```
%Problema de optimização dinâmica - Crescimento óptimo
%Modelo de Ramsey-Cass-Koopmans em equilíbrio centralizado com
%horizonte infinito.

function ramseymain

clear all
close all
clc

global rho delta alpha sigma gx n guessbvp NMax RelTol AbsTol kss k0 c0...
    kval cval c_dk_dt_0 kg t c k lambda

%Parâmetros base
gx = 0.02;          % Taxa de crescimento do progresso técnico A (aqui considera-s
n = 0.04;           % Taxa de crescimento da oferta de trabalho L
delta =0.2;         % Taxa de depreciação do capital físico k
alpha = 1/3;        % Fracção de k no produto Y

rho = 0.06;         % Taxa de desconto (factor de preferência pelo tempo)
sigma = 1/alpha;    % Elasticidade intertemporal do consumo


kss = (alpha/(delta+rho+gx/sigma))^(1/(1-alpha));       % k de steady-state
css = kss^alpha-(delta+n+gx)*kss;                       % c de steady-state
t0 = 0; tf = 50;                                        % t(0) e t(T)
k0 = kss*0.15;                                           % k(0) = k0 > 0
c0 = k0^alpha-(n+delta+gx)*k0;                          % c(0)= c0


%Escolher o método para resolver o problema de optimização dinâmica
ButtonName2 =questdlg...
('Modelo de Ramsey-Cass-Koopmans - Método Directo... (GPOPS) vs Indirecto (BVP5C)'
                        'Modelo de Ramsey','GPOPS', 'BVP5C','Cancelar','Cancelar'

if strcmp(ButtonName2,'GPOPS')==1


    %Método Directo - GPOPS
    iphase = 1;


    limits(iphase).time.min = [t0 tf];
    limits(iphase).time.max = [t0 tf];
    % Exclusão de trajectória explosiva que viola eq. de Euler k(T) = 0 dc/dt = +i
    limits(iphase).state.min = [k0 0 kss];
    limits(iphase).state.max = [k0 100 100];
    limits(iphase).control.min    =  0;
    limits(iphase).control.max    =  100;
    limits(iphase).parameter.min  = [];
    limits(iphase).parameter.max  = [];
    limits(iphase).path.min       = [];
```

```
limits(iphase).path.max       = [];
limits(iphase).event.min      = [];
limits(iphase).event.max      = [];
limits(iphase).duration.min   = [];
limits(iphase).duration.max   = [];

guess(iphase).time            = [t0; tf];
guess(iphase).state           = [k0; kss];
guess(iphase).control         = [c0; css];
guess(iphase).parameter       = [];

setup.name  = 'ramseymain-Problem';
setup.funcs.cost = 'ramseygpopsCost';
setup.funcs.dae = 'ramseygpopsDae';
setup.limits = limits;
setup.guess = guess;
setup.linkages = [];
setup.derivatives = 'finite-difference';
setup.autoscale = 'off';
setup.mesh.tolerance = 1e-3;
setup.mesh.iteration = 20;
setup.mesh.nodesPerInterval.min = 4;
setup.mesh.nodesPerInterval.max = 12;

[output] = gpops(setup);
solution = output.solution;
solutionPlot = output.solutionPlot;
assignin('base','solution',solution);
assignin('base','solutionPlot',solutionPlot);

t=solution.time;
k=solution.state;
c=solution.control;
lambda=solution.costate;

if sigma==1/alpha %Utilizar seguinte solução fechada:

    solucaoanalitica;

    plotfigures;

    figure(1);
    hold on;
    plot(t,ksol,'r','LineWidth',1.5);
    legend('aproximado','solução');
    hold off;

    figure(2);
    hold on,
    plot(t,csol,'r','LineWidth',1.5);
    legend('aproximado','solução');
    hold off;

    figure(4);
```

```
            hold on;
            if min(c_dk_dt_0)< min(c)
                    line([kss kss],[min(c_dk_dt_0) max(c_dk_dt_0)*1.2],[1 1],...
                        'Color','g','LineWidth',1.5) % Linha dc/dt = 0
            else
                    line([kss kss],[min(c) max(c_dk_dt_0)*1.2],[1 1],...
                        'Color','g','LineWidth',1.5)
            end



        else  %Utilizar linearização -> Nota: Só é fiável em pontos próximos do equilí

              f = warndlg...
              ('Impossível representar solução verdadeira. Só disponível quando sigma=

                    waitfor(f);

            kval = linspace(min(k),max(k)*1.1,15);
            cval = linspace(min(c),max(c)*1.,15);

            kg = linspace(min(k),max(k)*1.1,200);
            c_dk_dt_0 = kg.^alpha-(delta+n+gx).*kg; % dk/dt=0 => c(t)=(...)

            plotfigures;
            figure(4);
            hold on;
                if min(c_dk_dt_0)< min(c)
                    line([kss kss],[min(c_dk_dt_0) max(c_dk_dt_0)*1.2],...
                        [1 1],'Color','g','LineWidth',1.5) % Linha dc/dt = 0
                else
                    line([kss kss],[min(c) max(c_dk_dt_0)*1.2],...
                        [1 1],'Color','g','LineWidth',1.5)
                end


        % sigma=1/sigma; Solução linearizada -> implementar!!!
        %
        %    ksol = (1./((delta+n+gx).*1./sigma)+(k0.^(1-alpha)-...
        %   1./((delta+n+gx).*1./sigma))*exp(-(1-alpha).*(delta+n+gx).*t)).^(1./(1-alp
        %     csol =(1-sigma).*ksol.^alpha;
        end



    elseif strcmp(ButtonName2,'BVP5C')== 1

        %Método Indirecto - bvp5c
        NMax = 50;
        RelTol = 1e-6;
        AbsTol = 1e-3;
        guessbvp = [k0 c0];
        options = bvpset('FJacobian',@J,'RelTol',RelTol,...
```

```
    'AbsTol',AbsTol','NMax',NMax);
solinit = bvpinit(linspace(t0,tf,10),guessbvp);
sol = bvp5c(@eqdinamicas,@bcfun,solinit,options);

t = linspace(t0,tf);
y = deval(sol,t);
k = y(1,:)';
c = y(2,:)';
assignin('base','t',t);
assignin('base','c',c);
assignin('base','k',k);
assignin('base','kss',kss);
assignin('base','css',css);
if sigma==1/alpha %Utilizar seguinte solução fechada:

    solucaoanalitica;
    plotfigures;

    figure(1);
    hold on;
    plot(t,ksol,'r','LineWidth',1.5);
    legend('aproximado','solução');
    hold off;

    figure(2);
    hold on,
    plot(t,csol,'r','LineWidth',1.5);
    legend('aproximado','solução');
    hold off;

    figure(4);
    hold on;

    if min(c_dk_dt_0)< min(c)
            line([kss kss],[min(c_dk_dt_0) max(c_dk_dt_0)*1.2],[1 1],...
                'Color','g','LineWidth',1.5) % Linha dc/dt = 0
    else
            line([kss kss],[min(c) max(c_dk_dt_0)*1.2],[1 1],...
                'Color','g','LineWidth',1.5)
    end

else  %Utilizar linearização -> Nota: Só é fiável em pontos próximos do equilí

      f = warndlg...
      ('Impossível representar solução verdadeira. Só disponível quando sigma=

            waitfor(f);
kval = linspace(0.1,max(k)*1.2,15);
% Valor inicial diferente mas próximo de zero para diagrama de fase
cval = linspace(0.1,max(c)*1.2,15);

kg = linspace(0,max(k)*1.2,200);
c_dk_dt_0 = kg.^alpha-(delta+n+gx).*kg; % dk/dt=0 => c(t)=(...)
```

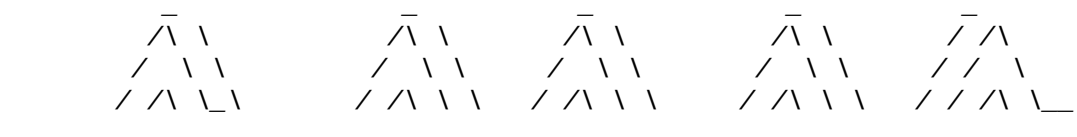
```
        plotfigures;
        figure(4);
        hold on;
            if min(c_dk_dt_0)< min(c)
                line([kss kss],[min(c_dk_dt_0) max(c_dk_dt_0)*1.2],...
