Q1

$y = F(x_0, x_1) = 100(x_1 - x_0^2)^2 + (1 - x_0)^2$

$F_{x_0} = \frac{dF}{dx_0}$, $F_{x_1} = \frac{dF}{dx_1}$

Need $F_{x_0} = F_{x_1} = 0$

$F_{x_0} = 400x_0(x_1 - x_0^2) + 2(1 - x_0) = 0$ (1)

$F_{x_1} = 200(x_1 - x_0^2) = 0$ (2)

Rearranging (2): $x_1 = x_0^2$

Subbing back into (1):

$400x_0(x_0^2 - x_0^2) + 2(1 - x_0) = 0$ (3)

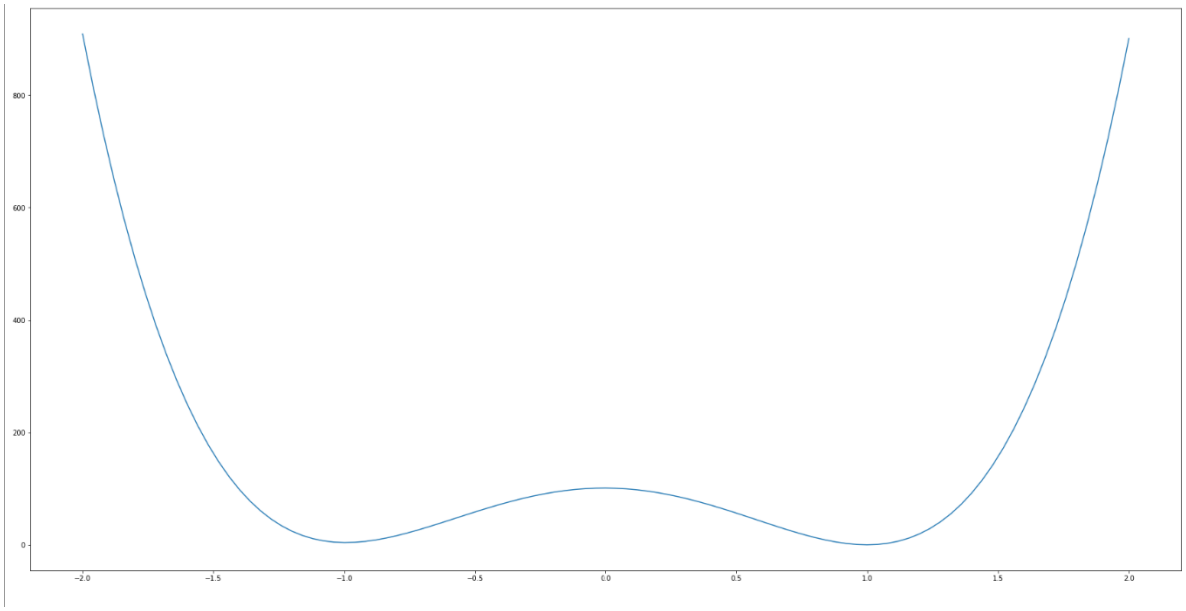
Rearranging (3): $x_0 = 1$ $\therefore x_1 = 1$

$y = F(1,1) = 100(1 - 1^2)^2 + (1 - 1)^2 = 0$

Therefore there is a stationary point of $y = 0$ at $(1, 1)$.

The point must be a minimum as the function is the sum of two squares, so can never fall below 0.

Q2)



This is a plot using python, with the source code given in the appendix. The x axis represents x_0, and the y axis represents the corresponding y values.

