# Zbiór rozwiązań problemów z kodem w języku Matlab/Octave z podręcznika *Programowanie dynamiczne i modele rekursywne w ekonomii*

Piotr Pieniążek

Łódź 2014

# Zbiór rozwiązań problemów z kodem w języku Matlab/Octave z podręcznika *Programowanie dynamiczne i modele rekursywne w ekonomii. Zagadnienia analityczne i metody numeryczne z przykładowymi implementacjami w języku Matlab/Octave,*

Piotr P. Pieniążek*

Ta wersja: październik 2015†
Pierwsza wersja: grudzień 2014

*Zbiór ten dedykuję wszystkim tym, którzy udostępniają swoje materiały dydaktyczne bez ograniczeń*




*Kontakt: piotr.p.pieniazek@gmail.com.

†Drobne poprawki, takie jak np. umieszczenie ilustracji autorstwa Marii Milosavljević na stronie tytułowej, dokonywane są na bieżąco.

# Spis treści

# 1 Wstęp

Niniejsza praca zawiera rozwiązania (i niekiedy, a nawet dość często, także ich rozszerzenia jeśli ich dodanie wydało się pouczające) dla wszystkich wymagających użycia kodu problemów z ćwiczeń z treści rozdziałów (rozdziały 1-6; problemy z dodatków pominięto) i zadań znajdujących się na ich końcu w *Części I* (*Zagadnienia deterministyczne*) i jak do tej pory jedynej (?) świetnej książki Klimy (2005)[1]. Stanowi więc dla niej pewnego rodzaju *solutions manual* z kodami w języku Matlab/Octave[1].

Zbiór ten powstał głównie z myślą o samoukach, jako złożenie kodów pisanych przeze mnie w procesie lektury książki Klimy. Potencjalnie skrywa liczne błędy w swej treści i formie, a przynajmniej zawarte w nim rozwiązania można przypuszczalnie pod wieloma względami poprawić, dlatego byłbym wdzięczny za przesłanie wszelkich uwag, które się do nich odnoszą i informacji o możliwych błędach.

W treści tego zbioru znajdują się kody, które wykorzystują napisane przez Klimę funkcje, a te czytelnik może już sam odnaleźć w jego pracy, dlatego nie zostały tu umieszczone (chyba że w zmienionej formie, jeśli ich modyfikacja okazała się niezbędna). Nie jest więc ta praca samowystarczalną. Choć nie ma na kartach tego zbioru także przepisanych poleceń czy danych rozpatrywanych problemów, to zawarte zostały jednak algebraiczne sformułowania i częściowe analityczne rozwiązania tych ćwiczeń i zadań, które akurat tego wymagały (a przynajmniej wydawało się to wskazane), by w ogóle zdefiniować problem do zakodowania.

[1] Do przeglądania i manipulacji kodami użyteczne są też (darmowe) edytory, takie jak np. Notepad++.

# 2 Rozdział Pierwszy. *Najprostszy problem programowania dynamicznego. Rekursja*

## 2.1 Ćwiczenie 1.6

Plik[2] *c16.m*

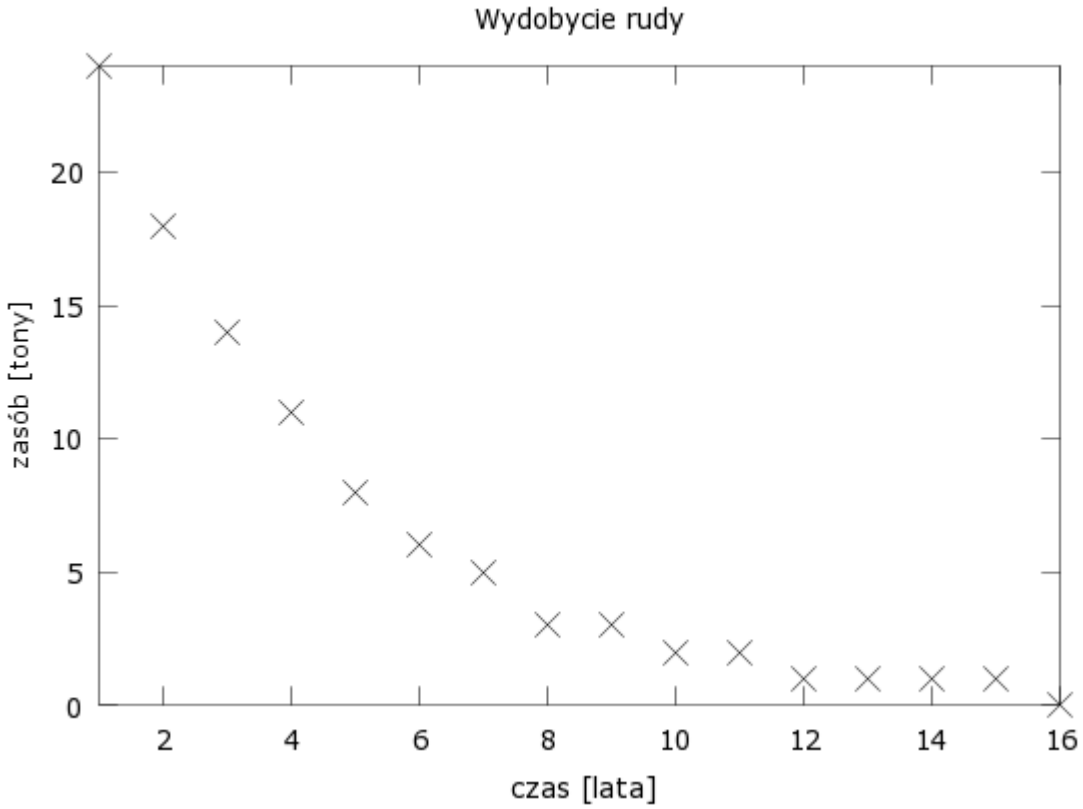
```
%========================================================================
%   Rozwiązanie ćwiczenia 1.6 z książki Klimy (2005).
%   Autor: Piotr Pieniążek, CC BY-NC-SA
%========================================================================

% kod należy dołączyć do tego już napisane przez Klimę w problemie 1.5.3

T=15;

x=(1:1:T+1);
y=zeros(1,T+1);
z=sim1(101, trans, c)-1;

for i=1:T
  y(1,i)=z(1,i)-z(1,i+1);
end

y(1,16)=0; % wynika z deklaracji y

plot(x,y,"xk")
title(['Wydobycie rudy'])
xlabel('czas [lata]')
ylabel('zasób [tony]')
```

[2]Przy kopiowaniu kodów należy być czujnym na to, że apostrofy mogą zmieniać swą formę, a tym samym stawać się nieczytelne dla Matlaba/Octave'a.

```
axis('tight')


%albo alternatywnie: dodać do pętli dla zmiennej 't' w sim1.m:
%
%y(t)=control(x(t),t);
%
%a na końcu:
%
%y(1,16)=0;
%Axx=(1:time+1)
%plot(Axx,y)
```

Koniec pliku *c16.m*

Wykonanie programu powinno zwrócić następujący wykres:



## 2.2 Ćwiczenie 1.7

Na przykład:

```
term = [ones(1,11)*(-inf), 0]';
```

albo

```
term = [ones(11,1)*(-inf); 0];
```

## 2.3 Zadanie 1.1

1.

Funkcja celu ma postać:

$$\max_{C}\{U(C)\},$$

gdzie $C$ to dwudziestoelementowy wektor konsumpcji zboża w czasie (horyzont czasowy wynosi 20 lat), a $U$ to addytywna międzyokresowa funkcja użyteczności:

$$U = \sum_{t=0}^{T} \beta^t u(c_t),$$

przy czym $T = 19$, $\theta = 0.1$, a $\beta = (1+\theta)^{-1}$. Zaś wewnątrzokresowa funkcja użyteczności dana jest wzorem:

$$u(c_t) = \ln c_t. \tag{1}$$

Równanie ewolucji zasobu (względnie: funkcję przejścia czy równanie ruchu: $s_{t+1} = g(s_t, c_t)$) można zapisać:

$$s_{t+1} = \min\{40, 2i_t\}, \tag{2}$$

gdzie $s_t$ to zboże do dyspozycji ($s_0 = 10$), a

$$i_t = s_t - c_t \tag{3}$$

to zasiane zboże. Poza tym $c_t, s_t, i_t \geq 0$ wyrażone są w tonach.

2.

Do równania Bellmana:

$$V_t(s_t) = \max_{c_t}\{u(c_t) + \beta V_{t+1}\big(s_{t+1}(s_t, c_t)\big)\}$$

można podstawić 1 i 2, uprzednio do 2 podstawiając 3.

3.

Plik *z11.m*

```
%=========================================================================
%    Rozwiązanie zadania 1.1 z książki Klimy (2005).
%    Autor: Piotr Pieniążek, CC BY-NC-SA
%=========================================================================

close all
clear all
clc

%-------------------------------------------------------------------------
% 1. Deklaracja danych problemu
%-------------------------------------------------------------------------

d = 0.2;                    % stopa dyskontowa;
%zmodyfikowano oryginalne dane (d=0,1) dla jaskrawości przykładu
dprim = 0.01;               % alternatywna stopa dyskontowa
r = d/(1-d);                % stopa procentowa ze wzoru NPV=NPV*(1+r)*(1-d)
beta = (1+r)^-1;            % czynnik dyskontujący (= 1-d)
betaprim = 1-dprim;         % czynnik dyskontujący dla d = 0,01
sstart = 10;                % zasób początkowy
smax = 40;                  % zasób maksymalny
smin = 0;                   % zasób minimalny
H = 20;                     % horyzont czasowy problemu;
```