                    [1 1],'Color','g','LineWidth',1.5) % Linha dc/dt = 0
            else
                line([kss kss],[min(c) max(c_dk_dt_0)*1.2],...
                    [1 1],'Color','g','LineWidth',1.5)
            end

    end



else

    return  % Cancelar programa

end

% Adicionar as variáveis ao "Workspace"
assignin('base','t',t);
assignin('base','c',c);
assignin('base','k',k);
assignin('base','kss',kss);
assignin('base','css',css);


function plotfigures


figure(1);
pp = plot(t,k,'-o');
set(pp,'LineWidth',1.5);
set(gca,'FontName','Times','FontSize',16);
xl = xlabel('$t$','Interpreter','latex');
yl = ylabel('$k(t)$','Interpreter','latex');
set(xl,'FontSize',18);
set(yl,'FontSize',18);
grid on;

figure(2);
pp = plot(t,c,'-o');
set(pp,'LineWidth',1.5);
set(gca,'FontName','Times','FontSize',16);
xl = xlabel('$t$','Interpreter','latex');
yl = ylabel('$c(t)$','Interpreter','latex');
set(xl,'FontSize',18);
set(yl,'FontSize',18);
grid on;


if isempty(lambda)
```

```
    figure(4);
    pp = plot(k,c);
    set(pp,'LineWidth',1.5);
    set(gca,'FontName','Times','FontSize',16);
    xl = xlabel('$k(t)$','Interpreter','latex');
    yl = ylabel('$c(t)$','Interpreter','latex');
