Candidate Number: 21777

Q1)

```
> f := proc(x, y)
    333.75·y^6 + x^2·(11·x^2·y^2 − y^6 − 121·y^4 − 2) + 5.5·y^8 + x/(2·y);
  end proc:
>
> for i from 5 to 30 do
    Digits := i:
    fEstimate := f(77617, 33096) :
    print(fEstimate)
  end do:
```

$$1.\ 10^{32}$$
$$-3.\ 10^{31}$$
$$1.172604$$
$$1.1726039$$
$$1.17260394$$
$$1.172603940$$
$$-1.\ 10^{26}$$
$$1.17260394005$$
$$3.\ 10^{24}$$
$$-1.\ 10^{23}$$
$$1.\ 10^{22}$$
$$-2.\ 10^{21}$$
$$-2.\ 10^{20}$$
$$1.17260394005317863$$
$$-9.99999999999999988\ 10^{17}$$
$$1.00000000000000000117\ 10^{17}$$
$$1.1726039400531786318 6$$
$$-9.99999999999999988273961\ 10^{14}$$
$$-9.999999999999998827396060\ 10^{13}$$
$$2.0000000000000117260394 01\ 10^{13}$$
$$-2.999999999999988273960599 47\ 10^{12}$$
$$1.172603940053178631858834 9$$
$$1.1726039400531786318588349 0$$
$$-9.999999988273960599468213681\ 10^{8}$$
$$1.000000011726039400531786318 6\ 10^{8}$$
$$1.0000000117260394005317863185 88\ 10^{7} \quad (1)$$

This shows that the number of significant figures used greatly affects the final answer given for f (77617,33096) because the digits function changes the number of significant figures used for all the variables within the function. In a function with such large numbers being generated (33096^8) if the number of significant figures used is too low then these numbers will not be accurate and so any further calculations with these numbers will also lose their accuracy. The value given in C using double precision is 1.172604 which matches the result for Digits equal to 7 and the value using single precision is -4445074789835319090322027315 2.000000.

> $Digits := 50$: *#set Digits equal to 50 to find the exact value of f(77617,33096)*

> $f(77617, 33096)$

$$-0.827396059946821368141165095479816291999033115784 \quad (2)$$

This exact value does not match any of the values generated by the loop with significant figures between 5 and 30. It takes 37 significant figures to achieve an accurate value of f(77617,33096). This matches the number of significant figures needed to perfectly calculate the value of 33096^8.

> **for** i **from** 35 **to** 39 **do**
$Digits := i$:
$fEstimate := f(77617, 33096)$:
$print(fEstimate)$
end do:

$$-298.82739605994682136814116509547982$$
$$21.172603940053178631858834904520183 7$$
$$-0.8273960599468213681411650954798162 92$$
$$-0.82739605994682136814116509547981629 20$$
$$-0.827396059946821368141165095479816292 00 \quad (3)$$

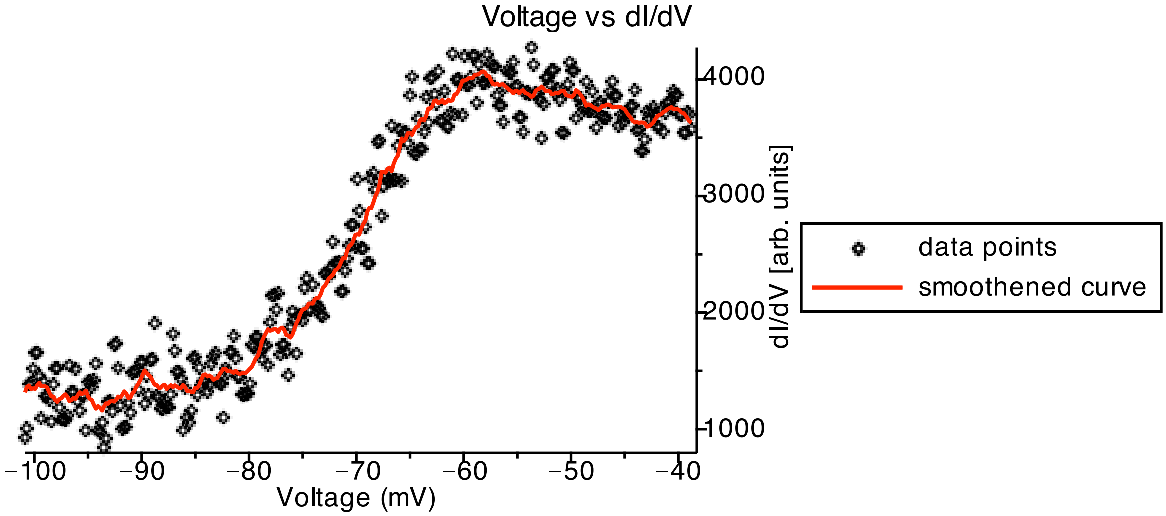
>

Q2)

