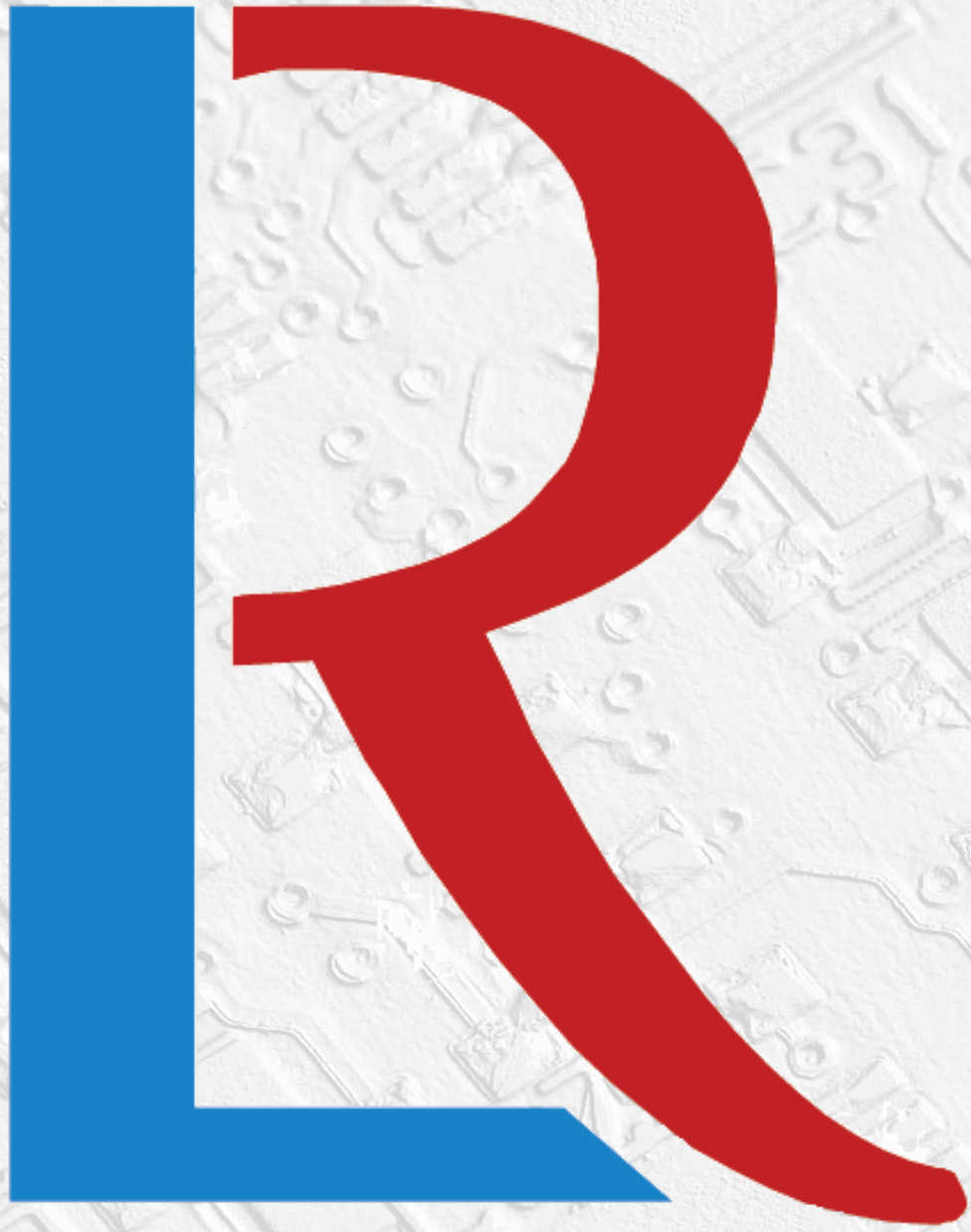


LifeRacing

Math Manual

Document Date: 2015-08-12

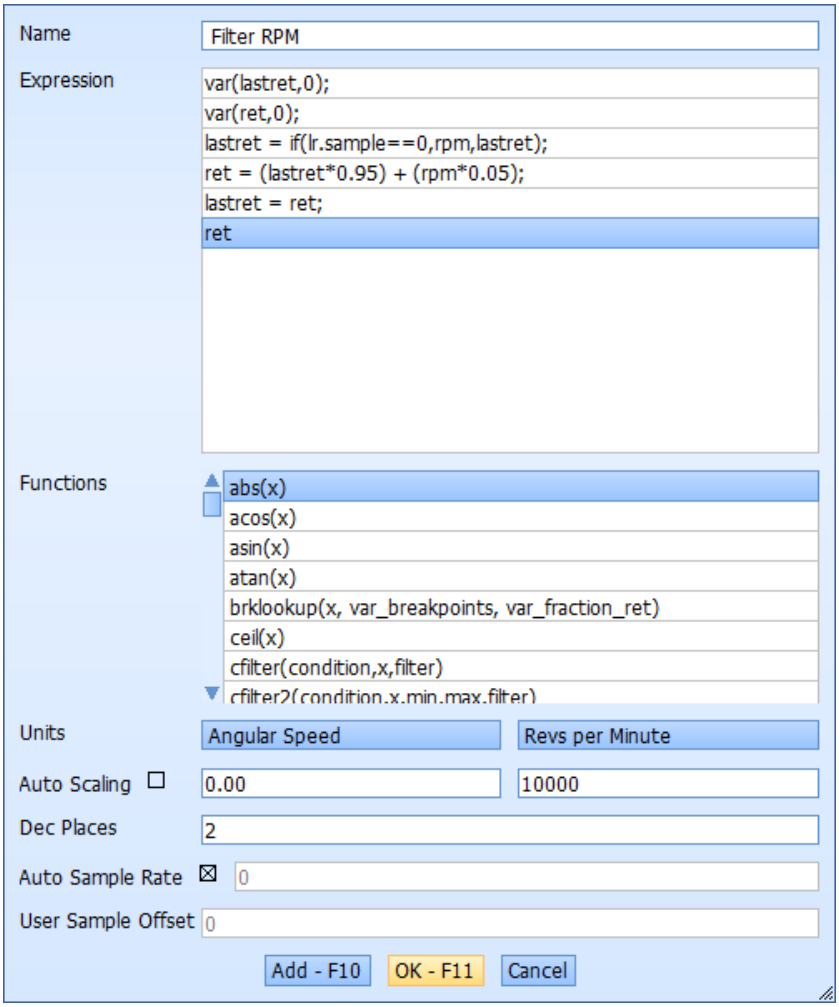
Author: M.H

1 Introduction

Math can be used to manipulate monitoring items on several Life Racing applications. It can be used to process downloaded track data in LifeView or used on live data with LifeMon. Life Racing displays can also function in a similar way to LifeMon when math items are created in LifeDash (Dash math functions are currently limited). There is a large number of mathematical functions, capable of a wide range of tasks, resulting in a flexible and powerful tool.

2 Writing an Expression

Math expressions can vary in complexity from simple multiplications to multidimensional lookup tables. Care must be taken to correctly write expressions to avoid errors, particularly as they become more complicated. Expressions can consist of several lines and several functions. A new line will continue from the previous line. To mark the end of a particular function, use a semi-colon (;). The final function does not require this as this will be taken as the final value.



All useable functions (to date of document) are described in this document in the following format:

Name
Description
Example

Contents

2.1 Operators

Several standard operators can be used within expressions and include the following:

(	Open bracket	<	Less than	&	Logical AND	&	Bitwise AND
)	Close bracket	<=	Less than or equal	||	Logical OR	|	Bitwise OR
*	Multiply	>	More than	!	Logical NOT	^	Bitwise XOR
/	Divide	>=	More than or equal	?	Then	~	Bitwise NOT
%	Modulus	==	Equal to	:	Else	<<	Bitwise Left Shift
+	Add	!=	Not Equal	,	Comma operator	>>	Bitwise Right Shift
-	Subtract	pi	π, 3.14159265…				

2.2 Definitions and References

lr. Functions (LifeView only)
lr. functions are fixed variables that return values for:

