[6.6] Linear regression through a fixed point

It can be useful or necessary to force a regression equation through a fixed point. For example, you might be modelling a system, and you know from the physical laws that the function must pass through (0,0). The 89/92+ built-in regression functions do not address this requirement. This function will find the fit coefficients for y = bx + a through a fixed point *(h,k).*

```
linreghk(xl,yl,h,k)
func
©return {b,a} for y=b*x+a through (h,k)
©14mar00/dburkett@infinet.com
local s1,s2,s3

sum(xl^2)→s3

if h≠0 then
sum(xl)→s2
(h*k*s2-k*s3-h^2*sum(yl)+h*sum(xl*yl))/(2*h*s2-s3-h^2*dim(xl))→s1
return {(k-s1)/h,s1}

else
return {sum(xl*yl)/s3,0}

endif

Endfunc
```

The code is a straightforward implementation of the regression equations from the book *Curve Fitting for Programmable Calculators*, William W. Kolb, Syntek, Inc., which is unfortunately out of print.

xl and *yl* are lists that specify the (x,y) data points to fit. *h* and *k* specify the point (h,k) through which to force the best-fit line. *linreghk()* returns a list that contains the coefficients {b,a}, where y = bx + a. Note that this list can be used directly with *polyeval()* to evalute the function at some point:

```
polyeval({b,a},x)
```

To force the fit curve through a specified a-value, use

```
linreghk(xl,yl,h,0)
```

To force the fit curve through the origin (0,0), use

```
linreghk(xl,yl,0,0)
```

Two different methods are needed, depending on whether or not h = 0. This routine does no error checking, but these criteria must be met:

1. The lists *xl* and *yl* must be the same length.
2. If h=0, then *xl* and *yl* must have at least two elements
3. If h is not equal to zero, then *xl* and *yl* must have at least three elements.
4. The function should be executed in Approx mode.

This data can be used to test the routine for a fit through the origin (0,0):

xl = {11,17,23,29}
yl = {15,23,31,39}

which returns {1.3483146067416,0}

This data can be used to test the routine for a fit through an arbitrary point:

```
xl = {100,200,300,400,500}
yl = {140,230,310,400,480}
h = 300
k = 310
```

which returns {0.85, 55}

This function can be used to find the unadjusted correlation coefficient for the curve fits:

```
corrhk(f1,xl,yl)
func
©(f1,xlist,ylist)return r^2 for y=f1(x)
©15mar00/dburkett@infinet.com

1-sum((seq(f1|x=xl[k],k,1,dim(xl))-yl)^2)/sum((seq(yl[k],k,1,dim(xl))-mean(yl))^
2)

Endfunc
```

For example, if the correlation equation coefficients are {b,a} as returned by *linreghk(),* use

```
corrhk(polyeval({b,a},x),xlist,ylist)
```

where *xlist* and *ylist* are the lists of data point coordinates.