# Particle Swarm Optimization

October 3, 2023

## 0.1 Inicialização

Nesta seção, inicializamos todos os parâmetros necessários, posições e velocidades das partículas. Também definimos a função objetivo que desejamos minimizar.

[1]:
```
import numpy as np

# Parâmetros
w = 0.70
c1 = 0.20
c2 = 0.60
r1 = 0.4657
r2 = 0.5319
iterations = 3  # Número de iterações, pode ser ajustado conforme necessário

# Posições e Velocidades Iniciais
X = np.array([0.4657, 0.8956, 0.3877, 0.4902, 0.5039])
V = np.array([0.5319, 0.8185, 0.8331, 0.7677, 0.1708])

# Função objetivo
def f(x):
    return 1 + 2*x - x**2

print("Posições Iniciais:", X)
print("Velocidades Iniciais:", V)
```

```
Posições Iniciais: [0.4657 0.8956 0.3877 0.4902 0.5039]
Velocidades Iniciais: [0.5319 0.8185 0.8331 0.7677 0.1708]
```

## 0.2 Algoritmo PSO para Minimização

Aqui, implementamos o algoritmo PSO (Particle Swarm Optimization) para minimizar nossa função. As partículas são atualizadas iterativamente com base em suas melhores posições locais e na melhor posição global.

[2]:
```
P_best = X.copy()
G_best_position = X[np.argmin(f(X))]  # Consideramos o mínimo valor de f(x)
G_best_value = f(G_best_position)
```

```python
for iteration in range(1, iterations + 1):
    # Atualizar a velocidade e posição
    V = w * V + c1 * r1 * (P_best - X) + c2 * r2 * (G_best_position - X)
    X = X + V

    # Atualizando as melhores posições locais e globais se necessário
    for i, x in enumerate(X):
        if f(x) < f(P_best[i]):
            P_best[i] = x

    new_best_index = np.argmin(f(X))
    if f(X[new_best_index]) < G_best_value:
        G_best_position = X[new_best_index]
        G_best_value = f(X[new_best_index])

    print(f"\nIteração {iteration}")
    print("Velocidades:", V)
    print("Posições:", X)
    print("Melhores posições locais:", P_best)
    print("Melhor posição global:", G_best_position)
    print("Melhor valor de fitness global:", G_best_value)
```

```
Iteração 1
Velocidades: [0.34743708 0.41085879 0.58317    0.50467815 0.08247593]
Posições: [0.81313708 1.30645879 0.97087    0.99487815 0.58637593]
Melhores posições locais: [0.4657     1.30645879 0.3877     0.4902     0.5039
]
Melhor posição global: 0.3877
Melhor valor de fitness global: 1.62508871

Iteração 2
Velocidades: [ 0.07507168 -0.00561153  0.16778967  0.11249415 -0.01335409]
Posições: [0.88820876 1.30084727 1.13865967 1.1073723  0.57302184]
Melhores posições locais: [0.4657     1.30645879 0.3877     0.4902     0.5039
]
Melhor posição global: 0.3877
Melhor valor de fitness global: 1.62508871

Iteração 3
Velocidades: [-0.14653466 -0.29482723 -0.19215288 -0.20841374 -0.07492948]
Posições: [0.7416741  1.00602004 0.94650679 0.89895856 0.49809235]
Melhores posições locais: [0.4657     1.30645879 0.3877     0.4902
0.49809235]
Melhor posição global: 0.3877
Melhor valor de fitness global: 1.62508871
```

[ ]: