Examen

Nom, prénom :

Durée 2H. Tous les exercices sont indépendants. Le barème prévisionnel est indiqué pour chaque exercice. Documents autorisés.

▷ **Exercice 1** (8 points)**.** On considère le problème à temps final $t_f > 0$ fixé

$$-q^2(t_f) + \int_0^{t_f} u^2(t)\,\mathrm{d}t \to \min$$

pour la dynamique

$$\ddot{q}(t) = u(t), \quad t \in [0, t_f],$$

où $q(t)$ et $u(t)$ sont dans $\mathbf{R}$, et où $q(0) = \dot{q}(0) = 0$ sont fixés. On laisse $q(t_f)$ et $\dot{q}(t_f)$ libres.

1.1. Mettre la dynamique sous la forme $\dot{x}(t) = f(x(t), u(t))$ avec f que l'on précisera.

► $f(x, u) = (x_2, u)$

1.2. Mettre le coût sous forme de Lagrange avec f^0 que l'on précisera. [Nota bene : $(\mathrm{d}/\mathrm{d}t)(q^2) = 2q\dot{q}$]

► $f^0(x, u) = -2x_1x_2 + u^2$

1.3. Donner le hamiltonien du problème. (En l'absence de contrainte terminale, on pourra poser $p^0 = -1/2$.)

► $H(x, p, u) = x_1x_2 - u^2/2 + p_1x_2 + p_2u$

1.4. Déterminer le système adjoint.

► $\dot{p}_1 = -x_2$, $\dot{p}_2 = -x_1 - p_1$

1.5. Écrire les conditions de transversalité.

► $p_1(t_f) = p_2(t_f) = 0$

1.6. Déterminer le contrôle en fonction de l'état et de l'état adjoint à l'aide de la condition de maximisation.

► $u(t) = p_2(t)$

1.7. En déduire le contrôle optimal.

► On voit que $\ddot{p}_2 = -\dot{x}_1 - \dot{p}_1 = -x_2 + x_2 = 0$, d'où l'on déduit que p_2, et donc u, sont des fonctions affines. Comme $p_2(t_f) = 0$, il existe $a \in \mathbf{R}$ tel que $u(t) = a(t_f - t)$.

1.8. Soit $a \in \mathbf{R}$ une constante, calculer le coût associé au contrôle $u(t) = a(t_f - t)$. Que peut-on en conclure ?

► Le coût vaut $(1/3)a^2 t_f^3(1 - t_f^3/3)$, que l'on peut rendre arbitrairement petit en choisissant a suffisamment grand quand $t_f > \sqrt[3]{3}$. On en déduit, dans ce cas, que le problème ne possède pas de solution.

▷ **Exercice 2** (6 points). On considère le problème à temps final $t_f > 0$ fixé

$$\int_0^{t_f} (x_2^2(t) + u_1^2(t) + u_2^2(t))\,\mathrm{d}t \to \min$$

pour la dynamique

$$\dot{x}_1(t) = -x_2(t) + u_2(t), \quad \dot{x}_2(t) = x_1(t) - u_1(t), \quad t \in [0, t_f],$$

où $x(t)$ et $u(t)$ sont dans $\mathbf{R}^2$, et où $x(0) = x_0$ est fixé. On laisse $x(t_f)$ libre.

2.1. Donner le hamiltonien du problème. (En l'absence de contrainte terminale, on pourra poser $p^0 = -1/2$.)

► $H(x, p, u) = (-1/2)(x_2^2 + u_1^2 + u_2^2) + p_1(-x_2 + u_2) + p_2(x_1 - u_1)$

2.2. Déterminer le système adjoint.

► $\dot{p}_1 = -p_2$, $\dot{p}_2 = x_2 + p_1$

2.3. Écrire les conditions de transversalité.