Current sample number.
lr.sample

Time into session in seconds.
lr.time

The time taken for current sample.
lr.sampleSize

const(name,value)
Define a constant and its initial value.
const(a,3)

var(name,value)
Declare variable and initial constant value.
var(x,0)
Multi-dimensional variables allow entire tables of data to be defined. When recalling, note that entry numbers begin at 0.
var(x[5],1,2,3,4,5);
x[3] =4
var(x[5,2],
0,1,2,3,4,
5,6,7,8,9);
x[3,1] =8

undefined()
Allows blank data.
undefined() =###

isdefined(x)
Returns 1 if x is defined. Returns 0 if x is set as undefined or there is a gap in the data. Useful for flagging sensor errors.
lam1(x)

math("math fn name")
Use result from other math function.
math(acceleration)/9.81

previous(initial_const_value)
Returns the previous expression result. Initial value must be specified.
previous(0)

previous(initial_const_value,sample)
Returns the previous nth result from the expression.
previous(0,5)

previousx(x,initial_const_value)
Returns the previous value of channel x.
previous(rpm,0)

previousx(x,initial_const_value,sample)
Returns the previous nth value of channel x.
previous(rpm,0,10)

itemvalueat(monitem, absolute time)
Returns monitoring item value at the specified time. Time is defined in seconds.
itemvalueat(rpm,500)

2.3 Numerical Analysis

abs(x)
Absolute value (magnitude) of x. Useful for inverting negative values.
abs(-10) *=10*

sign(x)
Sign value of x. returns 1 or -1 depending on sign.
sign(ignbase1)

round(x)
Round to nearest integer.
round(1.2) *=1.0*
round(1.8) *=2.0*

floor(x)
Round down to nearest integer.
floor(1.2) *=1.0*
floor(1.8) *=1.0*

ceil(x)
Round up to nearest integer.
ceil(1.2) *=2.0*
ceil(1.8) *=2.0*

fmod(x,y)
Remainder of x/y.
fmod(22,6) *=4*

min(x,y,...)
Returns the lowest value of bracketed numbers or items.
min(tps1A,tps1B)

max(x,y,...)
Returns the highest value of bracketed numbers or items.
max(tps1A,tps1B)

2.4 Trigonometry

sin(x)
Sine of x where x is in radians.
sin(pi) *=0*

cos(x)
Cosine of x where x is in radians.
cos(pi) *=-1*

tan(x)
Tangent of x where x is in radians.
tan(pi) *=0*

asin(x)
Arcsine of x (reverse sine) in radians range -pi/2 – pi/2.
asin(1) *=pi/2*

acos(x)
Arccosine of x (reverse cosine) in radians range 0 – pi.
acos(1) *=0*

atan(x)
Arctangent of x (reverse tangent) in radians range -pi/2 – pi/2.
atan(1) *=pi/4*

2.5 Mathmatical Operators

exp(x)
Exponential value of x, e^x.
exp(2) *=e^2=7.39*

log(x)
Log of x with base 10.
log(500) *=2.7*

pow(x,y)
Raise x to the power of y.
pow(2,3) *=8*

sqrt(x)
Square root of x.
sqrt(9) *=3*

derivative(x)
Differentiate x by time.
derivative(vehicleSpeed) *=acceleration*

integral(x)
Integrate x by time.
integral(vehicleSpeed) *=distance*

integral(x,condition)
Integrate x when conditions are true. Resets to 0 when condition is not met.
integral(vehicleSpeed,laptime()>0) *=lap distance*

if(condition,then,else)
Classic if statement.
if(rpm>200,1,0)

2.6 Session Timing

time()
Current log time in seconds.
time()

ctime()
Comparative time of current lap to fastest lap. Requires distance channel.
ctime()

ctime(lap)
Comparative time of current lap to a defined other in the current session. Requires distance channel.
ctime(3)

ctime(session,lap)
Comparative lap time to another in specified session when more than one file is loaded. Sessions start at 0. Requires distance channel.
ctime(1,3)

laptime()
Time into current lap.
laptime()

laptime(session)
Time into current lap in specified session when more than one file is loaded.
laptime(3)

laptime(session,lap)
Total duration of specific lap time.
laptime(1,3)

fastestlap(session)
Identity of fastest lap in specified session. Sessions start at 0.
fastestlap(1)

fastestlapsession()
Identity of session with fastest lap.
fastestlapsession()

getlap()
Returns lap number.
getlap()

getbeacontimes(lap, var_starttime, var_endtime)
Sets two variables as lap start and end time.
var(x,0);
var(y,0);
getbeacontimes(2,x,y)

