1-The minimum of the Rosenbrock valley

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

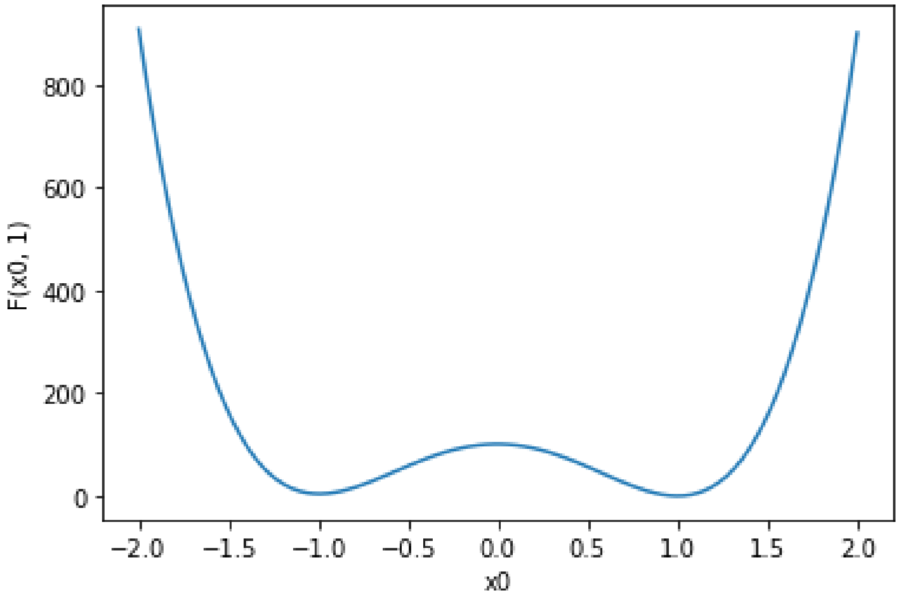
y is composed of two squares and therefore $y \geq 0$

Assuming the minimum occurs at $y = 0$ => $x_1 - x_0^2 = 0$ and $1 - x_0 = 0$

By substituting and solving for x_1 and x_0 we find that the minimum occurs at $F(1,1)$

$x_0 = 1$ => $x_1 - (1)^2 = 0$ => $x_1 = 1$

2-Rosenbrock's parabolic valley numerically



3-Downhill simplex

1st iteration:

$P_0 = \binom{0}{0}$, $P_1 = \binom{2}{0}$, $P_2 = \binom{0}{2}$

$y_0 = 1$, $y_1 = 1601$, $y_2 = 401$ => $y_0 = y_l$, $y_1 = y_h$, $y_2 = y_i$ and $P_0 = P_l$, $P_1 = P_h$, $P_2 = P_i$

$\overline{P} = \frac{P_i + P_l}{2} = \binom{0}{1}$

$P^* = 2\overline{P} - P_h = \binom{-2}{2}$ => $y^* = 409$

$y^* > y_l$ => $y^* > y_i$ => $y^* < y_h$

$P_h = P^* = \binom{-2}{2}$ => $P_0 = \binom{0}{0}$, $P_1 = \binom{-2}{2}$, $P_2 = \binom{0}{2}$

$P^{**} = \frac{(P_h + \overline{P})}{2} = \binom{-1}{1.5}$ => $y^{**} = 29$

$y^{**} < y_h$

$P_h = P^{**} = \binom{-1}{1.5} = P_1$

$n = 2, \ \sqrt{\sum_i \frac{(y_i - \overline{y})^2}{n}} \approx 832.7$

$832.7 > 10^{-8}$

2nd iteration:

$P_0 = \binom{0}{0}, \ P_1 = \binom{-1}{1.5}, \ P_2 = \binom{0}{2}$

$y_0 = 1, \ y_1 = 29, \ y_2 = 401$ => $y_0 = y_l, \ y_1 = y_i, \ y_2 = y_h$ and $P_0 = P_l, \ P_1 = P_i, \ P_2 = P_h$

