Pruebas Programación Dinámica (espacio estados finito)

Presentaciones en vídeo:

http://personales.upv.es/asala/YT/V/dp2d1.html , http://personales.upv.es/asala/YT/V/dp2d2.html , http://personales.upv.es/asala/YT/V/dp2d3.html .

Este código funcionó correctamente con Matlab **`R2021a`**

Objetivos: plantear un problema de programación dinámica en un mundo 2D (una especie de "laberinto") y explorar las soluciones clásicas del mismo (exactas, aproximando la función de valor por una tabla de datos, un elemento por estado).

Tabla de Contenidos

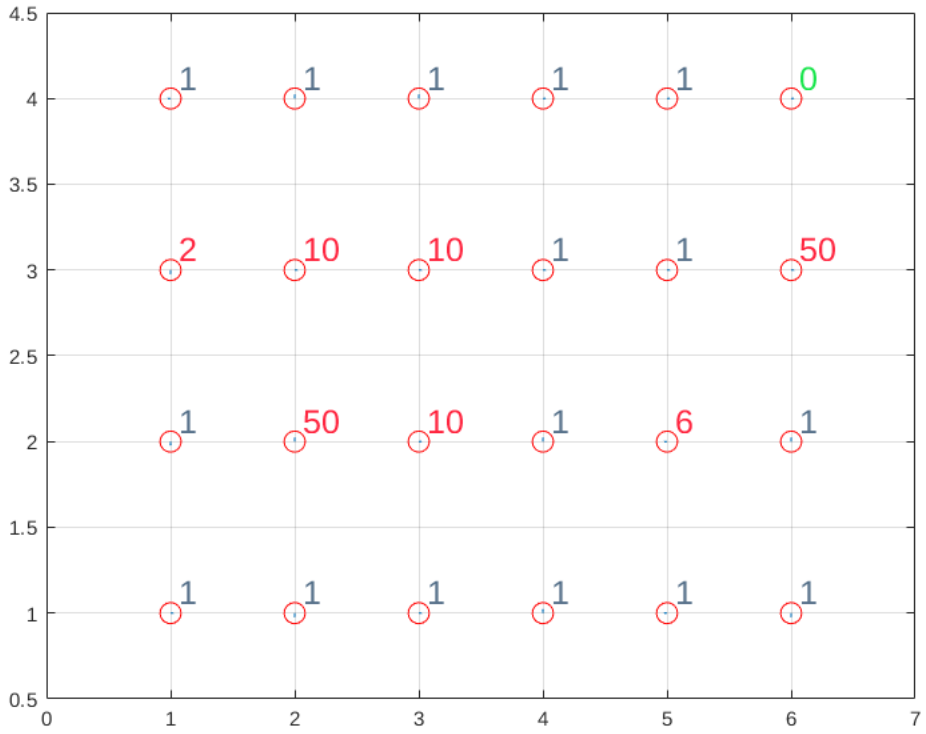
Modelado y planteamiento del problema

El espacio de estados será un mundo de 4x6

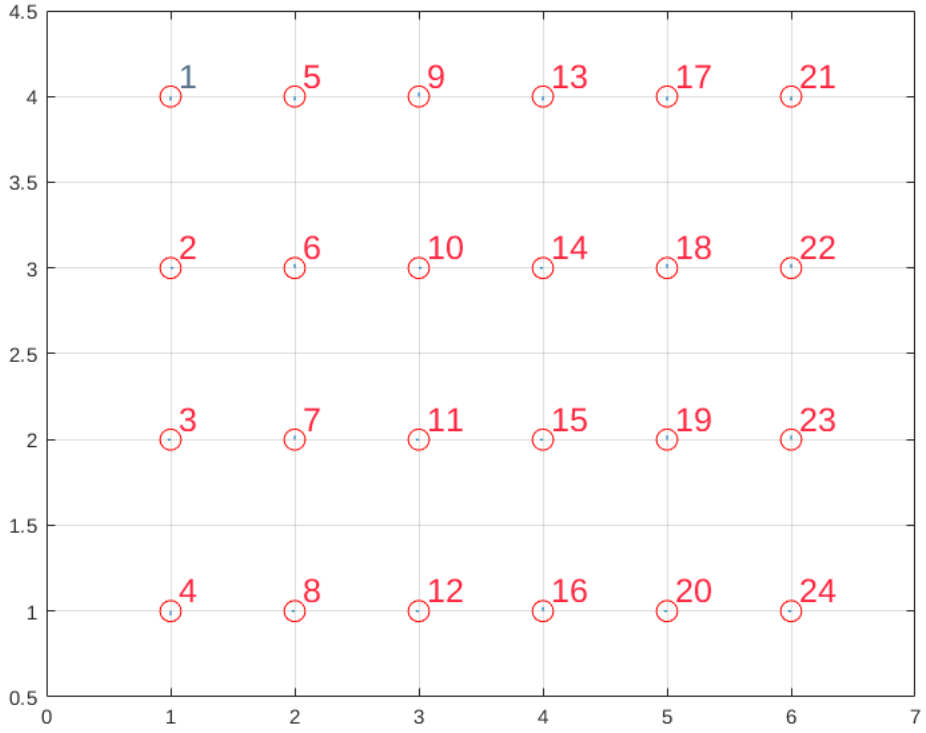
```
X=1:24; Nestados=length(X);
```

El coste "inmediato" por estar en una celda x será 1 y en algunas habrá más coste... y luego añadiremos coste en u:

```
L=ones(4,6); L(1,6)=0; L(2:3,2:3)=10; L(3,2)=50; L(2,6)=50; L(2,1)=2; L(3,5)=6;
dibujaPolitica([],L,false)
```




```
dibujaPolitica([],1:24,false) %números asociados a cada celda
```



Las acciones posibles son (1:izquierda, 2:derecha, 3:arriba, 4:abajo).

```
U=1:4; Ninputs=length(U);
CosteU=[0 0 0.2 0]; %ponemos que cuesta un poco más ir hacia arriba
```

Nota: "chocar" con la pared simplemente no mueve, no tiene coste.

El descuento de la programación dinámica será:

```
gam=0.99995;
factor_no_moverse= 1/(1-gam)
```

```
factor_no_moverse = 2.0000e+04
```

```
log(0.05)/log(gam)
```

```
ans = 5.9913e+04
```

Formemos tabla de transiciones con Xorigen--Ucontrol--Sucesor, y también lista de coste inmediato:

```
NTransiciones=Nestados*Ninputs;
X_U_Suc=zeros(NTransiciones,3);