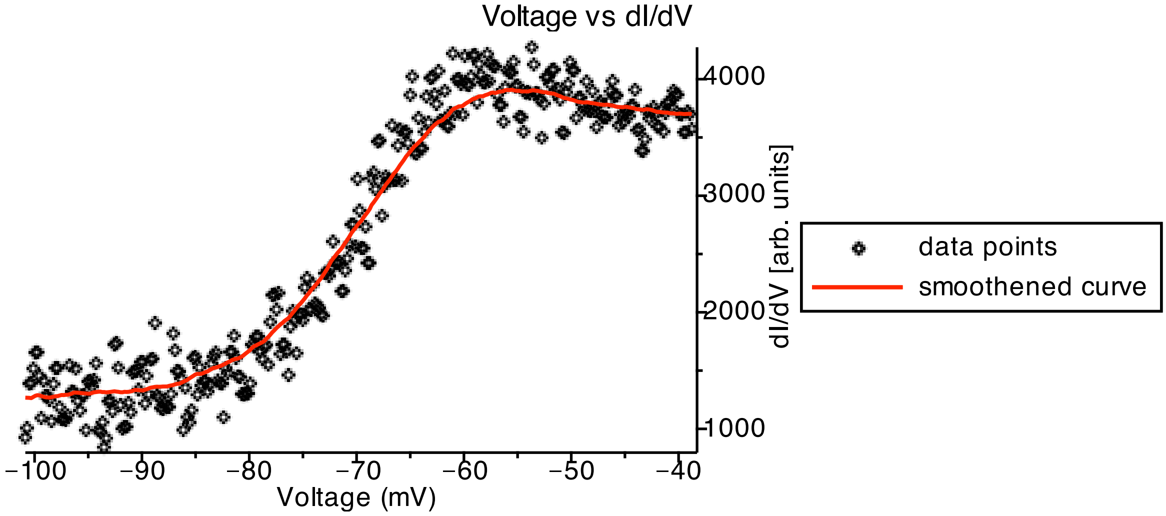
```
> restart : with(plots) :
> dataSmooth := proc(FileName :: string, n :: integer, smoothened :: boolean)

        local A, length, D1, D2, L, i; #declare the local variables required
         global B;
       #assigns the smoothened or unsmoothened values so that they are available to the user

        A := readdata(FileName, [float, float]); #read in the data from the file
        L := nops(A); #find the number of pieces of data
        B := readdata(FileName, [float, float]); #read in the data from the file
        B := convert(B, Array); #convert from a list to an array so values can be edited

        if smoothened = true then

            #smoothens the beginning of the data series from 1->n
            for i from 1 to n do
                B[i, 2] := (sum(A[j][2], j = 1 .. i + n)) / (i + n)
            end do;

            #smoothens main section of the data series from (n+1)->(L-n)
            for i from (n + 1) to (L - n) do
                B[i, 2] := (1 / (2 * n + 1)) * (sum(A[j][2], j = i-n .. i + n))
                end do;

            #smoothens the end of the data series from (L-(n-1))->L
            for i from L - (n-1) to L do
                B[i, 2] := (sum(A[j][2], j = i-n .. L)) / ((2 * n + 1) + ((L-n)-i))
                end do;

            #plots the original data series as points
            D1 := listplot(A, style = point, color = black, labels = ["Voltage (mV)",
        "dI/dV [arb. units]"], labeldirections = [horizontal, vertical], labelfont = ["Helvetica",
        8], title = "Voltage vs dI/dV", titlefont = ["Helvetica", 9], legend = "data points",
        legendstyle = [font = ["Helvetica", 8], location = right], size = [550, 200], axesfont
       = ["Helvetica", 8]);
            #plots the smoothened data series as a red line if smoothening has been selected
            D2 := listplot(B, style = line, color = red, legend = "smoothened curve");
            display(D1, D2);

        else

            #plots the original data series as points if smoothening has not been selected
             listplot(A, style = point, color = black, labels = ["Voltage (mV)",
        "dI/dV [arb. units]"], labeldirections = [horizontal, vertical], labelfont = ["Helvetica",
        8], title = "Voltage vs dI/dV", titlefont = ["Helvetica", 9], legend = "data points",
        legendstyle = [font = ["Helvetica", 8], location = right], size = [400, 200], axesfont
       = ["Helvetica", 8]);

        end if
    end proc:
>
```

