Nelder-Mead optimization:

Goal:

Find the global minimum of the following function:

$$f(x) = (x-3)^2 + 20 \tag{0}$$

Analytical solution:

- find the local minimum for x axis

$$(x-3)^2 + 20 = x^2 - 6x + 9 + 20 = x^2 - 6x + 9 + 29$$

$$\frac{df(x)}{dx} = 0$$

$$\frac{df(x^2 - 6x + 9 + 29\,)}{dx} = 2x - 6 \tag{1}$$

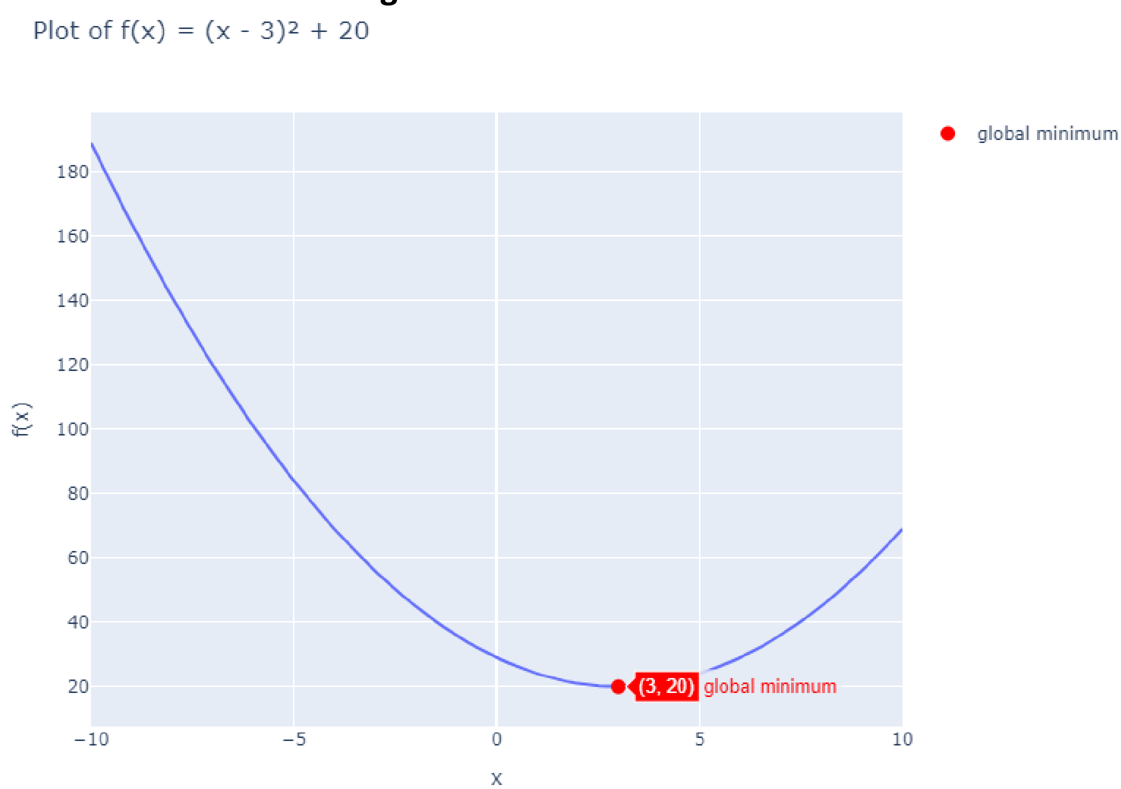
$$2x - 6 = 0$$

$$x = \mathbf{3}$$

- find the local minimum for f(x):

$$\boldsymbol{f(x = 3)} = (x-3)^2 + 20 = (3-3)^2 + 20 = \mathbf{20} \tag{2}$$

Plot the function with the global minimum:



Nelder-Mead algorithm:
- order the 3 points
- calculate a centroid between the best and second worst point

1) Reflexion

$$x_{ref} = x_{centroid} + \alpha(x_{centroid} - x_{worst})$$

2) IF $f(x_{ref}) < f(x_{best})$

The reflection point is better than the best point.

- **Expansion**, calculate expansion point $\mathbf{x_{exp}}$ (with expansion coefficient ϒ = 2)

$$x_{exp} = x_{centroid} + \gamma\left(x_{ref} - x_{centroid}\right) \tag{3}$$

  - **IF** $f(x_{exp}) < f(x_{ref})$
    - replace the worst point with the $\mathbf{x_{exp}}$
  - **ELSE**
    - replace the worst point with the $\mathbf{x_{ref}}$

3) ELIF $f(x_{best}) < f(x_{ref}) < f(x_{second\text{-}worst})$

The reflection point is **better than the second-worst** but **not better than the best point.**

- replace the worst point with x_{ref}

4) ELIF $f(x_{ref}) < f(x_{second\text{-}worst})$

The reflexion point is worse than the second worse.