$\overline{P} = \frac{P_i + P_l}{2} = \binom{-0.5}{0.75}$

$P^* = 2\overline{P} - P_h = \binom{-1}{-0.5}$ => $y^* = 229$

$y^* > y_l$ => $y^* > y_i$ => $y^* < y_h$

$P_h = P^* = \binom{-1}{-0.5}$ => $P_0 = \binom{0}{0}, \ P_1 = \binom{-1}{1.5}, \ P_2 = \binom{-1}{-0.5}$

$P^{**} = \frac{(P_h + \overline{P})}{2} = \binom{-0.75}{0.125}$ => $y^{**} = \frac{1421}{64} \approx 22.2$

$y^{**} < y_h$

$P_h = P^{**} = \binom{-0.75}{0.125} = P_2$ => $P_0 = \binom{0}{0}, \ P_1 = \binom{-1}{1.5}, \ P_2 = \binom{-0.75}{0.125}$

$n = 2, \ \sqrt{\sum_i \frac{(y_i - \overline{y})^2}{n}} \approx 223.3$

$223.3 > 10^{-8}$

Coordinates at the end of the 2nd iteration:

$P_0 = \binom{0}{0}, \ P_1 = \binom{-1}{1.5}, \ P_2 = \binom{-0.75}{0.125}$

$y_0 = 1, \ y_1 = 29, \ y_2 \approx 22.2$

Comparing to the code:

```
Final vertices:
(0.000000, 0.000000)
(-1.000000, 1.500000)
(-0.750000, 0.125000)

Evaluations:
1.000000
29.000000
```

22.203125

Iterations: 2

These coordinates match with those calculated previously.

The full code returns:

Final vertices:

(1.000108, 1.000217)

(0.999851, 0.999698)

(1.000017, 1.000041)

Evaluations:

0.000000

0.000000

0.000000

Iterations: 57

Appendix:

2-Rosenbrock's parabolic valley numerically

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

typedef struct {    //struct containing position inputs
    double x0;
    double x1;
} Point;

double rosenbrock(Point vertex) {   //rosenbock parabolic valley function
    double x0 = vertex.x0, x1 = vertex.x1;
    return 100 * pow(x1 - pow(x0, 2), 2) + pow(1 - x0, 2);
}

void file_data(char *textfile, double xlow, double xhigh, int values) { //writing data into
textfile

    double x0 = xlow;
    double step = (xhigh - xlow) / (values - 1);
```

```c
    FILE *fp = fopen(textfile, "w");

    for (int i = 0; i < 100; i++, x0 += step) {
        Point P = {x0, 1};
        fprintf(fp, "%.3lf, %.3lf\n", x0, rosenbrock(P));
    }

    fclose(fp);
}

int main() {

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

    return 0;
}
```

Python:

```python
import matplotlib.pyplot as plt
import numpy as np
data = np.loadtxt("data.txt", delimiter = ', ')
x = np.array(data[:,0])
F = np.array(data[:,1])
plt.plot(x, F)
plt.xlabel("x0")
plt.ylabel("F(x0, 1)")
plt.draw()
plt.show()
```

3-Downhill simplex

```c
#include <stdio.h>
#include <math.h>

typedef struct {    //struct containing position inputs
    double x0;
    double x1;
} Point;

double rosenbrock(Point vertex) {   //rosenbock parabolic valley function
    double x0 = vertex.x0, x1 = vertex.x1;
    return 100 * pow(x1 - pow(x0, 2), 2) + pow(1 - x0, 2);
}

void sortPoints(double Y[], Point P[]) {    //reordering arrays from small to big
    double temp;
    Point pTemp;
    for (int i = 0; i < 2; i++) {
        for (int j = i + 1; j < 3; j++) {
            if (Y[i] > Y[j]) {
                temp = Y[i];
```

