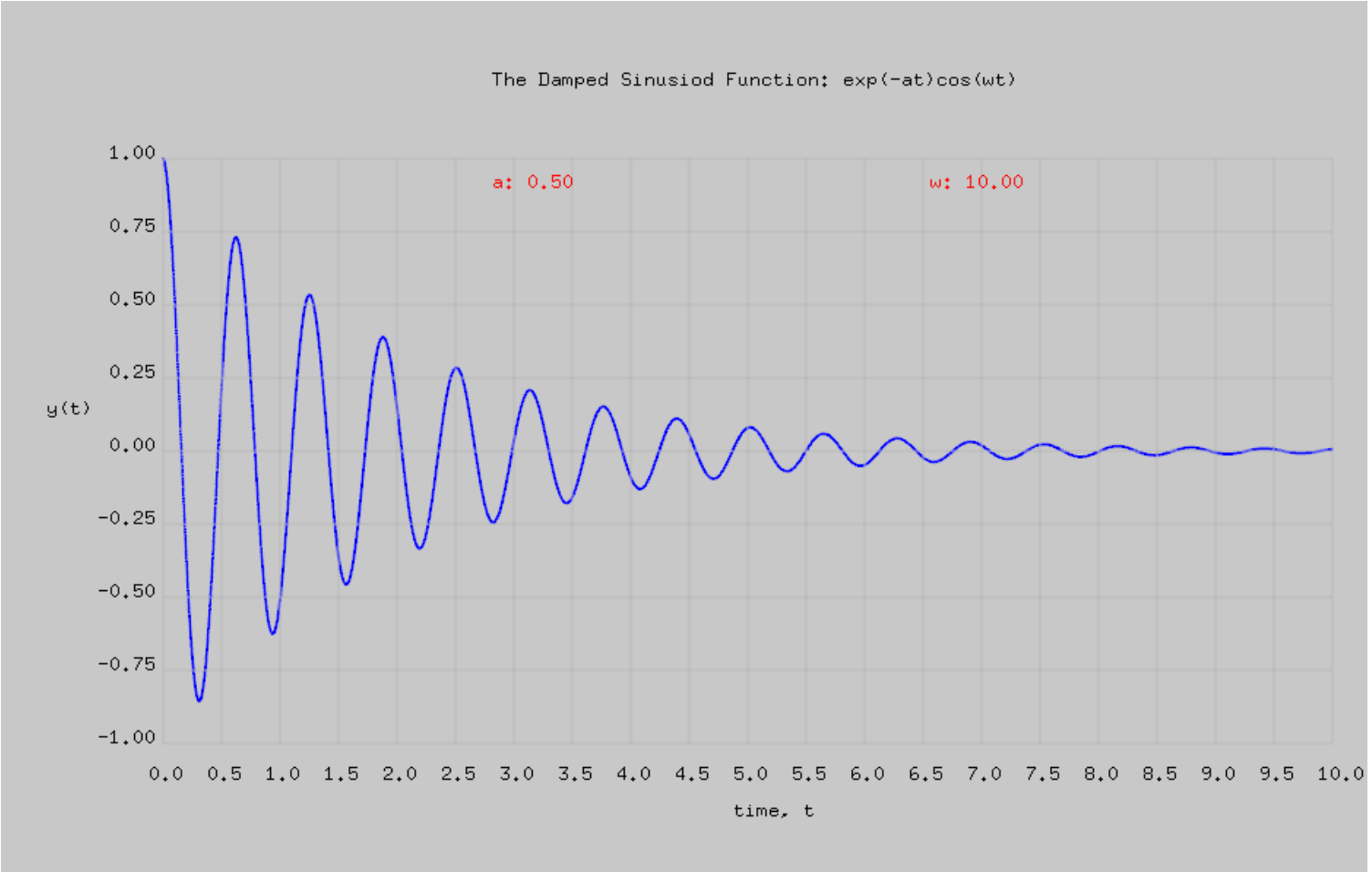
$$b = outMax - m * inMax \tag{6}$$

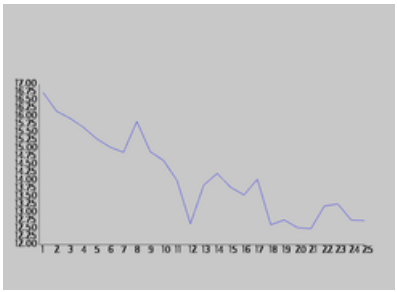
Lastly, it plugs the values into slope intercept form:

$$out = m * in + b \tag{7}$$

## 3 Testing and Output

The sample I used for the x-values is from 0 to 10 with a step size of 0.05. The parameters of the y-value are -1 to 1. Alpha is 0.5. The value for w is 10.0.