2.7 Data Analysis

extrapolate(channel, x1,y1, x2,y2 ...)
Find an estimated value of y from x (channel) from xy pairs. x values higher or lower than the list will continue the trend at that point.
extrapolate(1.3,0,0,1,2,2,4,3,6,4,8)==2.6
extrapolate(5,0,0,1,2,2,4,3,6,4,8)==10

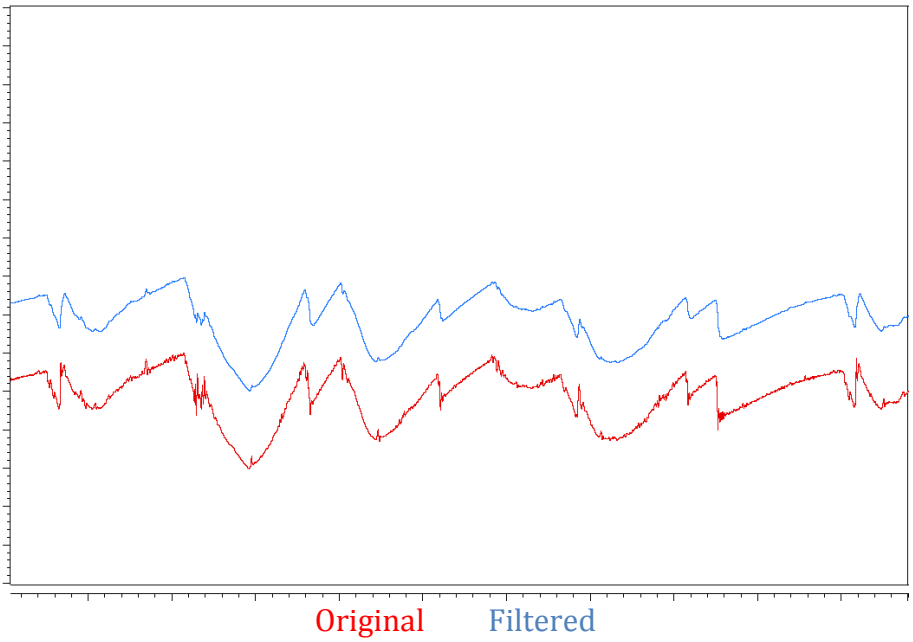
interpolate(channel, x1,y1, x2,y2, ...)
Find an estimated value of y for 'channel' from xy pairs. Item values higher or lower than the list are cut.
interpolate(1.3,0,0,1,2,2,4,3,6,4,8)==2.6
interpolate(5,0,0,1,2,2,4,3,6,4,8)==8

lookup(x, y0,y1,y2...)
Returns the xth value in the array defined by y0,y1,y2 ect.
lookup(3, 0,2000,3000,4000,5000)==4000

