Zeit-diskretes Kontrollsystem

$$x_{k+1} = f(x_k, u_k), \quad k = 0, 1, 2, \ldots, \tag{1}$$

$$J(x, (u_k)) := \sum_{k=0}^{\infty} c(x_k, u_k)$$

$$V(x) := \inf_{(u_k)} J(x, (u_k))$$

Bellman-Gleichung:

$$V(x) = \inf_{u \in U} \{c(x, u) + V(f(x, u))\}, \quad x \in \Omega \tag{2}$$

Beispiel: invertiertes Pendel:

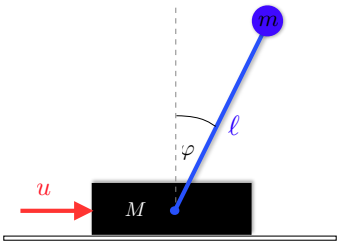

Figure 1: Model of the inverted pendulum on a cart.

Zustandsraum: $x = (x_1, x_2) := (\varphi, \dot{\varphi}) \in \mathbb{R}^2$.

$$m_r \cos^2(\varphi)\, \ddot{\varphi} - \frac{1}{2} m_r \sin(2\varphi)\, \dot{\varphi}^2 + \frac{g}{\ell} \sin(\varphi) - \frac{4}{3} - \frac{m_r}{m\,\ell} \cos(\varphi)\, u = 0, \tag{3}$$

Kružkov-Transformation: $V \mapsto v = \exp(-V(\cdot))$, wobei $\exp(-\infty) := 0$.
Mit dieser Transformation lautet die Bellman-Gleichung:

$$v(x) = \sup_{u \in U} \left\{ e^{-c(x,u)} v(f(x, u)) \right\}, \quad x \in \Omega \tag{4}$$

Prof. Dr. O. Junge, A. Schreiber
Zentrum Mathematik - M3
Technische Universität München

GITTERFREIE METHODEN - WINTERSEMESTER 2014/15

Übungsblatt zum Workshop 2 – 2015-01-13

A1 (Adaptive Optimale Steuerung des invertierten Pendels)
Aus der Vorlesung kennen Sie das Skript `pendulum.m`, das die Lösung des Optimalsteuerungsproblems für das invertierte Pendel mit Shepards Methode und der Kružkov-Transformation näherungsweise löst. In dieser Aufgabe wollen wir diesen Algorithmus so erweitern, dass adaptiv Zentren in Bereichen hinzugefügt werden, in denen das Residuum groß ist.

a) Argumentieren Sie kurz, warum es nicht sinnvoll ist, weiterhin einen globalen Shapeparameter zu verwenden.

b) Implementieren Sie eine Funktion `function shapevec = getShapevec (X,ep)`, die für eine Zentrenmenge X einen Vektor von Shapeparametern $\rho = \{\rho(x_i)\}$ berechnet, so dass für jedes x_i weitere 10 Zentren im Träger von $\phi_i(x) = \phi(\rho(x_i)(x - x_i))$ liegen (und $\rho(x_i)$ dabei möglichst groß ist).

c) Implementieren Sie die Funktion

```
function v = valueIteration(f,c,phi,shepard,Xa,shapevec,U,ep)
```

um den Code (im Folgenden) übersichtlicher zu halten.

d) Implementieren Sie die Funktion

```
function [Y, h] = refine(X,h)
```

die die Zentren in X "unterteilt", d.h. wie in der Grafik das mittlere Zentrum durch vier neue Zentren ersetzt und dabei auch die Einträge der *Füllmatrix* h (Abstände von den Zentren zu den Rechteckseiten; h hat jeweils die gleiche Dimension wie X) halbiert.

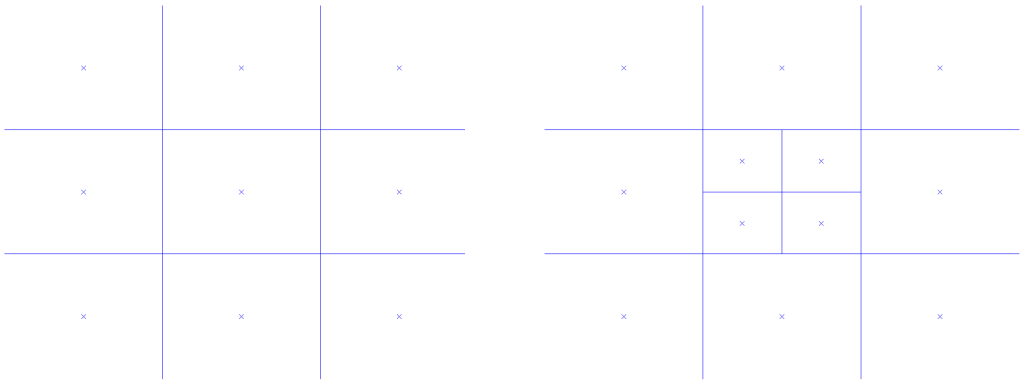

Abbildung 1: Unterteilung: Ein Zentrum links hat großes Residuum und wird daher "unterteilt", also durch vier neue Zentren wie in der Abbildung rechts ersetzt.

e) Implementieren Sie nun eine adaptive Version von `pendulum.m` unter Verwendung der zuvor definierten Hilfsfunktionen. Dabei sollen jeweils Zentren, an denen das (absolute) Residuum mindestens halb so groß wie bei seinem Maximum ist, "unterteilt" werden.