Lxu=zeros(NTransiciones,1);
k=0;
for i=1:Nestados
    for j=1:Ninputs
        k=k+1;
        X_U_Suc(k,:)=[X(i) U(j) Modelo(X(i),U(j))];
        Lxu(k)=L(i)+CosteU(j); %coste por estar en X(i) y aplicar U(j)
    end
end
```

Solución 1: Value Iteration (iteración de valor)

Nos basamos en construir una tabla de datos para $V(x)$, esto es, $\{V_1, V_2, \dots, V_{24}\}$, de modo que iteraremos

$$V^{opt}(x) = \min_u \left(L(x,u) + \gamma V^{[opt]}(f(x,u))\right)$$

$$V^{[iter+1]}(x) = \min_u \left(L(x,u) + \gamma V^{[iter]}(f(x,u))\right)$$

hasta que V converja (umbral $\|V^{[iter+1]}(x) - V^{[iter]}(x)\| \leq 5 \cdot 10^{-5}$).

```
V=zeros(24,1); %Función de valor inicial (arbitraria)
maxiter=1500; %realmente en este problema tardaremos mucho menos de 1500 iteraciones.
tic
for iter=1:maxiter
```

```
    Vxnext=V(X_U_Suc(:,3)); %Es 96x1
    Q=Lxu+gam*Vxnext; %Ec. de Bellman. Q es 96x1
    Vold=V;
    %hay que poner Q en forma adecuada (Ncontrol x Nestados)
    Qxu=reshape(Q,[4 24]); %si empezáramos aquí sería aprender Q, las iteraciones son '
    [V,indiceminimo]=min(Qxu); %minimizamos sobre las acciones de control
    V=V';
    if norm(V-Vold)<=5e-5  %comprobamos si hemos convergido
        fprintf("VI Converged to less than 1e-4 increment in %d iterations. ",iter), to
        break
    end
    %figure()
    %dibujaPolitica(indiceminimo,round(V,2)), title("Iteración "+num2str(iter))
end
```

```
VI Converged to less than 1e-4 increment in 10 iterations. Elapsed time is 0.002730 seconds.
```

```
if(iter==maxiter)
    disp("No converge Value Iteration")
end
Varray=reshape(V,[4 6]) %para verlo en forma de "mundo 2D"
```

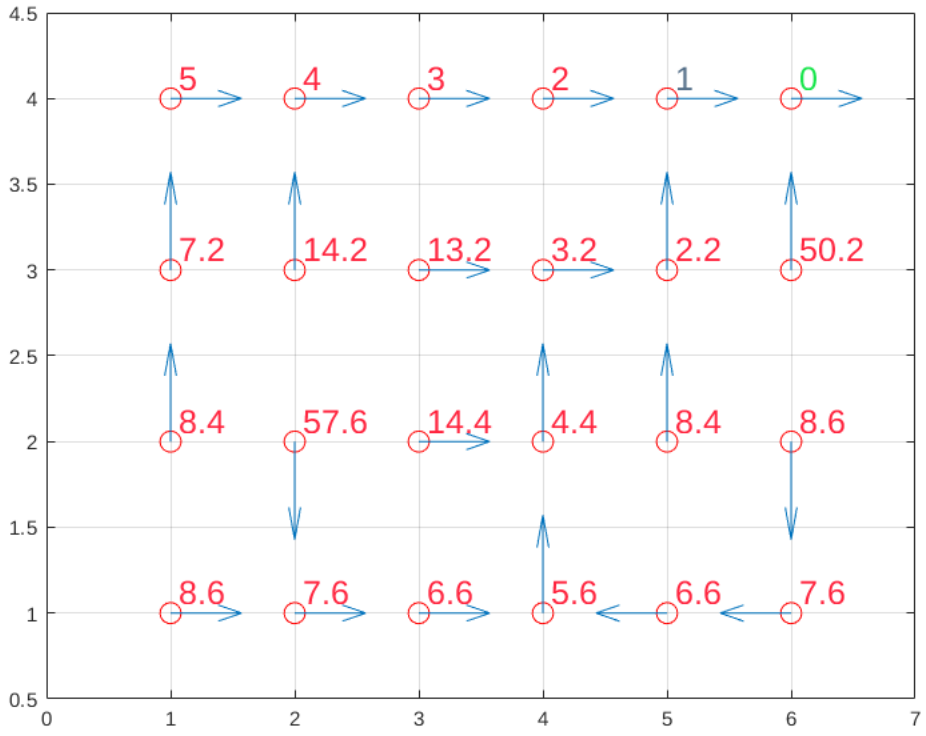
```
Varray = 4×6
    4.9995    3.9997    2.9999    2.0000    1.0000         0
    7.1993   14.1995   13.1997    3.1998    2.1999   50.2000
    8.3989   57.5985   14.3995    4.3997    8.3998    8.5985
    8.5985    7.5989    6.5992    5.5995    6.5992    7.5989
```

La política óptima es el número de "fila" de Qxu (que corresponde a la acción de control) que ha dado lugar al mínimo:

```
[~,piopt]=min(Qxu);
OptU=reshape(piopt,[4,6]) %son índices al vector de acciones de control
```

```
OptU = 4×6
     2     2     2     2     2     2
     3     3     2     2     3     3
     3     4     2     3     3     4
     2     2     2     3     1     1
```

```
dibujaPolitica(piopt,round(V,2)) %aquí vemos el "movimiento" que significa cada índice.
```



Solución 2: Programación Lineal

$V(x) \leq L(x,u) + \gamma V(f(x,u))$, para todo x, y todo u.

