```
> restart:
  with(LinearAlgebra):
```

```
> CREATEVANMAT := proc(paramA::posint)

  local n,xVals,i,j,vanMat,valIn:
  n := paramA:

  xVals := [seq(x[i],i=1..n)]:
  vanMat := Matrix(n,n):

  for i from 1 to n do:
      local k:
      k := 1:
      valIn := xVals[i]:
      for j from 1 to n do:
          if j=1 then:
              vanMat[i,j] := 1:
          else:
              vanMat[i,j] := valIn^k:
              k := k+1:
          fi:
      od:
  od:

  return vanMat:
  end proc:
```

```
> n := 4:
  vanMat := CREATEVANMAT(n):
  vanMat;
```

$$\begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ 1 & x_4 & x_4^2 & x_4^3 \end{bmatrix} \qquad (1)$$

```
> alphaB := [seq(i,i=1..n)]:
  alphaB;
```

$$[1, 2, 3, 4] \tag{2}$$

```
> BBVAN := proc(paramA::list(integer))

  local alpha,p,mat,det,res,i,temp:

  alpha := paramA:

  temp := vanMat:
  temp := subs([seq(x[i]=alpha[i],i=1..n)],temp):
  res := Determinant(temp):

  return res:
  end proc:
```

```
> BZ := proc(paramA::integer)

  local alpha,res,i:

  alpha := paramA:
  res := BBVAN([alpha,seq(alphaB[i],i=2..n)]):

  return res:
  end proc:
```

```
> GETDEGREE := proc(paramA::procedure)

  local BB,alpha,gK,k,m,yK,vK,mEval:

  BB := paramA:
  gK := 0:
  k := 0:
  m := 1:
  while true do:
      alpha := rand():
      mEval := eval(m,x=alpha):
      while mEval=0 do:
          alpha := rand():
          mEval := eval(m,x=alpha):
      od:
      yK := BB(alpha):
      vK := ((yK-eval(gK,x=alpha))/mEval):
      if vK=0 then:
          return gK:
      fi:
```

```
      gK := gK+(vK*m):
      m := expand(m*(x-alpha)):
       k := k+1:
  od:
  end proc:
```

```
> res := GETDEGREE(BZ):
  res;
```

$$-2 x^3 + 18 x^2 - 52 x + 48 \tag{3}$$

```
> res := eval(diff(res,x),x=alphaB[1]):
  res;
```

$$-22 \tag{4}$$

```
> mapRes := eval(diff(Determinant(vanMat),x[1]),[seq(x[i]=alphaB
  [i],i=1..n)]):
  mapRes;
```

$$-22 \tag{5}$$

```
> sumEval := expand(sum(8*k-2,k=1..d+2));
```

$$sumEval := 4 d^2 + 18 d + 20 \tag{6}$$