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


> N:=6;
  KK:=6;
  ### 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)
  U:=vector(N,[]):
  alpha:=vector(N,[]):
  K:=vector(N,[]):
  a:=vector(N,[]):
  bb:=vector(N+1,[]):
  c:=vector(KK,[]):
  for j from 1 to N do
  U[j]:=0;
  od:
  alpha[1]:=0.3:K[1]:=1:U[K[1]]:=0:
  alpha[2]:=0.5:K[2]:=2:U[K[2]]:=0:#
  alpha[3]:=0.4:K[3]:=3:U[K[3]]:=0: #
  alpha[4]:=0.4:K[4]:=4:U[K[4]]:=1:
  alpha[5]:=0.5:K[5]:=5:U[K[5]]:=1: #
  alpha[6]:=0.3:K[6]:=6:U[K[6]]:=1:
  i:='i':

  beta:=N-KK+sum(alpha[i],i=1..KK);i:='i';
  delta1:=(xw,yw)-> piecewise(xw<=yw,0,1):
  for j from 1 to N do
  b[j]:=(j-1-sum((1-alpha[i])*delta1( j,K[i]) ,i=1..KK))/beta;
  bb[j]:=b[j];
   od: i:='i':
  b[N+1]:=1:  bb[N+1]:=1:
  for j from 1 to N do
  a[j]:=(j-1-sum((1-alpha[i])*delta1( j,K[i]-U[j]) ,i=1..KK));
   od:


  for j from 1 to KK do
  if U[K[j]]=0 then    c[j]:=b[K[j]+1];
              else  c[j]:=b[K[j]];fi;
   #print(`c[`,j,`] = `,c[j]) ;
  od:
  print(`alpha = `,alpha);
  print(`K = `,K);
  print(`b = `,bb);
  print(`a = `,a);
```

```
   print(`c = `,c);

>
   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)];
   for j from 1 to KK do
   if U[K[j]]=0 then    Tc:=T(c[j]);
                  else   Tc:=uT(c[j])fi;
```

```
    print(`T(c[`,j,`])  =`,  Tc)
    od;


    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]], x=0..1, thickness=[2, 1, 1, 1, 1,
    1, 1]);
```

$$N := 6$$

$$KK := 6$$

$$\beta := 2.4000000000$$

$$i := i$$

$alpha$ = ,
[0.3000000000, 0.5000000000, 0.4000000000, 0.4000000000, 0.5000000000,
0.3000000000]

$$K = , \left[\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array}\right]$$

b = , [0.0000000000, 0.1250000000, 0.3333333333, 0.5000000000, 0.6666666667,
0.8750000000, 1]

a = ,
[0.0000000000, 0.3000000000, 0.8000000000, 0.6000000000, 1.1000000000,
1.4000000000]

c = ,
[0.1250000000, 0.3333333333, 0.5000000000, 0.5000000000, 0.6666666667,
0.8750000000]

$$0.5000000000$$

$$\beta max := 3.8000000000$$

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

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

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

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

$$T(c[, 1,]) =, 0.3000000000$$

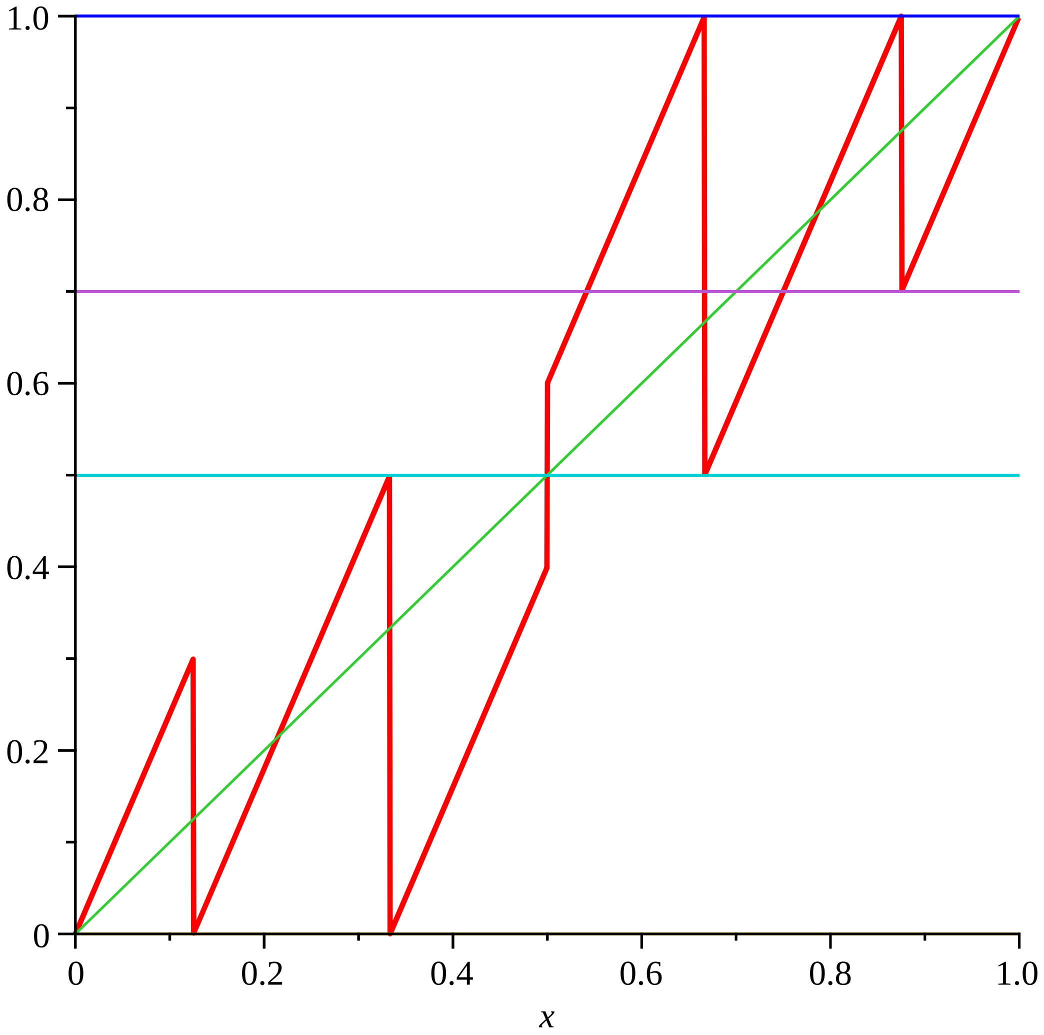
$$T(c[, 2,]) =, 0.5000000000$$

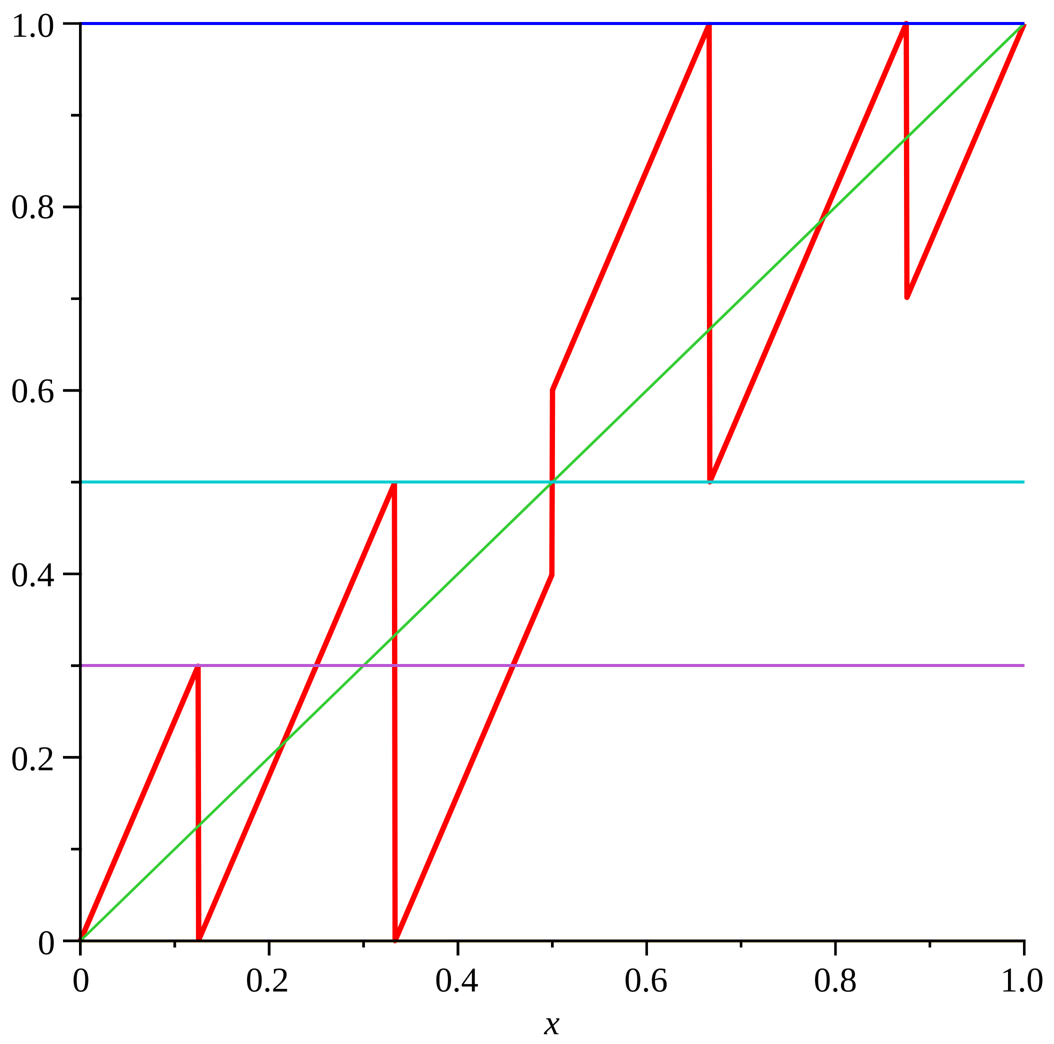
$$T(c[, 3,]) =, 0.4000000000$$

$$T(c[, 4,]) =, 0.6000000000$$

$$T(c[, 5,]) =, 0.5000000000$$

T(c[, 6,]) =, 0.7000000000





```
>
> ud:=vector(50):Digits:=100;NN:=50;
  d:=vector(50):
  xx:=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 := 0.3957188605
[0.8000000000, 0.3000000000, 0.0000000000, 0.3000000000, 0.0000000000, 0.0000000000,
    0.3000000000, 0.3000000000, 0.8000000000, 0.0000000000, 0.3000000000,
    0.3000000000, 0.0000000000, 0.3000000000, 0.3000000000, 0.3000000000,
    0.8000000000, 0.0000000000, 0.0000000000, 0.3000000000, 0.3000000000,
    0.8000000000, 0.0000000000, 0.3000000000, 0.8000000000, 0.0000000000,
    0.3000000000, 0.8000000000, 0.0000000000, 0.3000000000, 0.3000000000,
    0.8000000000, 0.0000000000, 0.0000000000, 0.3000000000, 0.3000000000,
    0.3000000000, 0.8000000000, 0.3000000000, 0.8000000000, 0.3000000000,
    0.8000000000, 0.3000000000, 0.8000000000, 0.3000000000, 0.0000000000,
    0.3000000000, 0.3000000000, 0.8000000000, 0.8000000000]
                       uIs_it_x := 0.3957188605
[0.8000000000, 0.3000000000, 0.0000000000, 0.3000000000, 0.0000000000, 0.0000000000,
    0.3000000000, 0.3000000000, 0.8000000000, 0.0000000000, 0.3000000000,
    0.3000000000, 0.0000000000, 0.3000000000, 0.3000000000, 0.3000000000,
    0.8000000000, 0.0000000000, 0.0000000000, 0.3000000000, 0.3000000000,
    0.8000000000, 0.0000000000, 0.3000000000, 0.8000000000, 0.0000000000,
    0.3000000000, 0.8000000000, 0.0000000000, 0.3000000000, 0.3000000000,
    0.8000000000, 0.0000000000, 0.0000000000, 0.3000000000, 0.3000000000,
    0.3000000000, 0.8000000000, 0.3000000000, 0.8000000000, 0.3000000000,
    0.8000000000, 0.3000000000, 0.8000000000, 0.3000000000, 0.0000000000,
    0.3000000000, 0.3000000000, 0.8000000000, 0.8000000000]
                       Is_it_x := 0.3957188605
```

$$terr := 1.478538300\ 10^{-20}$$

$$err := 1.478538300\ 10^{-20} \qquad \textbf{(1)}$$