    set(xl,'FontSize',18);
    set(yl,'FontSize',18);
    grid on;
    hold on;
    ypv = [gradient(k) gradient(c)];
    u = ypv(:,1);
    v = ypv(:,2);
    %Representação gráfica; Desenha setas nos vectores de velocidade
    quiver(k,c,u,v,'b','LineWidth',1.5);
    axis tight;
    ppp = plot(kg,c_dk_dt_0,'k');
    set(ppp,'LineWidth',1.5)
    campovectorial(@eqdinamicas,kval,cval);

else

    figure(3);
    pp = plot(t,lambda,'-o');
    set(pp,'LineWidth',1.5);
    set(gca,'FontName','Times','FontSize',16);
    xl = xlabel('$t$','Interpreter','latex');
    yl = ylabel('$\lambda(t)$','Interpreter','latex');
    set(xl,'FontSize',18);
    set(yl,'FontSize',18);
    grid on;

    figure(4);
    pp = plot(k,c);
    set(pp,'LineWidth',1.5);
    set(gca,'FontName','Times','FontSize',16);
    xl = xlabel('$k(t)$','Interpreter','latex');
    yl = ylabel('$c(t)$','Interpreter','latex');
    set(xl,'FontSize',18);
    set(yl,'FontSize',18);
    grid on;
    hold on;
    ypv = [gradient(k) gradient(c)];
    u = ypv(:,1);
    v = ypv(:,2);
    %Representação gráfica; Desenha setas nos vectores de velocidade
    quiver(k,c,u,v,'b','LineWidth',1.5);
    axis tight;
    ppp = plot(kg,c_dk_dt_0,'k');
