Robótica Espacial
Aula prática nº 13
Planeamento de Caminhos com Campos de Potencial

Vitor Santos

Universidade de Aveiro

29 maio 2025

Exercício 1 - Criação de um mapa simples

- Criar um mapa simples com três obstáculos circulares
- Medidas do mapa

```
WW=150; %Width  (# of columns)
HH=100; %Height (# of rows)
```

- Posição dos obstáculos:

```
PP =[
    40  55  65   %X
    80  35  20   %Y
    ];
```

- Raio dos obstáculos

```
r=4; %radius of obstacles
```

- Usar a função `meshgrid` do Matlab e recorrer à equação de um círculo: $(x - x_0)^2 + (y - y_0)^2 < r^2$

- Inicializar um mapa vazio ...
- ... e preenchê-lo com os obstáculos.
- Completar o código seguinte ***:

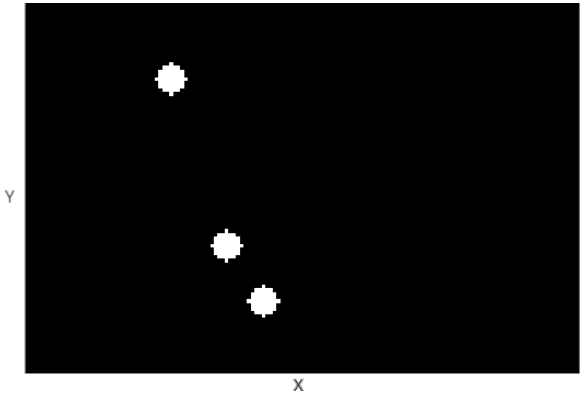
```
X = 1:WW;  % span of X coordinate
Y = 1:HH;  % span of Y coordinate
[XX, YY] = meshgrid(X, Y);
map = zeros(***, ***); %initial empty map

% For every obstacle in the PP matrix,
% fill appropriately the map with values "1"
for k=1:size(PP,2)
  XX0 = XX - PP(***,***);
  YY0 = YY - PP(***,***);
  map((***.^2 + ***.^2) <= ***^2) = 1;
end
```

- O resultado esperado é o que se mostra na página seguinte:

Exercício 1 (cont.) - Representação do mapa

- O mapa fica guardado na matriz `map`.
- Uma forma fácil de representar esse mapa é visualizar a matriz como uma imagem.
- Note-se porém que para termos a variável Y a crescer de baixo para cima, deve-se usar o comando `axis xy`.



Exercício 2 - Cálculo do potencial repulsor

- Calcular do potencial repulsor em todos os pontos do mapa
- Parâmetros a ajustar:
  - k_{rep} – constante de proporcionalidade
  - D_{max} – Distância máxima de influência do potencial repulsor
- O potencial repulsor num ponto P do mapa causado pelo obstáculo O_i é dado por:

$$U_{rep_i}(P) = \begin{cases} \frac{1}{2} k_{rep} \left(\frac{1}{||P - O_i||} - \frac{1}{D_{max}} \right)^2 & \Leftarrow ||P - O_i|| < D_{max} \\ 0 & \Leftarrow ||P - O_i|| \geqslant D_{max} \end{cases}$$

- O potencial causado por todos os N obstáculos no ponto P é dado por:

$$U_{rep}(P) = \sum_{i=1}^{N} U_{rep_i}(P)$$

- Notas:
  - Num mapa discretizado, um obstáculo é cada uma das células ocupadas.
  - Portanto, na fórmula acima, O_i representa efetivamente uma célula ocupada.

Exercício 2 (cont.) - Implementação do potencial repulsor

- Por uma questão de eficiência, os cálculos podem ser feitos de forma matricial para paralelizar as operações em vez de recorrer aos ciclos `for` (embora sejam aceitáveis).
- Completar o código abaixo (***) para calcular o potencial repulsor em todo o mapa.
- A matriz `Urep` terá os valores do potencial repulsor.
- Os valores de `Dmax` e `krep` são sugestões alteráveis.

```
Dmax = 20;                    %maximum distance effectivity
krep = 30;                    %proportional repeller potential constant

[yobst,xobst]=find(***);      %x and y coordinates of all obstacles (cells)
Urep=zeros(size(map));        %initial empty potential map
for k=1:numel(xobst)          %check all obstacle cells one by one
  maskDmax = zeros(***); %initial empty mask the same size of the map
  allDistsToObst=sqrt((***).^2 + (***).^2); % All distances to current obstacle
  maskDmax(allDistsToObst <= ***) = 1;      % Mask to filter the points to be affected
  Urep_parcial= 0.5*krep*(1./(***) - 1/***).^2; %the rep potential for all points on map
  Urep_parcial(***)=0;   %Apply mask to cancel influence on distant points
  Urep=Urep+Urep_parcial;%Update repeller potential map for this obstacle
end
```