The three tests show two examples with smoothening, one with a lower level (n smaller) than the other, and one where smoothening has been turned off.

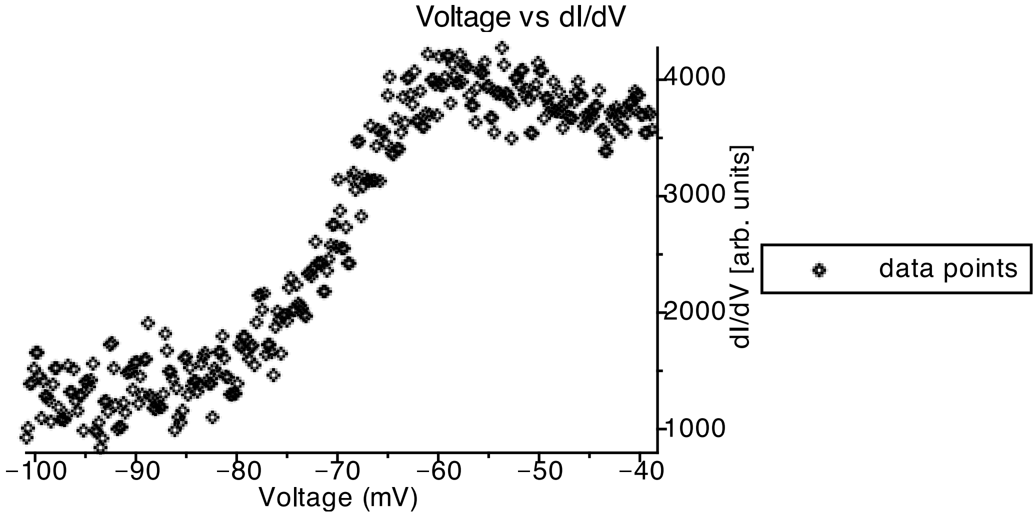
> *dataSmooth*("stm2.dat", 5, *true*)



> *dataSmooth*("stm2.dat", 40, *true*)



> *dataSmooth*("stm2.dat", 0, *false*)