1st Iteration

$y = F(x_0, x_1) = 100(x_1 - x_0^2)^2 + (1 - x_0)^2$

> $P_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ $P_1 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$ $P_2 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$ } $P_i \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$

$y_0 = 1$ $y_1 = 1601$ $y_2 = 401$

y_l y_h y_i

> $\bar{P} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

> $P^* = 2\bar{P} - P_h = \begin{pmatrix} -2 \\ 2 \end{pmatrix}$

> $y^* = 409$

> is $y^* < y_l \rightarrow 409 > 1$ ∴ NO

↳ is $y^* > y_i \rightarrow 409 > 401$ ∴ YES

↳ is $y^* > y_h \rightarrow 409 < 1601$ ∴ NO

↳ Replace P_h by $P^* \rightarrow P_h = \begin{pmatrix} -2 \\ 2 \end{pmatrix}$, $y_h = 409$

> $P^{**} = (P_h + \bar{P})/2 = \begin{pmatrix} -1 \\ 3/2 \end{pmatrix}$

> $y^{**} = 29$

> is $y^{**} > y_h \rightarrow 29 < 409$ ∴ NO

↳ Replace P_h by $P^{**} \rightarrow P_h = \begin{pmatrix} -1 \\ 3/2 \end{pmatrix}$, $y_h = 29$

> $\bar{y} = 101$

> error = 229 ∴ min not reached

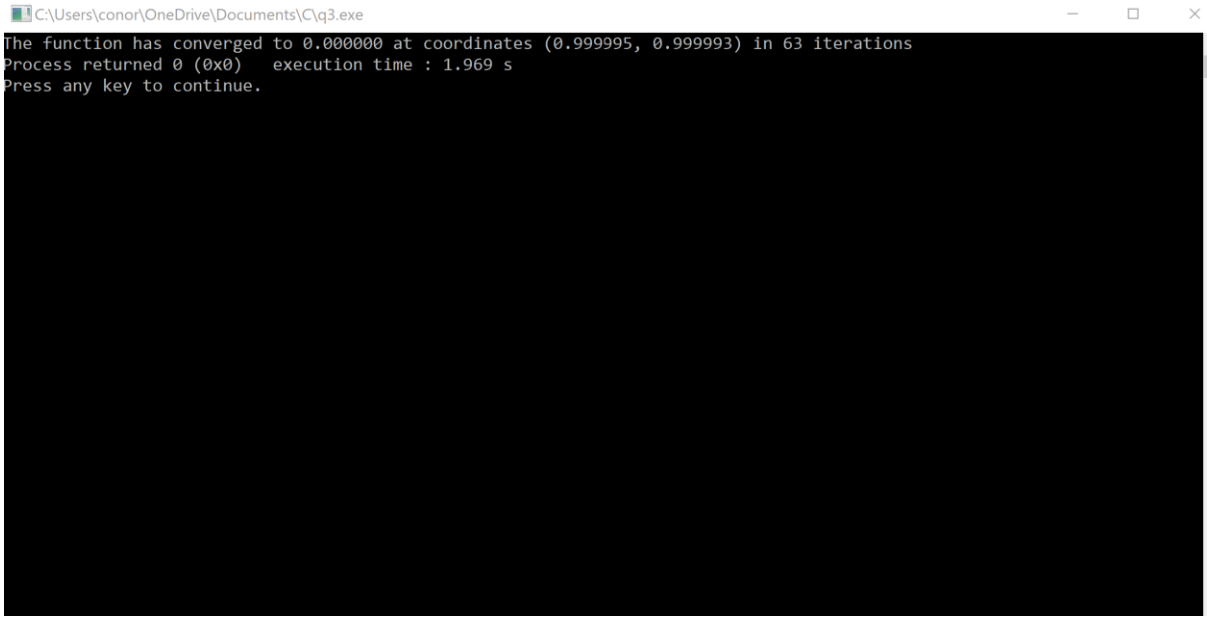
Actions applied to triangle:
reflection + contraction

