```
> with(plots):Digits:=100:interface(displayprecision=10):with
  (linalg):


> N:=6;
  KK:=5;
  ### Change of notation: Now the indices K[i] are in order not
  alpha's
  # vector U shows  if the branch is up (1) or down (0)
  K:=vector(N,[]):
  U:=vector(N,[]):
  UU:=vector(N,[]):
  alpha:=vector(N,[]):
  gamm:=vector(N,[]):#heiths of lower ends of hanging branches
                          # alpha[i]+gamma[i]<1   !!!!!!!
                          #if gamma[i]>0 then U[i]:=0, UU[i]:=1
  for j from 1 to N do
  U[j]:=0.0:
  UU[j]:=0:
  gamm[j]:=0:
  od:
  alpha[1]:=0.4:K[1]:=1:U[K[1]]:=1.0:
  alpha[2]:=0.5:K[2]:=3:U[K[2]]:=0.0:gamm[K[2]]:=0.3:UU[K[2]]:=1:#

  alpha[3]:=0.3:K[3]:=4:U[K[3]]:=0.0:  #
  alpha[4]:=0.6:K[4]:=5:U[K[4]]:=1.0:
  alpha[5]:=0.7:K[5]:=6:U[K[5]]:=0.0:gamm[K[5]]:=0.2:UU[K[5]]:=1:
  print(`alpha =`,alpha);
  print(`K =`,K);
  print(`U =`,U);
  print(`UU =`,UU);
  i:='i':
  beta:=N-KK+sum(alpha[i],i=1..KK);i:='i':
  delta1:=(xw,yw)-> piecewise(xw<=yw,0,1):
  bb:=vector(N+1,[]):
  for j from 1 to N do
  b[j]:=(j-1-sum((1-alpha[i])*delta1( j,K[i]) ,i=1..KK))/beta:
   od:  i:='i':
  b[N+1]:=1:
  ag:=vector(N,[]):
  al:=vector(N,[]):
  a:=vector(N,[]):
  c:=vector(N,[]):
  for j from 1 to N do
  bb[j]:=b[j];
  ag[j]:=beta*b[j];
  al[j]:=-1+beta*b[j+1];
```

```
od:
for j from 1 to N do
a[j]:=(j-1-sum((1-alpha[i])*delta1( j,K[i]-U[j]) ,i=1..KK)-gamm
[j]);
 od:

print(`b =`,bb);
print(`ag =`,ag);
print(`al  =`,al);
print(`a =`,a);
print(`gamma =`,gamm);
```

$$N := 6$$

$$KK := 5$$

$$alpha =, \left[0.4000000000 \quad 0.5000000000 \quad 0.3000000000 \quad 0.6000000000 \quad 0.7000000000 \quad \alpha_6 \right]$$

$$K =, \left[1 \quad 3 \quad 4 \quad 5 \quad 6 \quad K_6 \right]$$

$U =,$ [1.0000000000, 0.0000000000, 0.0000000000, 0.0000000000, 1.0000000000, 0.0000000000]

$$UU =, \left[0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \right]$$

$$\beta := 3.5000000000$$

$b =,$ [0.0000000000, 0.1142857143, 0.4000000000, 0.5428571429, 0.6285714286, 0.8000000000, bb_7]

$ag =,$ [0.0000000000, 0.4000000000, 1.4000000000, 1.9000000000, 2.2000000000, 2.8000000000]

$al =,$ [-0.6000000000, 0.4000000000, 0.9000000000, 1.2000000000, 1.8000000000, 2.5000000000]

$a =,$ [-0.6000000000, 0.4000000000, 1.1000000000, 1.9000000000, 1.8000000000, 2.6000000000]

$$gamma =, \left[0 \quad 0 \quad 0.3000000000 \quad 0 \quad 0 \quad 0.2000000000 \right]$$

```
> # ag shows maximal digit (greedy)
  #  al shows minimal digit (lazy)  #### if ag[j]=al[j] then j is
  onto branch and there is
                                                                  #
  no choice there
  # a shows digits assigned automatically using the vector U: U(j)
  =1 lazy
  #                                                          U(j)=
  0 greedy
  # we can assign digit a arbitrarily between minimum and maximum
```

```
   and then put 2 into vector U

   #a[3]:=1.0: U[3]:=2.0:
   #a[5]:=1.90:U[5]:=2.0:
   #a[6]:=2.70:U[6]:=2.0:
   #print(`U =`,U);
   #print(`a =`,a);

   # Now we will name points c[i] (there is KK + number of 2's in U
   points c[i])
   # and create a vectors sidec[], ineqc[],signc[] which shows the
   character of the point c[i]
   Kc:=KK:# new number of c points
   for j from 1 to N do if UU[j]=1 then Kc:=Kc+1 fi od:
   print(`Kc =`,Kc);
   c:=vector(2*N,[]):
   sidec:=vector(2*N,[]):# 1 left (use uT), 0 right (use T)


   cj:=1:# this is the new index for c points
   for j from 1 to KK do
   if (U[K[j]]=0 and UU[K[j]]=0) then   c[cj]:=b[K[j]+1]; sidec[cj]
   :=0;cj:=cj+1 fi;
    if U[K[j]]=1 then  c[cj]:=b[K[j]]; sidec[cj]:=1;cj:=cj+1 fi;
    if (U[K[j]]=0 and UU[K[j]]=1) then  c[cj]:=b[K[j]]; sidec[cj]:=
   1;cj:=cj+1 ;
                        c[cj]:=b[K[j]+1]; sidec[cj]:=0;cj:=cj+1 ;
   fi;
   od;
   print(`c =`,c);
   print(`sidec =`,sidec);


>
   maa:=a[2]-a[1]:# maximum a[i+1]-a[i]
   for i from 3 to N do
   if (a[i]-a[i-1])>maa then maa:=(a[i]-a[i-1]) fi
   od;#
   maa;
   beta_max:=evalf(1+(a[N]-a[1])/maa);
> if beta> beta_max then print("ERROR") fi;


>
   uint_of_x:=x->piecewise(x<b[2],1,# This function needs additions
   by hand for
                                    # N>9 . Automathic procedure
```

```
causes plotting problems
                                        # but is used in other
programs
                              x<b[3],2,
                              x<b[4],3,
                              x<b[5],4,
                              x<b[6],5,
                              x<b[7],6,
                              x<b[8],7,
                              x<b[9],8,
                                   9);
int_of_x:=x->piecewise(x<=b[2],1,# This function needs additions
by hand for
                                        # N>9 . Automathic procedure
causes plotting problems
                                        # but is used in other
programs
                              x<=b[3],2,
                              x<=b[4],3,
                              x<=b[5],4,
                              x<=b[6],5,
                              x<=b[7],6,
                              x<=b[8],7,
                              x<=b[9],8,
                                   9);
x:='x':
uT:=x->beta*x-a[uint_of_x(x)];
T:=x->beta*x-a[int_of_x(x)];
Tc:=vector(2*N,[]):
for j from 1 to Kc do
if sidec[j]=0 then    Tc[j]:=T(c[j]);
              else    Tc[j]:=uT(c[j])fi;
od:
print(`Tc = `, Tc);

plot([uT(x),x,0,1,1-alpha[1],1-alpha[2]],x=0..1,thickness=[2,1,
1,1,1,1,1]);
plot([T(x),x,0,1,alpha[1],alpha[2],0.2],x=0..1,thickness=[2,1,1,
1,1,1,1,1]);
```

$$Kc =, 7$$

$$c =, \left[0.0000000000, 0.4000000000, 0.5428571429, 0.6285714286, 0.6285714286, 0.8000000000, 1, c_8, c_9, c_{10}, c_{11}, c_{12}\right]$$

$$sidec =, \left[\begin{array}{ccccccc} 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{array} \; sidec_8 \; sidec_9 \; sidec_{10} \; sidec_{11} \; sidec_{12}\right]$$

1.0000000000

$\beta max := 4.2000000000$

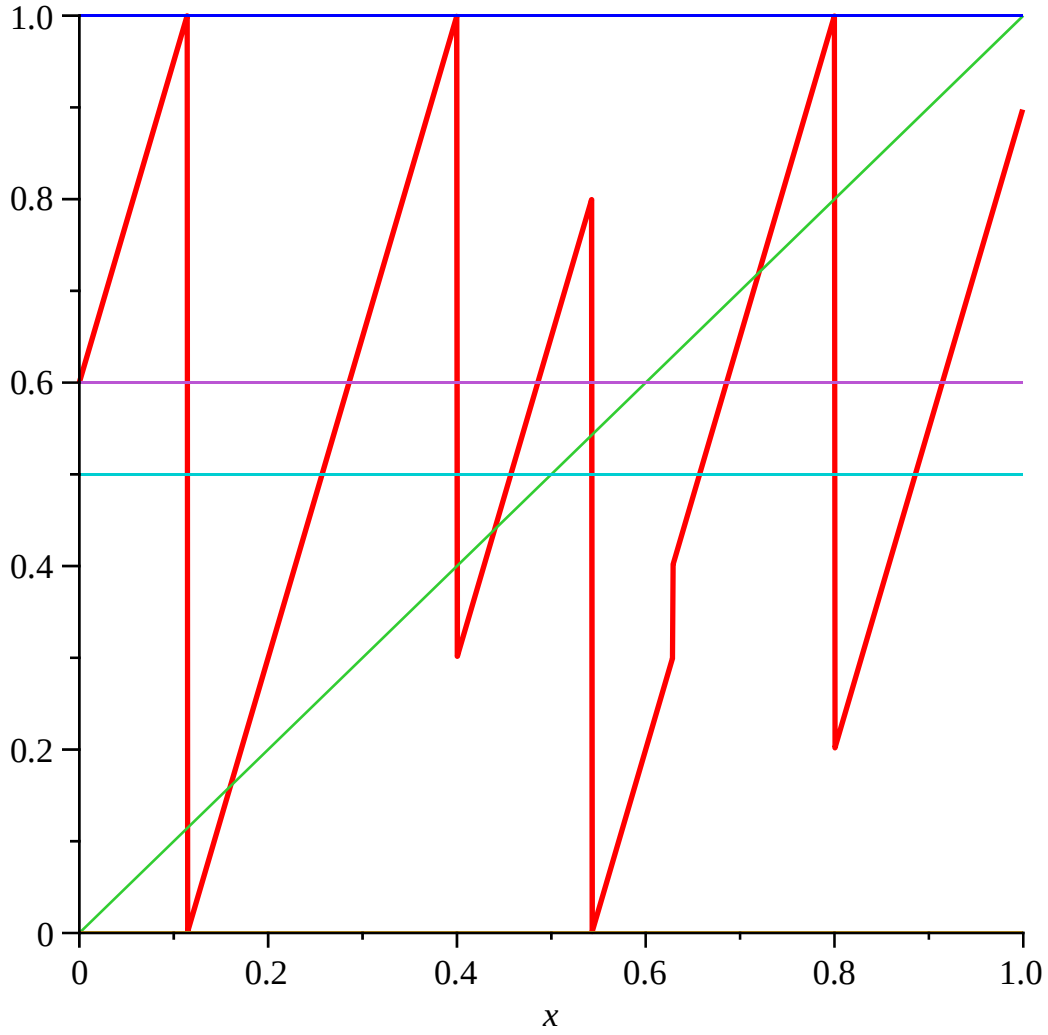
$uint\_of\_x := x \rightarrow piecewise(x < b_2, 1, x < b_3, 2, x < b_4, 3, x < b_5, 4, x < b_6, 5, x < b_7, 6, x < b_8, 7, x < b_9, 8, 9)$

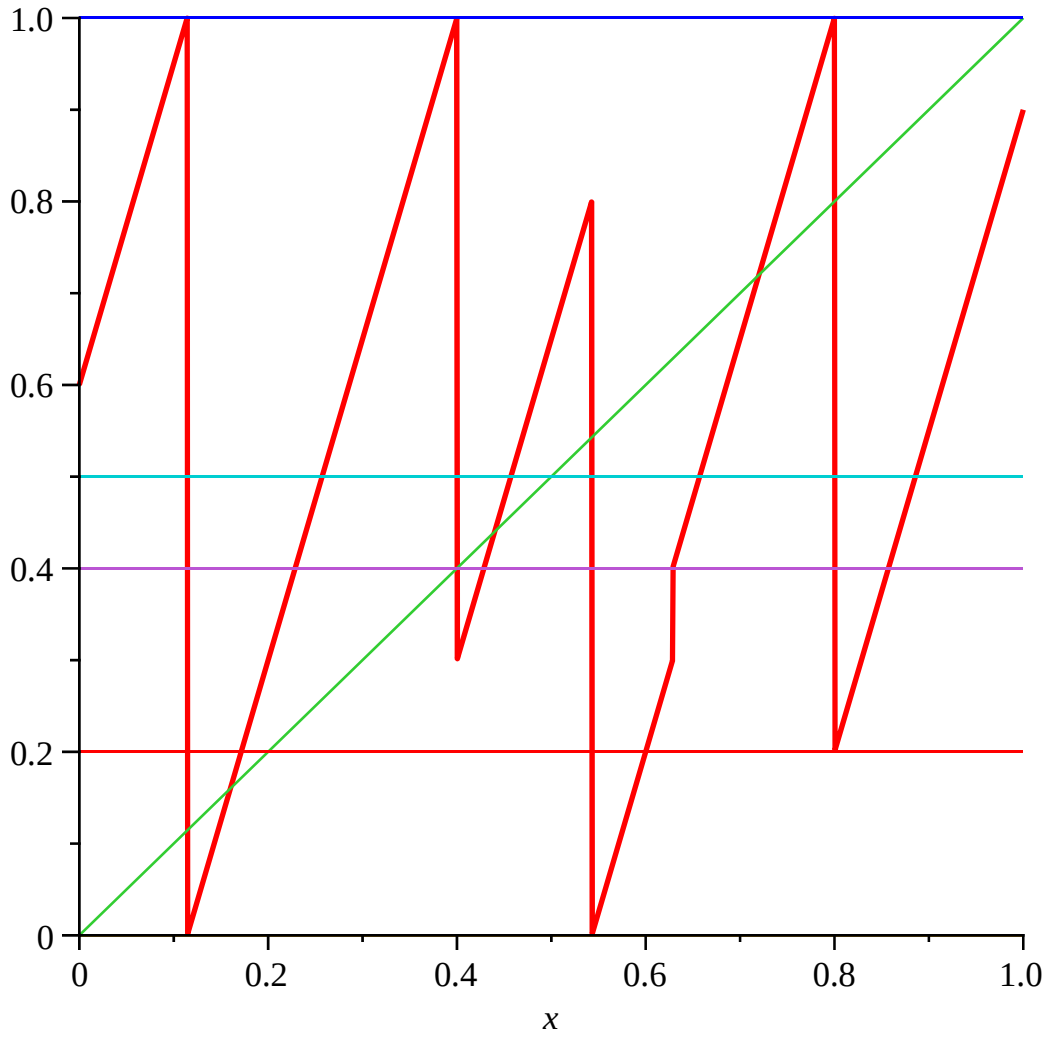
$int\_of\_x := x \rightarrow piecewise(x \leq b_2, 1, x \leq b_3, 2, x \leq b_4, 3, x \leq b_5, 4, x \leq b_6, 5, x \leq b_7, 6, x \leq b_8, 7, x \leq b_9, 8, 9)$

$$uT := x \rightarrow \beta x - a_{uint\_of\_x(x)}$$

$$T := x \rightarrow \beta x - a_{int\_of\_x(x)}$$

$Tc = , [0.6000000000, 0.3000000000, 0.8000000000, 0.3000000000, 0.4000000000, 0.2000000000, 0.9000000000, Tc_8, Tc_9, Tc_{10}, Tc_{11}, Tc_{12}]$






```
>
> ud:=vector(50):Digits:=100;NN:=50;
  d:=vector(50):
  xx:=c[3];#evalf(rand()/10^12);
  xxt:=xx:
  for i from 1 to NN do
  ud[i]:=a[uint_of_x(xxt)];
  xxt:=uT(xxt);
  od:
  xxt:=xx:
  for i from 1 to NN do
  d[i]:=a[int_of_x(xxt)];
  xxt:=T(xxt);
  od:
  print(ud);
  uls_it_x:=evalf(sum(ud[j1]/beta^j1,j1=1..NN));
  print(d);
  ls_it_x:=evalf(sum(d[j1]/beta^j1,j1=1..NN));
  terr:=xx-uls_it_x;
  err:=xx-ls_it_x;
```

$$Digits := 100$$

$$NN := 50$$

$$xx := 0.5428571429$$

[1.9000000000, −0.6000000000, 1.9000000000, 0.4000000000, 0.4000000000, 1.8000000000,
1.1000000000, 1.9000000000, −0.6000000000, 2.6000000000, 0.4000000000,
1.8000000000, 2.6000000000, 2.6000000000, 1.1000000000, 1.1000000000,
1.9000000000, −0.6000000000, 1.8000000000, 2.6000000000, 1.1000000000,
1.8000000000, 1.1000000000, 1.9000000000, −0.6000000000, 2.6000000000,
1.1000000000, 1.1000000000, 1.8000000000, 1.8000000000, 1.8000000000,
1.8000000000, 1.9000000000, 0.4000000000, 1.1000000000, 1.8000000000,
1.9000000000, 0.4000000000, 1.1000000000, 1.1000000000, 1.1000000000,
1.8000000000, 1.8000000000, 1.8000000000, 1.1000000000, 1.1000000000,
1.1000000000, 1.1000000000, 1.9000000000, −0.6000000000]

$$uIs\_it\_x := 0.5428571429$$

[1.1000000000, 1.8000000000, 2.6000000000, 2.6000000000, 1.9000000000, −0.6000000000,
1.8000000000, 1.9000000000, 0.4000000000, 0.4000000000, 2.6000000000,
1.9000000000, 0.4000000000, 1.1000000000, 1.1000000000, 1.8000000000,
1.1000000000, 1.8000000000, 1.9000000000, −0.6000000000, 2.6000000000,
1.8000000000, 1.8000000000, 2.6000000000, 1.8000000000, 1.8000000000,
1.8000000000, 2.6000000000, 1.8000000000, 1.9000000000, −0.6000000000,
1.8000000000, 2.6000000000, 1.8000000000, 1.1000000000, 1.1000000000,
1.8000000000, 1.9000000000, 0.4000000000, 0.4000000000, 0.4000000000,
2.6000000000, 0.4000000000, 1.8000000000, 1.9000000000, 0.4000000000,
0.4000000000, 1.9000000000, 0.4000000000, 0.4000000000]

$$Is\_it\_x := 0.5428571429$$

$$terr := 4.252662205\ 10^{-28}$$

$$err := 3.455028762\ 10^{-28} \quad \textbf{(1)}$$

```
>
> NN:=50; chi:=(x1,x2,t)->piecewise(t<x1, 0, t<=x2, 1, 0);
  uchi:=(x1,x2,t)->piecewise(t<x1, 0, t<x2, 1, 0);

  #Expansion of c1, c2 ... and all the S's

  for i from 1 to Kc do
  xxt:=c[i];
       for n from 1 to NN+1 do
        if sidec[i]=1 then intx:=uint_of_x(xxt) else intx:=
  int_of_x(xxt) fi;
          dc[i,n]:=a[intx];
          ic[i,n]:=intx-1;
         if sidec[i]=0 then
              for ii from 1 to Kc do
                   if xxt>c[ii] then cc[i,ii,n]:=1 else cc[i,ii,n]
  :=0 fi;
```

```
                od;
                        else
                for ii from 1 to Kc do
                     if xxt<c[ii] then cc[i,ii,n]:=1 else cc[i,ii,n]
:=0 fi;
                od;
        fi;
        valc[i,n]:=xxt;
        if sidec[i]=1 then xxt:=uT(xxt) else xxt:=T(xxt)fi;
        od:
Is_it_x:=sum(dc[i,j1]/beta^j1,j1=1..NN);
S[i]:=sum(ic[i,j1+1]/beta^(j1+1),j1=1..NN);
od;
for i from 1 to Kc do
for j from 1 to Kc do
SS[i,j]:=sum(cc[i,j,j1+1]/beta^(j1+1),j1=1..NN);

#print(`SS[`,i,j,`] =`,SS[i,j]):
od;od:
```

$$NN := 50$$

$$\chi := (x1, x2, t) \rightarrow piecewise(t < x1, 0, t \leq x2, 1, 0)$$

$$uchi := (x1, x2, t) \rightarrow piecewise(t < x1, 0, t < x2, 1, 0)$$

$$xxt := 0.0000000000$$

$$Is\_it\_x := -3.615709297\ 10^{-28}$$

$$S_1 := 0.2841287774$$

$$xxt := 0.4000000000$$

$$Is\_it\_x := 0.4000000000$$

$$S_2 := 0.1948632372$$

$$xxt := 0.5428571429$$

$$Is\_it\_x := 0.5428571429$$

$$S_3 := 0.4829575063$$

$$xxt := 0.6285714286$$

$$Is\_it\_x := 0.6285714286$$

$$S_4 := 0.1948632372$$

$$xxt := 0.6285714286$$

$$Is\_it\_x := 0.6285714286$$

$$S_5 := 0.2189405168$$

$$xxt := 0.8000000000$$

$$Is\_it\_x := 0.8000000000$$

$$S_6 := 0.1373078637$$

$$xxt := 1$$

$Is\_it\_x := 1.0000000000$

$S_7 := 0.4876580231$

```
>
   MM:=matrix(Kc,Kc,[]):
   for i from 1 to Kc do
   for j from 1 to Kc do

   MM[j,i]:=-SS[i,j];
   od;od;
   print(`MM = `,MM);
   print(`1/beta =`, 1/beta);

   print(`eigenvalues MM =`,eigenvalues(MM));

   ve:=vector(Kc,[]):
   for i from 1 to Kc do
   ve[i]:=1/beta;

   MM[i,i]:=MM[i,i]+1/beta;
   od:

   print(MM);
   print(ve);



   DD:=linsolve(MM,ve);
```

$MM = , [[-0.0000000000, -0.0000000000, -0.1142857143, -0.1142857143,$
$-0.0000000000, -0.0000000000, -0.1142857143],$
$[-0.0300355383, -0.0822210584, -0.1137253258, -0.0320646559, -0.0234917310,$
$-0.1051243840, -0.1074209549],$
$[-0.0305795604, -0.0888853291, -0.1137252941, -0.0254003852, -0.1070284614,$
$-0.1070284614, -0.1074205675],$
$[-0.1123676414, -0.0907893219, -0.1117766222, -0.0234963924, -0.1075724593,$
$-0.1075724593, -0.0835493357],$
$[-0.1123676414, -0.0907893219, -0.1117766222, -0.0234963924, -0.1075724593,$
$-0.1075724593, -0.0835493357],$
$[-0.1142726459, -0.1141256269, -0.0299885410, -0.0001600874, -0.1142399750,$
$-0.1142399750, -0.0816453409],$
$[-0.1142857143, -0.1142857143, -0.0000000000, -0.0000000000, -0.1142857143,$
$-0.1142857143, -0.0000000000]]$