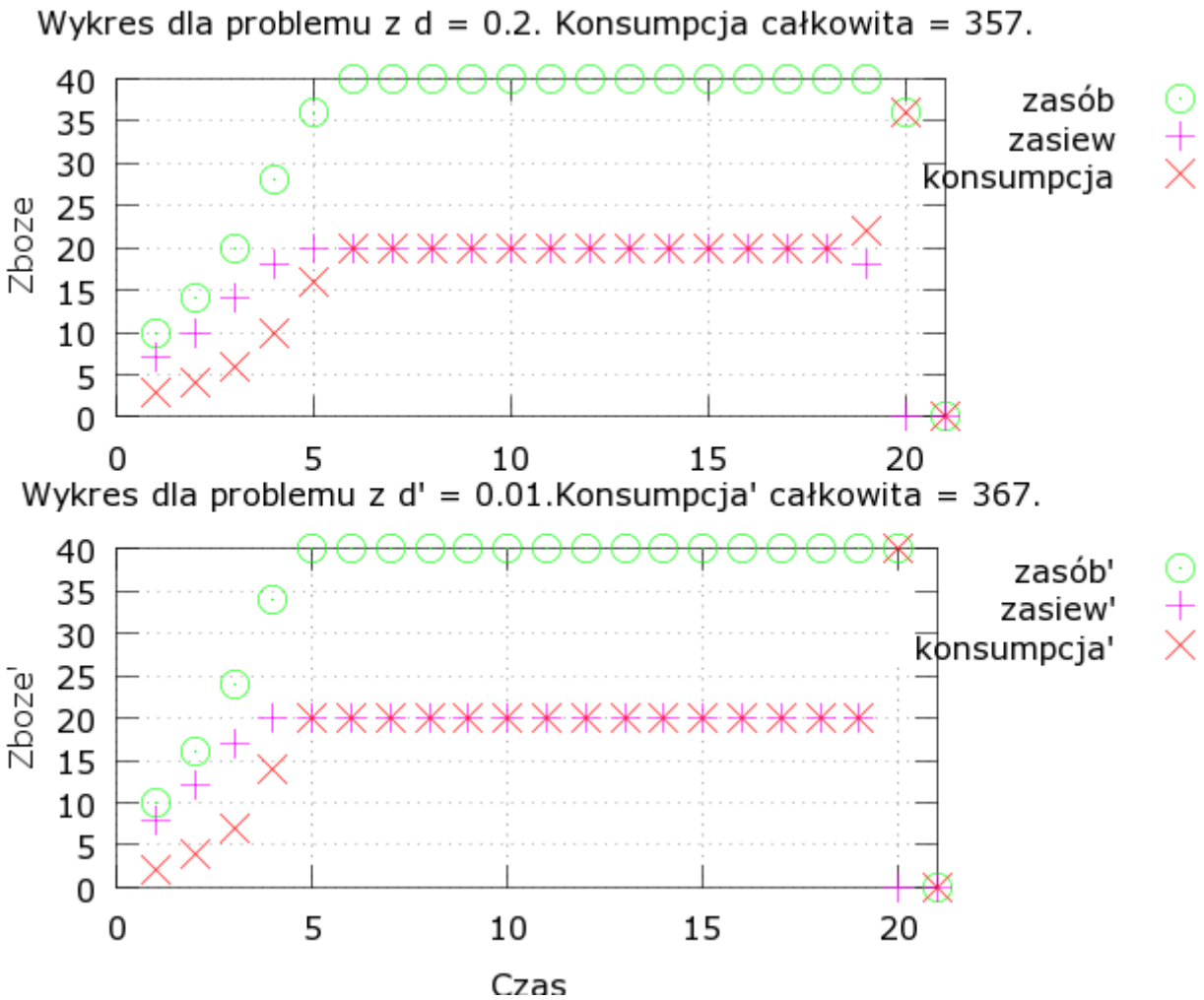
```
%zmieniony, by na wykresach wyraźniejszą stała się różnica ścieżek
%czasowych dla różnych wariantów preferencji czasowych
term = zeros(smax+1,1);      % wektor wypłat końcowych (wartości końcowej)
trans = zeros(smax+1,smax+1);% macierz przejścia
rew = zeros(smax+1,smax+1);  % macierz wypłat

%--------------------------------------------------------------------------
% 2. Wypełnienie macierzy przejścia i wypłat
%--------------------------------------------------------------------------

for i = 1:smax+1 % pętla stanu

  for j = 1:smax+1 % pętla polityki

    trans(i,j) = min(smax+1,max(smin+1,2*(i-j)+1));

    if (i >= j)
    % działa jeśli ln(0)=-inf; jeśli ln(0) nie istnieje, to wtedy "if (i>j)"

       rew(i,j) = log(j-1);

    else

       rew(i,j) = -inf; % metoda kar, by wyeliminować niedopuszczalne sterowania

    end

  end

end
```

```
%-------------------------------------------------------------------------
% 3. Wykorzystanie funkcji dprecur1.m i sim1.m do rozwiązania problemu
%    polityki optymalnego wykorzystania zasobu i symulacji jego ścieżki
%    ewolucji w czasie dla dwóch różnych stóp dyskontowych.
%-------------------------------------------------------------------------

[C, V] = dprecur1(H, beta, rew, trans, term);
s = sim1(sstart+1, trans, C)-1;

[Cprim, Vprim] = dprecur1(H, betaprim, rew, trans, term);
sprim = sim1(sstart+1, trans, Cprim)-1;

%-------------------------------------------------------------------------
% 4. Wizualizacja ścieżek wielkości zasobu, zasiewu oraz konsumpcji (zboża).
%-------------------------------------------------------------------------

axX = 1:H+1;

c = zeros(1,H+1);
z = zeros(1,H+1);

cprim = zeros(1,H+1);
zprim = zeros(1,H+1);

for h = 1:H % c(1,h+1)=0 wynika z deklaracji c

    c(1,h) = C(s(h)+1,h)-1;
    % C w kategoriach indeksów elementów macierzy, c w kat. realnych
    z(1,h) = s(h)-c(h); % w kategoriach realnych (tony)
```

```
    cprim(1,h) = Cprim(sprim(h)+1,h)-1; % to samo dla problemu z d = 0,01
    zprim(1,h) = sprim(h)-(cprim(h)); % to samo dla problemu z d = 0,01

end

sumc = sum(c); % obliczanie całkowitej konsumpcji
sumcprim = sum(cprim);

figure % wykresy ścieżek dla różnych "d"

subplot(2,1,1);
plot(axX,s,"go",axX,z,"+m",axX,c,"xr");
legend('zasób','zasiew','konsumpcja',-1)
ylabel('Zboze')
title(['Wykres dla problemu z d = ',num2str(d),'.
Konsumpcja całkowita = ',num2str(sumc),'.']) %title ciągiem, bez "return"
axis([0 H+1])
grid on

subplot(2,1,2);
plot(axX,sprim,"go",axX,zprim,"+m",axX,cprim,"xr");
legend('zasób''','zasiew''','konsumpcja''',-1)
xlabel('Czas')
ylabel('Zboze''')
title(['Wykres dla problemu z d'' = ',num2str(dprim),'.Konsumpcja''
całkowita = ',num2str(sumcprim),'.']) %title ciągiem bez "return"
axis([0 H+1])
grid on
```

Koniec pliku *z11.m*

Wykonanie programu powinno wygenerować następujące wykresy:



## 2.4 Zadanie 1.2

1.

Funkcja celu ma postać:

$$\max_{Q}\{\Pi(Q)\},$$

gdzie $Q$ to piętnastoelementowy wektor $q_t \geq 0$ wydobycia rudy [w tonach] w czasie (horyzont czasowy wynosi 15 lat), a $\Pi$ to (addytywna) wielookresowa funkcja zysku:

$$\Pi = \sum_{t=0}^{T} \beta^t \pi(q_t),$$

przy czym $T = 15$, $r = 0.1111$, a $\beta = (1+r)^{-1}$. Zaś natychmiastowa funkcja zysku dana jest standardowo jako różnica między przychodem całkowitym a kosztem całkowitym:

$$\pi(q_t) = TR(q_t) - TC(q_t). \tag{4}$$

przy czym:

$$TR = p_t q_t \quad \text{i} \quad TC = \frac{q_t^2}{1+s_t} \tag{5}$$

gdzie $s_t$ to pozostały w okresie $t$ możliwy do wydobycia zasób naturalny. Ograniczenie stanowi funkcja popytu:

$$q_t = a - bp_t, \tag{6}$$

gdzie $a \in \{60, 61, 62, ..., 70\}$, zaś $b \in \{35, 36, 37, ..., 45\}$, natomiast $p_t \geq 0$ to zmienna w czasie cena rynkowa rozważanego zasobu, względem której można przekształcić 6, by otrzymać funkcję odwróconego popytu:

$$p_t = \frac{a - q_t}{b}. \tag{7}$$

Można ją (7) podstawić do $TR$ w 5, a następnie oba wyrażenia 5 do 4, aby dostać:

$$\pi(q_t) = \frac{q_t(a - q_t)}{b} - \frac{q_t^2}{1+s_t}. \tag{8}$$

Równanie ewolucji zasobu wygląda natomiast następująco:

$$s_{t+1} = \max\{0, s_t - q_t\}, \tag{9}$$

albowiem $s_{t-1} \geq s_t \geq 0$.

2.

Do równania Bellmana:

$$V_t(s_t) = \max_{q_t}\{\pi(q_t) + \beta V_{t+1}(s_{t+1})\}$$

można podstawić 8 i 9.

3.

Plik *z12.m*

```
%========================================================================
%    Rozwiązanie zadania 1.2 z książki Klimy (2005).
%    Autor: Piotr Pieniążek, CC BY-NC-SA
%========================================================================

close all
clear all
clc

tic

%------------------------------------------------------------------------
% 1. Deklaracja danych problemu
%------------------------------------------------------------------------

sstart = 100;                       % zasób początkowy
```

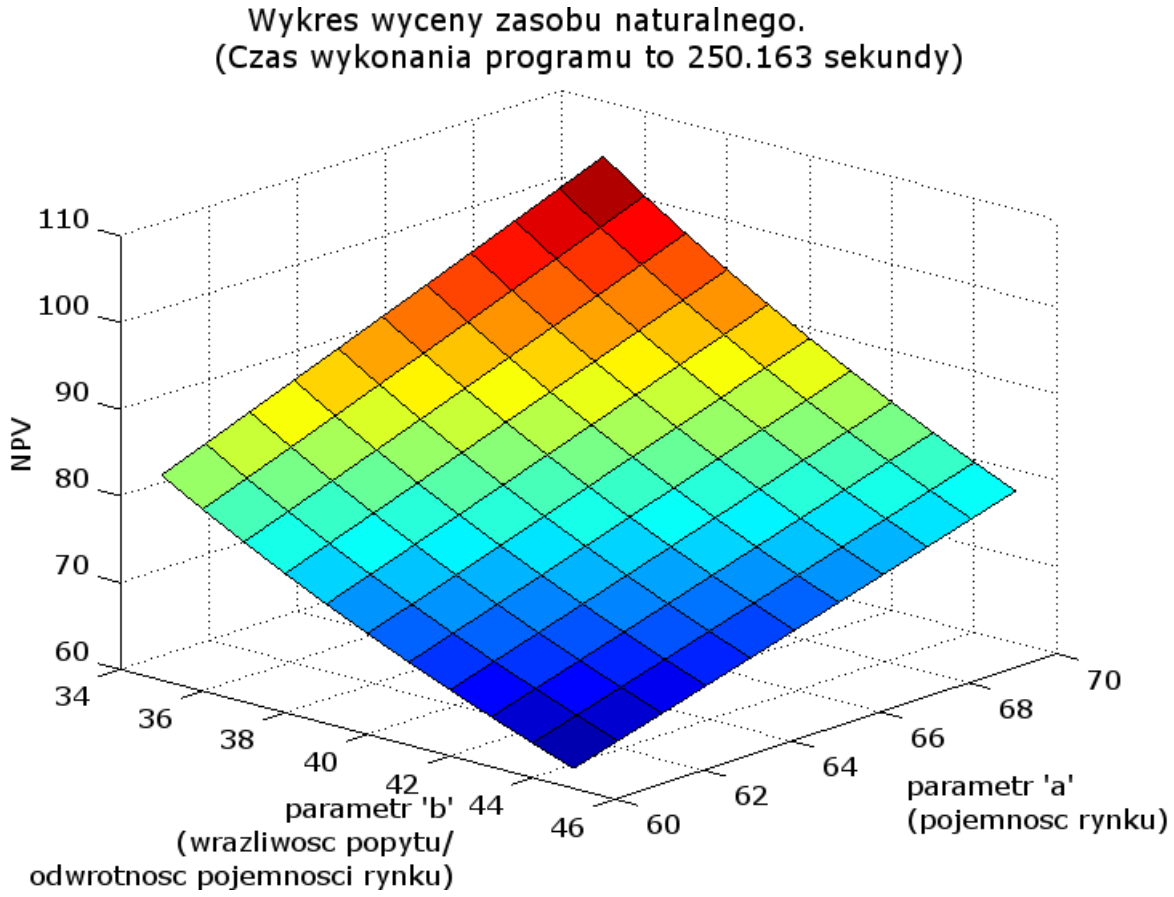
```
T = 15;                              % horyzont czasowy problemu
r = 0.1111;                          % stopa procentowa
beta = (1+r)^-1;                     % czynnik dyskontujący
a = (60:70);                         % proporcjonalne do pojemności rynku
b = (35:45);                         % wrażliwość popytu/odwrotność pojemności rynku
trans = zeros(sstart+1,sstart+1); % macierz przejścia
rew = zeros(sstart+1,sstart+1);   % macierz wypłat
term = zeros(sstart+1,1);         % wektor wypłat końcowych
NPV = zeros(size(a,2),size(b,2)); % macierz wycen początkowych

%------------------------------------------------------------------------
% 2. Wypełnienie macierzy przejścia i wypłat
%------------------------------------------------------------------------

for m = 1:size(a,2) % pętla problemu dla coraz pojemniejszego rynku

  for l = 1:size(b,2) % pętla problemu dla coraz wrażliwszego popytu/mniej
                        pojemnego rynku
    for i = 1:sstart+1 % pętla stanu

      for j = 1:sstart+1 % pętla polityki

        if ((l==1)&(m==1)) % obecność tego warunku gwarantuje w obrębie tych
        czterech pętli wypełnienie macierzy przejścia tylko raz zamiast 121
        razy, co skraca czas wykonania całego programu o około 7 sekund

            trans(i,j) = max(1,i-j+1);

        end
```

```
        if (i >= j)

          rew(i,j) = [(j-1)*(a(m)-j+1)/b(l)]-[(j-1)^2]/i;

        else

          rew(i,j) = -inf;

        end

      end

    end

%-------------------------------------------------------------------------
% 3. Wykorzystanie funkcji dprecur1.m do rozwiązania problemu polityki
%    optymalnej sprzedaży zasobu w czasie
%-------------------------------------------------------------------------

    [C, V] = dprecur1(T, beta, rew, trans, term);

    NPV(m,l) = V(sstart+1,1);

  end

end

%-------------------------------------------------------------------------
% 4. Wizualizacja wyceny zasobu w zależności od parametrów funkcji popytu
%-------------------------------------------------------------------------
```

```
surf(b,a,NPV)
xlabel({'parametr ''b''', '(wrazliwosc popytu/', 'odwrotnosc pojemnosci rynku)'});
ylabel({'parametr ''a''', '(pojemnosc rynku)'});
zlabel('NPV')
title(['Wykres wyceny zasobu naturalnego.';'(Czas wykonania programu to
',num2str(toc),' sekundy)'])
```

Koniec pliku *z12.m*

Wykonanie tego zawierającego w samej tylko głównej funkcji aż cztery pętle programu (stąd umieszczony licznik czasu wykonania operacji) powinno zwrócić:




W celu skrócenia czasu wykonania[3] tego programu można przeprowadzić wektory-

[3] Wykres wygenerowano przy użyciu programu Octave 3.8.2 na 64-bitowym systemie operacyjnym

zację, lecz wymaga ona ingerencji w funkcję `dprecur1`, dlatego poniżej zaprezentowano zmodyfikowany kod nie tylko głównej funkcji, ale także zwektoryzowaną wersję tej zagnieżdżonej.

Plik *z12vect.m*

```
%=========================================================================
%    Rozwiązanie (z wektoryzacją) zadania 1.2 z książki Klimy (2005).
%    Autor: Piotr Pieniążek, CC BY-NC-SA
%=========================================================================

close all
clear all
clc

tic

%-------------------------------------------------------------------------
% 1. Deklaracja danych problemu
%-------------------------------------------------------------------------

sstart = 100;                          % zasób początkowy
T = 15;                                % horyzont czasowy problemu
r = 0.1111;                            % stopa procentowa
beta = (1+r)^-1;                       % czynnik dyskontujący
a = (60:70);                           % proporcjonalne do pojemności rynku
b = (35:45);                           % wrażliwość popytu/odwrotność pojemności rynku
trans = zeros(sstart+1,sstart+1);      % macierz przejścia
rew = zeros(sstart+1,sstart+1,size(a,2),size(b,2));    % czterowymiarowa
                                         struktura wypłat
```

Windows 8.1 i procesorze Intel®Core™i5-4590 i 8 GB RAM. Natomiast temu samemu programowi w wersji 3.6.4. na 64-bitowym systemie operacyjnym Windows 7 z procesorem Intel®Core™2 Duo CPU E7500 2.93GHz i 4 GB RAM zajmuje to około 800 sekund.

```
term = zeros(sstart+1,1,size(a,2),size(b,2));          % czterowymiarowa
                                        struktura wypłat końcowych
NPV = zeros(1,1,size(a,2),size(b,2));                  % czterowymiarowa
                                        struktura wycen początkowych

%------------------------------------------------------------------------
% 2. Wypełnienie macierzy przejścia i wypłat
%------------------------------------------------------------------------

for m = 1:size(a,2) % pętla problemu dla coraz pojemniejszego rynku

  for l = 1:size(b,2) % pętla problemu dla coraz bardziej wrażliwego popytu/mniej
                        pojemnego rynku
    for i = 1:sstart+1 % pętla stanu

      for j = 1:sstart+1 % pętla polityki

        if (l==1)&(m==1)

          trans(i,j) = max(1,i-j+1);

        end

        if (i >= j)

          rew(i,j,m,l) = [(j-1)*(a(m)-j+1)/b(l)]-[(j-1)^2]/i;

        else

          rew(i,j,m,l) = -inf;
```

```
        end

      end

    end

  end

end

%-------------------------------------------------------------------------
% 3. Wykorzystanie funkcji dprecur1vect.m do rozwiązania problemu polityki
%    optymalnej sprzedaży zasobu w czasie.
%-------------------------------------------------------------------------

[C, V] = dprecur1vect(T, beta, rew, trans, term); % funkcja rozwiązująca
problem rekursywny z dyskretną przestrzenią stanów i skończonym
horyzontem czasowym

NPV = V(sstart+1,1,:,:);
NPV = squeeze(NPV); % usunięcie zbędnych (1-elementowych) wymiarów

%-------------------------------------------------------------------------
% 4. Wizualizacja początkowej wyceny zasobu w zależności od stopy procentowej
%-------------------------------------------------------------------------

surf(b,a,NPV)
xlabel({'parametr ''b''', '(wrazliwosc popytu/', 'odwrotnosc pojemnosci rynku)'});
ylabel({'parametr ''a''', '(pojemnosc rynku)'});
```

```
zlabel('NPV')
title(['Wykres wyceny zasobu naturalnego.';'(Czas wykonania programu to
',num2str(toc),' sekundy)'])
```

Koniec pliku *z12vect.m*

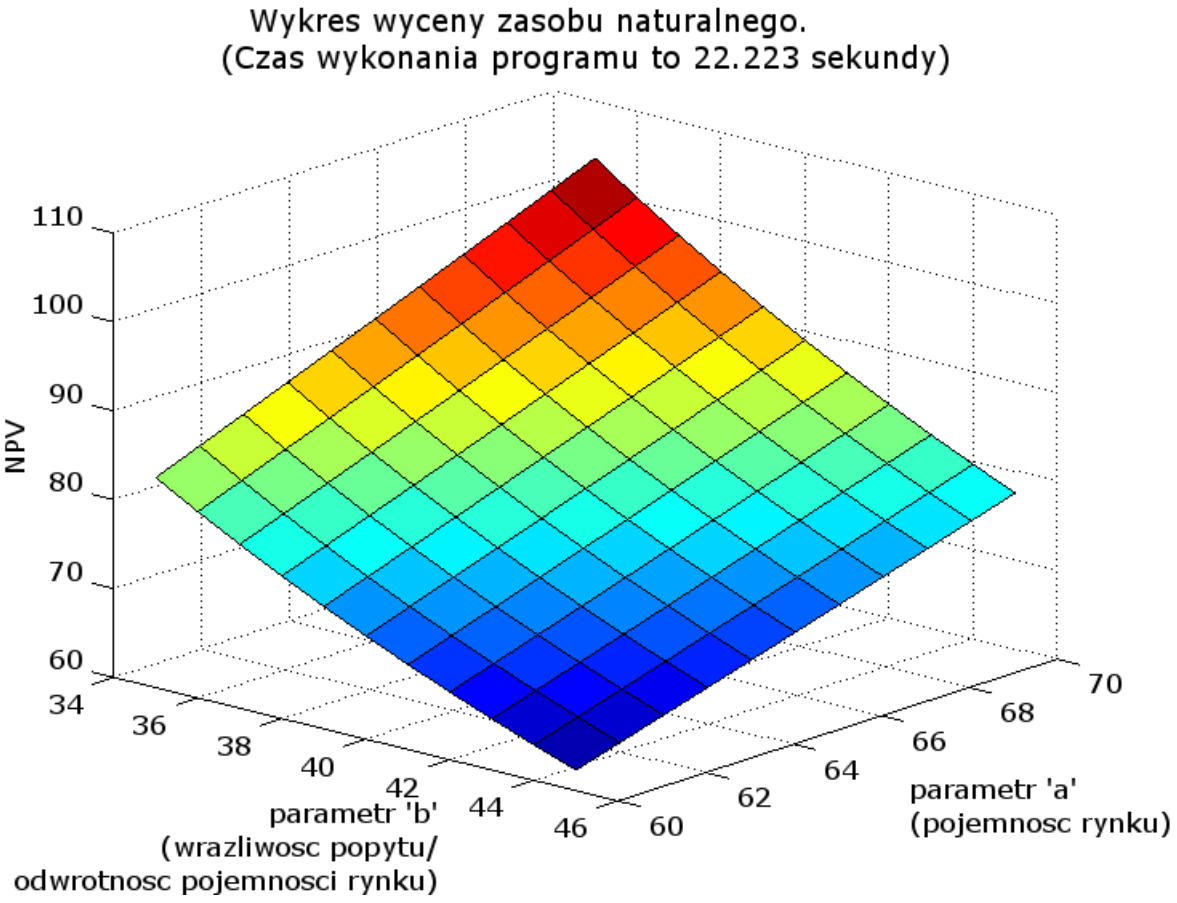
Plik *deprecur1vect.m*

```
% Zwektoryzowana wersja funkcji deprecur1.m autorstwa (c) Grzegorza Klimy 2004

function [C, V] = dprecur1vect(time, discount, reward, transition, terminal)

[n, m, a, b] = size(reward);
control = zeros(n,time,a,b);
value = zeros(n,time+1,a,b);
value(:,time+1,:,:) = terminal;
policy_values = zeros(m,a,b);

for t = time:-1:1

  for i = 1:n

    for j = 1:m

      policy_values(j,:,:) = squeeze(reward(i,j,:,:)) +
                             discount*squeeze(value(transition(i,j),t+1,:,:));

    end

    [value(i,t,:,:), control(i,t,:,:)] = max(policy_values);

  end
```

```
end


C=control;
V=value;
```

