**Laboratório 3 – Métodos de Otimização de Busca Local**

**Inteligência Artificial para Robótica Móvel – CT-213**

**Aluno: Caio Graça Gomes**

**Professor: Marcos Ricardo Omena de Albuquerque Maximo**

## Introdução:

Nesse laboratório, teve-se por objetivo implementar três algoritmos de otimização baseados em busca local, foram eles: *Gradient Descent, Hill Climbing* e *Simulated Annealing*. O problema a ser resolvido foi a minimização do valor de uma função matemática de custo para encontrar o caminho que melhor ajusta um certo conjunto de pontos, sendo a função a ser minimizada análoga à do método dos mínimos quadrados (MMQ). No caso do teste, o problema foi encontrar o coeficiente de desaceleração de uma bola em movimento num campo de futebol de robôs. A bola em movimento perde energia devido a um fenômeno conhecido como *rolling friction*.

## Metodologia e descrição em alto nível dos algoritmos utilizados:

Para o problema em questão, deve-se determinar o coeficiente de desaceleração de uma bola num campo de futebol de robôs. Os robôs possuem câmeras que são capazes de determinar a posição $(x, y)$ da bola no campo, o que permite o cálculo numérico da velocidade da bola.

Sabendo que na teoria a velocidade da bola em função do termo seria da forma $v(t) = v_o - f \cdot t$, podemos definir uma função custo $J: \mathbb{R}^2 \rightarrow \mathbb{R}$ tal que $J([v_0, f] = \sum_{k=1}^{n} (v_0 + ft[k] - v[k])^2$. Assim, queremos minimizar a função $J$ otimizando os seguintes parâmetros $\theta[0] = v_0$ e $\theta[1] = f \Rightarrow \theta = [v_0 \ f]^T$. Para isso foram utilizados três métodos de otimização de funções matemáticas.

### 1) *Gradient Descent*:

O método do *"Gradient Descent*" para o problema em questão de baseou no seguinte algoritmo:

1) Fornecer uma estimativa inicial $\theta_0$ para o ponto de minimização da função custo;
2) Calcular o gradiente da função custo no ponto $\theta$ em questão;
3) Caminhar no sentido oposto ao vetor gradiente da função $J$ por um hiperparâmetro $\alpha$. ($\theta_{k+1} = \theta_k - \alpha \cdot \nabla J$);
4) Se a função custo $J$ for menor que um dado ε ou a quantidade de iterações for maior que um dado número parar, caso contrário, voltar ao passo 2.

## 2) *Hill Climbing*:

O método do *"Hill Climbing"* para o problema em questão de baseou no seguinte algoritmo:

1) Fornecer uma estimativa inicial $\theta_0$ para o ponto de minimização da função custo;
2) Analisar os vizinhos do ponto $\theta$ em questão como um *"grid"* 8-conectado, ou seja os pontos que distam um hiperparâmetro $\delta$ e a projeção das retas que ligam $\theta$ a estes vizinhos formam um ângulo de $k * \pi/4,\ k \in Z$, com a horizontal;
3) Caminhar para o melhor vizinho de $\theta$, isto é, o vizinho de $\theta$ que minimiza a função custo $J$. ($\theta_{k+1} =$ melhor vizinho de $\theta_k$);
4) Se a função custo $J$ for menor que um dado ε ou a quantidade de iterações for maior que um dado número parar, caso contrário, voltar ao passo 2.

## 3) *Simulated Annealing*:

O método do *"Simulated Annealing"* para o problema em questão de baseou no seguinte algoritmo:

1) Fornecer uma estimativa inicial $\theta_0$ para o ponto de minimização da função custo e definir uma função *"schedule"* $T(i) = T_0/(1 + \beta \cdot i^2)$ que fornece a "temperatura" na iteração $i$ utilizando os hiperparâmetros $T_0$ e $\beta$ ;
2) Tomar um vizinho aleatório do $\theta$ em questão, isto é, sortear de maneira uniforme um ângulo aleatório no intervalo $[-\pi, \pi]$ e tomar o vizinho à uma distância de um hiperparâmetro $\delta$ tal que a projeção da reta que ligam $\theta$ a este vizinho forma o ângulo sorteado com a horizontal ;
3) Defina $\Delta E = J(vizinho) - J(\theta)$. Se $\Delta E < 0$, caminhar para o vizinho ($\theta_{k+1} =$ vizinho de $\theta$).
4) Caso contrário, sortear uniformemente um número $r$ no intervalo $(0, 1)$, se $r \leq exp(-\Delta E/T)$, caminhar para o vizinho ($\theta_{k+1} =$ vizinho de $\theta$) ;