>

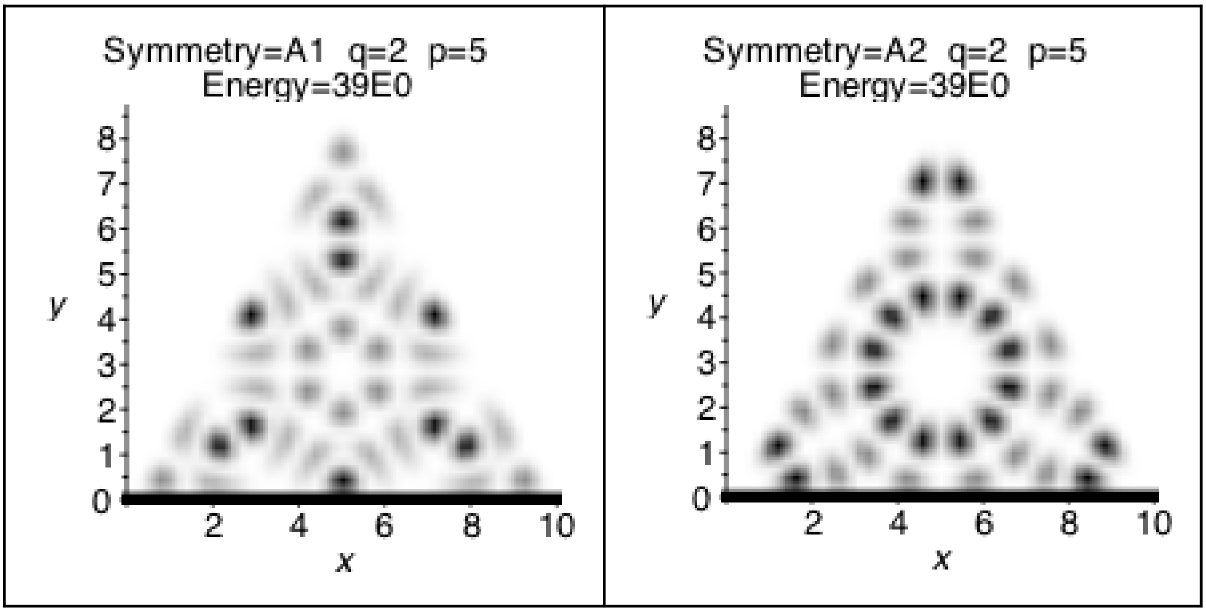
Q3)

```
> restart : with(plots) :
> triangle_psi := proc(sym :: string, q, p)

      #declare the local variables required
      local a, A, u, v, w, x, y, Cq, C, D, E, S1, S2, eps, epsConj, psi, psi2, psiSquared,
     psiSquaredTri, intCheck;

       a := 10; #arbitrary length of the side of the triangle
      A := sqrt(3) * a/2; #height of the triangle
      u := (2 * Pi/A) * y; #boundary conditions
      v := (2 * Pi/A) * (sqrt(3) * x/2 - y/2);
      w := 2 * Pi - u - v;
      E := (p^2 + p * q + q^2); #Energy/E0

      if sym = "A1" then

          intCheck := type([q, p], ['integer','integer']); #make sure q & p are integers

          if intCheck = true and q ≥ 0 and p > q then
     #make sure q & p are suitable for A1 symmetry

              if q = 0 then #assign the correct value of Cq dependant on q
                  Cq := 1/(sqrt(2 * sqrt(3)));
              else
                  Cq := 1/(sqrt(sqrt(3)));
              end if;

              #normalised wave function for A1 symmetry
              psi(x, y) := Cq * (sin(p * u-q * v) + sin(p * v-q * w) + sin(p * w-q * u)
      + sin(p * v-q * u) + sin(p * w-q * v) + sin(p * u-q * w));
          else
              ERROR("p and q are not valid for A1 symmetry");
          end if;

      elif sym = "A2" then

          intCheck := type([q, p], ['integer','integer']); #make sure q & p are integers

          if intCheck = true and q ≥ 1 and p > q then
     #make sure q & p are suitable for A2 symmetry

              C := (1/3)^(1/4); #assign the value of C

              #normalised wave function for A2 symmetry
              psi(x, y) := C * (cos(p * u-q * v) + cos(p * v-q * w) + cos(p * w-q * u)
     -cos(p * v-q * u)-cos(p * w-q * v)-cos(p * u-q * w));
          else
              ERROR("p and q are not valid for A2 symmetry");
          end if;

      elif sym = "E" then
```

```
        #make sure q and p are either both an integer+1/3 or both an integer+2/3
        S1 := type([q + 2/3, p + 2/3], ['integer','integer']);
        S2 := type([q + 1/3, p + 1/3], ['integer','integer']);

        #make sure q & p are suitable for either of the E symmetries
        if (S1 = true or S2 = true) and q > 0 and p > q then

            D := (1/12)^(1/4); #assign the value of D

            #assign the correct value of epsilon dependent on which E symmetry
            if S1 = true then
                eps := exp(2*Pi*I/3);
            else
                eps := exp(-2*Pi*I/3);
            end if;

            epsConj := conjugate(eps); #find the complex conjugate of epsilon

            #normalised wave function for E symmetry
            psi(x, y) := D*(exp((p*u-q*v)*I) + eps*exp((p*v-q*w)*I)
    + epsConj*exp((p*w-q*u)*I)-exp(-1*(p*v-q*u)*I)-eps*exp(-1*(p*w
    -q*v)*I)-epsConj*exp(-1*(p*u-q*w)*I));

             #the second E symmetry soltuion can found by taking the
             #complex conjugate of psi however as it is (mod(psi))^2
             #that is plotted the same graphical solution will be
             #found for both and hence is not plotted
            psi2(x, y) := conjugate(psi(x, y));
        else
            ERROR("p and q are not valid for E symmetry");
        end if;

    else
        ERROR("Symmetry must be A1, A2 or E");
    end if;

    psiSquared(x, y) := (abs(psi(x, y)))^2;

    psiSquaredTri(x, y) := psiSquared(x, y)*Heaviside(u)*Heaviside(v)
    *Heaviside(w);

    #plots the visualisation of (mod(psi))^2 in an equilateral triangle
    densityplot(psiSquaredTri(x, y), x = 0..a, y = 0..A, title = cat("Symmetry=", sym,
    "  q=", q, "  p=", p, "  Energy=", E, "E0"), titlefont = ["Helvetica", 10], style
    = patchnogrid, colorscheme = ["white", "black"], size = [350, 350], labelfont
    = ["Helvetica", 10], axesfont = ["Helvetica", 10]);

end proc:
>
```