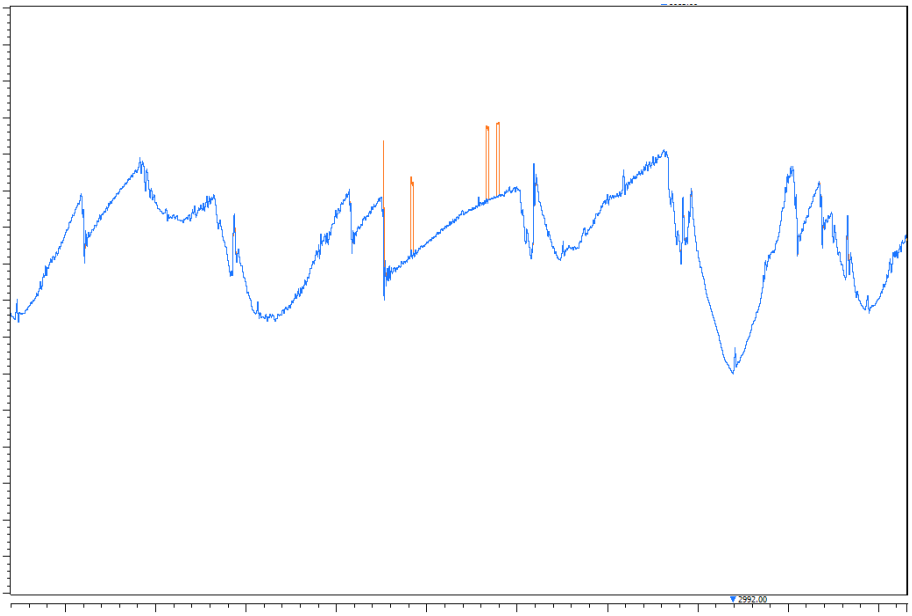
brklookup(x, var_breakpoints, var_fraction_ret)
Returns the lower extrapolated value and fraction to next breakpoint.
var(xbrk[5],0,2000,3000,4000,5000);
var(frac,0);
brklookup(2800,xbrk,frac)==1, frac==0.8

2.8 Filters

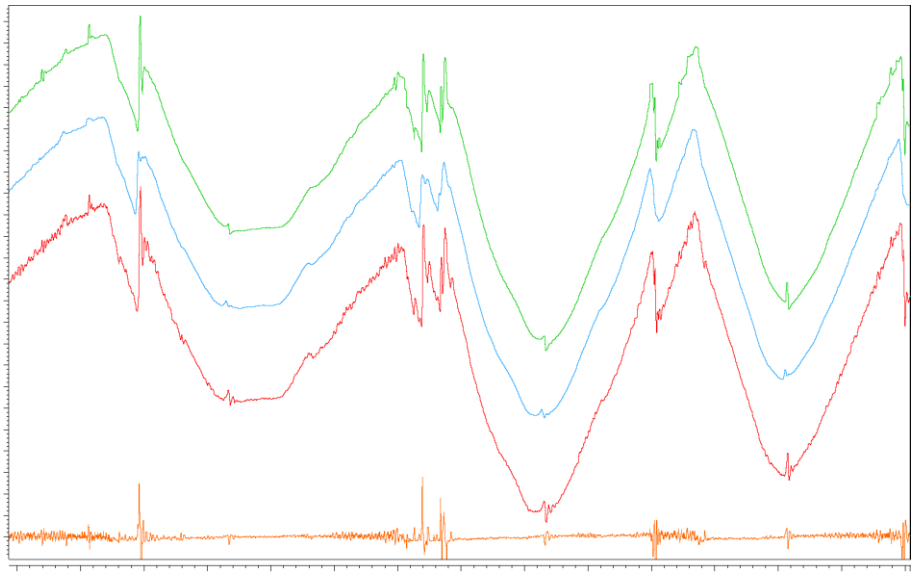
filter(x,filter)
Smooths the x trace at a magnitude of filter (between 0 and 1).
filter(rpm,0.9)



nfilter(x,delta)
Noise filter of x where delta is max difference allowed between samples. Useful for removing spikes in data.
nfilter(rpm,500)



cfilter(condition,x,filter)
Smooths the x trace when condition is true at a magnitude of filter (between 0 and 1).
cfilter(abs(derivative(rpm))<2800,rpm,0.9)



cfilter2(condition,x,min,max,filter)
Smooths the x trace when conditions are met and only between the specified min and max values at a magnitude of filter (between 0 and 1).
cfilter2(abs(derivative(rpm))<2800,rpm,0,6000,0.9)