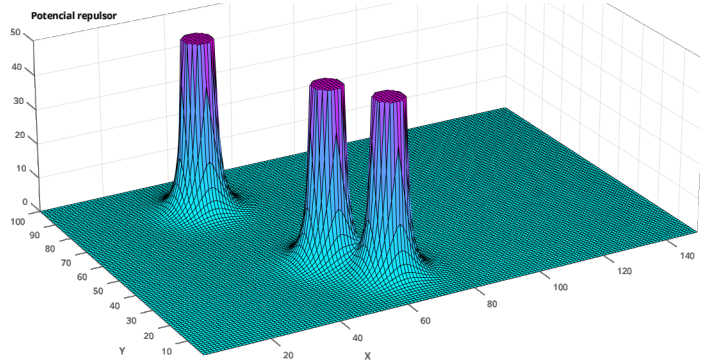
Exercício 2 (conc.) - Resultado e vizualização

- Note-se que o potencial repulsor sobre os obstáculos é infinito e para permitir uma melhor visualização pode-se limitar os valores. Uma sugestão para limitar a 50 é:

```
Urep = min(Urep,50);  % Limit the infinites to a reasonable value
```

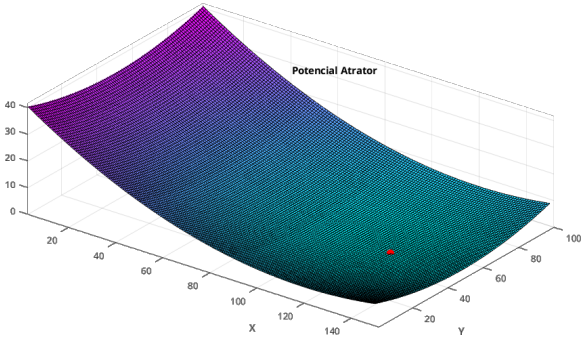
- Para visualizar o campo de potencial repulsor pode-se usar o comando `surf`:

```
surf(XX,YY,Urep); colormap("cool"); xlabel('X'), ylabel('Y'), axis equal
```



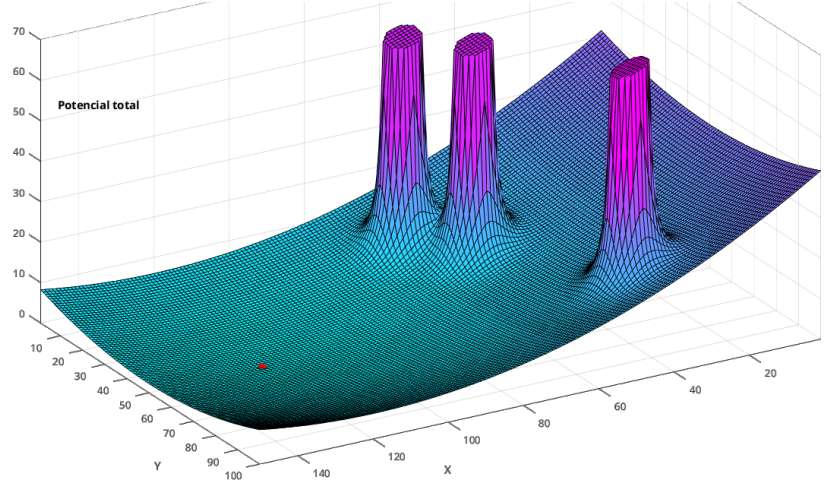
Exercício 3a - Criação do potencial atrator

- O ponto de destino T (*target*) é único e exerce uma atração criando um potencial atrator pela seguinte expressão: $U_{att}(P) = \frac{1}{2} k_{att} ||P - T||^2$
- A constante k_{att} é adaptada a cada problema, e em geral é função das dimensões do mapa e do potencial máximo que se pretende nos pontos mais distantes.
- Para um valor de `T = [120 50]'` e $k_{att} = 0.005$ obter o seguinte resultado:



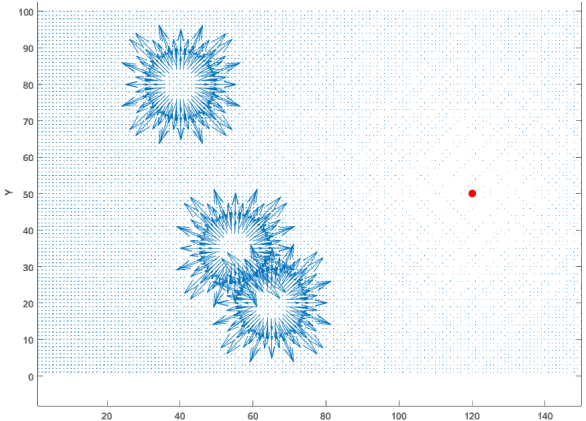
Exercício 3b - Campo de potencial total

- A combinação dos potenciais repulsor e atrator resultam no campo de potencial total que dá o seguinte resultado:



Exercício 4 - Gradientes descendentes no campo de potencial

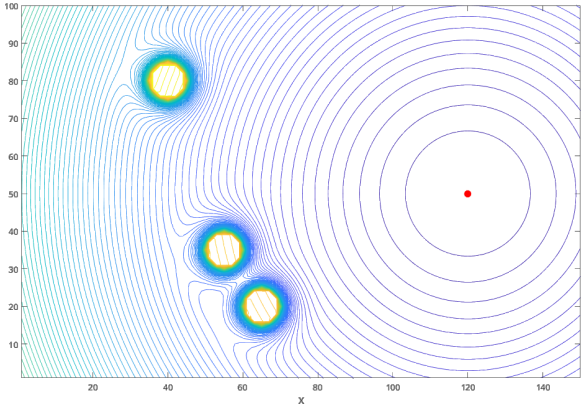
- Tendo o campo de potencial, deve-se determinar a direção rumo aos potenciais mais baixos (dado que o destino estará no potencial mínimo absoluto).
- Isso faz-se usando o negativo do gradiente da função potencial (gradientes descendentes).
- Assim, calcula-se com $[G_x, G_x] = -\vec{\nabla}(U)$ que em matlab se faz com a função `gradient`.



