MATLAB - Comando `fminsearch`

Universidade do Minho, Escola de Engenharia, Departamento de Produção e Sistemas

Ana Maria A. C. Rocha

FMINSEARCH - usa o método de Nelder-Mead.

```
[x,fval,exitflag,output] = fminsearch(fun,x0,options,...)
```

O comando `fminsearch` tem como argumentos de entrada:

`fun` - é a função objetivo a minimizar

`x0` - é o ponto inicial

`options` - opções para o controlo do processo de otimização.

O comando `fminsearch` tem como argumentos de saída:

`x` - é a solução ótima do problema.

`fval` - é o valor da função objetivo em `x`.

`exitflag` - descreve valores de saída do `fminsearch`	
1	indica que a função convergiu para a solução x
0	indica que o número de iterações excedeu o `MaxIter` ou o `MaxFunEvals`.
-1	indica que a função não convergiu para uma solução

`output` - estrutura que contém informação acerca do processo de otimização	
`output.iterations`	indica o número de iterações realizadas
`output.funcCount`	indica o número de cálculos da função
`output.algorithm`	indica o algoritmo usado
`output.message`	indica a mensagem de saída

Para ver as opções disponíveis fazer:

```
optimset('fminsearch')
```

`Options` - definição de parâmetros	
`Display`	Nível de apresentação
	`off` - não apresenta nada
	`iter` - apresenta resultados em cada iteração
	`notify` - apresenta resultado se a função não convergir
	`final` - (default) apresenta apenas o resultado final
`MaxFunEvals`	Número máximo de cálculos da função - (default = 200*numberofvariables)
`MaxIter`	Número máximo de iterações - (default = 200*numberofvariables)
`OutputFcn`	Especifica uma ou mais funções que o processo de otimização pode invocar, em cada iteração.
`PlotFcns`	Representa graficamente várias medidas do progresso do algoritmo, ao longo das iterações.
	`@optimplotx` - desenha o ponto actual.
	`@optimplotfunccount` - desenha o número de cálculos da função objetivo.
	`@optimplotfval` - desenha o valor da função objetivo.
`TolFun`	Tolerância de paragem relativamente à função objetivo - (default = 1e-4)
`TolX`	Tolerância de paragem relativamente a x - (default = 1e-4)
`FunValCheck`	Verifica se os valores da função objetivo não são números complexos, Inf ou NaN.
	`on` - apresenta erro
	`off` - (default) não apresenta erro

1. Determine o mínimo da seguinte função não diferenciável

$$f(x_1, x_2) = \max(|x_1|, |x_2 - 1|)$$

através do método do simplex de Nelder-Mead (usando o comando `fminsearch`). Inicie o processo com $x^{(1)} = (1, 1)^T$.

(a) Sem usar qualquer opção.

```
[x,fval,exitflag,output]=fminsearch(@NM1,[1,1])
function [ f ] = NM1( x )
    f=max(abs(x(1)),abs(x(2)-1));
end
```

(b) Visualizando os resultados obtidos em cada iteração.

```
op=optimset('Display','iter');
[x,fval,exitflag,output]=fminsearch(@NM1,[1,1],op)
function [ f ] = NM1( x )
    f=max(abs(x(1)),abs(x(2)-1));
end
```

(c) Representando graficamente os valores da função objetivo ao longo das iterações.

```
op=optimset('PlotFcns',@optimplotfval);
[x,fval,exitflag,output]=fminsearch(@NM1,[1,1],op)
function [ f ] = NM1( x )
    f=max(abs(x(1)),abs(x(2)-1));
end
```

(d) Visualizando os resultados obtidos em cada iteração e a representação gráfica dos valores da função objetivo ao longo das iterações.

```
op=optimset('Display','iter','PlotFcns',@optimplotfval);
[x,fval,exitflag,output]=fminsearch(@NM1,[1,1],op)
function [ f ] = NM1( x )
    f=max(abs(x(1)),abs(x(2)-1));
end
```

(e) Usando como tolerância de paragem $TolX = 10^{-10}$.

```
op=optimset('TolX',1e-10);
[x,fval,exitflag,output]=fminsearch(@NM1,[1,1],op)
function [ f ] = NM1( x )
    f=max(abs(x(1)),abs(x(2)-1));
end
```

(f) Usando como tolerância de paragem $TolFun = 10^{-12}$.

```
op=optimset('TolFun',1e-12);
[x,fval,exitflag,output]=fminsearch(@NM1,[1,1],op)
function [ f ] = NM1( x )
    f=max(abs(x(1)),abs(x(2)-1));
end
```

(g) Usando como tolerância de paragem 20 iterações.

```
op=optimset('MaxIter',20);
[x,fval,exitflag,output]=fminsearch(@NM1,[1,1],op)
function [ f ] = NM1( x )
    f=max(abs(x(1)),abs(x(2)-1));
end
```

(h) Usando como tolerâncias de paragem $TolX = 10^{-4}$, $TolFun = 10^{-2}$ e 50 como máximo de iterações.

```
op=optimset('TolX',1e-3,'TolFun',1e-2,'MaxIter',50);
[x,fval,exitflag,output]=fminsearch(@NM1,[1,1],op)
function [ f ] = NM1( x )
    f=max(abs(x(1)),abs(x(2)-1));
end
```

2. Calcule o máximo da seguinte função não diferenciável

$$f(x_1, x_2) = -|x_1 x_2| - x_2^2$$

usando o comando `fminsearch`. Inicie o processo com $x^{(1)} = (-1, 1)^T$.

(a) Sem usar qualquer opção.

```
[x,fval,exitflag,output]=fminsearch(@NM1,[-1 1])
function [ f ] = NM1( x )
    f=abs(x(1)*x(2))+x(2)^2;
end
```

(b) Usando como tolerância de paragem $TolX = 10^{-8}$.

```
op=optimset('TolX',1e-8);
[x,fval,exitflag,output]=fminsearch(@NM1,[-1 1],op)
function [ f ] = NM1( x )
    f=abs(x(1)*x(2))+x(2)^2;
end
```

(c) Usando como tolerância de paragem $TolFun = 10^{-6}$.

```
op=optimset('TolFun',1e-6);
[x,fval,exitflag,output]=fminsearch(@NM1,[-1 1],op)
function [ f ] = NM1( x )
     f=abs(x(1)*x(2))+x(2)^2;
end
```

(d) Usando como tolerâncias de paragem o máximo de iterações de 50.

```
op=optimset('Maxiter',50);
[x,fval,exitflag,output]=fminsearch(@NM1,[-1 1],op)
function [ f ] = NM1( x )
     f=abs(x(1)*x(2))+x(2)^2;
end
```

3. Determine o mínimo da seguinte função não diferenciável

$$f(x_1, x_2) = \max(\, (x_1 - 1)^2 \,,\, x_2^2 + x_1 \,,\, 4(x_2 - 1)^2 \,)$$

através do comando `fminsearch`. Inicie o processo com $x^{(1)} = (1, 0)^T$.

(a) Sem usar qualquer opção.

```
[x,fval,exitflag,output]=fminsearch(@NM1,[1, 0])
function [ f ] = NM1( x )
    u=[(x(1)-1)^2 , x(2)^2+x(1) , 4*(x(2)-1)^2];
    f=max(u);
end
```

(b) Use como tolerâncias de paragem $TolX = 10^{-6}$, $TolFun = 10^{-8}$.

```
op=optimset('TolX',1e-6,'TolFun',1e-8);
[x,fval,exitflag,output]=fminsearch(@NM1,[1, 0],op)
function [ f ] = NM1( x )
u=[(x(1)-1)^2 , x(2)^2+x(1) , 4*(x(2)-1)^2];
f=max(u);
end
```

(c) Use como tolerâncias de paragem $TolX = 10^{-6}$, $TolFun = 10^{-8}$ e 100 como máximo de iterações.

```
op=optimset('TolX',1e-6,'TolFun',1e-8,'maxiter',100);
[x,fval,exitflag,output]=fminsearch(@NM1,[1, 0],op)
function [ f ] = NM1( x )
    u=[(x(1)-1)^2 , x(2)^2+x(1) , 4*(x(2)-1)^2];
    f=max(u);
end
```