► $p_1(t_f) = p_2(t_f) = 0$

2.4. Déterminer le contrôle en fonction de l'état et de l'état adjoint à l'aide de la condition de maximisation.

► $u(t) = (-p_2(t), p_1(t))$

2.5. Injecter ce contrôle dans le hamiltonien pour déterminer le hamiltonien maximisé (fonction de l'état et de l'état adjoint seulement) et compléter le code ci-dessous.

```
# Hamiltonian
function h(t, x, p)
    r = 0.0 # **** TO BE UPDATED ****
    return r
end

# Makes flow from Hamiltonian
f = Flow(h)

# Shooting function
function shoot(p0)
    xf, pf = f(t0, x0, p0, tf)
    s = zeros(2) # **** TO BE UPDATED ****
    return s
end
```

►

```
    r = 0.5*(p[1]^2+p[2]^2-x[2]^2)-p[1]*x[2]+p[2]*x[1] # in h
    ...
    s = pf # in shoot
```

▷ **Exercice 3** (6 points).

3.1. Dans le code MPC ci-dessous, entourer la ou les expressions qui simulent le système réel.

```
while true

    w = drift(x1, y1) # current sampled at (x1,y1)
    us, τs = solve(x1, y1, θ1, xf, yf, θf, w, P, print_level=0)
    ts = [ t1+τs[1], t1+τs[1]+τs[2] ]
    tf = t1+τs[1]+τs[2]+τs[3]
    if (t1+Δt < tf)
        t2 = t1+Δt
    else
        t2 = tf
        println("t2=tf: ", t2)
    end
    sol = trajectory((t1, t2), x1, y1, θ1, us, ts, drift)
    t = [ t ; sol.t ]
    x = [ x ; sol[1, :] ]
    y = [ y ; sol[2, :] ]
    θ = [ θ ; sol[3, :] ]
```

```
    uu = zeros(length(sol.t))
    uu[ findall(sol.t .< ts[1]) ] .= us[1]
    uu[ findall(ts[1] .<= sol.t .< ts[2]) ] .= us[2]
    uu[ findall(sol.t .>= ts[2]) ] .= us[3]
    u = [ u ; uu ]

    t1 = t2
    x1 = x[end]
    y1 = y[end]
    θ1 = θ[end]
    err = norm([ x1, y1, θ1 ] - [ xf, yf, θf ])
    println("N: ", N, "\t err: ", err)
    ((err > ε) && (N <= Nmax)) || break
    N = N+1

end
```

3.2. Dans ce même code, expliquer le rôle de la ligne
`uu[ findall(ts[1] .<= sol.t .< ts[2]) ] .= us[2]`

► Pour les indices qui correspondent aux temps entre les deux switchs (deuxième arc), on met à jour la valeur du contrôle avec la valeur déterminée pour cet arc par `solve`.

3.3. Dans le code Hexapawn ci-dessous, entourer la ou les expressions qui détectent l'absence de coup possible pour la machine.

```
% Move 1: player2
u1 = play2(X1, lX1, lu1);
X2 = f2(X1, u1); % no possible win after u1
if dsp, disp('Player 2 move:'); disp(X2); end;

% Move 2: player1
u2 = play1(X2, inter);
X3 = f1(X2, u2);
if dsp, disp('Player 1 move:'); disp(X3); end;

if win1(X3) || isempty(play2(X3, lX3, lu3))
  winner = 1;
  lu1 = reinforce(X1, u1, lX1, lu1);
```

3.4. On considère une partie d'Hexapawn pendant laquelle la machine vient de jouer le coup ci-dessous :

```
[ 0 2 0        [ 0 0 0
```

```
  2 1 1   ---->   2 1 2
  0 0 0 ]         0 0 0 ]
```

La liste de coups de la machine associée à l'état précédent (avant son dernier coup) est

```
{ [ 2 1   , [ 1 2
    3 1 ]     2 3 ] }
```

Comment cette liste doit-elle être mise à jour par renforcement ?

► La machine perd immédiatement puisque le seul contrôle admissible pour son adversaire,

```
[ 2 2
  1 2 ]
```

le fait gagner. Le dernier coup joué par la machine est donc supprimé par le renforcement de sorte que la liste de coups devient

```
{ [ 2 1
    3 1 ] }
```

qui, de fait, conduit à une victoire de la machine.