# 4 Code

```
//ofApp.h 4 part a)
//My Variables

 float step = 0.001;



 float xMin = 0.00;

 float xMax = 10.0;



 float yMin = -1.00;

 float yMax = 1.00;



 float xPixMin = 200;

 float xPixMax = 1000;



 float yPixMin = 600;

 float yPixMax = 200;



 int n; //number of samples



 float x[N_MAX];

 float y[N_MAX];







 float xPix[N_MAX];

 float yPix[N_MAX];









 float xDrawMax[N_MAX];

 float xDrawMin[N_MAX];

 float yDrawMax[N_MAX];

 float yDrawMin[N_MAX];
```

```
    //My Functions
    int getXSamples (float xMin, float xMax, float step, float x[]);
    void getDampedCosSample (int n, float x[], float y[], float alpha, float w);
    float map (float in, float inMin, float inMax, float outMin, float outMax);
    void map_vec (int n, float in[], float out[], float inMin, float inMax, float outMin, float outMax);
    void printArray (int dim, float x[], char label[]);
};
//--------------------------------------------------------------------------

//--ofApp.cpp 4 part a)

#include "ofApp.h"

int ofApp::getXSamples (float xMin, float xMax, float step, float x[])

{

    int i;

    int numSamples;

    numSamples = (xMax - xMin) / step + 1;


    for (i = 0; i < numSamples; ++i)

    {

        x[i] = xMin + i * step;

    }


    return numSamples;

}

void ofApp::getDampedCosSample (in


    for(i = 0; i < n; ++i)

    {

        y[i] = exp(-alpha * x[i]) * cos(w * x[i]);

    }

}

float ofApp::map (float in, float inMin, float inMax, float outMin, float outMax)

{

    float m;

    float b;
```

```
    float out;

    //slope
    m = (outMax - outMin) / (inMax - inMin);
    //y-inter
    b = outMax - m * inMax;
    //y = mx + b
    out = m * in + b;

    return out;

}
void ofApp::map_vec (int n, float in[], float out[], float inMin, float inMax, float outMin, float outMax)
{
    int i;

    for (i = 0; i < n; ++i)
    {
        out[i] = map (in[i], inMin, inMax, outMin, outMax);
    }
}
void ofApp::printArray (int dim, float x[], char label[])
{
    int i;

    printf("\n%s: ", label);

    for (i = 0; i < dim; ++i)
    {
        printf("%.2f ", x[i]);
    }
}
```

```
//--------------------------------------------------------------
void ofApp::setup()
{
    float xSamples[N_MAX];
    float ySamples[N_MAX];
    float alpha = 0.50;
    float omega = 10.00;

    n = getXSamples(xMin, xMax, step, xSamples);
    printArray(n, xSamples, "X values");

    getDampedCosSample (n, xSamples, ySamples, alpha, omega);
    printArray(n, ySamples, "Y values");

    map_vec(n, xSamples, xPix, xMin, xMax, xPixMin, xPixMax);
    printArray(n, xPix, "XPix");

    map_vec(n, ySamples, yPix, yMin, yMax, yPixMin, yPixMax);
    printArray(n, yPix, "YPix");
}

//--------------------------------------------------------------
void ofApp::update()
{

}

//--------------------------------------------------------------
void ofApp::draw()
{
    ofSetBackgroundColor(255, 255, 255);

    float xIntercept;
```

```
float yIntercept;
int i;

for(i = 0; i < n; ++i)
{
    ofSetBackgroundAuto(false);
    ofSetColor(0, 0, 255);
    ofSetLineWidth(10.0f);
    ofCircle(xPix[i], yPix[i], 1);
}

xDrawMax[0] = 1000;
xDrawMin[0] = 200;
yDrawMax[0] = 200;
yDrawMin[0] = 200;

//code the plot y-axis
yIntercept = (yMax - yMin) / YSTEP + 1;

for(i = 0; i < yIntercept; i++)
{
    ofSetBackgroundAuto(false);

    ofSetColor(169, 169, 169);
    ofSetLineWidth(0.001f);
    ofLine(xDrawMin[0], yDrawMin[i], xDrawMax[0], yDrawMax[i]);

    ofSetColor(0, 0, 0);

    yDrawMax[i+1] = yDrawMax[i] + 50;
    yDrawMin[i+1] = yDrawMin[i] + 50;
}
```

```
xDrawMax[0] = 200;

xDrawMin[0] = 200;

yDrawMax[0] = 200;

yDrawMin[0] = 600;


//code the plot x-axis

xIntercept = (xMax - xMin) / XSTEP + 1;


for(i = 0; i < xIntercept; i++)

{

    ofSetBackgroundAuto(false);


    ofSetColor(169, 169, 169);

    ofSetLineWidth(0.001f);

    ofLine(xDrawMin[i], yDrawMin[0], xDrawMax[i], 200);


