

Selección actuadores/referencias

© 2021, 2023 Antonio Sala Piqueras, Universitat Politècnica de València. Todos los derechos reservados.

Este código ejecutó sin errores en Matlab R2022b

Vídeo presentación en <http://personales.upv.es/asala/YT/V/sarefp1.html> y <http://personales.upv.es/asala/YT/V/sarefp2.html>

Colección completa de materiales en <http://personales.upv.es/asala/YT>

```
%modelo linealizado
G=[3 4 6;2 -4 -5];
%unidades físicas (incremento)
limsyarriba=[2 1];
limsyabajo=[-2 -1.25];
limsuarriba=[0.5 1 1.5]; %en incremento, claro
limsuabajo=[-0.5 -1 -4];%podría ser NO simétrico

verticesy= ...
[limsyarriba(1) limsyarriba(1) limsyabajo(1) limsyabajo(1); ...
 limsyabajo(2) limsyarriba(2) limsyabajo(2) limsyarriba(2)]
```

```
verticesy = 2x4
    2.0000    2.0000   -2.0000   -2.0000
   -1.2500    1.0000   -1.2500    1.0000
```

Vamos a ver si la cosa satura o no con mínimos cuadrados.

Nota: no está escalado, con lo que no se está considerando que los incrementos de actuadores no son "iguales"; al final del código se discute la solución escalada.

```
usol=pinv(G)*verticesy
```

```
usol = 3x4
    0.1114    0.5869   -0.6397   -0.1642
    0.1266   -0.0384    0.0568   -0.1083
    0.1932    0.0655   -0.0513   -0.1790
```

```
find(usol<limsuabajo')
```

```
ans = 7
```

```
find(usol>limsuarriba')
```

```
ans = 4
```

```
%nos pasamos por los dos lados. NO vale PINV...
```

Se sale, pero la solución no es única con 3 actuadores...

```
nohacenada=null(G)
```

```
nohacenada = 3x1
```

```
-0.1182
-0.7979
0.5911
```

```
estoescero=G*nohacenada
```

```
estoescero = 2x1
10-15 x
-0.4441
0
```

```
G*usol(:,2)
```

```
ans = 2x1
2.0000
1.0000
```

```
nuevaU2=usol(:,2)+5421*nohacenada
```

```
nuevaU2 = 3x1
103 x
-0.6402
-4.3256
3.2042
```

```
G*(nuevaU2)
```

```
ans = 2x1
2.0000
1.0000
```

Factibilidad linprog/quadprog

```
for nvert=1:4
    disp('probamos vértice:')
    vert=verticesy(:,nvert)
    U=quadprog(eye(3),zeros(3,1),[],[],G,vert,limsuabajo,limsuarriba);%optimizamos cual
end
```

```
probamos vértice:
```

```
vert = 2x1
2.0000
-1.2500
```

Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in feasible directions, to within the value of the optimality tolerance, and constraints are satisfied to within the value of the constraint tolerance.

```
<stopping criteria details>
```

```
probamos vértice:
```

```
vert = 2x1
2
1
```

Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in feasible directions, to within the value of the optimality tolerance, and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>

probamos vértice:

```
vert = 2x1
    -2.0000
    -1.2500
```

Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in feasible directions, to within the value of the optimality tolerance, and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>

probamos vértice:

```
vert = 2x1
    -2
     1
```

Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in feasible directions, to within the value of the optimality tolerance, and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>

```
for nvert=1:4
    disp('probamos vértice:')
    vert=verticesy(:,nvert)
    U=linprog([1 1 1],[],[],G,vert,limsuabajo,limsuarriba) %optimizamos cualquier cosa
end
```

probamos vértice:

```
vert = 2x1
    2.0000
    -1.2500
```

Optimal solution found.

```
U = 3x1
   -0.0556
   -1.0000
    1.0278
```

probamos vértice:

```
vert = 2x1
     2
     1
```

Optimal solution found.

```
U = 3x1
    0.4444
   -1.0000
    0.7778
```

probamos vértice:

```
vert = 2x1
   -2.0000
   -1.2500
```

Optimal solution found.

```
U = 3x1
   -0.5000
    1.0000
   -0.7500
```

probamos vértice:

```
vert = 2x1
```

```

-2
1
Optimal solution found.
U = 3x1
-0.2963
-1.0000
0.4815

```

Efecto del escalado

En la solución pseudoinversa sin escalar, el actuador 1 ha saturado porque todos eran tratados "por igual", pero ese era el de menor incremento hasta saturación. La solución con un escalado de modo que todos los actuadores tengan incremento de 1 hasta saturación será:

```
Eu=diag(min(abs(limsuarriba),abs(limsuabajo)))
```

```
Eu = 3x3
0.5000    0    0
0    1.0000    0
0    0    1.5000

```

```
Gesc=G*Eu;
uesc_pinv=pinv(Gesc)*verticesy %en unidades "escaladas"
```

```
uesc_pinv = 3x4
0.1940    1.1098   -1.2115   -0.2958
0.0298   -0.2545    0.2861    0.0018
0.1767    0.1504   -0.1475   -0.1737

```

```
u_pinv2=Eu*uesc_pinv %en unidades "físicas" originales
```

```
u_pinv2 = 3x4
0.0970    0.5549   -0.6058   -0.1479
0.0298   -0.2545    0.2861    0.0018
0.2650    0.2256   -0.2212   -0.2606

```

```
find(u_pinv2<limsuabajo')
```

```
ans = 7
```

```
find(u_pinv2>limsuarriba')
```

```
ans = 4
```

En definitiva, seguimos teniendo el mismo problema de saturación aún escalando la entrada para que todas ellas tengan un significado "homogéneo".