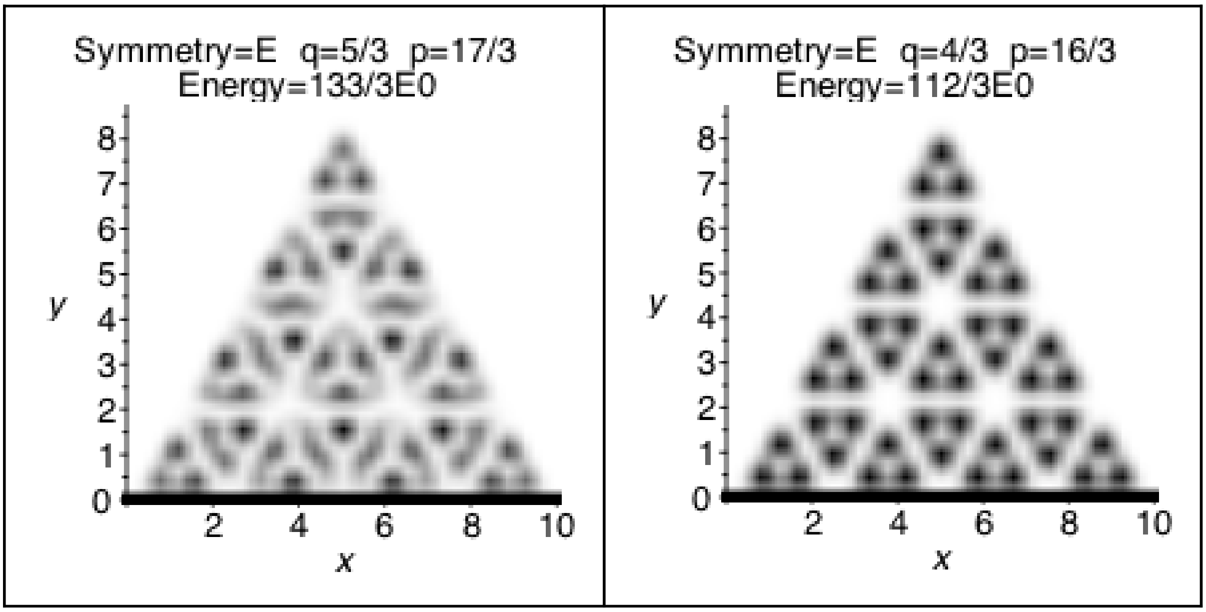
The first two figures show the functionality of the A1 and A2 symmetry and highlights the difference between the two for the same values of p and q.

> $a := triangle\_psi(\text{"A1"}, 2, 5) : b := triangle\_psi(\text{"A2"}, 2, 5) : display(\langle a|b \rangle);$



These two figures show the difference between the two type of E symmetry and includes ("E",5/3, 17/3) state which is the one shown on the front page of the assignment document.

> $c := triangle\_psi\left(\text{"E"}, \frac{5}{3}, \frac{17}{3}\right) : d := triangle\_psi\left(\text{"E"}, \frac{4}{3}, \frac{16}{3}\right) : display(\langle c|d \rangle);$



This test shows the error message displayed if the symmetry entered does not match A1, A2 or E.