Koniec pliku *dprecur1vect.m*

Program powinien wygenerować następujący wykres:




Jak widać, zabieg wektoryzacji wydatnie - albowiem ponad dziesięciokrotnie - przyspieszył wykonanie instrukcji[4].

[4] Jak pokazują Aruoba i Fernández-Villaverde (2015) [2], nie jest to jednak regułą, np. jeśli wektoryzacja prowadzi do pominięcia pewnych elementów kodu, dlatego nie powinno się jej stosować bezkrytycznie.

# 3 Rozdział Drugi. *Ciągła przestrzeń stanów. Równania Eulera. Nieskończony horyzont*

## 3.1 Ćwiczenie 2.15

Plik *c215.m*

```
%=========================================================================
%   Rozwiązanie ćwiczenia 2.15 z książki Klimy (2005).
%   Autor: Piotr Pieniążek, CC BY-NC-SA
%=========================================================================

close all
clear all
clc

%-------------------------------------------------------------------------
% 1. Deklaracja danych problemu.
%-------------------------------------------------------------------------

zasob = 101;                % zasób początkowy
trans = zeros(zasob,zasob); % macierz przejścia
rew = zeros(zasob,zasob);   % macierz wypłat
rmin = 0.01;                % kres dolny stóp procentowych
rmax = 0.2;                 % kres górny stóp procentowych
dr = 0.01;                  % różnica między stopami procentowymi
r = (rmin:dr:rmax);         % wektor stóp procentowych
d = (r+1).^-1;              % wektor czynników dyskontujących
eks = zeros(1,size(r,2));   % wektor z okresami eksploatacji (jako elementami)
wart = zeros(1,size(r,2));  % wartość początkowa zasobu
czas = 100;                 % horyzont czasowy
```

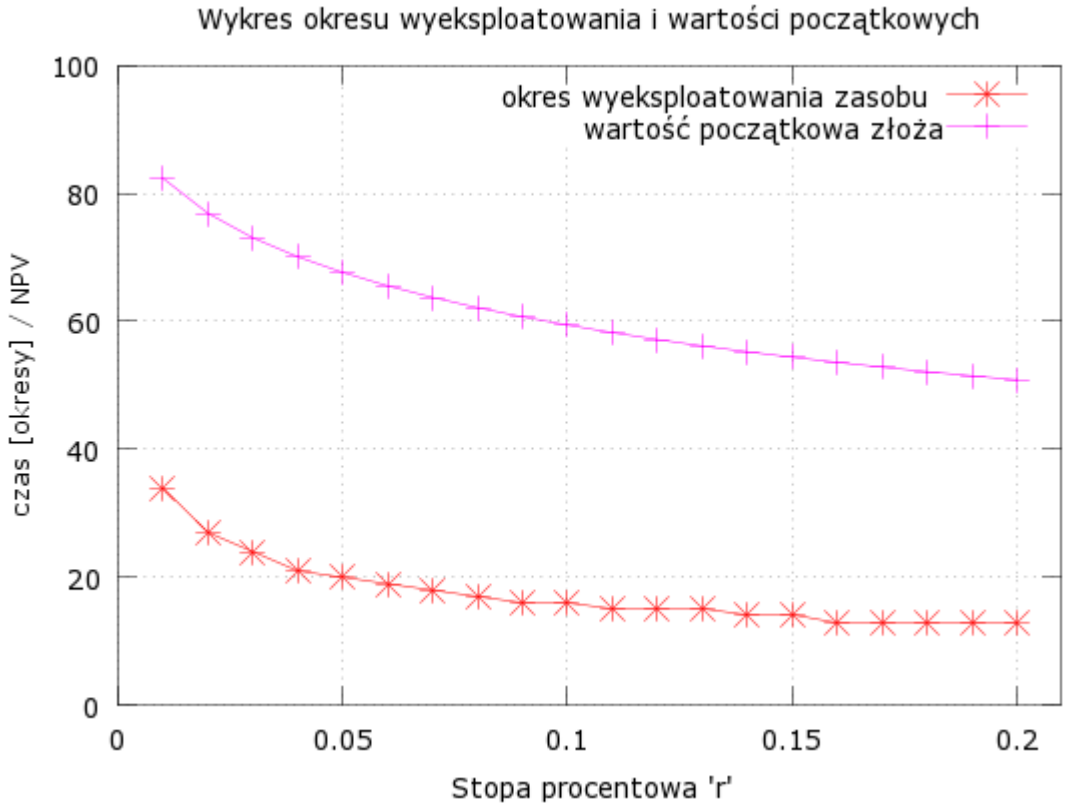
```
%-------------------------------------------------------------------------
% 2. Wypełnienie macierzy przejścia i wypłat
%-------------------------------------------------------------------------

for i = 1:zasob % pętla stanu

  for j = 1:zasob % pętla polityki

    trans(i,j) = max(0,i-j)+1;

    if (i >= j)

      rew(i,j) = (j-1)-(j-1)^2/i;

    else

      rew(i,j) = -inf;

    end

  end

end

%-------------------------------------------------------------------------
% 3. Pętla wykonująca algorytm kolejnych przybliżeń dla problemu rekursywnego
%    z nieskończonym horyzontem czasowym, a następnie symulująca ścieżkę
%    stanu i obliczająca okres, po którym zasoby zostają wyczerpane, a
%    także wartość złoża w okresie początkowym.
%-------------------------------------------------------------------------
```

```
for k = 1:size(r,2) % pętla stóp procentowych

  [C, V] = dpsa2(d(k), rew, trans);

  z = sim2(zasob, czas, trans, C)-1;

  wart(k) = V(zasob); % wartość nieeksploatowanego jeszcze złoża

    for w = 1:czas % pętla czasu

    if (z(w) == 0) % jeśli w okresie 'w' zasób wyniesie 0, to złoże
                   % zostanie wyeksploatowane w okresie 'w-1'

      eks(k) = w-1;

      break % przerwanie pętli 'for' po ustaleniu okresu wyeksploatowania

    end

  end

end

%-------------------------------------------------------------------------
% 4. Generowanie wykresu okresu eksploatacji i V(t=0) złoża dla każdej ze
%    stóp procentowych.
%-------------------------------------------------------------------------

plot(r,eks,"*r-;okres wyeksploatowania zasobu;",r,wart,"+m-;wartość
```

```
początkowa złoża;")
% plot ciągiem bez "return"
xlabel('Stopa procentowa ''r''')
ylabel('czas [okresy] / NPV')
title('Wykres okresu wyeksploatowania i wartości początkowych')
axis( [r(1)-dr r(20)+dr 0 ceil(max(wart(1),eks(1))/100)*100] )
grid on
```

Koniec pliku *c215.m*

Wykonanie programu powinno wygenerować następujący wykres:




Komentarz intuicyjnie wyjaśniający uzyskany wynik:

> Im wyższa stopa procentowa (preferencja czasowa), tym kompletne wydobycie zasobu następuje wcześniej, bo aktor jest bardziej niecierpliwy w swym działaniu i w każdym okresie wydobywania, wydobywa więcej, niezrażony

nawet tym, że koszt krańcowy owego wydobywania jest rosnący (nie kompensuje całkowicie wpływu efektu wyższej preferencji czasowej).

Gdyby zmodyfikowano kod napisanego przez Klimę programu `dpsa2`, wykorzystując przy tym właściwość monotoniczności funkcji polityki i warunek obwiedni, to (tak jak w kodzie Aruoby i Fernándeza-Villaverde (2015) [2]) zmniejszono by ilość wykonanych iteracji w algorytmie kolejnych przybliżeń. Co więcej, w tym samym celu można obrać rozsądniejszą wartość elementów wektora pierwotnej funkcji wartości, nie we wszystkich iteracjach aktualizować sterowanie (jest to poprawka Howarda) czy w końcu wykorzystywać interpolację do znajdowania dokładniejszych wartości funkcji wartości niż tych wynikłych z przeszukiwania wyłącznie pierwotnej kraty sterowań (zob. np. Heer i Maußner (2009) [3], sekcja 4.1).

## 3.2 Zadanie 2.3

Wykorzystany w poniższym kodzie wzór na kapitał w stanie ustalonym można uzyskać rozwiązując **Ćwiczenie 2.14**.

