A5

Daehun Kang(213023759) and Ghislain Rutayisire(218416842)

> **We wanted to show how maple can simplify solving complex problems that can require many tedious steps and techniques from different areas of mathematics. We created a procedure that uses summations, matrix operations and linear algebra for statistical purposes. With concepts of regression analysis for polynomials, our procedure takes as inputs two data lists and an x-value X. Computes the sums needed to create a system of 3 linear equations which the uknowns yeild the coefficients of a least squares polynomial of degree 2. It then uses Gaussian elimination and backward substitution to find the unknowns and the resulting polynomial. It evaluates the quadratic model at the given x-value and outputs the result and a graph of the data points, including the point with the given x-value on the curve. This procedure can be used to predict any y-value corresponding to a given x-value for sets of data points that are appropriate for 2nd degree polynomial least squares approximation.**

```
> with(Student[LinearAlgebra]): #to bring the LinearAlgebra package
  to solve linear algebra problems.

  lsp:=proc(vecx,vecy,X); #lsp procedure takes in two data lists
  and an x-value X, creates a least suqares polynomial of degree 2
  with the given data, evaluates the least suqares polynomial at
  the given x-value and outputs the result and a graph of the data
  points on the polynomial.

  local sumx,sumx2,sumx3,sumx4,sumy,sumxy,sumx2y,a,b,c,w,x;

  if numelems(vecx)<>numelems(vecy) then print("Data lists must be
  of same length.")else #since data lists(vecx and vecy) must be of
  same length, we need to use an if statement to check it.

  #to create elements of 3 linear equations.
  sumx:=sum(vecx[n],n=1..numelems(vecx)); #sum of elements of
  vector x.
  sumx2:=sum(vecx[n]^2,n=1..numelems(vecx)); #sum of squared
  elements of vector x.
  sumx3:=sum(vecx[n]^3,n=1..numelems(vecx));
  sumx4:=sum(vecx[n]^4,n=1..numelems(vecx));
  sumy:=sum(vecy[n],n=1..numelems(vecy)); #sum of elements of
  vector y.
  sumxy:=sum(vecx[n]*vecy[n],n=1..numelems(vecx));  #sum of
  elements of vector x multiplied by coressponding elements of
```

```
vector y.
sumx2y:=sum((vecx[n]^2)*vecy[n],n=1..numelems(vecx));

a:=Matrix([[numelems(vecx),sumx,sumx2,sumy],[sumx,sumx2,sumx3,
sumxy],[sumx2,sumx3,sumx4,sumx2y]]); #putting 3 linear equations
in the form of matrix.

b:=ReducedRowEchelonForm(a); #This will reduce the matrix 'a' to
row echelon form.

c:=evalf(BackwardSubstitute(b)); #This will output the
coefficients of the regression quadratic - c(3),c(2),c(1).

w:=plots:-pointplot([seq([x,c(3)*x^2+c(2)*x+c(1)],x=min(vecx)..X)
]); #This will generate point-style plot of the regression
quadratic 'y=c(3)*x^2+c(2)*x+c(1)' when x=vec(x) to X.

return c(3)*X^2+c(2)*X+c(1),w; #The procedure will return the
value of y when x=X and the plot.

end if;
end proc;
```

$lsp := \mathbf{proc}(vecx, vecy, X)$ **(1)**

local $sumx, sumx2, sumx3, sumx4, sumy, sumxy, sumx2y, a, b, c, w, x$;

if $numelems(vecx) <> numelems(vecy)$ **then**

$print($"Data lists must be of same length."$)$

else

$sumx := sum(vecx[n], n = 1 ..numelems(vecx));$

$sumx2 := sum(vecx[n]\text{^}2, n = 1 ..numelems(vecx));$

$sumx3 := sum(vecx[n]\text{^}3, n = 1 ..numelems(vecx));$

$sumx4 := sum(vecx[n]\text{^}4, n = 1 ..numelems(vecx));$

$sumy := sum(vecy[n], n = 1 ..numelems(vecy));$

$sumxy := sum(vecx[n] * vecy[n], n = 1 ..numelems(vecx));$

$sumx2y := sum(vecx[n]\text{^}2 * vecy[n], n = 1 ..numelems(vecx));$

$a := Matrix([[numelems(vecx), sumx, sumx2, sumy], [sumx, sumx2, sumx3, sumxy], [sumx2, sumx3, sumx4, sumx2y]]);$

$b := Student\text{:-}LinearAlgebra\text{:-}ReducedRowEchelonForm(a);$

$c := evalf(Student\text{:-}LinearAlgebra\text{:-}BackwardSubstitute(b));$

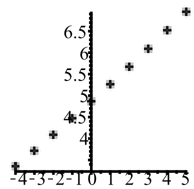
$w := plots\text{:-}pointplot([seq([x, c(3) * x\text{^}2 + c(2) * x + c(1)], x = \min(vecx) ..X)]);$

return $c(3) * X\text{^}2 + c(2) * X + c(1), w$

end if

end proc

```
> lsp([-4,-3,-2,-1,0,1,2,3],[3.36,3.72,4.10,4.46,4.86,5.28,5.69,
6.08],5);
```

6.954940477, 


```
> lsp([-4,-3,-2,-1,0,1,2],[3.36,3.72,4.10,4.46,4.86,5.28,5.69,
  6.08],5);
```

"Data lists must be of same length." **(2)**

To verify if our result is correct we will check with the built in LeastSquares command.

```
> with(CurveFitting):
> LeastSquares([-4,-3,-2,-1,0,1,2,3],[3.36,3.72,4.10,4.46,4.86,
  5.28,5.69,6.08],x,curve=a*x^2+b*x+c);
```

$$4.86684523809524 + 0.395297619047619\, x + 0.00446428571428567\, x^2 \quad \textbf{(3)}$$

```
> eval(%,x=5);
```

$$6.95494047619048 \quad \textbf{(4)}$$

This confirms our result.