2nd Iteration

- $P_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ $P_1 = \begin{pmatrix} -1 \\ 3/2 \end{pmatrix}$ $P_2 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$
- $y_0 = 1$ $y_1 = 29$ $y_2 = 401$
  y_l y_i y_h
- $\bar{P} = \begin{pmatrix} -1/2 \\ 3/4 \end{pmatrix}$
- $P^* = \begin{pmatrix} -1 \\ -1/2 \end{pmatrix}$
- $y^* = 229$
- is $y^* < y_l \rightarrow 229 > 1 \therefore$ NO
  - is $y^* > y_i \rightarrow 229 > 29 \therefore$ YES
    - is $y^* > y_h \rightarrow 229 < 401 \therefore$ NO
    - Replace P_h by $P^* \rightarrow P_h = \begin{pmatrix} -1 \\ -1/2 \end{pmatrix}$, $y_h = 229$
- $P^{**} = \begin{pmatrix} -3/4 \\ 1/8 \end{pmatrix}$
- $y^{**} = 1421/64$
- is $y^{**} > y_h \rightarrow 1421/64 < 401 \therefore$ NO
  - Replace P_h by $P^{**} \rightarrow P_h = \begin{pmatrix} -3/4 \\ 1/8 \end{pmatrix}$, $y_h = 1421/64$

- $\bar{y} = 27.25$
- error $= 18.9 \therefore$ min not reached

Actions applied to triangle:
reflection + contraction



This is a screenshot of the print screen when the code is run. This shows that the minimum of 0 was reached after 63 iterations.

The code was written using a series of ‘if’ statements to replicate the conditional behaviour of the flowchart. These were nested within a ‘do while’ loop with the ‘while’ statement used to cap the number of iterations and threshold error value, thus printing the final coordinates and number of iterations. Coordinates were stored and modified in arrays, and all variables (apart from the counter) were stored as doubles.

Appendix:

Q2)

```
#include <stdio.h>
#include <stdlib.h>

double F(double x) // Define F and x
{

  return 100 * pow((1 - pow(x, 2)), 2) + pow((1 - x), 2); // return the function needed
}

void write_file(char *filename, double a, double b, int n)
{

  int i; // define i to be used as a loop counter
  double x; // define x which gives the current position
  FILE *fpointer;
  fpointer = fopen(filename, "w");

i = 0; // set i to begin at 0
    do {
    x = a + i * (b - a) / n; //
    fprintf(fpointer, "%e, %e\n", x , F(x) );
    i++;
}   while (i <= 100);


  fclose(fpointer);
}

int main()
```

```
{

  write_file("data.txt", -2., 2., 100);

  return 0;
}
```