> $triangle\_psi(\text{"A"}, 1, 3)$

Error, (in triangle_psi) Symmetry must be A1, A2 or E

These show the error messages displayed if p and q are not valid for the A1 and A2 symmetries. This includes entering negative integers, non-integers, if q>p and also the q=0 and p=1 state which the A1 symmetry should accept and the A2 symmetry should not.

> $triangle\_psi(\text{"A1"}, -1, 5)$

```
Error, (in triangle_psi) p and q are not valid for A1 symmetry
```

> $triangle\_psi(\text{"A1"}, 3, 2)$

```
Error, (in triangle_psi) p and q are not valid for A1 symmetry
```

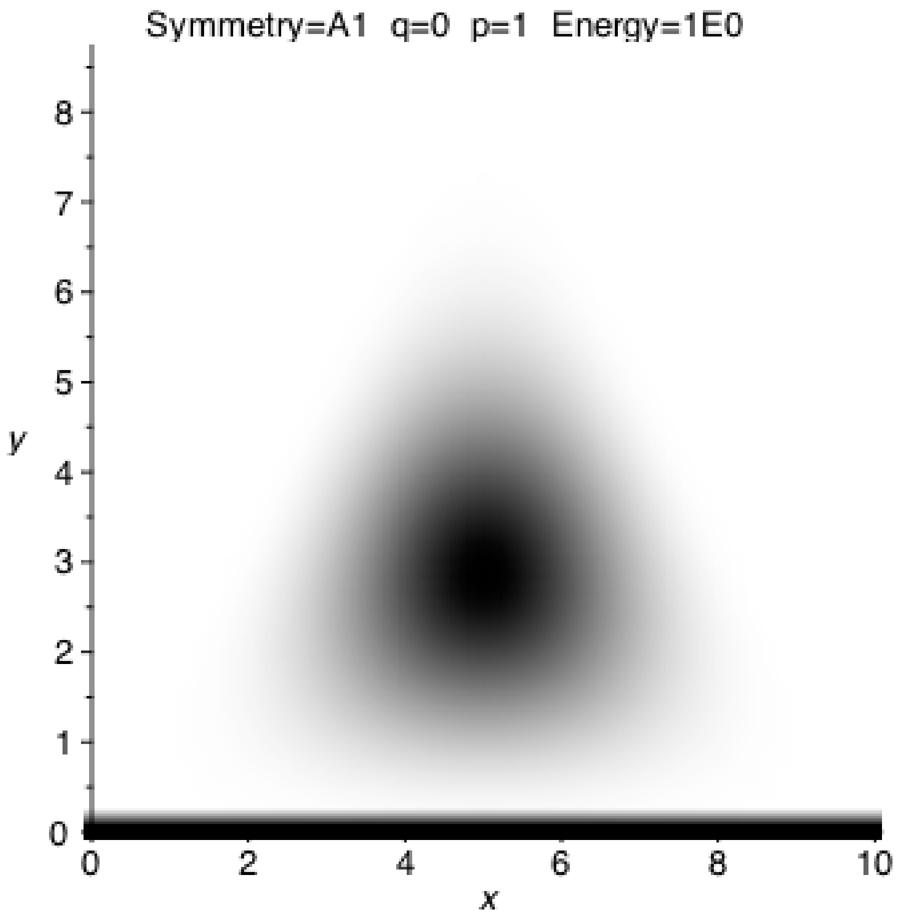
> $triangle\_psi\left(\text{"A2"}, \frac{1}{3}, \frac{4}{3}\right)$

```
Error, (in triangle_psi) p and q are not valid for A2 symmetry
```

> $triangle\_psi(\text{"A2"}, 0, 1)$

```
Error, (in triangle_psi) p and q are not valid for A2 symmetry
```

> $triangle\_psi(\text{"A1"}, 0, 1)$



These show the error messages displayed if p and q are not valid for the E symmetries. This includes entering negative numbers, integers and a combination of the p and q values that each should accept.

> $triangle\_psi\left(\text{"E"}, -\frac{1}{3}, \frac{4}{3}\right)$

```
Error, (in triangle_psi) p and q are not valid for E symmetry
```

> $triangle\_psi(\text{"E"}, 1, 4)$

```
Error, (in triangle_psi) p and q are not valid for E symmetry
```

> $triangle\_psi\left(\text{"E"}, \frac{1}{3}, \frac{5}{3}\right)$

```
Error, (in triangle_psi) p and q are not valid for E symmetry
```