Appendix A

Appendix essay I

A.1 Sosialt optimum

Vi skal her vise at $PS + CS + GS = \pi(a, y) - D(a)$. (Se fig. 2.1). Vi har

$$PS = y^\star \cdot C_p(a, y^\star) - \int_0^{y^\star} C_p(a, x)\, dx. \tag{A.1}$$

$$CS = \int_0^{y^\star} P(a, x)\, dx - y^\star\big(C_p(a, y^\star) + \tau(a, y^\star)\big) \tag{A.2}$$

$$GS = y^\star\tau - D(a) - \int_0^{y^\star} \big(C_s(a, x) - C_p(a, x)\big)\, dx. \tag{A.3}$$

Dermed får vi

$$\begin{aligned} PS + CS + GS &= y^\star \cdot C_p(a, y^\star) - \int_0^{y^\star} C_p(a, x)\, dx \\ &\quad + \int_0^{y^\star} P(a, x)\, dx - y^\star\big(C_p(a, y) + \tau(a, y^\star)\big) \\ &\quad + y^\star\tau - D(a) - \int_0^{y^\star} \big(C_s(a, x) - C_p(a, x)\big)\, dx \\ &= \int_0^{y^\star} \big(P(a, x) - C_s(a, x)\big)\, dx - D(a) \\ &= \pi(a, y^\star) - D(a). \end{aligned} \tag{A.4}$$

A.2 Eksistensbevis

Vi skal her vise at det alltid eksisterer løsning for optimeringsproblemet definert i (2.6) og (2.7). I den sammenheng begrenser vi oss til å vise at betingelsene i *Filipov-Cesaris eksistensteorem*, som vi finner som teorem 6.18 i Seierstad og Sydsæter (1987), er oppfylte.

Kravet om at mengden $N(a, y, t)$ er konveks, utgjør sammen med kravet om en øvre grense for tilstandsvariablene, de eneste ikke-trivielle betingelser stilt i dette teoremet.

Når det gjelder det siste kravet har vi at $\dot{a} + \dot{s} = y - f(a) + (-y) = -f(a) \leq 0$, slik at $0 \leq a + s \leq a_0 + s_0$. Følgelig er dette tilfredsstilt.

Teoremet forutsetter at mengden

$$N(a, y, t) = \left\{(e^{-rt}[\pi(a, y) - D(a)] + \gamma, y - f(a), -y) : \gamma \leq 0, y \in [0, \hat{y}]\right\} \tag{A.5}$$

er konveks for alle $(a, t) \in R \times [0, T]$. (I (A.5) er $\hat{y}$ gitt ved $\pi_y(a, \hat{y}) = 0$).

Bevis: Fikser (a, t), la $z_i \equiv e^{-rt}[\pi(a, y_i) - D(a)] + \gamma$ for $i = 1, 2$, og la y_3, z_3 være en konveks kombinasjon av y_1, y_2 og z_1, z_2.

Vi får $\lambda(z_1, y_1 - f(a), -y_1) + (1 - \lambda)(z_2, y_2 - f(a), -y_2) = (\lambda z_1 + (1 - \lambda)z_2, \lambda y_1 + (1 - \lambda)y_2 - f(a), -\lambda y_1 - (1 - \lambda)y_2) = (z_3, y_3 - f(a), -y_3)$. Konkaviteten til $\pi(a, y)$ gir $z_3 = e^{-rt}[\lambda\pi(a, y_1) + (1 - \lambda)\pi(a, y_2) - D(a)] + \lambda\gamma_1 + (1 - \lambda)\gamma_2 \leq e^{-rt}[\pi(a, y_3) - D(a)] + \lambda\gamma_1 + (1 - \lambda)\gamma_2 \Rightarrow \gamma_3 \leq \lambda\gamma_1 + (1 - \lambda)\gamma_2 \leq 0 \quad \Rightarrow \quad (z_3, y_3 - f(a), -y_3) \in N(a, y, T)$.

A.3 Formulering av modell

I det følgende skisserer vi først kort i seksjon A.3.1 hvordan vi kommer frem til en differensialligning som gir en feedback-form for den optimale kontrollen, $y = y(a)$. Deretter gjør vi i seksjon A.3.2 rede for hvordan vi benytter oss av denne differensialligningen til å finne en numerisk løsning av modellen.

Optimeringsproblemet i ligningene (2.6) og (2.7) kan formuleres slik at det strekker seg over to perioder - før og etter teknologiskiftet. Kontrollvariabelen er uttaket, y, og vi har et tilpasningsproblem i tidspunktet T.

Hamiltonfunksjonen med tilstandsligninger, multiplikatorligninger og det indre optimalitetskravet er oppsummert i tab. A.1.

Beskrivelse	Første periode	Sluttperiode
Tid	$0 \leq t \leq T$	$T < t \leq \infty$
produksjon	$y > 0, \quad y(T) = y_T > 0$	$y = 0$
Sosial velferd	$\pi(a, y) - D(a)$	$\widehat{\pi}(t) - D(a)$
Tilstandligning 1	$\dot{a} = y - f(a)$	$\dot{a} = -f(a)$
Tilstandsligning 2	$\dot{s} = -y$	
Hamilton	$\mathcal{H} = \pi(a, y) - D(a) + m(y - f(a)) - ny$	$\mathcal{H} = \widehat{\pi}(t) - D(a) - mf(a)$
Indre optimum	$\mathcal{H}_y = 0 \Leftrightarrow m - n = -\pi_y(a, y)$	
Kofaktor 1	$\dot{m} = (r + f'(a))m - \pi_a(a, y) + D'(a)$	$\dot{m} = (r + f'(a))m + D'(a)$
Kofaktor 2	$\dot{n} = rn$	$\dot{n} = rn, ns = 0$ og $n, s \geq 0$

Table A.1: Førsteordens betingelser.

Vi krever at denne Hamilton-funksjonen, tilstandsvariablene og multiplikatorene er kontinuerlige i $t = T$.

Kofaktor 2: Løser vi ligningen for kofaktor 2 i tab. A.1, får vi

$$n(t) = n_0 e^{rt}, \quad \text{der} \quad n_0 = n(0). \tag{A.6}$$

Dette kombinerer vi med (2.13) og får to muligheter for skyggeprisen $n(t)$:
Tilfelle 1: $s(T) > 0$ og $n(t) = 0$ for alle t.
Tilfelle 2: $s(T) = 0$ og $n(t) = n_0 e^{rt}$.

A.3.1 Feedback-formulering

I det følgende skal vi skal vi uten å gå inn på detaljer skissere hvordan vi utleder en 1. ordens differensialligning for feedback-kurven til y for tilfelle 1 og tilfelle 2. For begge tilfellene vil den dynamiske egenskapen

$$\frac{d}{dt}\mathcal{H} = r(\mathcal{H}_a \dot{a} + \mathcal{H}_s \dot{s}) \tag{A.7}$$

bli brukt. (En utledning av dette resultatet finnes blant annet i Sandal og Berge (2000)). Videre setter vi kravet om indre løsning, se tab. A.1, inn i Hamilton-funksjonen. Dette gir

$$\mathcal{H}(a, y) = \pi(a, y) - D(a) - y\pi_y - mf(a). \tag{A.8}$$

Tilfelle 1: $s(t) > 0$ og $n(t) = 0$ for alle t. Det at $n = 0$ forenkler ligningssystemet (A.7) og (A.8). Vi kommer frem til differensialligningen

$$y'(a)(y - f)\pi_{yy} = \pi_a - D' - (y - f)\pi_{ay} + (r + f')\pi_y \tag{A.9}$$

ved å løse dette systemet. Dette er vist i Aanestad, Sandal og Berge (2003).

Tilfelle 2: $s(T) = 0$ og $n(t) = n_0 e^{rt}$. I dette tilfellet blir feedback-formen mer komplisert. I tillegg til (A.7) og (A.8) må vi trekke inn differensialligningen for kofaktor m. (Se tab. A.1). Da kommer vi etter noen tekniske detaljer frem til en feedback-ligning. Istedenfor å presentere denne, presenterer vi den tidseksplisitte versjon som er noe enklere rent notasjonsmessig. Vi har

$$\frac{dY}{dt} = \begin{bmatrix} \dot{a} \\ \dot{y} \\ \dot{m} \end{bmatrix} = \begin{bmatrix} y - f(a) \\ G_1(a, y, m)(y - f(a)) \\ (r + f'(a))m + D'(a) \end{bmatrix}, \tag{A.10}$$

med

$$G_1(a, y, m) \equiv y'(a) = \frac{\pi_a - D' - y\pi_a y - mf)(y - f) + r(y\pi_y + mf)}{y\pi_y y(y - f)}.$$

Detaljene for hvordan vi kommer frem til denne ligningen er presentert i Aanestad, Sandal og Berge (2003).