Q3)

```
#include <stdio.h>
#include <stdlib.h>

double y(double x0, double x1){ // y, x0 & x1
    return 100 * pow((x1 - pow(x0, 2)), 2) + pow((1 - x0), 2); // define function for y(x0, x1)
}
double v0[] = {0, 0}; // define coordinate vectors
double v1[] = {2, 0};
double v2[] = {0, 2};
double y_0; // define y values for coordinates v0, v1 & v2
double y_1;
double y_2;
double y_h; // define y values for flowchart
double y_l;
double y_i;
double y_hat;
double y_star;
double y_star2;
double vh[2]; // define vectors for flowchart
double vl[2];
double vi[2];
double vhat[2];
double vstar[2];
double vstar2[2];
double errorcheck; // define error
int i = 0;

int main()
```

```
{
do { // open do while loop
y_0 = y(v0[0], v0[1]); // determine y values for v0, v1 & v2
y_1 = y(v1[0], v1[1]);
y_2 = y(v2[0], v2[1]);

   if (y_0 >= y_1 && y_0 >= y_2){ // if else loop to determine y_h & vh
         y_h = y_0;
         vh[0] = v0[0];
         vh[1] = v0[1];
   } else{
      if (y_1 >= y_0 && y_1 >= y_2){
            y_h = y_1;
            vh[0] = v1[0];
            vh[1] = v1[1];
      } else{
         y_h = y_2;
         vh[0] = v2[0];
         vh[1] = v2[1];
      }
   }

    if (y_0 <= y_1 && y_0 <= y_2){ // if else loop to determine y_l & vl
         y_l = y_0;
         vl[0] = v0[0];
         vl[1] = v0[1];
   } else{
      if (y_1 <= y_0 && y_1 <= y_2){
```

```
            y_l = y_1;
            vl[0] = v1[0];
            vl[1] = v1[1];
        } else{
            y_l = y_2;
            vl[0] = v2[0];
            vl[1] = v2[1];
        }
    }

    if (y_0 != y_h && y_0 != y_l){ // if else loop to determine y_i & vi
            y_i = y_0;
            vi[0] = v0[0];
            vi[1] = v0[1];
    } else{
        if (y_1 != y_h && y_1 != y_l){
            y_i = y_1;
            vi[0] = v1[0];
            vi[1] = v1[1];
        } else{
            y_i = y_2;
            vi[0] = v2[0];
            vi[1] = v2[1];
        }
    }

vhat[0] = (vl[0] + vi[0]) / 2; // calculate v^
vhat[1] = (vl[1] + vi[1]) / 2;
vstar[0] = 2 * vhat[0] - vh[0]; // calculate v*
```

```
vstar[1] = 2 * vhat[1] - vh[1];
y_star = y(vstar[0], vstar[1]); // calculate y*

if (y_star < y_l){ // is y*<yl YES
   vstar2[0] = 2 * vstar[0] - vhat[0];
   vstar2[1] = 2 * vstar[1] - vhat[1];
   y_star2 = y(vstar2[0], vstar2[1]);

   if(y_star2 < y_l){ // is y**<yl YES
      vh[0] = vstar2[0];
      vh[1] = vstar2[1];

   } else{ // is y**<yl NO
      vh[0] = vstar[0];
      vh[1] = vstar[1];
   }
} else { // is y*<yl NO
   if(y_star > y_i){ // is y*>yi YES
      if(y_star <= y_h){ // is y*>yh NO
         vh[0] = vstar[0];
         vh[1] = vstar[1];
         y_h = y(vh[0], vh[0]);
         }
      vstar2[0] = (vh[0] + vhat[0]) / 2;
      vstar2[1] = (vh[1] + vhat[1]) / 2;
      y_star2 = y(vstar2[0], vstar2[1]);
      if(y_star2 > y_h){ // is y**>yh YES
         vh[0] = (vh[0] + vl[0]) / 2;
         vh[1] = (vh[1] + vl[1]) / 2;
```

```
            vl[0] = (vl[0] + vl[0]) / 2;
            vl[1] = (vl[1] + vl[1]) / 2;
            vi[0] = (vi[0] + vl[0]) / 2;
            vi[1] = (vi[1] + vl[1]) / 2;
        } else{ // is y**>yh NO
            vh[0] = vstar2[0];
            vh[1] = vstar2[1];
        }
 } else { // is y*>yi NO
     vh[0] = vstar[0];
     vh[1] = vstar[1];
    }
}
y_h = y(vh[0], vh[1]); // determine y values for errorcheck
y_l = y(vl[0], vl[1]);
y_i = y(vi[0], vi[1]);
y_hat = y(vhat[0], vhat[1]);
v0[0] = vh[0]; // define vectors for next iteration
v0[1] = vh[1];
v1[0] = vi[0];
v1[1] = vi[1];
v2[0] = vl[0];
v2[1] = vl[1];

errorcheck = sqrt((pow((y_h - y_hat), 2) + pow((y_i - y_hat), 2) + pow((y_l - y_hat), 2)) / 2); //
define function for errorcheck

i ++;
}
while(i < 1000 && errorcheck >= pow(10, -8)); // conditions for minimum reached
```

```
printf("The function has converged to %f at coordinates (%f, %f) in %i iterations", y_h, vh[0], vh[1], i); // print final coordinates


    return 0;

}
```