$1/beta =, 0.2857142857$

eigenvalues MM =, −0.5245883691, −0.0280369835 + 0.0238555228 I, −0.0280369835 − 0.0238555228 I, 0.0746292026 + 0.0211520154 I, 0.0746292026 − 0.0211520154 I, −0.0098512486, $-6.938893904\ 10^{-18}$

[[0.2857142857, −0.0000000000, −0.1142857143, −0.1142857143, −0.0000000000, −0.0000000000, −0.1142857143],
[−0.0300355383, 0.2034932273, −0.1137253258, −0.0320646559, −0.0234917310, −0.1051243840, −0.1074209549],
[−0.0305795604, −0.0888853291, 0.1719889916, −0.0254003852, −0.1070284614, −0.1070284614, −0.1074205675],
[−0.1123676414, −0.0907893219, −0.1117766222, 0.2622178933, −0.1075724593, −0.1075724593, −0.0835493357],
[−0.1123676414, −0.0907893219, −0.1117766222, −0.0234963924, 0.1781418264, −0.1075724593, −0.0835493357],
[−0.1142726459, −0.1141256269, −0.0299885410, −0.0001600874, −0.1142399750, 0.1714743107, −0.0816453409],
[−0.1142857143, −0.1142857143, −0.0000000000, −0.0000000000, −0.1142857143, −0.1142857143, 0.2857142857]]

[0.2857142857, 0.2857142857, 0.2857142857, 0.2857142857, 0.2857142857, 0.2857142857, 0.2857142857]

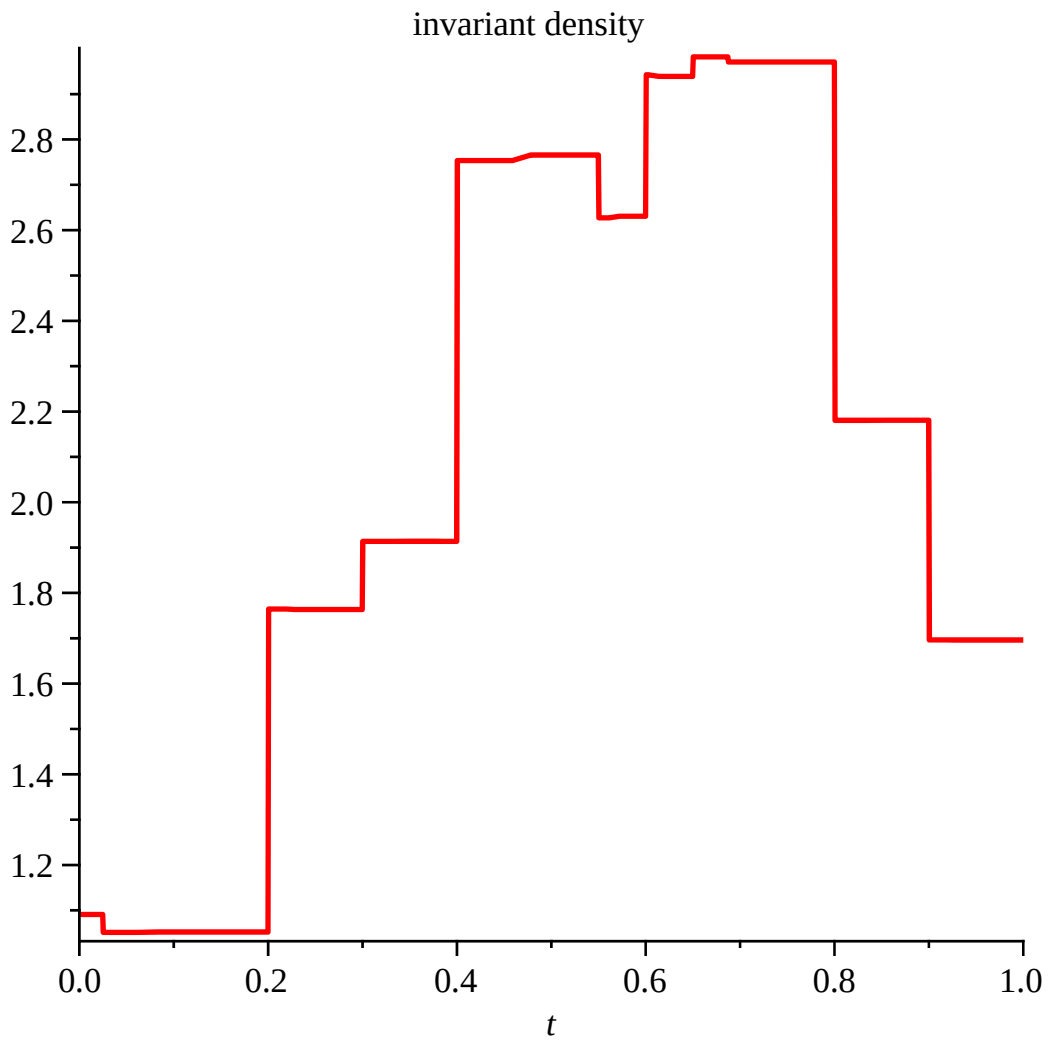
DD := [−0.6130909327, −1.0755319036, −1.5543991481, −1.6518894256, −1.6518894256, −1.2255846333, −0.8264387581] **(2)**

```
>
        density:=proc(t) local j, den;
                    den:=1/beta;
                   for j from 1 to Kc do
                   if sidec[j]=0 then
                     den:=den+ DD[j]*sum((chi(0,valc[j,i1+1],t))
   /beta^(i1+1),i1=1..50) fi;

                    if sidec[j]=1 then
                     den:=den+ DD[j]*sum((uchi(valc[j,i1+1],1,t))
   /beta^(i1+1),i1=1..50) fi;


                    od;
                   return den;
              end proc;
   #Normalizing factor

   NC:=1/beta:
   for j from 1 to KK do
   if sidec[j]=0 then
    NC:=NC+DD[j]*sum((valc[j,i1+1])/beta^(i1+1),i1=1..50) fi;
```

```
    if  sidec[j]=1  then
    NC:=NC+DD[j]*sum((1-valc[j,i1+1])/beta^(i1+1),i1=1..50)  fi;

     od:

    print(`NC = `,NC);

    plot([(1/NC)*density(t)],t=0..1-0.00000000001,title="invariant
    density",thickness=2);
```

```
density := proc(t)
    local j, den;
    den := 1/beta;
    for j to Kc do
        if sidec[j] = 0 then
            den := den + DD[j]*(sum(chi(0, valc[j, i1 + 1], t)/beta^(i1 + 1), i1 = 1..50))
        end if;
        if sidec[j] = 1 then
            den := den + DD[j]*(sum(uchi(valc[j, i1 + 1], 1, t)/beta^(i1 + 1), i1 = 1..50))
        end if
    end do;
    return den
end proc
```