A.3.2 To randverdiproblem

I seksjon A.3.1 fant vi differensialligninger for optimale løsningskurver for hvert av de to spesialtilfellene av optimeringsproblemet vårt. Nå skal vi finne tilhørende randbetingelser tilstrekkelige for å finne løsningskurvene for hvert av tilfellene.

Fremgangsmåten blir i praksis at man først antar at der ikke er noen ressursknapphet, altså tilfelle 1. Man finner tilhørende optimale løsningskurver for dette tilfellet og kontrollerer at uttaket gir $s(T) \geq 0$. Deretter søker man en løsning hvor ressursen brukes helt opp. (Tilfelle 2). Hvis der eksisterer potensielle løsninger for begge de to tilfeller, må man numerisk sammenligne overskuddet knyttet til løsningene.

Tilfelle 1: $s_T > 0$ og $n(t) = 0$ for alle t. I dette tilfellet er randkravene initialbetingelsen $a(0) = a_0$, samt overgangsbetingelsene i $t = T$. Siden $n(t) = 0$, er Hamiltonfunksjonen vår $\mathcal{H} = \pi(a, y) - D(a) + m(y - f(a))$.

Ligningen for kofaktor 1 sammen med betingelsen $\lim_{t\to\infty} a(t) = 0$, det indre optimalitetskravet og kravet om kontinuerlig hamilton-funksjon gir oss følgende betingelser i teknologiskiftetidspunktet: (Se Aanestad, Sandal og Berge (2003) for en utledning av disse betingelsene).[1]

$$m_T = m(\infty) + \int_0^{a_T} \frac{dm}{da}\, da \equiv G(a_T), \quad t \geq T \tag{A.11}$$

$$\mathcal{H}_y = 0 \Rightarrow \pi_y(a_T, y_T) + m_T = 0 \tag{A.12}$$

$$\widehat{\pi}(T) + \gamma(a_T) y_T^2 = 0. \tag{A.13}$$

Størrelsene som skal bestemmes er T, a_T, y_T og m_T. I tillegg til de tre overgangsbetingelsene i teknologiskiftetidspunktet, (A.11)-(A.13), krever vi at initialbetingelsen er oppfylt. Vi har dermed fire randkrav og fire ukjente størrelser.

Siden vi har eksplisitt tidsavhengighet i den alternative profittfunksjonen, $\widehat{\pi}(t)$, lønner det seg å bruke eksplisitt tidsavhengighet også når vi skal finne de optimale løsningskurvene for y og a. Vi har

$$\frac{d\mathbf{Y}}{dt} = \begin{bmatrix} \dot{a} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} (y - f(a)) \\ y'(a)\big(y - f(a)\big) \end{bmatrix}, \tag{A.14}$$

der $y'(a)$ er gitt ved (A.9) og vi har randkravene (A.11) - (A.13) samt $a(0) = a_0$.

Det viser seg at $\frac{d\mathbf{Y}}{dt}$ er numerisk ustabil når man integrerer fra $t = 0$ til $t = T$. Går man motsatt vei fra $t = T$ til $t = 0$ blir systemet stabilt.

Siden man har randkrav både i $t = 0$ og $t = T$, er det ikke et trivielt problem å løse (A.14) med tilhørende randkrav. Vi ble nødt for å utvikle en egen shooting-metode som kunne håndtere dette randverdiproblemet numerisk.

[1] Vi har i det følgende brukt $a(T) = a_T$, $y(T) = y_T$, $s(T) = s_T$ og $m(T) = m_T$.

Tilfelle 2: $n(t) = n_0 e^{rt}$ og $s(T) = 0$. Vi bruker følgende randkrav:

$$\begin{aligned} &a(0) - a_0 = 0 \\ &\widehat{\pi}(T) + \gamma(a_T) y_T^2 = 0 \\ &m_T - M(a_T) = 0 \\ &s(T) = 0. \end{aligned} \tag{A.15}$$

Dessuten innfører vi parameteren $\Omega(t)$ med initialkrav $\Omega(0) = 0$. Denne betegner samlet neddiskontert overskudd ved tidspunkt t, og er tatt med fordi differensialligningene våre med tilhørende randkrav i noen tilfeller kan ha flere løsninger slik at vi må sammenligne dem numerisk for å skille ut hvilken som er den beste. Antall mulige løsninger vil avhenge av a-avhengigheten i γ. (Se andre ligning i (A.15)). Dette vil vi imidlertid ikke komme noe mer inn på her.

I tillegg til de nevnte randkravene har vi $m_T - n_T = -\pi_y(a_T, y_T)$, men dette kravet bidrar ikke med noe nytt siden det innfører en ny ukjent, nemlig n_T.