    set(ppp,'LineWidth',1.5)
    campovectorial(@eqdinamicas,kval,cval);
end
```

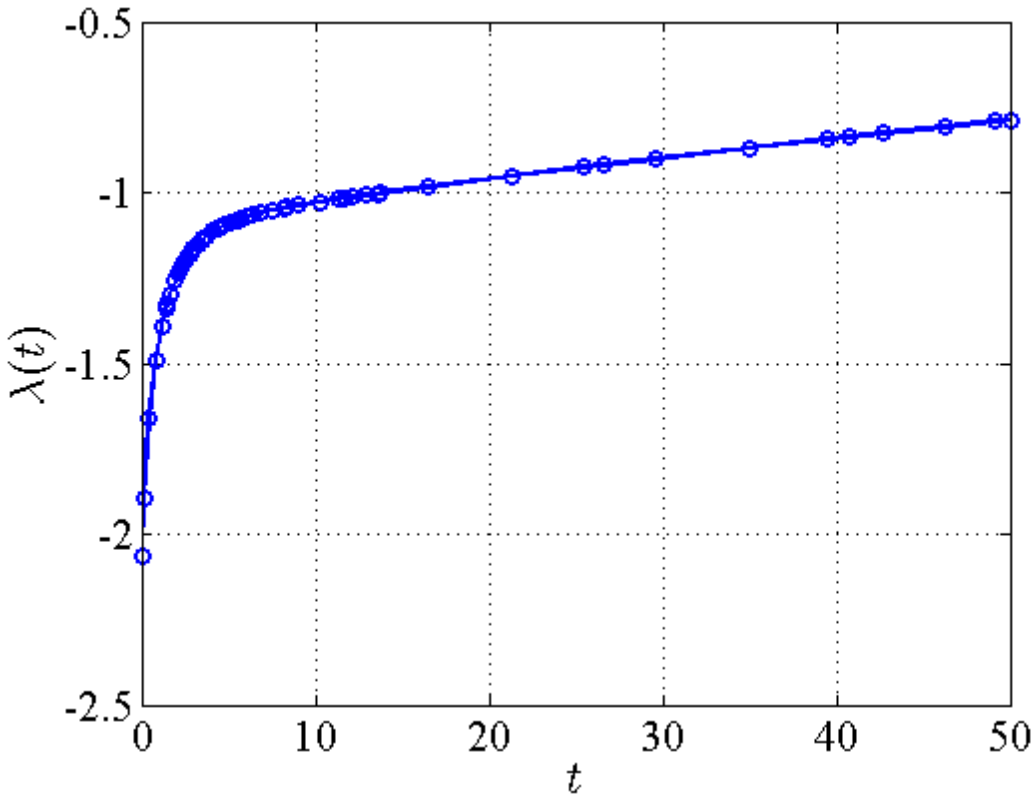
```
end

function ydot = eqdinamicas(~,y)



ydot=zeros(2,1);
 %dc/dt

ydot(1) = y(1).^alpha-(delta+n+gx).*y(1)-y(2); %dk/dt
ydot(2) = y(2).*sigma.*(alpha.*y(1).^(alpha-1)-delta-rho-gx/sigma); %dc/dt
end

function campovectorial(func,kval,cval)

n1 = length(kval);
n2 = length(cval);
%yp1 e yp2 são os vectores associados a yval e rval, respectivamente.
yp1 = zeros(n2,n1);   %Indiferente pois y1val e y2val têm que ser igualmente espaç
yp2 = zeros(n2,n1);
for i = 1:n1
  for j = 1:n2
     %yp1 e yp2 são os vectores associados a yval e rval, respectivamente.
    ypv = feval(func,0,[kval(i);cval(j)]);
    yp1(j,i) = ypv(1);
    yp2(j,i) = ypv(2);
  end
end
 %Representação gráfica; Desenha setas nos vectores de velocidade
quiver(kval,cval,yp1,yp2,'r','LineWidth',1.4);
axis tight;

end

function res = bcfun(ya,yb)


  res = [ ya(1) - k0
          yb(1) - kss];
end

function j = J(~,y)



j=zeros(2,2);
j(1,1)=alpha.*y(1,:).^(alpha-1)-(delta+n+gx);
j(1,2)=-1;
j(2,1)=y(2,:).*sigma.*(alpha.*(alpha-1)).*y(1,:).^(alpha-2);
j(2,2)=sigma.*(alpha.*y(1,:).^(alpha-1)-delta-rho-gx/sigma);
end