Maximizar V obtiene la igualdad con el mínimo de la derecha (ec. Bellman).

$$V^{opt}(x) = \min_u \left(L(x,u) + \gamma V^{opt}(f(x,u))\right)$$

esto es: $V(x) - \gamma V(f(x,u)) \leq L(x,u)$

```
tic
%formamos matrices
A=zeros(NTransiciones,Nestados); %nestados^2*nacciones elementos 96x24
B=Lxu;
for k=1:NTransiciones
    A(k,X_U_Suc(k,1))=A(k,X_U_Suc(k,1))+1;
    A(k,X_U_Suc(k,3))=A(k,X_U_Suc(k,3))-gam;
end
%resolvemos
VoptLP=linprog(-ones(Nestados,1),A,B); toc
```

```
Optimal solution found.

Elapsed time is 0.018123 seconds.
```

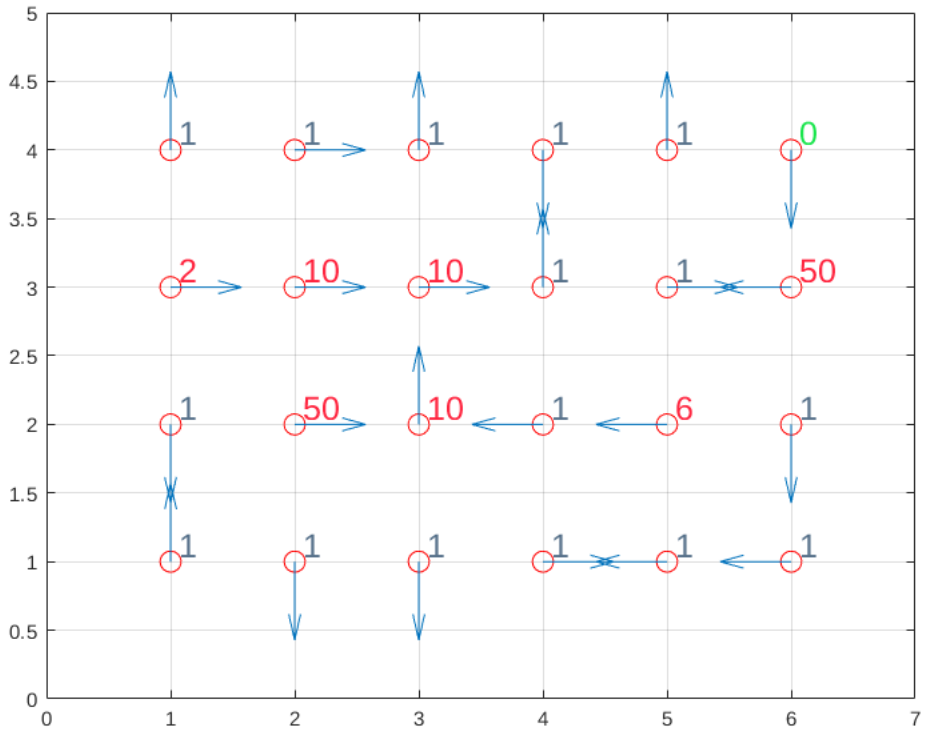
```
VarrayLP=reshape(VoptLP,[4 6]) %coincide con otras
```

```
VarrayLP = 4×6
    4.9995    3.9997    2.9999    1.9999    1.0000         0
    7.1993   14.1995   13.1997    3.1998    2.1999   50.2000
    8.3989   57.5985   14.3995    4.3997    8.3998    8.5985
    8.5985    7.5989    6.5992    5.5995    6.5992    7.5989
```

Solución 3: Policy Iteration (iteración de política)

Inicializamos una política $\pi(x)$ de coste finito, en este caso al azar:

```
pi_tmp=randi([1 4],Nestados,1); %empiezo al azar... con gam<1, coste es finito.
dibujaPolitica(pi_tmp,L)
```



Y lo que debemos iterar son los pasos de:

1.) Evaluar la función de valor de la política $V^{\pi}(x)$, resolviendo la ecuación de Bellman

$V^{\pi} = L(x,\pi(x)) + \gamma V^{\pi}(f(x,\pi(x)))$. *Notación:* en la iteración "iter" le llamaremos $V^{\pi^{[iter]}}(x)$.