    ofSetColor(0, 0, 0);


    xDrawMax[i+1] = xDrawMax[i] + 40;

    xDrawMin[i+1] = xDrawMin[i] + 40;

}


ofSetColor(0, 0, 0);

ofDrawBitmapString("The Damped Sinusiod Function: exp(-at)cos(wt)", 425, 150);

ofSetColor(255, 0, 0);

ofDrawBitmapString("a: 0.50", 425, 220);

ofDrawBitmapString("w: 10.00", 725, 220);


//plot y values

ofSetColor(0, 0, 0);

ofDrawBitmapString("y(t)", 200 - 80, 375);


ofDrawBitmapString(" 1.00", 200 - 45, 200);
```

```
    ofDrawBitmapString(" 0.75", 200 - 45, 250);
    ofDrawBitmapString(" 0.50", 200 - 45, 300);
    ofDrawBitmapString(" 0.25", 200 - 45, 350);
    ofDrawBitmapString(" 0.00", 200 - 45, 400);
    ofDrawBitmapString("-0.25", 200 - 45, 450);
    ofDrawBitmapString("-0.50", 200 - 45, 500);
    ofDrawBitmapString("-0.75", 200 - 45, 550);
    ofDrawBitmapString("-1.00", 200 - 45, 600);
    //plot x values
    ofDrawBitmapString("time, t", 600 - 10, 600 + 50);

    ofDrawBitmapString("0.0", 200 - 10, 600 + 25);
    ofDrawBitmapString("0.5", 240 - 10, 600 + 25);
    ofDrawBitmapString("1.0", 280 - 10, 600 + 25);
    ofDrawBitmapString("1.5", 320 - 10, 600 + 25);
    ofDrawBitmapString("2.0", 360 - 10, 600 + 25);
    ofDrawBitmapString("2.5", 400 - 10, 600 + 25);
    ofDrawBitmapString("3.0", 440 - 10, 600 + 25);
    ofDrawBitmapString("3.5", 480 - 10, 600 + 25);
    ofDrawBitmapString("4.0", 520 - 10, 600 + 25);
    ofDrawBitmapString("4.5", 560 - 10, 600 + 25);
    ofDrawBitmapString("5.0", 600 - 10, 600 + 25);
    ofDrawBitmapString("5.5", 640 - 10, 600 + 25);
    ofDrawBitmapString("6.0", 680 - 10, 600 + 25);
    ofDrawBitmapString("6.5", 720 - 10, 600 + 25);
    ofDrawBitmapString("7.0", 760 - 10, 600 + 25);
    ofDrawBitmapString("7.5", 800 - 10, 600 + 25);
    ofDrawBitmapString("8.0", 840 - 10, 600 + 25);
    ofDrawBitmapString("8.5", 880 - 10, 600 + 25);
    ofDrawBitmapString("9.0", 920 - 10, 600 + 25);
    ofDrawBitmapString("9.5", 960 - 10, 600 + 25);
    ofDrawBitmapString("10.0", 1000 - 10, 600 + 25);
}
```

```
//--------------------------------------------------------------------------------//

//--------------------------------------------------------------------------------//
//--------------------------------------------------------------------------------//

//--------------------------------------------------------------------------------//

//ofapp.h 4 part b)

float step = 0.1;


    float xMin = 0.00;

    float xMax = 168.1;


    float yMin = 0;

    float yMax = 350;


    float xPixMin = 1000;

    float xPixMax = 400;


    float yPixMin = 600;

    float yPixMax = 200;


    int n; //number of samples


    float xPix[N_MAX];

    float yPix[N_MAX];


    //My Functions

    int getXSamples (float xMin, float xMax, float step, float x[]);

    void getDampedCosSample (int n, float x[], float y[], float alpha, float w);

    float map (float in, float inMin, float inMax, float outMin, float outMax);

    void map_vec (int n, float in[], float out[], float inMin, float inMax, float outMin, float outMax);

    void printArray (int dim, float x[], char label[]);

//--------------------------------------------------------------------------------------------------

//--------------------------------------------------------------------------------------------------
//ofApp.cpp 4 paart b)

#include "ofApp.h"
```

```
int ofApp::getXSamples (float xMin, float xMax, float step, float x[])
{
    int i;
    int numSamples;
    numSamples = (xMax - xMin) / step + 1;

    for (i = 0; i < numSamples; ++i)
    {
        x[i] = xMin + i * step;
    }

    return numSamples;
}
void ofApp::getDampedCosSample (int n, float x[], float y[], float alpha, float w)
{
    int i;

    for(i = 0; i < n; ++i)
    {
        y[i] = exp(-alpha * x[i]) * cos(w * x[i]);
    }
}
float ofApp::map (float in, float inMin, float inMax, float outMin, float outMax)
{
    float m;
    float b;
    float out;

    //slope