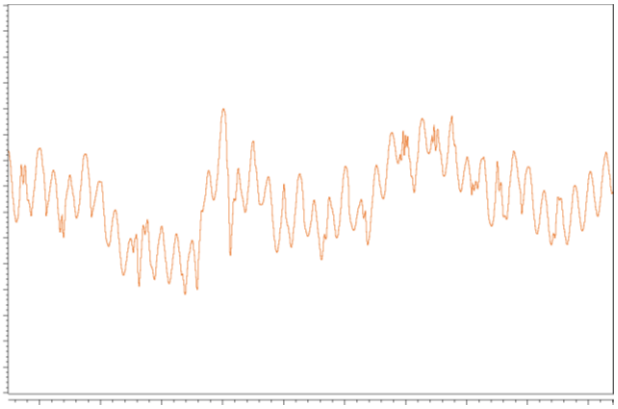
Running average functions are primarily used as noise filters. By setting the number of samples as the period of the noise frequency, the noise should cancel itself out. If the noise frequency changes a varying number of samples can be used. This should be a function of the cause of noise. Use the delta to work out the frequency and number of samples across the period making sure the sample rate is the same for the data and runavg filter. Running min/max functions are used in exactly the same way but return the lowest or highest values instead of the average.

runmin(x,num_samples)
Returns minimum value of x over the last n samples.

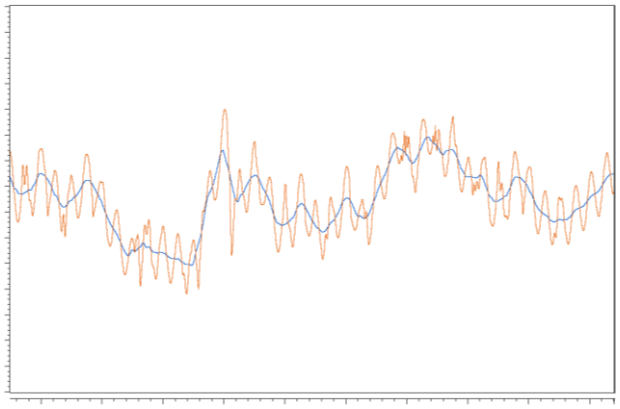
runmax(x,num_samples)
Returns maximum value of x over the last n samples.

runavg(x,num_samples)
Running average of x using n number of samples.

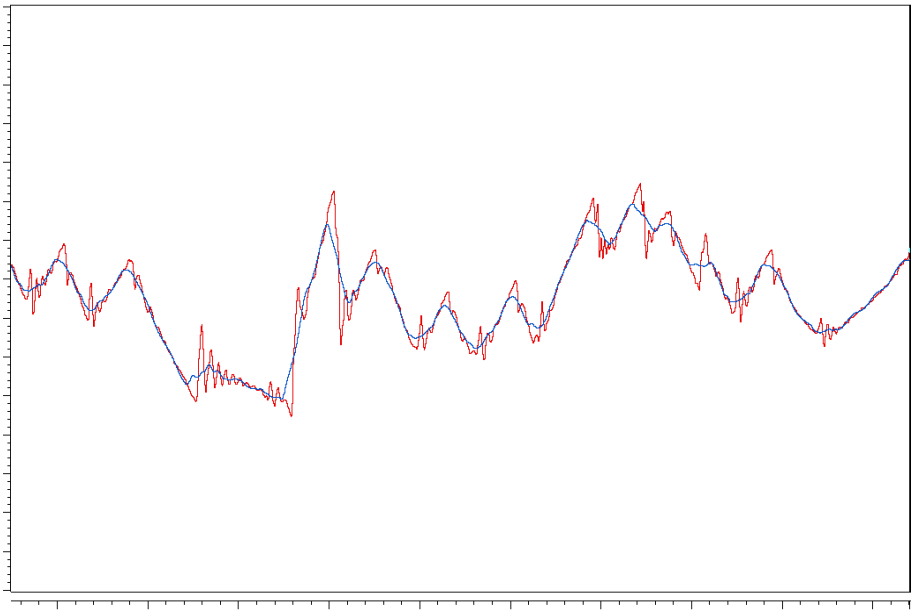
Constant frequency, measured delta 0.63 seconds at 100Hz, 63 samples
runavg(orange,63)