2.) Mejorarla a un paso con $\pi^{[iter+1]} := \arg\min_{u} L(x,u) + \gamma V^{\pi^{[iter]}}(f(x,u))$

Hasta que converja la política.

```
MaxItersPolicyIteration=150; maxiterpolicyev=400000;
```

```
Sucesores=zeros(Nestados,1);
Lpi=zeros(Nestados,1);
tic
opcion=2;
for pi_iter=1:MaxItersPolicyIteration
```

- Simulamos la política para obtener sucesor y coste inmediato, utilizados luego en evaluation/ improvement

```
    for i=1:Nestados
        Sucesores(i,:)=Modelo(X(i),pi_tmp(i)); %tamaño 24x1.
        Lpi(i)=L(i)+CosteU(pi_tmp(i));
    end
```

- Hacemos POLICY EVALUATION

```
    if(opcion==1) %usamos value iteration para evaluar la propia política
        if pi_iter==1
            Vpi=zeros(Nestados,1); %inicial valor estimado para política
        else
            %No hacemos nada, Vpi comienza valiendo el Vpi de la anterior
            %iteración
        end
        for iter=1:maxiterpolicyev
            Vold=Vpi;
            Vpi_next=Vpi(Sucesores);
            Vpi=Lpi+gam*Vpi_next;
            if norm(Vpi-Vold)<=1e-7  %comprobamos si hemos convergido
                fprintf("Policy Evaluation (via VI) Converged to less than 1e-4 increme
                break
            end
        end
        if(iter==maxiterpolicyev)
            disp("No converge Value Iteration")
        end
    else %opción con "ecuaciones" porque no hay aproximación
```

$V^{\pi}(x) = L(x,\pi(x)) + \gamma V^{\pi}(f(x,\pi(x)))$, esto es:

$$V^{\pi}(x) - \gamma V^{\pi}(f(x,\pi(x))) = L(x,\pi(x))$$

```
        A=zeros(Nestados,Nestados); %tamaño 24x24
        B=Lpi;
        for k=1:Nestados
            A(k,k)=A(k,k)+1;
            A(k,Sucesores(k))=A(k,Sucesores(k))-gam;
        end
        Vpi=A\B; %resolvemos para obtener el vector de valores de cada estado.
    end
```

```
%dibujaPolitica(pi_tmp,round(Vpi))
```

- Hacemos el policy IMPROVEMENT, calculando Q^{π} y minimizándolo:

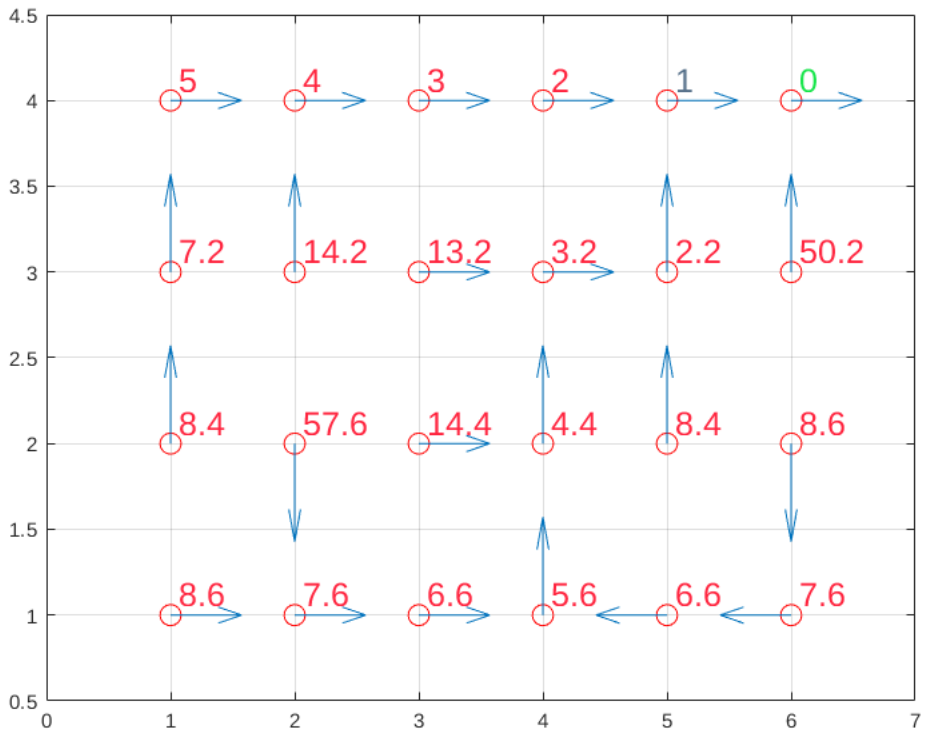
```
    Qpi=Lxu+gam*Vpi(X_U_Suc(:,3)); %Bellman, valor de ahora una acción y luego la polít
    Qpixu=reshape(Qpi,[4 24]); %Ncontrol x Nestados
    [~,newpi]=min(Qpixu); %aquí obtenemos la nueva política.

    if norm(newpi-pi_tmp)<=5e-5 %comprobamos si hemos convergido...
        %como pi es "discreta (1,2,3,4)" cuando esto se cumpla entonces
        %newpi será IGUAL a pi_tmp.
        fprintf("PI Converged to less than 1e-4 increment in %d iterations.",pi_iter),
        break
    end
    pi_tmp=newpi;
end
```

```
PI Converged to less than 1e-4 increment in 11 iterations.Elapsed time is 0.006671 seconds.
```

```
reshape(Vpi,[4 6])
```

```
ans = 4×6
    4.9995    3.9997    2.9999    2.0000    1.0000         0
    7.1993   14.1995   13.1997    3.1998    2.1999   50.2000
    8.3989   57.5985   14.3995    4.3997    8.3998    8.5985
    8.5985    7.5989    6.5992    5.5995    6.5992    7.5989
```

```
dibujaPolitica(pi_tmp,round(Vpi,2)) %coincide con Value Iteration, y con linprog.
```



```
function sucesor=Modelo(X,U)
Ncols=6;Nfilas=4;
fila=mod(X-1,Nfilas)+1;
col=floor((X-1)/Nfilas)+1;
switch U
    case 1 %izquierda
        col=max(1,col-1); %si se choca a la izquierda, mantiene col=1
    case 2 %derecha
        col=min(Ncols,col+1);%si se choca a la dcha., mantiene col=6
    case 3 %arriba
        fila=max(1,fila-1); %si se choca arriba, mantiene fila=1
    case 4 %abajo
        fila=min(Nfilas,fila+1); %si se choca abajo, mantiene fila=4
end
sucesor=1+(col-1)*Nfilas+(fila-1);
end


function dibujaPolitica(OptUvector,L,dibujaflechas)
arguments
    OptUvector
    L=[];
    dibujaflechas=true
end
if(dibujaflechas)
    Scaling=0.51;
else
    Scaling=0.02;
end
if isempty(OptUvector)
    OptUvector=randi([1 4],24,1);
end
OptU=reshape(OptUvector,[4 6]); %matriz
Urow=[0 0 1 -1]; %para quiver
Ucol=[-1 1 0 0]; %para quiver
flOU=flipud(OptU);%para gráficas
q=quiver(Ucol(flOU),Urow(flOU),Scaling); grid on %gráfico con flechas
q.Marker='o'; q.MarkerEdgeColor='r';q.MarkerSize=12;
if ~isempty(L) %etiquetas con color indicando los valores de L
    Lar=reshape(L,[4 6]);
    for row=1:4
        for col=1:6
            Val=Lar(row,col);
            if(Val<0.5)
                Col=[.1 .9 .3];
            elseif(Val<1.5)
                Col=[.35 .45 .55];
            else
                Col=[1 .15 .25];
```

```
            end
            text(col+.06,5-row+.13,num2str(Val),FontSize=19,Color=Col);
        end
    end
end
end
```