Plik *z23.m*

```
%=========================================================================
%    Rozwiązanie zadania 2.3 z książki Klimy (2005).
%    Autor: Piotr Pieniążek, CC BY-NC-SA
%=========================================================================

close all
clear all
clc

%-------------------------------------------------------------------------
% 1. Deklaracja danych problemu
%-------------------------------------------------------------------------

alpha = 1/3;                        % elastyczność w funkcji produkcji
```

```
                                    % typu Cobba-Douglasa
beta = 0.9;                         % czynnik dyskontujący
kstar = (alpha*beta)^[(1-alpha)^-1]; % kapitał (zm.stanu) w stanie ustalonym
kmin = 0.1*kstar;                   % wartość minimalna w wektorze kapitału
kmax = 1.3*kstar;                   % wartość maksymalna w wektorze kapitału
dk = 0.1*kstar;                     % róznica między kolejnymi możliwymi
                                    % realizacjami wartości zmiennej stanu
k = (kmin:dk:kmax);                 % wektor możliwych realizacji wartości
                                    % zmiennej stanu
t = 10;                             % horyzont czasowy symulacji
kss = zeros(1,t);                   % wektor wartości kapitału w stanie ustalonym
ks = zeros(1,t);                    % symulowana ścieżka ewolucji kapitału
kt = zeros(1,t);                    % teoretyczna ścieżka ewolucji kapitału
startK = 1;                         % numer indeksu wektora k jako
                                    % wartość startowa symulacji
stopT = 0;                          % okres, po którym następuje zbieżność
                                    % rozwiązania symulowanego do stanu ustalonego;
                                    % jeśli nie następuje, to stopT = 0

%-------------------------------------------------------------------------
% 2. Wypełnienie macierzy przejścia i wypłat.
%-------------------------------------------------------------------------

for i = 1:size(k,2) % pętla stanu

  for j = 1:size(k,2) % pętla polityki

    trans(i,j) = j;

    if (k(i)^alpha >= k(j))
```

```
      rew(i,j) = log(k(i)^alpha-k(j));

    else

      rew(i,j) = -inf;

    end

  end

end

%------------------------------------------------------------------------
% 3. Rozwiązania problemu polityki optymalnej akumulacji kapitału
%    i symulacji ścieżki ewolucji tego zasobu w czasie.
%------------------------------------------------------------------------

[C, V] = dpsa2(beta, rew, trans);

ks = sim2(startK, t-1, trans, C);

kt(1)=k(startK); % ustalenie wartości startowej kapitału teoretycznego

for l = 2:t  % pętla ścieżki kapitału teoretycznego

    kt(l)=alpha*beta*kt(l-1)^alpha;

end
```

```
if (abs(kstar-k(ks(1,t))) < 0.0001) % sprawdzenie czy różnicę między wartościami
                                    % rozwiązania symulowanego po t okresach i
                                    % stanu ustalonego da się zaokrąglić do zera

  roznica = 0;

  for l = 1:t % sprawdzenie po ilu okresach rozwiązanie symulowane zbiega
              % do stanu ustalonego

    if ((abs(kstar-k(ks(l))) < 0.0001)) % warunek zbieżności kapitału do
                                        % wartości stanu ustalonego

      stopT = l;

      break % przerwanie pętli for po ustaleniu okresu zbieżnosci rozwiązania

    end

  end

else

  roznica = kstar-k(ks(1,t));

end



%-------------------------------------------------------------------------
% 4. Wyświetlenie porównania rozwiązania symulacyjnego z wynikiem
```

```
%    teoretycznym i informacji o zbieżności rozwiązania.
%--------------------------------------------------------------------------

fprintf('\n')

text1 = ['Rozwiazanie symulowane po ',num2str(t),'. okresach dla kapitalu to
',num2str(k(ks(1,t))),', a stan ustalony to ',num2str(kstar),'. Tak wiec
roznica miedzy rozwiazaniami (kss - sym) wynosi ',num2str(roznica),'.'];
% text1 ciągiem, bez "return". Tekst prezentujący wyniki k symulowanego w
% ostatnim okresie i wartość stanu ustalonego oraz różnicę między nimi.

disp(text1);

fprintf('\n')

if (stopT == 1) % wartość startowa kapitału jest równa tej dla stanu ustalonego

  text1 = ['Wybierz inny kapital startowy niz ten ze stanu ustalonego, czyli
  nie #',num2str(startK),'. element wektora kapitału!'];
  % text1 ciągiem, bez "return".

  disp(text1);

else

    axX = 1:t;
    kss(1:t) = kstar;
    plot(axX,k(ks(1:t)),"m-;kapitał symulowany;",axX,kt,";kapitał teoretyczny;",
    axX,kss,"c;wartość kapitału w stanie ustalonym ;"); % plot ciągiem, bez
                                                        % "return".
```

```
    xlabel('Czas [okresy]')
    ylabel('Kapitał')
    title('Wykres zbiegania w czasie kapitału do wartości stanu ustalonego')
    axis([1 t 0 ceil(1.2*kstar*50)/50]])
    grid on

    if (stopT == 0)

      warning('Twoj symulowany kapital nie zbiega do stanu ustalonego!
      Zmien wartosc startowa albo zwieksz horyzont symulacji.');
      % warning ciągiem, bez "return".

    else

      text2 = ['Rozwiazanie symulowane zbiega do stanu ustalonego po
      ',num2str(stopT),'. okresach symulacji.'];
      % text2 ciągiem, bez "return". Tekst wyświetlający ilość okresów,
      % po których kapitał symulowany zbiega do stanu ustalonego.

      disp(text2);

    end

end

fprintf('\n')
```

Koniec pliku *z23.m*

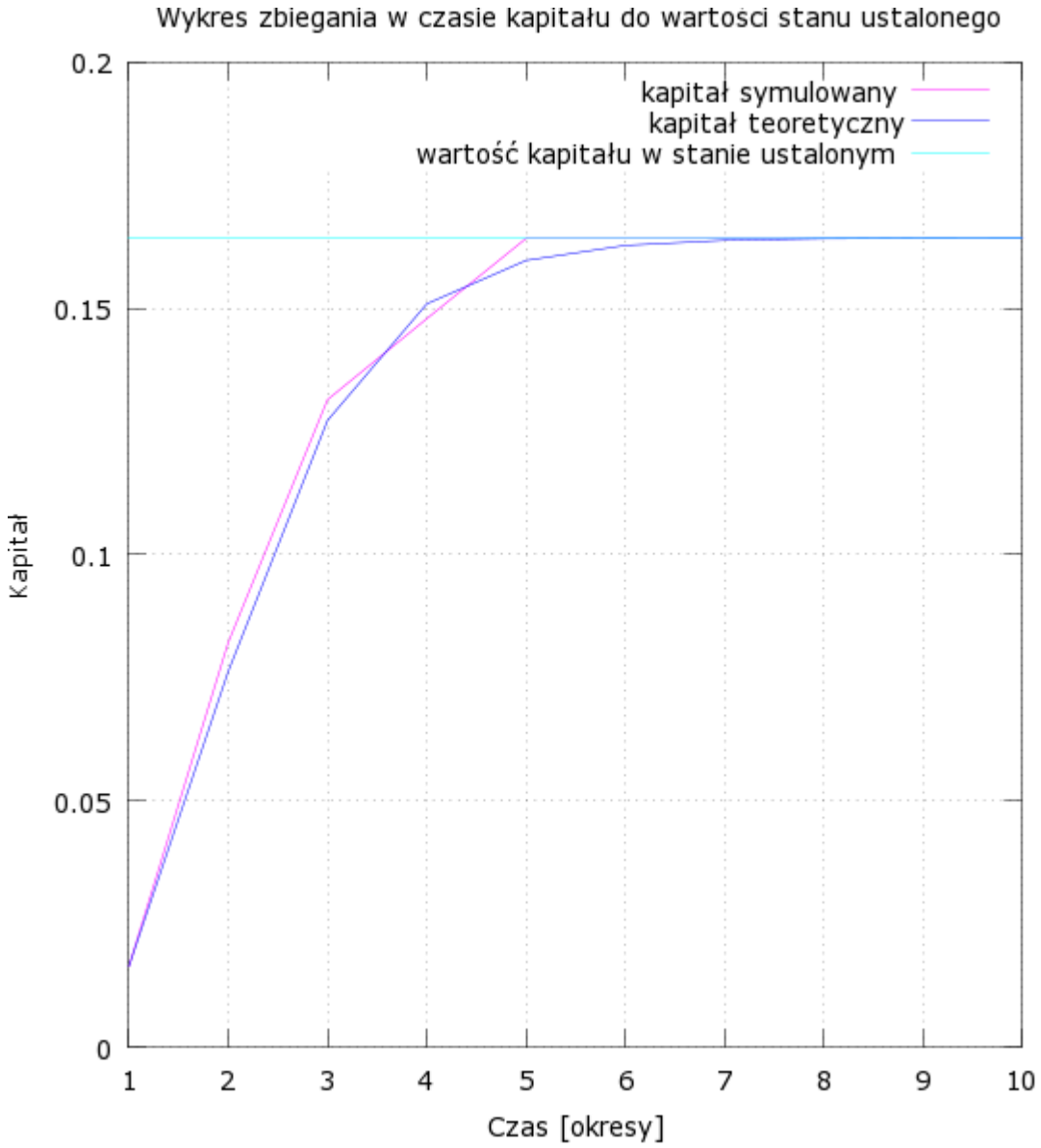
Wykonanie programu powinno wyświetlić następujące komunikaty:

```
Rozwiazanie symulowane zbiega do stanu ustalonego po 5. okresach symulacji.
```

i

```
Rozwiazanie symulowane po 10. okresach dla kapitalu to 0.016432, a stan ustalony
to 0.016432. Tak wiec roznica miedzy rozwiazaniami (kss - sym) wynosi 0.
```

Zwrócony winien zostać również poniższy wykres:




Ścieżka teoretyczna, jak można było oczekiwać, jest nieco bardziej gładka niż ta zdyskretyzowana (symulowana na kracie).