Differensialligningssytem (A.10) utvider vi med uttrykk for $\dot{s}$ og $\dot{\Omega}$. Dermed får vi:

$$\frac{dY}{dt} = \begin{bmatrix} \dot{a} \\ \dot{y} \\ \dot{m} \\ \dot{s} \\ \dot{\Omega} \end{bmatrix} = \begin{bmatrix} y - f(a) \\ G_1(a, y, m)(y - f(a)) \\ (r + f'(a))m + D'(a) \\ -y \\ e^{-rt}\big(\pi(a, y) - D(a)\big) \end{bmatrix}, \tag{A.16}$$

Vi ser at vi har fem randkrav og fem differensialligninger. Dette randverdiproblemet er like komplisert som tilsvarende problem under *tilfelle 1*. Vi brukte en såkalt kollokasjonsmetode for å løse problemet numerisk.

Appendix B

Appendix essay II

This appendix displays the numerical solver used to solve example 1 in essay II. The solver consists of a matlab-file that is runned with the command *toDimEks2(dx,dy,delta,feil_tol,iterasjoner),* where *dx, dy, ...* etc. are numerical values.

```
function [X0,Y0,V0,u0,feil_u,feil_V]= ...
    toDimEks2(dx,dy,delta,feil_tol,iterasjoner);

%Solves the 2D-model in ex. 5.3 of Grüne and Semmler (2004)
tic %Cmd that registers the time the prgm starts
epsilon=1e-10; %Used to avoid problems with 1/0.

h=1/200; beta=1-delta.*h;
xmin=0;ymin=0; xmax=7;ymax=2;
X=xmin:dx:xmax; Y=ymin:dy:ymax;
DX=[dx diff(X)]; DY=[dy diff(Y)];
mx=length(X); my=length(Y)

[X0,Y0]=meshgrid(X,Y);

% Initial conditions for the value function V0,
% the politics u0 and so on ...
V0=zeros(my,mx); %The value funct. along the state space grid
V_I=V0; Vxder=zeros(my,mx); Vyder=zeros(my,mx);
u0=zeros(my,mx);
x_ny=X0+fx(X0,Y0).*h;  y_ny=Y0+fy(u0);
feil_u=1;feil_V=1;feil=1; % Initiate 3 error terms

V_I=V0; Vxder=zeros(my,mx); Vyder=zeros(my,mx);
Vzder=Vyder; u0=zeros(my,mx);

x_ny=min(xmax,X0+fx(X0,Y0).*h); %Demand a closed state-space
```

```
y_ny=Y0+fy(u0).*h;

% A little procedure to make use of previous estimates
% (if such exist). These might be used as a
% better initial condition
try
    load toDim; %Download toDim.mat
    vv=size(Vgammel);
    if size(V0)==size(Vgammel)
        V0=Vgammel;
    else
        V0=interp2(Xgammel,Ygammel,Vgammel,X0,Y0);
    end
end

% Estimate partial derivatives of the value function in x
% and y-direction and make use of them to find starting
% values for the optimal controls.

[Vxder,Vyder]=gradient(V0); %Gradients (x and y-direction)
Vxder=Vxder./ (ones(my,1)*DX);
Vyder=Vyder./(DY'*ones(1,mx));

% Get u-values from sub-function "fangst"
u0=fangst(X0,Y0,Vxder,Vyder,beta,h);

% Find new x- and y-values corresponding (to u0)
x_ny=min(xmax,X0+fx(X0,Y0).*h);
y_ny=Y0+fy(u0).*h;

% Interpolate V-value outside Gx for x1=x_ny and x2=y_ny
V_I=interp2(X0,Y0,V0,x_ny,y_ny,'linear');
% The first policy it. is finished

% A combination of policy and value iterations until
% convergence or till maximum numb. of it. is exceeded
feil_old=1; V0_old=V0; it=1;
while  it<iterasjoner && feil>feil_tol
   it=it+1;
```

```
    if mod(it,20)==0  %A procedure to display number of it.
        disp('iterasjoner');
        disp(it);
        disp(feil_u);
        disp(feil_V);
    end

    % Value it: Fixed point it. on the opt. value func.
    [V0,feil_V]=verdiIt(X0,Y0,V0,x_ny,y_ny,u0, ...