5) Se a função custo $J$ for menor que um dado ε ou a quantidade de iterações for maior que um dado número parar, caso contrário, voltar ao passo 2.

## Resultados:

Ao final da implementação do código obteve-se os seguintes resultados:

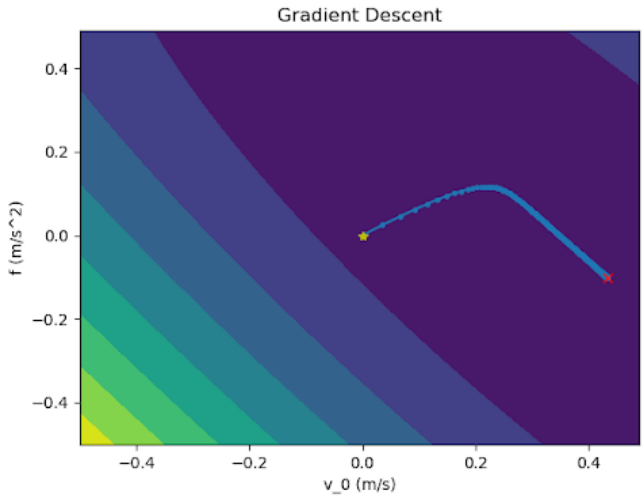


Figura 1: Caminho percorrido pelo *Gradient Descent* na função, com a estrela representando o ponto inicial e o "xis" vermelho representando o ponto final.

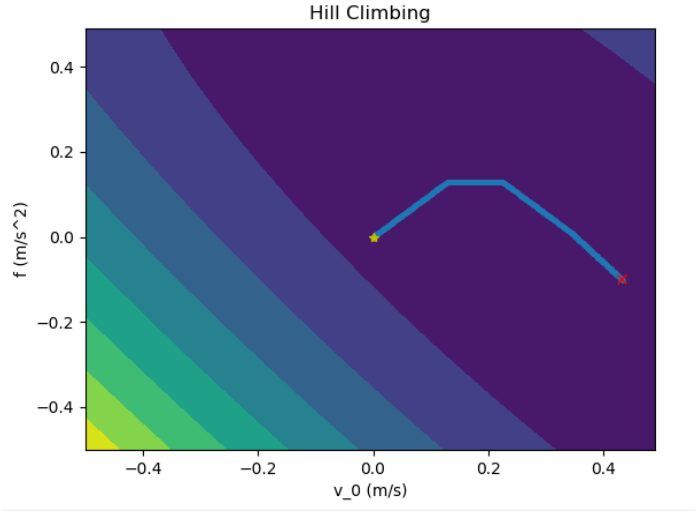


Figura 2: Caminho percorrido pelo *Hill Climbing* na função, com a estrela representando o ponto inicial e o "xis" vermelho representando o ponto final.

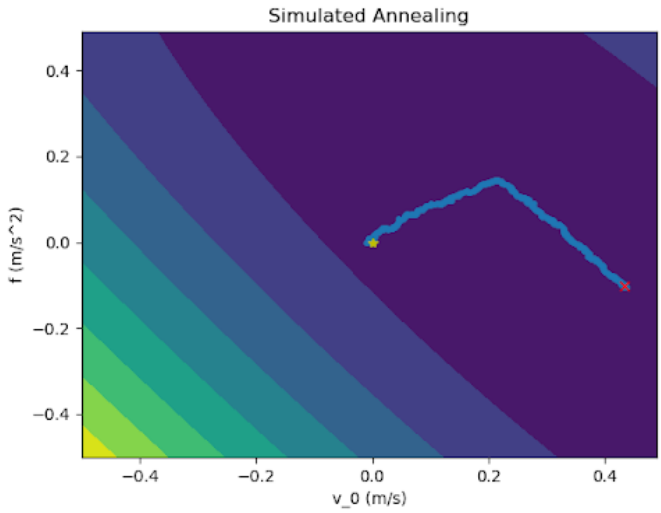


Figura 3: Caminho percorrido pelo *Simulated Annealing* na função, com a estrela representando o ponto inicial e o "xis" vermelho representando o ponto final.