```c
                Y[i] = Y[j];    //rearranging evaluations
                Y[j] = temp;

                pTemp = P[i];
                P[i] = P[j];    //rearranging corresponding positions
                P[j] = pTemp;
            }
        }
    }
}

void getCentroid(Point P[], Point *Pbar) {  //centroid
    Pbar->x0 = (P[0].x0 + P[1].x0) / 2;
    Pbar->x1 = (P[0].x1 + P[1].x1) / 2;
}

void reflect(Point *Ps, Point *Pbar, Point P[]) {   //P*
    Ps->x0 = 2 * Pbar->x0 - P[2].x0;
    Ps->x1 = 2 * Pbar->x1 - P[2].x1;
}

void expand(Point *Pss, Point *Ps, Point *Pbar) {   //P** expansion
    Pss->x0 = 2 * Ps->x0 - Pbar->x0;
    Pss->x1 = 2 * Ps->x1 - Pbar->x1;
}

void contract(Point *Pss, Point P[], Point *Pbar) { //P** contraction
    Pss->x0 = (P[2].x0 + Pbar->x0) / 2;
    Pss->x1 = (P[2].x1 + Pbar->x1) / 2;
}

void replacePoint(Point *new, Point *orig) {    //replace Ph with P* or P**
    new->x0 = orig->x0;
    new->x1 = orig->x1;
}

void getPi(Point P[]) { //replace Pi with (Pi+Pl)/2
    for (int i = 0; i < 3; i++) {
        P[i].x0 = (P[i].x0 + P[0].x0) / 2;
        P[i].x1 = (P[i].x1 + P[0].x1) / 2;
    }
}

_Bool MinCon(double Y[]) {  //condition for breaking loop
    double Ybar = 0, sum = 0;
    for (int i = 0; i < 3; i++) {
        Ybar += Y[i] / 3;
    }
    for (int i = 0; i < 3; i++) {
        sum += pow(Y[i] - Ybar, 2) / 2;
    }
    return (sqrt(sum) < pow(10, -8));
}
```

```
void algorithm(Point P[]) { //loop containing algorithm
    int a;
    for (a = 0; a < 1000; a++) {    //stops after 1000 iterations

        double Y[3];
        for (int i = 0; i < 3; i++) {
            Y[i] = rosenbrock(P[i]);    //positions evaluated
        }

        sortPoints(Y, P);   //sort values and positions small to big

        Point Pbar, Ps, Pss;

        getCentroid(P, &Pbar);  //centroid calculation
        reflect(&Ps, &Pbar, P); //P* calculation

        double Ys = rosenbrock(Ps); //y* evaluation

        if (Ys < Y[0]) {
            expand(&Pss, &Ps, &Pbar);   //P** expansion
            double Yss = rosenbrock(Pss);   //y** evaluation

            if (Yss < Ys) {
                replacePoint(&P[2], &Pss);  //replace Ph with P**
            }

            else {
                replacePoint(&P[2], &Ps);   //replace Ph with P*
            }
        }

        else if (Ys >= Y[1]) {

            if (Ys < Y[2]) {
                replacePoint(&P[2], &Ps);   //replace Ph with P*
            }
            contract(&Pss, P, &Pbar);   //P** contraction
            double Yss = rosenbrock(Pss);   //y** evaluation

            if (Yss < Y[2]) {
                replacePoint(&P[2], &Pss);  //replace Ph with P**
            }

            else {
                getPi(P);   //replace Pi with (Pi+Pl)/2
            }
        }

        else {
            replacePoint(&P[2], &Ps);   //replace Ph with P*
        }

        if (MinCon(Y)) {    //condition for breaking loop
            break;
```

```c
        }
    }
        printf("Final vertices:\n");
        for (int i=0; i<3; i++) {
            printf("(%lf, %lf)\n", P[i].x0, P[i].x1);
        }

        printf("\nEvaluations:\n");
        for (int i=0; i<3; i++) {
            printf("%lf\n", rosenbrock(P[i]));
        }

        printf("\nIterations: %d\n", a+1);
}

int main() {
    Point P[3] = {{0, 0}, {2, 0}, {0, 2}};  //initial positions
    algorithm(P);
    return 0;
}
```