data with noise



runavg filter over noisy data



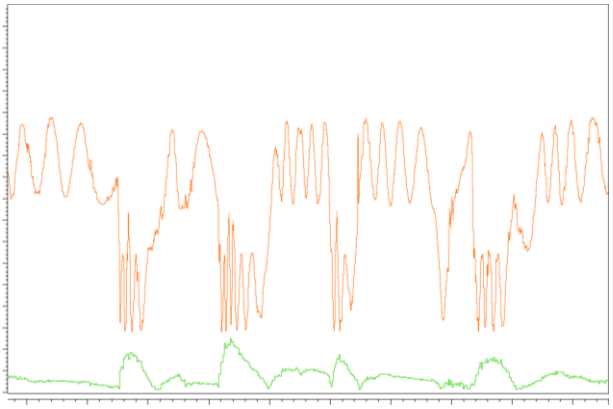
runavg filter over original

runmin(x,max_samples,var_num_samples)
Running minimum of x where the number of samples can vary.

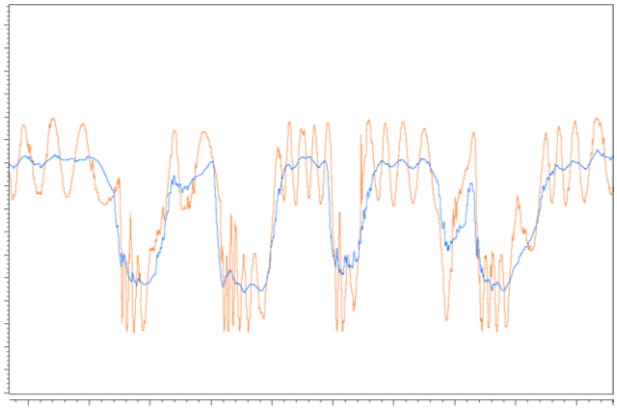
runmax(x,max_samples,var_num_samples)
Running maximum of x where the number of samples can vary.

runavg(x,max_samples,var_num_samples)
Running average of x where the number of samples can vary.

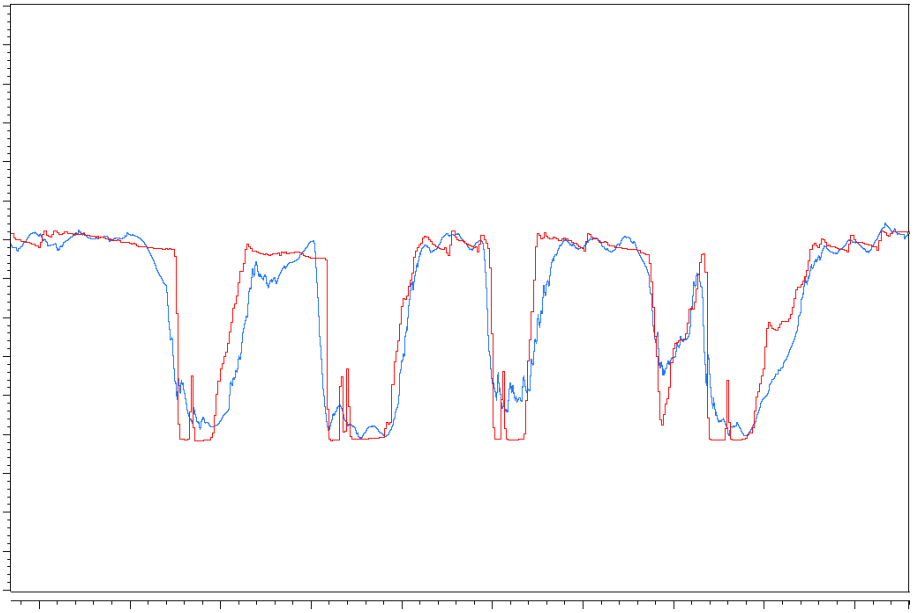
Variable frequency found to be dependent on *green*.
Largest period: delta=2.485s at 100Hz, 250 samples.
Random, average size period: delta=1.298s at 100Hz, 130 samples.
green = 16.95 at midpoint.
130x16.95=2203.5
filter(runavg(orange,250,2200/green),0.95)



data with noise and frequency dependant



runavg filter over noisy data



runavg filter over original

3 Document Revision History

2015-08-12 – MH V1.0 – Initial public release