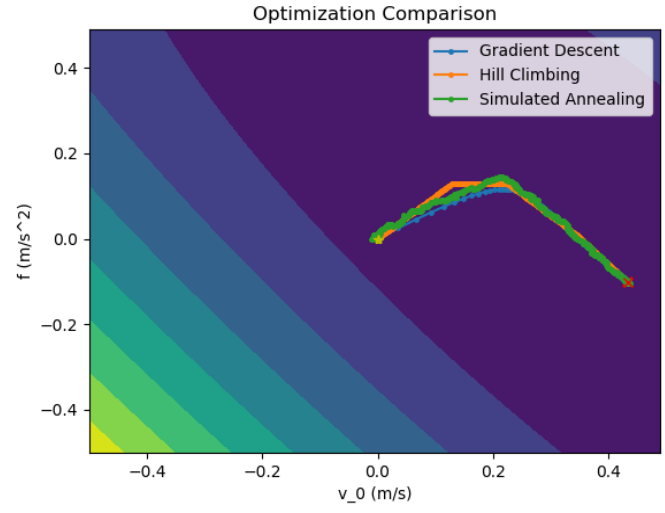


Figura 4: Comparação entre os caminhos percorridos utilizando cada método.

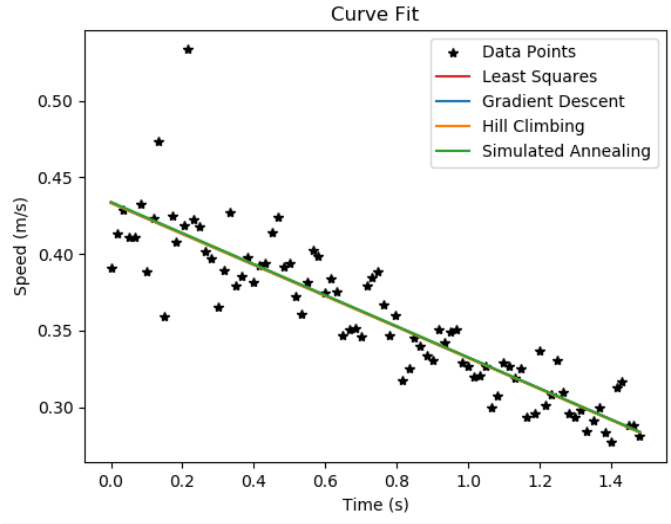


Figura 5: Ajuste linear da velocidade da bola em função do tempo utilizando cada método (os ajustes estão muito próximos).

| **Método** | ***Gradient Descent*** | ***Hill Climbing*** | ***Simulated Annealing*** | ***Least Squares*** |
|---|---|---|---|---|
| $v_0$ | 0.43337067 | 0.43341125 | 0.43397656 | 0.43337277 |
| $f$ | -0.10101846 | -0.10119596 | -0.10134529 | -0.10102096 |

Tabela 1: Comparação dos resultados obtidos de $v_0$ e $f$ pelos diferentes métodos

Assim, conclui-se que os três métodos foram extremamente eficientes na obtenção do $f$ ao comparar-se com o método do *"Least Squares"*, que obtém a solução ótima.No entanto, é válido ressaltar que o *"Gradient Descent"* , o *"Hill Climbing"* e o *"Simulated Annealing"* só foram muito eficientes para o problema em questão, que possuía poucos mínimos locais. Para um problema cuja função oscila muito e tem muitos pontos de mínimo é provável que não se obtenha o mínimo global ao utilizar o *"Gradient Descent"* ou o *"Hill Climbing"*, pois eles tendem a ficar "presos" em mínimos locais. Já o *"Simulated Annealing"* pode ser muito demorado a depender do problema.