function solucaoanalitica
```

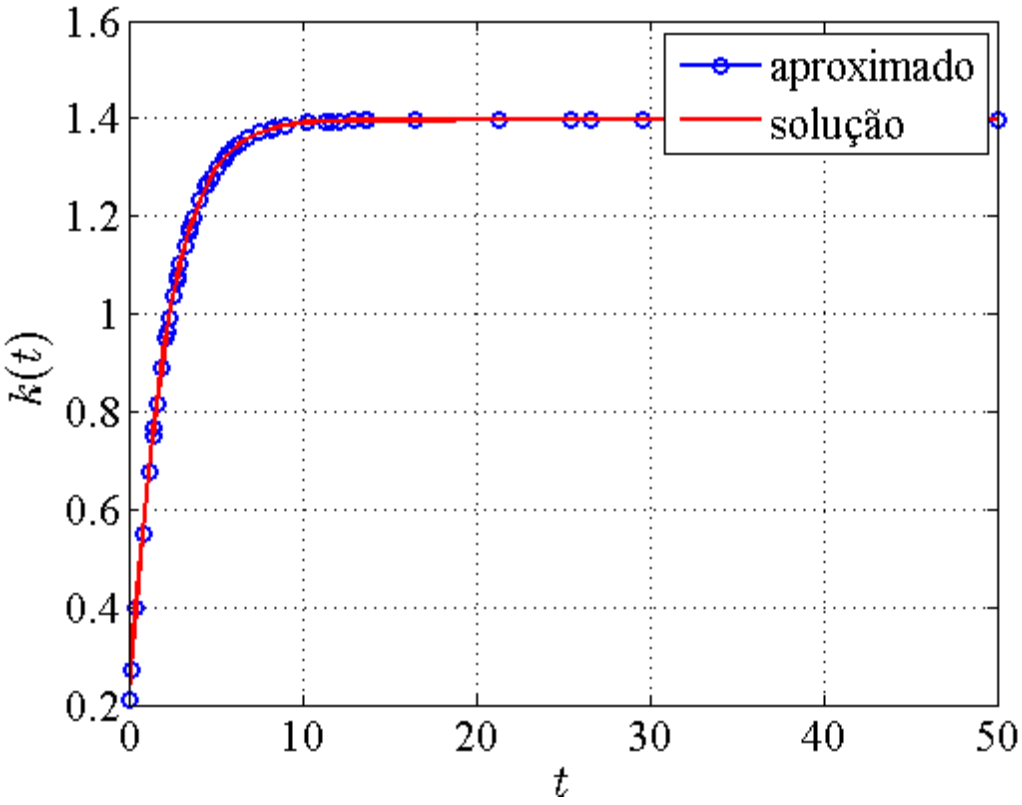
```
ksol = ((kss.^(1-alpha)+(k0.^(1-alpha)-kss.^(1-alpha)).*exp(-(1-alpha).*...
((rho+delta+gx/sigma)/alpha).*t)).^(1/(1-alpha)));

csol = (((rho-n-gx*(1-1/sigma))+(n+delta+gx)*(1-alpha)).*(((kss.^(1-alpha)+...
(k0.^(1-alpha)-kss.^(1-alpha)).*exp(-(1-alpha).*((rho+delta)/alpha).*t))...
.^(1/(1-alpha)))))/alpha; % dk/dt=0 => c(t)=(...)

%Valores para função campovectorial -> Diagrama de fase
% Valor inicial diferente mas próximo de zero para diagrama de fase
kval = linspace(0.1,max(ksol)*1.2,15);
% Valor inicial diferente mas próximo de zero para diagrama de fase
cval = linspace(0.1,max(csol)*1.2,15);

kg = linspace(0,max(ksol)*1.2,200);
c_dk_dt_0 = kg.^alpha-(delta+n+gx).*kg; % dk/dt=0 => c(t)=(...)


end
end
```

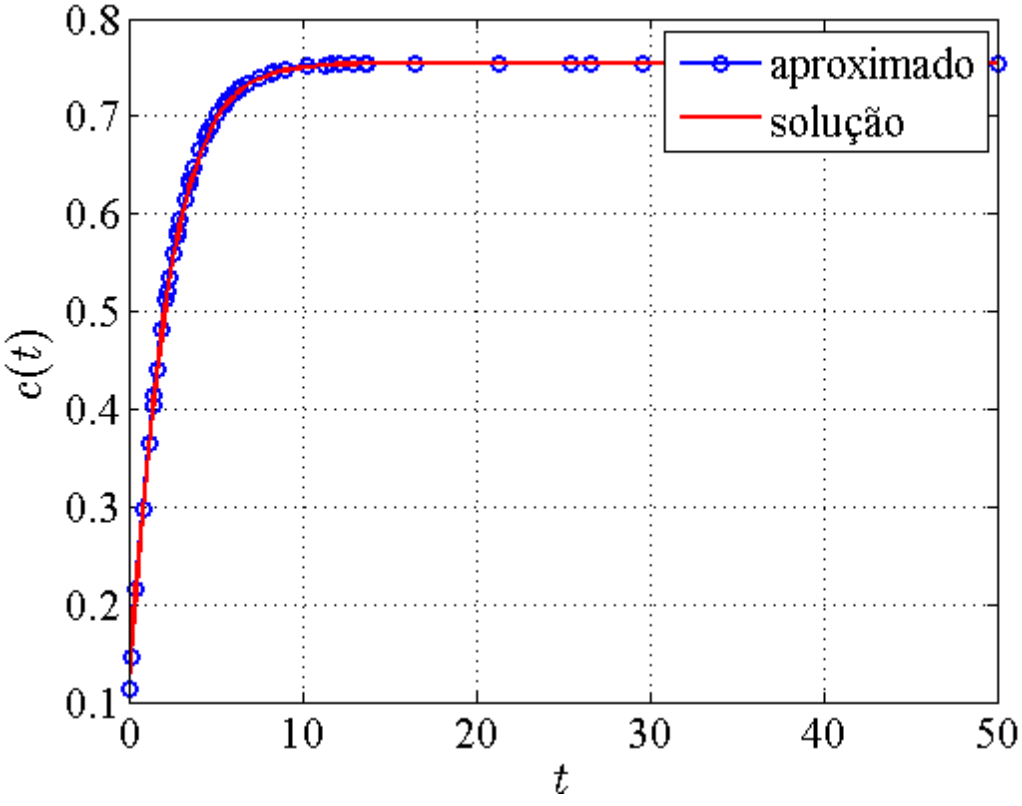
```
        ------------------------------------------------------------------------
```

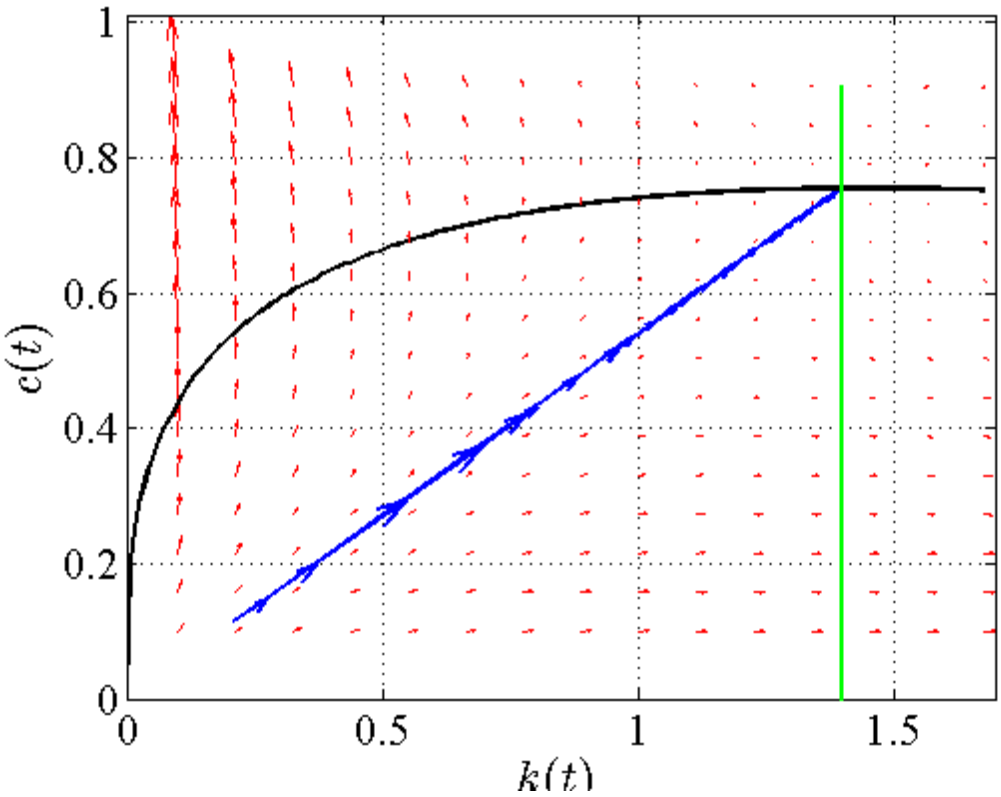


```
...
```




aproximado
solução
k(t)
t
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
10
20
30
40
50


aproximado
solução
c(t)
t
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10
20
30
40
50



Published with MATLAB® 7.13