$$NC = , -0.1605796837$$




```
> >
> 
#check density greedy
#preimages
for j6 from 1 to KK-1 do
y[j6]:=alpha[j6]+(alpha[j6+1]-alpha[j6])*rand()/10^12;
od:
y[0]:=alpha[1]*rand()/10^12:
y[KK]:=alpha[KK]+(1-alpha[KK])*rand()/10^12;
for j6 from 0 to KK do
for i3 from 1 to N do
pre[i3]:=(y[j6]+a[i3])/beta;
od;
#plot([T(t),0,1,y[j6],tT(tc[1]),tT(tc[2])],t=0..1,color=[red,
black,black,green,yellow,yellow]);
su:=0:
for i3 from 1 to N do
if (pre[i3]>=b[i3] and pre[i3]<=b[i3+1]) then
 su:=su+density(pre[i3])/beta;
print(i3);
fi;
od;
```

```
err[j6]:=density(y[j6])-su;
od;

for j6 from 0 to KK do
print(`y =`,y[j6]);
print(`err[`,j6,`]=`,err[j6]);
od;
```

$$y_5 := 0.9527868053$$

$$su := 0$$

2

4

$$err_0 := 2.738265462\ 10^{-29}$$

$$su := 0$$

2

3

5

6

$$err_1 := 2.738265462\ 10^{-29}$$

$$su := 0$$

2

3

5

6

$$err_2 := 2.738265462\ 10^{-29}$$

$$su := 0$$

2

3

6

$$err_3 := 2.738265462\ 10^{-29}$$

$$su := 0$$

1

2

3

5

6

$$err_4 := 2.738265462\ 10^{-29}$$

$$su := 0$$

1

2

$$err_5 := -4.430696884\ 10^{-29}$$

$$y =, 0.1710208227$$

$$err[, 0,]=, 2.738265462\ 10^{-29}$$

$$y =, 0.4395718861$$

$$err[, 1,]=, 2.738265462\ 10^{-29}$$

$$y =, 0.4613720367$$

$$err[, 2,]=, 2.738265462\ 10^{-29}$$

$$y =, 0.3067272511$$

$$err[, 3,]=, 2.738265462\ 10^{-29}$$

$$y =, 0.6800187484$$

$$err[, 4,]=, 2.738265462\ 10^{-29}$$

$$y =, 0.9527868053$$

$$err[, 5,]=, -4.430696884\ 10^{-29}$$

```
>
#check density greedy
#preimages
y:=evalf(rand()/10^12);#0.50100400000000;
for i2 from 1 to N do
pre[i2]:=(y+a[i2])/beta;
od;
plot([T(t),0,1,y,T(c[1]),T(c[2])],t=0..1,color=[red,black,black,
green,yellow,yellow]);
su:=0:
for i2 from 1 to N do
if (pre[i2]>=b[i2] and pre[i2]<=b[i2+1]) then
 su:=su+density(pre[i2])/beta;
print(i2);
fi;
od;
err2:=density(y)-su;
```