# 4 Rozdział Szósty. *Metoda kolokacji*

## 4.1 Ćwiczenie 6.1

Plik *c61.m*

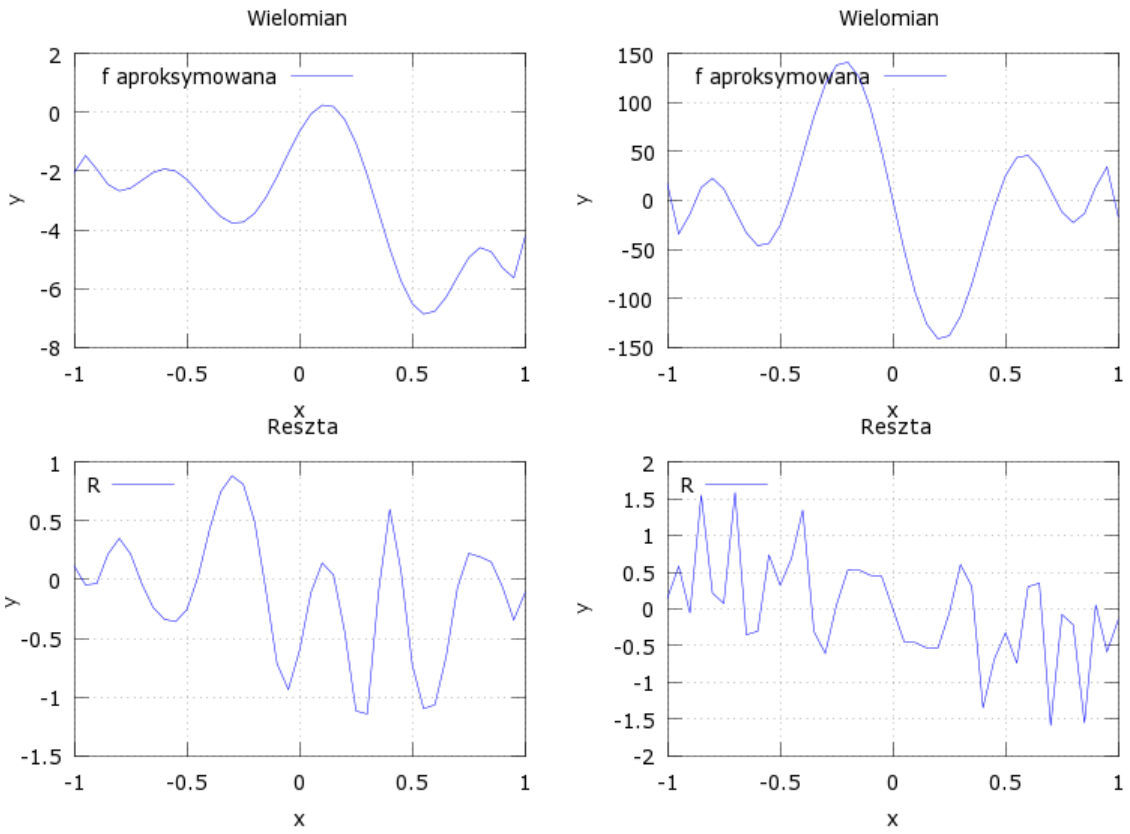
```
%=========================================================================
%    Rozwiązanie ćwiczenia 6.1 z książki Klimy (2005).
%    Autor: Piotr Pieniążek, CC BY-NC-SA
%=========================================================================

close all
clear all
clc

%-------------------------------------------------------------------------
% 1. Obliczanie wartości wielomianu i reszt dla dwóch alternatywnych
%    zestawów startowych wektorów szeregu Czebyszewa
%-------------------------------------------------------------------------

a=newton ('residasin', [-2 0 0 0 0 0 0 0 0 0]');
x=(-1:.05:1)';
y=chebval (a, x);
R=eqasin(x,y);

b=newton ('residasin', [0 10 0 0 0 0 0 0 0 0]');
z=chebval (b, x);
re=eqasin(x,z);

%-------------------------------------------------------------------------
% 2. Rysowanie wykresów wielomianów i reszt
%-------------------------------------------------------------------------
```

```
subplot(2,2,1);
plot(x,y);
legend('f aproksymowana',0)
xlabel('x')
ylabel('y')
title(['Wielomian'])
grid on

subplot(2,2,3);
plot(x,R);
legend('R',0)
xlabel('x')
ylabel('y')
title(['Reszta'])
grid on

subplot(2,2,2);
plot(x,z);
legend('f aproksymowana',0)
xlabel('x')
ylabel('y')
title(['Wielomian'])
grid on

subplot(2,2,4);
plot(x,re);
legend('R',0)
xlabel('x')
ylabel('y')
```

```
title(['Reszta'])
grid on
```

Koniec pliku *c61.m*

Wykonanie programu powinno wygenerować następujące wykresy:




## 4.2 Ćwiczenie 6.2

Plik *invdem.m*

```
function y=invdem(q, p)
y=q-p.^(-0.5)-3*p.^(-0.3333);
```

Koniec pliku *invdem.m*

Plik *residdem.m*

```
function ret=residdem(v)
ret=invdem(chebnod(10,1,4), chebphi(10)*v);
% o wyborze przedziału aproksymacji napisano poniżej
```

Koniec pliku *residdem.m*

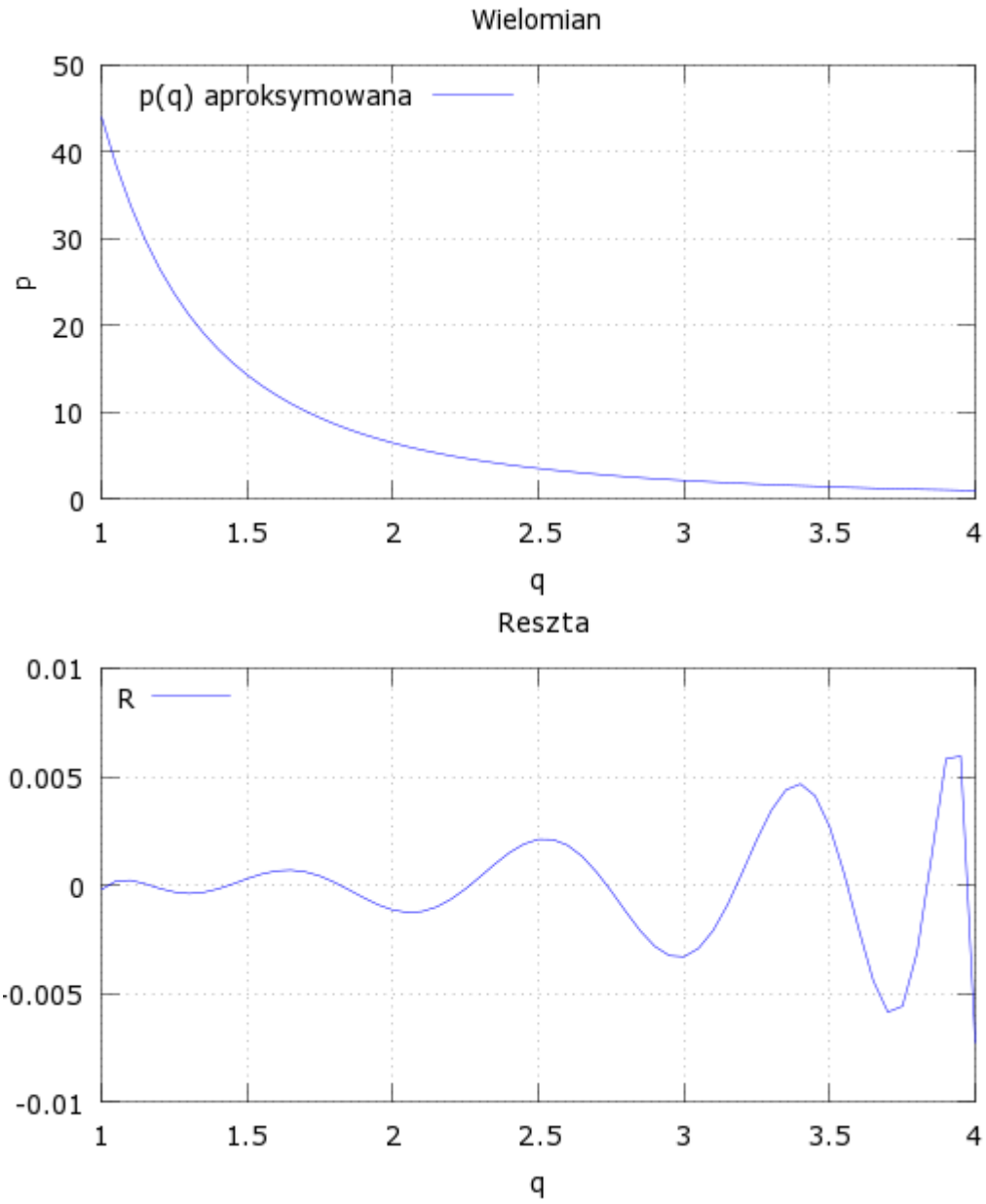
Plik *c62.m*

```
%=========================================================================
%   Rozwiązanie ćwiczenia 6.2 z książki Klimy (2005).
%   Autor: Piotr Pieniążek, CC BY-NC-SA
%=========================================================================

close all
clear all
clc

%-------------------------------------------------------------------------
% 1. Obliczanie wartości wielomianu i reszt dla odwróconej funkcji popytu
%-------------------------------------------------------------------------

a=newton ('residdem', [1 0 0 0 0 0 0 0 0 0]');

x=(1:.05:4)';
% przedział dla nieujemnych argumentów i nie rosnących ad infinitum wartości
% funkcji; w szczególności ze względu na ewentualną maksymalizację
% zysku przez monopolistę, najbardziej interesujący powinien być przedział o
% relatywnie nieelastycznym popycie, ale taki, w którym elastyczność cenowa
% popytu jest mniejsza od -1, by cena mogła być dodatnia.

y=chebval (a, x, 1, 4);
R=invdem(x,y);
```

```
%-------------------------------------------------------------------------
% 2. Rysowanie wykresu wielomianu i reszt dla odwróconej funkcji popytu
%-------------------------------------------------------------------------

subplot(2,1,1);
plot(x,y);
legend('p(q) aproksymowana',0)
xlabel('q')
ylabel('p')
title(['Wielomian'])
grid on

subplot(2,1,2);
plot(x,R);
legend('R',0)
xlabel('q')
title(['Reszta'])
grid on
axis([1 4 -.01 .01])
```

Koniec pliku *c62.m*

Wykonanie programu powinno zwrócić następujące wykresy:




### 4.3 Zadanie 6.1

1.

W modelu Ramseya z czasem w wersji dyskretnej i z endogeniczną podażą pracy, po wykorzystaniu danej tożsamości rachunkowej można uzyskać równanie ruchu w formie

funkcji wyłącznie jednej zmiennej stanu ($k_t$) i dwóch sterowań ($c_t(k_t)$ i $l_t(k_t)$):

$$\begin{aligned} k_{t+1} &= (1-\delta)k_t + i_t \quad [i_t \equiv f(k_t, l_t) - c_t] \\ &= (1-\delta)k_t + f(k_t, l_t) - c_t. \end{aligned} \tag{10}$$

Równanie Bellmana wygląda więc następująco:

$$V(k_t) = \max_{c_t, l_t} \{u(c_t, l_t) + \beta V[\underbrace{(1-\delta)k_t + f(k_t, l_t) - c_t}_{k_{t+1}}]\}, \tag{11}$$

zaś funkcje polityki (optymalne sterowania) to:

$$\begin{bmatrix} c(k_t) \\ l(k_t) \end{bmatrix} = \arg\max_{c_t, l_t} \{u(c_t, l_t) + \beta V[(1-\delta)k_t + f(k_t, l_t) - c_t]\}.$$

2.

Po obustronnym zróżniczkowaniu 11 po $k_t$ (na mocy twierdzenia o obwiedni) dostajemy:

$$V'(k_t) = \beta V'(k_{t+1})[(1-\delta) + f_1(k_t, l_t)]. \tag{12}$$

Następnie z warunków pierwszego rzędu dla prawej strony 11 mamy:

$$u_1(c_t, l_t) = \beta V'(k_{t+1}) \Rightarrow V'(k_{t+1}) = \beta^{-1} u_1(c_t, l_t) \tag{13}$$

i

$$u_2(c_t, l_t) + \beta V'(k_{t+1}) f_2(k_t, l_t) = 0. \tag{14}$$

Przesuwając indeks czasowy o jeden do przodu dla 12 otrzymujemy wyrażenie

$$V'(k_{t+1}) = \beta V'(k_{t+2})[(1-\delta) + f_1(k_{t+1}, l_{t+1})],$$

do którego podstawiamy drugą formę 13 i ponownie ją, ale z przesuniętym o jeden okres

do przodu indeksem czasu, by uzyskać:

$$\beta^{-1}u_1(c_t, l_t) = \beta\beta^{-1}u_1(c_{t+1}, l_{t+1})[(1-\delta) + f_1(k_{t+1}, l_{t+1})]. \tag{15}$$

Upraszając i porządkując 15, otrzymujemy warunek optymalizacji międzyokresowej (równanie Eulera):

$$\beta u_1(c_{t+1}, l_{t+1})[(1-\delta) + f_1(k_{t+1}, l_{t+1})] - u_1(c_t, l_t) = 0. \tag{16}$$

Z kolei do 14 również podstawiamy drugą formę 13, by, po uproszczeniu, dostać warunek optymalizacji wewnątrzokresowej:

$$u_2(c_t, l_t) + u_1(c_t, l_t)f_2(k_t, l_t) = 0. \tag{17}$$

Układ równań 16 i 17 jest domknięty[5] przez równanie ruchu 10.