Gegeben ist dazu die Datei `residual.m` mit der Funktion

```
function [e,V] = residual(v,f,c,phi,shepard,Y,X,shapevec,U,ep)
```

die das *Bellman-Residuum* für die Punkte in Y berechnet, wenn die Wertefunktion durch die Koeffizienten v auf X gegeben ist.

Das Bellman-Residuum e ist

$$e(x) := \inf_{u \in U} \{c(x,u) + V(f(x,u))\} - V(x).$$

Gegeben:

- pendulum.m:

```
%% Dynamic programming using radial basis functions
%% an inverted pendulum on a cart
M = 8; m = 2; mr = m/(m+M); l = 0.5; c = 9.8; h = 0.1;
f = @(x,u) kron(x,ones(length(u),1))+h*   ...
    [reshape(ones(length(u),1)*x(:,2)', length(u)*size(x,1),1),   ...
    reshape((c/l*ones(length(u),1)*sin(x(:,1)')   ...
    -1/2*mr*ones(length(u),1)*(x(:,2)'.^2.*sin(2*x(:,1)'))   ...
    -mr/m/l*u*cos(x(:,1)'))./(4/3-mr*ones(length(u),1)*cos(x(:,1)').^2),   ...
    length(u)*size(x,1),1)];
c = @(x,u) exp(-h*(1/2*reshape(0.1*ones(length(u),1)*x(:,1)'.^2 + ...
    0.05*ones(length(u),1)*x(:,2)'.^2 + 0.01*u.^2*ones(1,size(x,1)), ...
    length(u)*size(x,1),1)));
phi = @(r) max(spones(r)-r,0).^4.*(4*r+spones(r));     % Wendland function
phi = @(r) max(spones(r)-r,0).^10.*(429*r.^4+450*r.^3+210*r.^2+50*r+5*spones(r));
T = [0 0]; v_T = 1;                                    % boundary cond.
shepard = @(A) spdiags(1./sum(A')',0,size(A,1),size(A,1))*A; % Shepard op.
S = [8,10];                                            % radius of domain
L = 33; U = linspace(-128,128,L)';                     % control values
N = 50; X1 = linspace(-1,1,N);
[XX,YY] = meshgrid(X1*S(1),X1*S(2));
X = [XX(:) YY(:)];                                    % centers
ep = 1/sqrt((4*prod(S)*20/N^2)/pi);                    % shape parameter
fXU = f(X,U);
C = c(X,U);
A = shepard(phi(ep*sdistm(fXU,[T;X],1/ep)));   % Shepard matrix
%C = exp(-cXU);                                    % one step costs

%% value iteration
v = zeros(N^2+1,1); v0 = ones(size(v)); TOL = 1e-12;
while norm(v-v0,inf)/norm(v,inf) > TOL
    v0 = v;
    v = [v_T; max(reshape(C.*(A*v),L,N^2))'];          % Bellman operator
end

%% plot of the value function
clf; Mr = 500; X1r = linspace(-1,1,Mr);
[XXr,YYr] = meshgrid(X1r*S(1),X1r*S(2)); Xr = [XXr(:) YYr(:)];
R = shepard(phi(ep*sdistm(Xr,[T;X],1/ep)));
surf(XXr,YYr,reshape(min(-log(abs(R*v)),10),Mr,Mr)); axis tight; shading flat;
view(0,90); colorbar;  xlabel('angle'); ylabel('angular velocity');
```

- residual.m:

```
function [e,V] = residual(v,f,c,phi,shepard,Y,X,shapevec,U,ep)

nu = length(U);
ny = length(Y);

% V = value function evaluated on Y
R = shepard(phi(sdistm(Y,X,1/ep)*sparse(diag(shapevec))));
V = -log(R*v+realmin);

% LV = right hand side of the Bellman equation evaluated on Y
A = shepard(phi(sdistm(f(Y,U),X,1/ep)*sparse(diag(shapevec))));
C = c(Y,U);
LV = max(reshape(C.*(A*v),nu,ny));
LV = -log(LV'+realmin);

e = LV-V;                                             % Bellman residual
```