- Usar a função `quiver` para representar os gradientes como ilustrado.
- Representar os vetores com `quiver` usando uma escala maior, e.g. 10.
- Note-se que a dimensão das setas (vetores de gradiente) são muito grandes próximo dos obstáculos e muito pequenas próximo do destino.

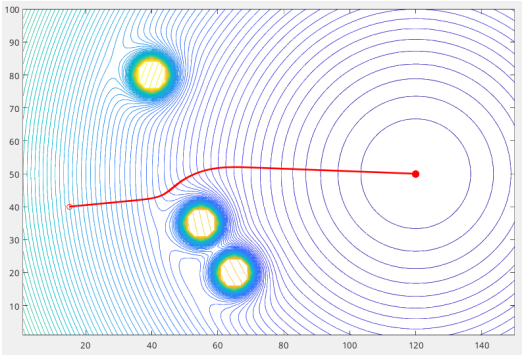
Exercício 5 - Curvas equipotencial

- Uma ferramenta potencialmente útil para uma análise visual dos potenciais são as isolinhas ou curvas equipotencial.
- Em matlab representam-se com o comando `contour`. Neste exemplo usou–se o comando `contour(XX,YY, U, 100)`. O 4º parâmetro é o número de curvas a representar.



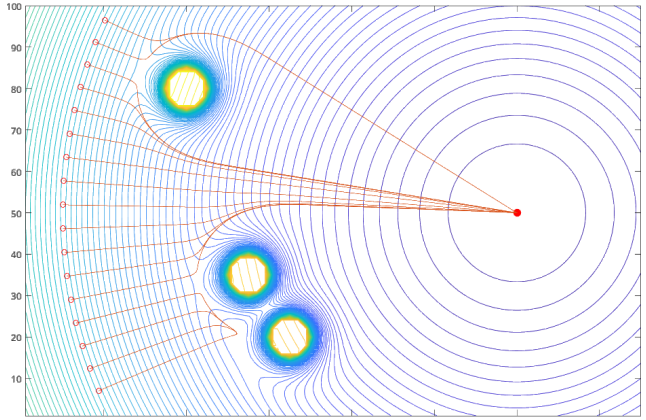
Exercício 6 - Cálculo de caminhos

- O cálculo de caminhos faz-se partindo de um qualquer ponto para o potencial mais baixo.
- O matlab tem uma função que faz essa deteção e que se designa `stream2()`.
- O caminho pode-se desenhar com a função `streamline()`.
- Para o ponto de partida (15,40) o percurso é o seguinte usando o comando:
  `streamline(stream2(X,Y,fx,fy,15,40));`



Exercício 7 - Caminhos que levam a mínimos locais

- Obter os caminhos que começam em pontos distanciados 110 unidades do destino nas direções entre 155° e 205°. Usar uma resolução de 3°, como ilustrado.
- Observar que há vários pontos de partida que levam a um mínimo local.

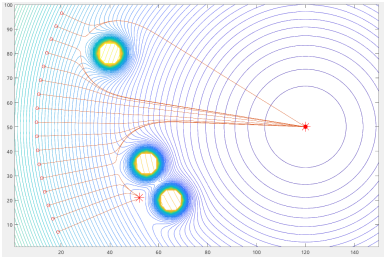


Exercício 8 - Deteção dos pontos de mínimo local

- Em certas circunstâncias é possível detetar se há pontos que são mínimos locais.
- Um mínimo local é um ponto (célula) cujo potencial é menor do que nos seus 8 vizinhos.
- Essa verificação pode ser feita por uma busca célula a célula e com cálculos locais, mas o matlab tem uma função que permite fazê-lo: a função `ordfilt2()`.
- Neste problema podia-se usar deste modo:

```
B = ordfilt2(U, 1, ones(3,3)); [ym,xm] = find(B == U); plot(xm,ym,'r*','MarkerSize',20)
```

- Ao aplicar neste problema verifica-se que há um mínimo local no ponto que já se tinha observado. E o processo deteta outro mínimo, que é o mínimo global no destino!



Exercício 9 - Alterar parâmetros para modificar os mínimos locais

- É possível alterar quer o D_{max} quer o k_{rep}, ou mesmo o k_{att}, mas é preciso garantir que a repulsão dos obstáculos é suficientemente forte para provocar os desvios necessários!
- Tentar encontrar soluções que resultem como as ilustradas (sem mínimos locais):