uM-FPU Application Note 3

Calculating Trend Lines

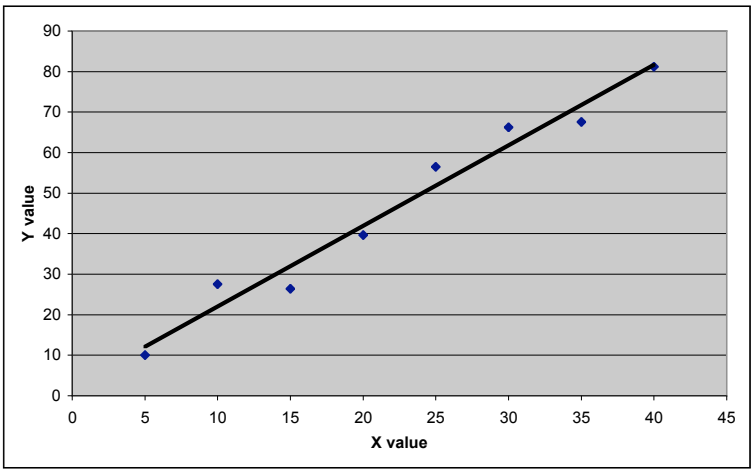
Micromega Corporation

This application note describes a method of calculating a trend line for a set of data points. A least-squares line fitting algorithm is implemented using the uM-FPU floating point coprocessor.

Introduction

A useful method of calibrating a device that has a linear response is to collect a series of data points where both the *x* and *y* values are known or can be measured, and then calculating a trend line which best fits the data to a straight line. Once the trend line is known, it can then be used to calculate *y* given *x*, or *x* given *y*. Figure 1 shows a series of data points that have been plotted on a graph, and the trend line that has been calculated for the data.

Figure 1 – Data Points with Trend Line



The Trend Line

It isn't necessary to understand all of the underlying math to use this algorithm. All you really need to know is that the equation for a line is defined as follows:

$$y = mx + b$$

Where *m* is the slope of the line and the *b* is the intercept. The algorithm calculates the value of *m* and *b* for the trend line that best represents the data points. If you want to look at the underlying math, refer to Appendix A for a description of the least squares algorithm.

Using the uM-FPU to find the Trend Line

The algorithm calculates the slope (*m*) and the intercept (*b)* for the trend line based on a set of *x* and *y* data points. The data points are stored on the microcontroller and sent to the uM-FPU as required by the algorithm. The uM-FPU code for the trend line algorithm is shown below. The following description uses

pseudo-code to reference the x and y values. For example, `x[i] low byte`, means the low byte of the `x` value for data point `i`. The specific method of storing *x* and *y* data point values will vary depending on the microcontroller and the size of the *x* and *y* values.

Microcontroller Variable

```
x[n]                    ; 16-bit x values
y[n]                    ; 16-bit y values
```

uM-FPU Register Definitions

```
M                       ; slope of the trend line (m)
B                       ; intercept of the trend line(b)
N                       ; number of data points
Sx                      ; sum of the x values
Sy                      ; sum of the y values
SxN                     ; sum of x divided by N
T                       ; temporary value t
Stt                     ; sum of t * t
```

Step1 Initialize the variables

```
SELECTA+N               ; N = number of data points
LOADBYTE
ndata
FSET

XOP, LOADZERO           ; register 0 = 0.0
SELECTA+Sx              ; Sx = 0.0
FSET
SELECTA+Sy              ; Sy = 0.0
FSET
SELECTA+Stt             ; Stt = 0.0
FSET
SELECTA+M               ; M = 0.0
FSET
```

Step 2 Calculate the sum of the x and y values

```
loop for i = 1 to n
{
   wait until uM-FPU is ready
   SELECTA+Sx           ; Sx =  Sx + xval(i)
   LOADWORD
   x[i] high byte
   x[i] low byte
   FADD

   SELECTA+Sy           ; Sy =  Sy + yval(i)
   LOADWORD
   y[i] high byte
   y[i] low byte
   FADD
}
```

Step 3 Calculate the sum of the x values divided by the number of data points

```
SELECTA+SxN             ; SxN = Sx / N
FSET+Sx
FDIV+N
```

Step 4 Calculate the slope (M)

```
loop for i = 1 to n (where n = number of data points)
{
   wait until uM-FPU is ready
   SELECTA+T            ; T = xval(i) - SxN
   LOADWORD
   x[i] high byte
   x[i] low byte
   FSET
   FSUB+SxN

   SELECTA              ; Stt = Stt + (T*T)
   FSET+T
   FMUL+T
   SSELECTA+Stt
   FADD

   SELECTA+T            ; M = M + T  * yval(i)
   LOADWORD
   y[i] high byte
   y[i] low byte
   FMUL
   SELECTA+M
   FADD+T
}
FDIV+Stt                ; M = M / Stt
```

Step 5 Calculate the intercept (B)

```
SELECTA+Sx              ; B = (Sy – Sx * M) / N
FMUL+M
SELECTA+B
FSET+Sy
FSUB+Sx
FDIV+N
```

Result

The uM-FPU *M* and *B* registers now contain the slope and intercept of the trend line.

In steps 2 and 4 of the algorithm described above, the `x` and `y` values are assumed to be 16-bit signed values and are loaded with the `LOADWORD` instruction. The `LOADBYTE`, `LOADUBYTE`, `LOADUWORD` and `WRITEB` instruction could also be used depending on data type used to store the `x` and `y` values. See the sample programs for specific examples.

Further Information

Sample programs that calculate the trend line shown in Figure 1 are available for various microcontrollers.

Check the Micromega website at www.micromegacorp.com for up-to-date information.

Appendix A - The Least Squares Algorithm

The least-squares curve fitting algorithm is a mathematical procedure for finding the best-fitting curve to a given set of data points by minimizing the sum of the squares of the offsets (the difference between the data points and a points on the curve). We will use this technique to determine the straight-line approximation for a set of data points, also referred to as the linear regression line or trend line.

$$n = \text{number of data points}$$

$$S_x = \sum_{i=1}^{n} x_i \qquad S_y = \sum_{i=1}^{n} y_i \qquad S_{xx} = \sum_{i=1}^{n} x_i^2 \qquad S_{xy} = \sum_{i=1}^{n} x_i y_i$$

$$m = \frac{nS_{xy} - S_x S_y}{nS_{xx} - (S_x)^2} \qquad b = \frac{S_{xx} S_y - S_x S_{xy}}{nS_{xx} - (S_x)^2}$$

When implemented as shown above, these formulas are susceptible to roundoff error, so they are rewritten and calculated as follows:

$$t_i = x_i - \frac{S_x}{n}, \quad \text{i} = 1, 2, \ldots, n \qquad S_{tt} = \sum_{i=1}^{n} t_i^2$$

$$m = \frac{1}{S_{tt}} \sum_{i=1}^{n} t_i y_i \qquad b = \frac{S_y - S_x m}{n}$$