- **IF** $f(x_{ref}) > f(x_{worst})$, reflexion point is not good enough
  - **Outside contraction** (with contraction coefficient β = 0.5):

$$x_{cont,out} = x_{centroid} + \beta\left(x_{ref} - x_{centroid}\right) \tag{4}$$

    - **ELIF** $f(x_{cont,out}) < f(x_{ref})$
      - replace the worst point with the $\mathbf{x_{cont,out}}$
    - **ELSE**
      - go to step 'Shrink'
- **ELSE**, reflexion failed to improve
  - **Inside contraction** (with contraction coefficient β = 0.5):

$$x_{cont,in} = x_{centroid} + \beta(x_{worst} - x_{centroid})$$

    - **IF** $f(x_{cont,in}) < f(x_{worst})$
      - Replace the worst point with $x_{cont,in}$
    - **ELSE**
      - go to step 'Shrink'

5) Shrink the points

If both contractions attempts fail, shrink the simplex towards the best point x_{best} For all points (except x_{best}).

$$x_i = x_{best} + \delta(x_i - x_{best}) \;\; for \;\; x_i \neq x_{best} \tag{5}$$

Nelder-Mead algorithm – Python code:

- code for evaluating the points

```python
def evaluate_points(f, points):
    # Compute function values for each point in the simplex
    values = [f(x) for x in points]
    values = np.array(values)

    # Sort indices based on function values (from the lowest to the highest)
    sorted_indices = np.argsort(values)

    # Return both sorted simplex points and their corresponding function values
    sorted_points = points[sorted_indices]
    sorted_values = values[sorted_indices]

    return sorted_points, sorted_values
```

- code for findingthe global minimum

```python
def nelder_meads_algorithm(points, max_iter=1000, alpha=1, gamma=2, beta=0.5, delta=0.5, tol=1e-4):
    for iteration in range(max_iter):
        # Step 0: Evaluate points and sort by function values
        points, f_values = evaluate_points(f=objective_function, points=points)

        print(f"iteration: {iteration}")
        print(f"f(x) values: {f_values}")
        print(f"points: {points}")

        # Step 1: Define best, second-worst, and worst points
        x_best, x_worst = points[0], points[-1]
        f_best, f_second_worst, f_worst = f_values[0], f_values[-2], f_values[-1]

        # Step 2: Calculate the centroid (excluding the worst point)
        x_centroid = np.mean(points[:-1], axis=0)

        # Step 3: Reflection
        x_ref = x_centroid + alpha * (x_centroid - x_worst)
        f_ref = objective_function(x_ref)

        if f_ref < f_best:  # Expansion
            x_exp = x_centroid + gamma * (x_ref - x_centroid)
            f_exp = objective_function(x_exp)
            points[-1] = x_exp if f_exp < f_ref else x_ref
        elif f_best <= f_ref < f_second_worst:  # Accept reflection
            points[-1] = x_ref
        else:
            # Contraction
            if f_ref < f_worst:  # Outside contraction
                x_cont = x_centroid + beta * (x_ref - x_centroid)
            else:  # Inside contraction
                x_cont = x_centroid + beta * (x_worst - x_centroid)

            f_cont = objective_function(x_cont)
            if f_cont < min(f_ref, f_worst):
                points[-1] = x_cont
            else:  # Shrink the simplex
                points = [x_best] + [x_best + delta * (point - x_best) for point in points if point != x_best]
                print('shrink')

        # Convergence check
        if np.std(f_values) < tol:
            print(f"Converged after {iteration} iterations.")
            break
```

- call the functions with intial points:

```
# Initial points
initial_points = np.array([20.5, 19.1, 18.3])

# Run the algorithm
nelder_meads_algorithm(points=initial_points)
```

```
iteration: 0
f(x) values: [254.09 279.21 326.25]
points: [18.3 19.1 20.5]
iteration: 1
f(x) values: [166.41 254.09 279.21]
points: [15.1 18.3 19.1]
iteration: 2
f(x) values: [ 99.21 166.41 254.09]
points: [11.9 15.1 18.3]
iteration: 3
f(x) values: [ 20.81  99.21 166.41]
points: [ 3.9 11.9 15.1]
```

.

.

.

.

```
iteration: 17
f(x) values: [20.00001857 20.00063603 20.00072253]
points: [3.00430908 2.97478027 2.97312012]
iteration: 18
f(x) values: [20.00001857 20.00003563 20.00063603]
points: [3.00430908 3.00596924 2.97478027]
iteration: 19
f(x) values: [20.00001857 20.00003563 20.00010081]
points: [3.00430908 3.00596924 2.98995972]
Converged after 19 iterations.
```

Conclusion:

- we demonstrated that results from the numerical solution is eqaul to the results we got from numerical solution in Python
- numerical solution converges when the deviation of all 3 point are lower than defined convergence error