Podstawiając postacie funkcyjne pochodnych funkcji użyteczności i produkcji do 17 i rozwiązując względem konsumpcji, mamy:

$$c_t = \mathcal{A}(1-l_t)\left(\frac{k_t}{l_t}\right)^{\alpha}, \tag{18}$$

gdzie $\mathcal{A} \equiv \frac{\tau(1-\alpha)}{1-\tau}$. Dokonując odpowiednich podstawień postaci funkcyjnych w 16 i dodatkowo wymieniając $c_t$ na prawą stronę 18, a następnie upraszając, uzyskujemy ostatecznie równanie Eulera z tylko jednym sterowaniem (podażą pracy):

$$\beta\left[\mathcal{A}\left(\frac{k_{t+1}}{l_{t+1}}\right)^{\alpha}\right]^{\mathcal{B}}(1-l_{t+1})^{-\theta}\left[(1-\delta) + \alpha\left(\frac{k_{t+1}}{l_{t+1}}\right)^{(\alpha-1)}\right] - \left[\mathcal{A}\left(\frac{k_t}{l_t}\right)^{\alpha}\right]^{\mathcal{B}}(1-l_t)^{-\theta} = 0, \tag{19}$$

przy czym $\mathcal{B} \equiv -\theta\tau + \tau - 1$. Zabiegu tego samego rodzaju można dokonać także na

[5] Warunek transwersalności:

$$\lim_{t\to\infty}\beta^t V'(k_t)k_t = 0,$$

(który, wykorzystując 13, można przedstawić jako $\lim_{t\to\infty}\beta^t\beta^{-1}u_1(c_{t-1}, l_{t-1})k_t = 0$, co poddane skróceniu oraz podstawieniu postaci funkcyjnej pochodnej funkcji użyteczności wygląda następująco: $\lim_{t\to\infty}\beta^{t-1}\left(c_{t-1}^{\tau}(1-l_{t-1})^{1-\tau}\right)^{-\theta}(1-l_{t-1})^{1-\tau}\tau c_{t-1}^{\tau-1}k_t = 0$) jest spełniony dla przyjętej parametryzacji, dopuszczalnych wartości podaży pracy i skończonej wartości kapitału w stanie ustalonym (która implikuje także skończoną konsumpcję).

równaniu ruchu 10:

$$k_{t+1} = (1-\delta)k_t + k_t^{\alpha} l_t^{1-\alpha} - \mathcal{A}(1-l_t)\left(\frac{k_t}{l_t}\right)^{\alpha}. \tag{20}$$

Wyrażenie 19 w postaci równania funkcyjnego wygląda następująco:

$$\beta\left\{\mathcal{A}\left[\frac{g\big(k_t, l(k_t)\big)}{l\Big(g\big(k_t, l(k_t)\big)\Big)}\right]^{\alpha}\right\}^{\mathcal{B}}\Big[1-l\Big(g\big(k_t, l(k_t)\big)\Big)\Big]^{-\theta}\left\{(1-\delta)+\alpha\left[\frac{g\big(k_t, l(k_t)\big)}{l\Big(g\big(k_t, l(k_t)\big)\Big)}\right]^{(\alpha-1)}\right\}$$
$$-\left\{\mathcal{A}\left[\frac{k_t}{l(k_t)}\right]^{\alpha}\right\}^{\mathcal{B}}[1-l(k_t)]^{-\theta}=0, \tag{21}$$

gdzie $k_{t+1}$ zostało wyeliminowane przez podstawienie uproszczonego równania ruchu 20 również przepisanego w formie równania funkcyjnego:

$$g(k_t, l(k_t)) = (1-\delta)k_t + k_t^{\alpha} l_t^{1-\alpha} - \mathcal{A}(1-l_t)\left(\frac{k_t}{l_t}\right)^{\alpha}. \tag{22}$$

Przydatne w następnym podpunkcie stają się wartości stanu ustalonego, które można (dla wielkości kapitału i podaży pracy) uzyskać rozwiązując złożony z dostosowanych równań 21 i 22 układ[6]:

$$\begin{cases} \beta\left[\mathcal{A}\left(\frac{k^*}{l^*}\right)^{\alpha}\right]^{\mathcal{B}}(1-l^*)^{-\theta}\left[(1-\delta)+\alpha\left(\frac{k^*}{l^*}\right)^{(\alpha-1)}\right]-\left[\mathcal{A}\left(\frac{k^*}{l^*}\right)^{\alpha}\right]^{\mathcal{B}}(1-l^*)^{-\theta}=0, \\ k^* = g(k^*, l^*). \end{cases}$$

Mając jego rozwiązanie, konsumpcję w stanie ustalonym można obliczyć korzystając np. z równania 18.

3. & 4.

W rozwiązaniu pominięto odpowiednik funkcji `utilramsey`, ponieważ ze względu na to, że w równaniu Eulera, eliminując konsumpcję, doszło do tak znaczących jego modyfikacji, wartości funkcji użyteczności i jej pochodnych stały się zbędne.

[6] Aby ułatwić rozwiązywanie tego układu równań analitycznie, pierwsze z nich można skrócić (dalej uproszczając), zaś drugie rozpisać, lecz wówczas postać ta nie będzie odpowiadała tej użytej do rozwiązania numerycznego z kodu w kolejnym podpunkcie.

Plik *prodframseyel.m*

```
function [alpha, f, f1, f2]=prodframseyel(k,l)

alpha=1/3;

f=k.^alpha.*(l.^(1-alpha));
f1=alpha*(k./l).^(alpha-1);
f2=(1-alpha)*(k./l).^alpha;
```

Koniec pliku *prodframseyel.m*

Plik *transramseyel.m*

```
function [g, g1, elem, A, delta]=transramseyel(k,l)

delta=.05;
tau=.35;
theta=2;

[alpha, f, f1, ~]=prodframseyel(k,l);

A=(tau/(1-tau))*(1-alpha);
elem=((A*(k./l).^alpha).^(-theta*tau+tau-1)).*(1-l).^(-theta);
g=(1-delta)*k+f-A*(1-l).*(k./l).^alpha;
g1=(1-delta)+f1;
```

Koniec pliku *transramseyel.m*

Plik *eqeulerramseyel.m*

```
function y=eqeulerramseyel(k1, l1, k2, l2)

discount=1/1.04;

[~, ~, elem1]=transramseyel(k1, l1);
```

```
[~, gprim2, elem2]=transramseyel(k2, l2);

y=discount*elem2.*gprim2-elem1;
```

Koniec pliku *eqeulerramseyel.m*

Plik *ramseyelstst.m*

```
function y=ramseyelstst(x)
y=zeros(2,1);
y(1)=eqeulerramseyel(x(1), x(2), x(1), x(2));
y(2)=transramseyel(x(1), x(2))-x(1);
```

Koniec pliku *ramseyelstst.m*

Plik *exrelss.m*

```
function [kss, lss, css]=exrelss

ss=zeros(2,1);
[ss]=newton ('ramseyelstst', [1 0.5]'); % l zawiera się w przedziale [0,1]
kss=ss(1);
lss=ss(2);
[~, ~, ~, A]=transramseyel(kss,lss);
css=A*(1-lss)*(kss/lss)^prodframseyel(kss,lss); % (pierwszy) wynik ostatniej
funkcji to alfa
```

Koniec pliku *exrelss.m*

Plik *resideulerramseyel.m*

```
function ret=resideulerramseyel(v)

kss=exrelss;
kmin=ceil(kss)/2;
kmax=ceil(kss);
```

```
k1=chebnod(10, kmin, kmax);
l1=chebphi(10)*v;
k2=transramseyel(k1, l1);
l2=chebval(v, k2, kmin, kmax);
ret=eqeulerramseyel(k1, l1, k2, l2);
```

Koniec pliku *resideulerramseyel.m*

Plik *exrelcol.m*

```
%=========================================================================
%    Rozwiązanie zadania 6.1.(3&4) z książki Klimy (2005).
%    Autor: Piotr Pieniążek, CC BY-NC-SA
%=========================================================================
close all
clear all
clc
%-------------------------------------------------------------------------
% 1. Obliczanie wartości współczynników wielomianu aproksymującego funkcję
% polityki dla podaży pracy. Przy wyborze wartości startowych należy mieć na
% uwadze wartości stanu ustalonego. W startowym przybliżeniu dla k=2.25 "l"
% będzie równe 0.25, a współczynnik kierunkowy ujemny, bo wtedy funkcja
% polityki dla konsumpcji jest - zgodnie z intuicją ekonomiczną i przykładem
% Klimy - rosnąca. Odpowiada to przypadkowi, w którym efekt dochodowy
% dominuje nad efektem substytucyjnym, gdy czynniki produkcji są
% komplementarne (df2/dk>0). Dla jakościowo różnych (od tych poniżej
% przyjętych) wartości startowych otrzymywane są funkcje wychodzące w pobliżu
% stanu ustalonego kapitału poza przedział [0,1] i powodujące, że wartości
% bezwzględne reszt w równaniu Eulera są zdecydowanie wyższe, a zmienne
% modelu nie zbiegają wówczas do swego stanu ustalonego. Natomiast dla nawet
% nieco innych ilościowo (od tych przyjętych poniżej) wartości startowych,
% reszty w węzłach kolokacji się niepokojąco nie zerują (czy wynika to z
% braku precyzji maszynowej?).
```

```
%-------------------------------------------------------------------------

a=newton ('resideulerramseyel', [0.25 -0.1 0 0 0 0 0 0 0 0]');

kss=exrelss;
kmin=ceil(kss)/2;
kmax=ceil(kss);
step=0.01;
k=(kmin:step:kmax)';
l=chebval (a, k, kmin, kmax);
subplot(5,2,1);
plot(k,l);
xlabel('k')
ylabel('l')
title(['Aproksymowana wielomianem funkcja polityki l(k)'])
axis('tight')
grid on

z=transramseyel (k,l);
R=eqeulerramseyel (k, l, z, chebval (a, z, kmin, kmax));
subplot(5,2,2);
plot(k,R);
xlabel('k')
title(['Reszta równania Eulera'])
axis('tight')
grid on

alpha=prodframseyel(k,l);
[~, ~, ~, A]=transramseyel(k,l);
c=A*(1-l).*(k./l).^alpha;
```

```
subplot(5,2,3);
plot(k,c);
xlabel('k')
ylabel('c')
title(['Aproksymowana wielomianem funkcja polityki c(k)'])
axis('tight')
grid on

%------------------------------------------------------------------------
% 2. Symulacja ścieżek czasowych zmiennych sterujących i stanu.
%------------------------------------------------------------------------

T=100;
t=1:1:T;

K=zeros(T,1);
L=zeros(T,1);
K(1)=kmin;
L(1)=chebval (a, K(1), kmin, kmax);

for i=(2:T)

   K(i)=transramseyel (K(i-1), L(i-1));
   L(i)=chebval (a, K(i), kmin, kmax);

end

subplot(5,2,4);
plot(t,K);
legend('K',1)
```