$$y := 0.4122862858$$

$$pre_1 := -0.0536324898$$

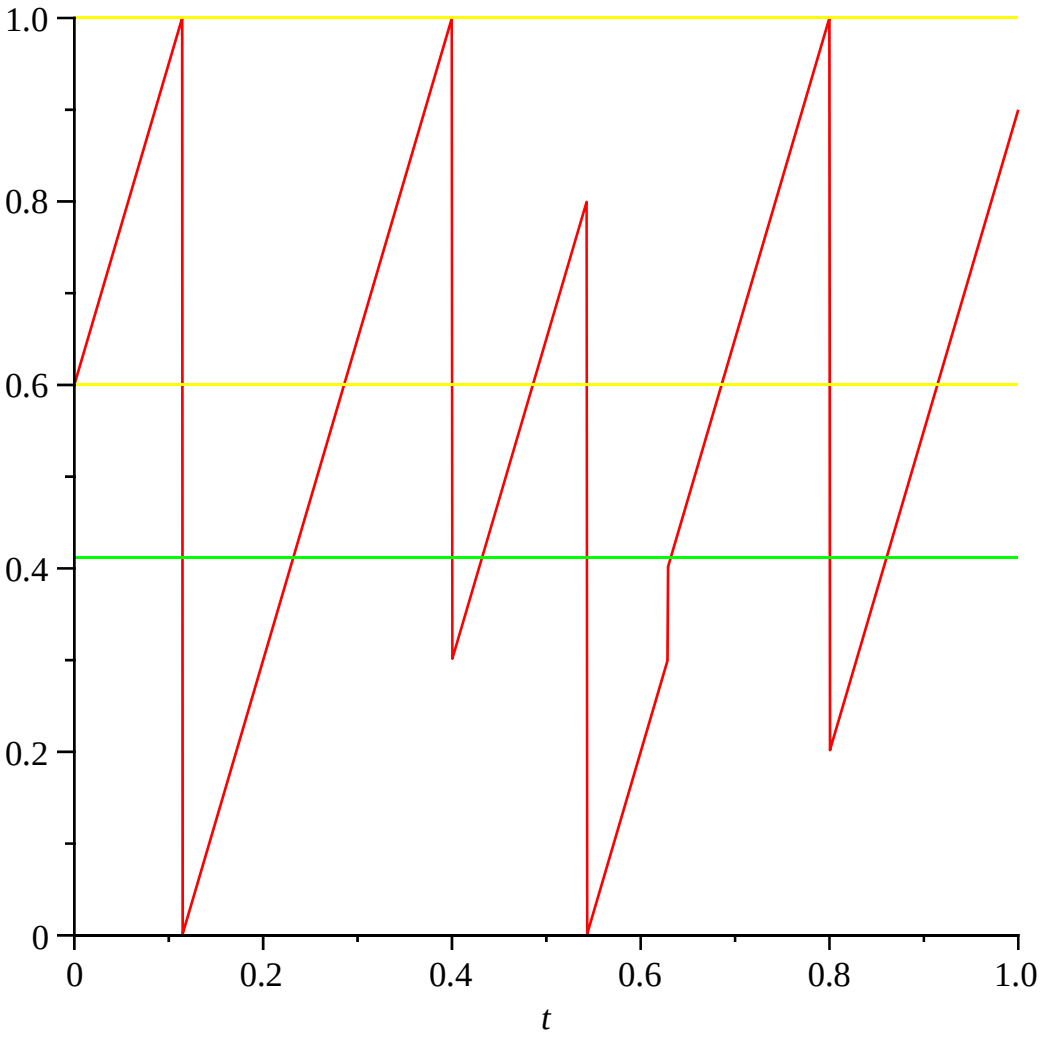
$$pre_2 := 0.2320817960$$

$$pre_3 := 0.4320817960$$

$$pre_4 := 0.6606532245$$

$$pre_5 := 0.6320817960$$

$$pre_6 := 0.8606532245$$



2

3

5

6

$err2 := 2.738265462\ 10^{-29}$

>

>