    m = (outMax - outMin) / (inMax - inMin);
    //y-inter
    b = outMax - m * inMax;
    //y = mx + b
```

```
    out = m * in + b;


    return out;


}
void ofApp::map_vec (int n, float in[], float out[], float inMin, float inMax, float outMin, float outMax)
{
    int i;


    for (i = 0; i < n; ++i)
    {
        out[i] = map (in[i], inMin, inMax, outMin, outMax);
    }
}
void ofApp::printArray (int dim, float x[], char label[])
{
    int i;


    printf("\n%s: ", label);


    for (i = 0; i < dim; ++i)
    {#include "ofApp.h"
int ofApp::getXSamples (float xMin, float xMax, float step, float x[])
{
    int i;
    int numSamples;
    numSamples = (xMax - xMin) / step + 1;


    for (i = 0; i < numSamples; ++i)
    {
        x[i] = xMin + i * step;
    }
```

```
    return numSamples;
}
void ofApp::getDampedCosSample (int n, float x[], float y[], float alpha, float w)
{
    int i;

    for(i = 0; i < n; ++i)
    {
        y[i] = exp(-alpha * x[i]) * cos(w * x[i]);
    }
}
float ofApp::map (float in, float inMin, float inMax, float outMin, float outMax)
{
    float m;
    float b;
    float out;

    //slope
    m = (outMax - outMin) / (inMax - inMin);
    //y-inter
    b = outMax - m * inMax;
    //y = mx + b
    out = m * in + b;

    return out;

}
void ofApp::map_vec (int n, float in[], float out[], float inMin, float inMax, float outMin, float outMax)
{
    int i;

    for (i = 0; i < n; ++i)
    {
```

```
        out[i] = map (in[i], inMin, inMax, outMin, outMax);
    }
}
void ofApp::printArray (int dim, float x[], char label[])
{
    int i;

    printf("\n%s: ", label);

        printf("%.2f ", x[i]);
    }
}

//--------------------------------------------------------------
void ofApp::setup()
{
    float xSamples[N_MAX];
    float xData[N_MAX];
    int i;
    int k;
    int j;
    int isReading;
    FILE *fp;

    fp = fopen("C:\\Users\\shawn\\Google Drive\\Programming\\OpenFramworks\\examples\\Assignments\\7 assignment
        4.2\\bin\data\\TeslaData.csv", "r");

    isReading = 1;
    j = 0;

    while(isReading == 1)
    {
        k = fscanf(fp, "%f", &xData[j]);
```

```
        if (k == EOF)
            isReading = 0;

        else
            j++;
    }

    //printArray(j, xData, "Data values");

    n = getXSamples(xMin, xMax, step, xSamples);
   // printArray(n, xSamples, "X values");

    map_vec(n, xSamples, xPix, xMin, xMax, xPixMin, xPixMax);
    //printArray(n, xPix, "XPix");

    map_vec(n, xData, yPix, yMin, yMax, yPixMin, yPixMax);
    //printArray(n, yPix, "YPix");

    fclose(fp);
}

//--------------------------------------------------------------
void ofApp::update()
{

}

//--------------------------------------------------------------
void ofApp::draw()
{
    ofSetBackgroundColor(255, 255, 255);
```

```
int i;


for(i = 0; i < n; ++i)
{
    ofSetBackgroundAuto(false);
    ofSetColor(0, 0, 255);
    ofSetLineWidth(10.0f);
    ofFill();
    ofCircle(xPix[i], yPix[i], 1);
}


ofSetColor(169, 169, 169);
ofSetLineWidth(0.001f);

ofSetColor(0, 0, 0);
ofDrawBitmapString("Tesla (TSLA)", 600, 275);
ofDrawBitmapString("Price", 275, 450);
ofDrawBitmapString("Year", 750, 650);


ofDrawBitmapString("300", 350, 255);
ofDrawBitmapString("275", 350, 285);
ofDrawBitmapString("250", 350, 315);
ofDrawBitmapString("225", 350, 345);
ofDrawBitmapString("200", 350, 375);
ofDrawBitmapString("175", 350, 405);
ofDrawBitmapString("150", 350, 435);
ofDrawBitmapString("125", 350, 465);
ofDrawBitmapString("100", 350, 495);
ofDrawBitmapString("75", 350, 525);
ofDrawBitmapString("50", 350, 555);
ofDrawBitmapString("25", 350, 585);
ofDrawBitmapString("0", 350, 615);


ofDrawBitmapString("2011", 450, 625);
```

```
    ofDrawBitmapString("2012", 530, 625);
    ofDrawBitmapString("2013", 610, 625);
    ofDrawBitmapString("2014", 690, 625);
    ofDrawBitmapString("2015", 770, 625);
    ofDrawBitmapString("2016", 850, 625);
    ofDrawBitmapString("2017", 930, 625);
}
```