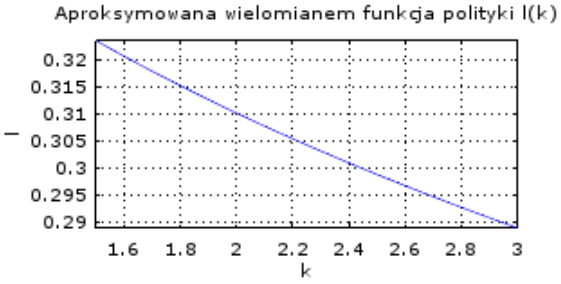
```
xlabel('czas [okresy]')
ylabel('k')
title(['Ewolucja zasobu kapitału'])
grid on

subplot(4,2,5);
plot(t,L);
legend('L',1)
xlabel('czas [okresy]')
ylabel('l')
title(['Ewolucja nakładu pracy'])
grid on

C=A*(1-L).*(K./L).^alpha;
subplot(5,2,6);
plot(t,C);
legend('C',1)
xlabel('czas [okresy]')
ylabel('c')
title(['Ewolucja konsumpcji'])
grid on

[~, Y]=prodframseyel (K,L);
subplot(5,2,7);
plot(t,Y);
legend('Y',1)
xlabel('czas [okresy]')
ylabel('y')
title(['Ewolucja produkcji'])
grid on
```

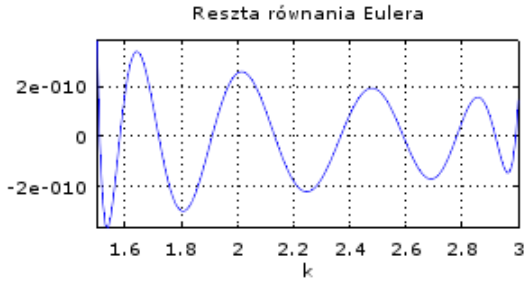
```
I=Y-C;
subplot(5,2,8);
plot(t,I);
legend('I',1)
xlabel('czas [okresy]')
ylabel('i')
title(['Ewolucja nakładów inwestycyjnych'])
grid on

[~, ~, ~, ~, d]=transramseyel (K,L);
D=ones(T,1)*d;
r=alpha*(K.^(alpha-1)).*L.^(1-alpha)-D;
subplot(5,2,9);
plot(t,r);
xlabel('czas [okresy]')
ylabel('r')
title(['Ewolucja stopy procentowej (netto)'])
grid on

w=(1-alpha)*(K./L).^(alpha);
subplot(5,2,10);
plot(t,w);
xlabel('czas [okresy]')
ylabel('w')
title(['Ewolucja płacy'])
grid on
```

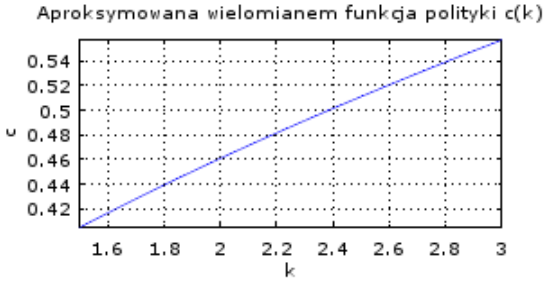
Koniec pliku *exrelcol.m*

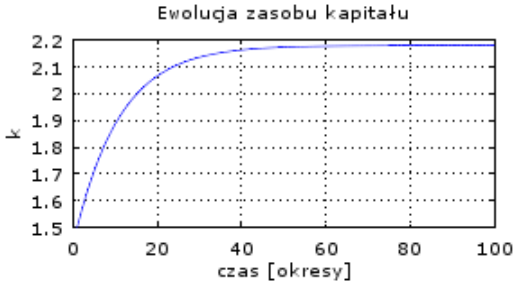
Wykonanie programu powinno wygenerować następujące wykresy[7]:

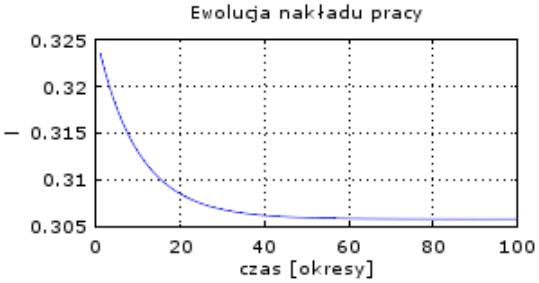
[7]Dla alternatywnej parametryzacji (np. $\alpha = 0,5$) czy niższych wartości kapitału startowego (np. `kmin=ceil(kss)/4`) można uzyskać wybrzuszoną krzywą ewolucji inwestycji (tzw. *hump-shaped*).

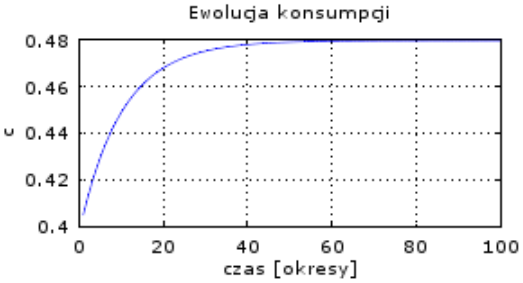

Aproksymowana wielomianem funkcja polityki l(k)
l
0.32
0.315
0.31
0.305
0.3
0.295
0.29
1.6 1.8 2 2.2 2.4 2.6 2.8 3
k

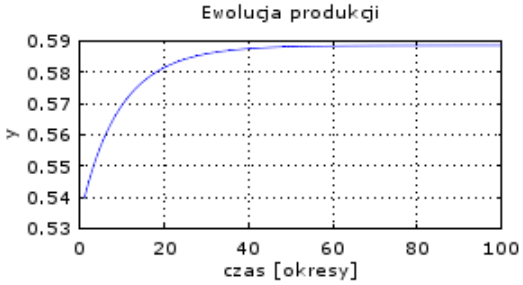

Reszta równania Eulera
2e-010
0
-2e-010
1.6 1.8 2 2.2 2.4 2.6 2.8 3
k

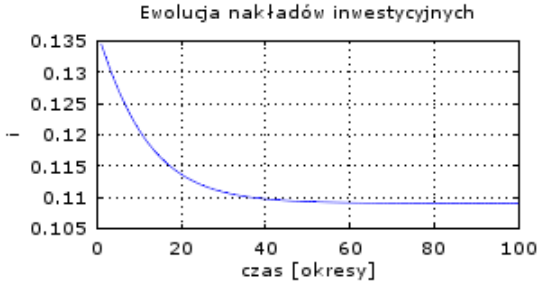

Aproksymowana wielomianem funkcja polityki c(k)
c
0.54
0.52
0.5
0.48
0.46
0.44
0.42
1.6 1.8 2 2.2 2.4 2.6 2.8 3
k

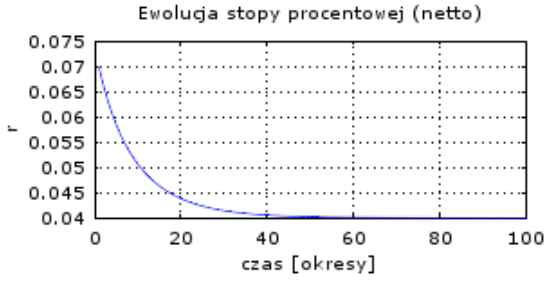

Ewolucja zasobu kapitału
k
2.2
2.1
2
1.9
1.8
1.7
1.6
1.5
0 20 40 60 80 100
czas [okresy]

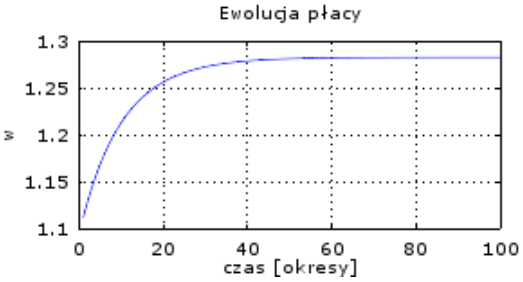

Ewolucja nakładu pracy
l
0.325
0.32
0.315
0.31
0.305
0 20 40 60 80 100
czas [okresy]


Ewolucja konsumpcji
c
0.48
0.46
0.44
0.42
0.4
0 20 40 60 80 100
czas [okresy]


Ewolucja produkcji
y
0.59
0.58
0.57
0.56
0.55
0.54
0.53
0 20 40 60 80 100
czas [okresy]


Ewolucja nakładów inwestycyjnych
i
0.135
0.13
0.125
0.12
0.115
0.11
0.105
0 20 40 60 80 100
czas [okresy]


Ewolucja stopy procentowej (netto)
r
0.075
0.07
0.065
0.06
0.055
0.05
0.045
0.04
0 20 40 60 80 100
czas [okresy]


Ewolucja płacy
w
1.3
1.25
1.2
1.15
1.1
0 20 40 60 80 100
czas [okresy]

# 5 Bibliografia

[1] Klima, Grzegorz, *Programowanie dynamiczne i modele rekursywne w ekonomii. Zagadnienia analityczne i metody numeryczne z przykładowymi implementacjami w języku Matlab/Octave*, *Materiały i Studia*, Zeszyt nr 201, 2005. 4

[2] Aruoba, S.B., J. Fernández-Villaverde, *A Comparison of Programming Languages in Macroeconomics*, *Journal of Economic Dynamics & Control*, 58, ss. 265–273, 2015. Wcześniejsze wolnodostępne wersje znajdują się tu i tu. 22, 27

[3] Heer, B., A. Maußner, *Dynamic General Equilibrium Modeling. Computational Methods and Applications*, $2^{\text{gie}}$ wydanie, Springer, Berlin i Heidelberg 2009. 27