        V_I,beta,h,feil_tol,feil_V,epsilon);

    %policy iterations
   [u0,V0,V_I,x_ny,y_ny,feil_u,feil_V]=policyit(X0,Y0,u0, ...
       V0,feil_tol,feil_u, beta,h,mx,my,DX,DY,epsilon);

    feil=max(feil_u,feil_V); % Update size of error
end

%_____________________________________________________________%
% Have reached a solution. Save it for later use and
% display errors and time to convergence.
disp(['iterasjoner: ' 'rel_feil_u: ' 'rel_feil_V:' 'tid:']);
disp([it feil_u feil_V toc]);
Vgammel=V0; Xgammel=X0; Ygammel=Y0;
x=X0; y=Y0;u=u0;

save toDim Xgammel Ygammel Vgammel;

% Select a cruder grid to be plotted
x=0:2.5*dx:6;  y=0:2*dy:1.6;
[x,y]=meshgrid(x,y);
V=interp2(X0,Y0,V0,x,y,'linear'); %Interpol on new grid
u=interp2(X0,Y0,u0,x,y,'linear');

%% Figures
figure(1) % The value function V(x,y)
box off; legend('boxoff')
surf(x,y,V);
set(gca,'FontSize',14,'FontWeight','bold')
```

```
xlabel('x_1'); ylabel('x_2'); zlabel('V(x_1,x_2)')

figure(2) % Optimal policy
surf(x,y,u);
box off; legend('boxoff')
set(gca,'FontSize',14,'FontWeight','bold');
xlabel('x_1'); ylabel('x_2'); zlabel('u(x_1,x_2)');

%% All sub-functions
%%- State eq. 1
function tilst1=fx(x,y)
sigma=0.25;
tilst1=y-sigma.*x;

%%-State eq. 2
function tilst2=fy(u)
tilst2=u;

%%- Utility/profit function
function out=g(x,y,u)
k1=2;k2=0.0117;c1=0.75;c2=2.5;alfa=12;
out=k1.*x.^0.5-x./(1+k2.*x.^4)-c1.*y-c2.*y.^2./2-alfa.*u.^2./2;

%%- Function for optimal control, u
function u=fangst(x,y,Vxder,Vyder,beta,h) %
alfa=12;
u=beta.*Vyder./alfa; % F.o.c. (inner solution)
u=max(-1,u);
u=max(u,-y./h);
u=min(1,u);

%%- Value iterations
function [V0,feil_V]= ...
    verdiIt(X0,Y0,V0,x_ny,y_ny,u0,V_I,beta,h,feil_tol, ...
    feil_V_old,epsilon)

    fiksPi=g(X0,Y0,u0); %Utility function
    feil_V=feil_V_old; V0_old=V0;it=1;
```

```
    % Use g(x,u_fixed) to est. val. func. with fixed pnt. it
    V0=h.*fiksPi+beta.*V_I;
    % Improve old V_I
    V_I=interp2(X0,Y0,V0,x_ny,y_ny,'linear');

    while feil_V>feil_tol && it<45

        % Use g(x,u_fixed) to estimate value func.
        V0=max(0,h.*fiksPi+beta.*V_I);
        V_I=interp2(X0,Y0,V0,x_ny,y_ny,'linear');

        %Rel. change val. func.
        feil_V=max(max(abs(V0_old-V0)./(V0+epsilon)));
        V0_old=V0;
        it=it+1;
    end

%%- Policy-iterations
function [u0,V0,V_I,x_ny,y_ny,feil_u,feil_V]= ...
    policyit(X0,Y0,u0,V0,feil_tol,feil_old,beta,h, ...
    mx,my,DX,DY,epsilon)

    xmax=6;feil_u=1;feil_V=1;it=0;

    while  (feil_u>feil_tol || it<1) && it<6
        it=it+1;
        u_0=u0; V_0=V0;

        [Vxder,Vyder]=gradient(V0);
        Vxder=Vxder./(ones(my,1)*DX);
        Vyder=Vyder./(DY'*ones(1,mx));
        u0=fangst(X0,Y0,Vxder,Vyder,beta,h);
        x_ny=min(xmax,X0+fx(X0,Y0).*h);
        y_ny=Y0+fy(u0).*h;

        % Est. val. func. for new x-vals.
        V_I=interp2(X0,Y0,V0,x_ny,y_ny,'linear');

        % Updts val. funct. for old x-vals.
```

```
        V0=max(0,h.*g(X0,Y0,u0)+beta.*V_I);

        % Rel. change u
        feil_u=max(max(max(abs(u_0-u0)./(u_0+1e-14))));
        % Rel. change of V
        feil_V=max(max(abs(V_0-V0)./(V_0+epsilon)));
    end
```