```
> 
> 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 KK do
  xxt:=c[i]; upflag:=U[K[i]];
       for n from 1 to NN+1 do
        if upflag=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 upflag=0 then
              for ii from 1 to KK do
                   if xxt>c[ii] then cc[i,ii,n]:=1 else cc[i,ii,n]
  :=0 fi;
```

```
            od;
        fi;
        if upflag=1 then
            for ii from 1 to KK 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 upflag=0 then xxt:=T(xxt) else xxt:=uT(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 KK do
for j from 1 to KK 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;
#for i from 1 to 30 do
#print(cc[2,1,i],cc[2,2,i]) od;
```

$$NN := 50$$

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

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

$$xxt := 0.1250000000$$

$$upflag := 0$$

$$Is\_it\_x := 0.1250000000$$

$$S_1 := 0.3684413480$$

$$xxt := 0.3333333333$$

$$upflag := 0$$

$$Is\_it\_x := 0.3333333333$$

$$S_2 := 0.5304751832$$

$$xxt := 0.5000000000$$

$$upflag := 0$$

$$Is\_it\_x := 0.5000000000$$

$$S_3 := 0.4398071063$$

$$xxt := 0.5000000000$$

$$upflag := 1$$

$$Is\_it\_x := 0.5000000000$$

$$S_4 := 1.0482881318$$

$$xxt := 0.6666666667$$

$$upflag := 1$$

$$Is\_it\_x := 0.6666666667$$

$$S_5 := 0.9576200549$$

$$xxt := 0.8750000000$$

$$upflag := 1$$

$$Is\_it\_x := 0.8750000000$$

$$S_6 := 1.1196538901$$

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

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

print(`eigenvalues of  -S =`,eigenvalues(MM));

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

MM[i,i]:=MM[i,i]+1/beta;
od:
det(MM);
print(MM);
print(ve);
print(`1/beta(beta-1) =`,1/(beta*(beta-1)));



DD:=linsolve(MM,ve);
for i from 1 to 10 do
print(i,1/(beta^i*(beta-1)));od;
SS[2,1]/SS[2,2];
```

```
MM = , [[-0.2961033701, -0.2845212912, -0.2661844323, -0.0000000000,
    -0.0000000000, -0.0000000000],
    [-0.0723379780, -0.2459538919, -0.1736226739, -0.0000000000, -0.0000000000,
    -0.0000000000],
    [-0.0000000000, -0.0000000000, -0.0000000000, -0.0000000000, -0.0000000000,
    -0.0000000000],
```

[−0.0000000000, −0.0000000000, −0.0000000000, −0.0000000000, −0.0000000000,
−0.0000000000],
[−0.0000000000, −0.0000000000, −0.0000000000, −0.1736226739, −0.2459538919,
−0.0723379780],
[−0.0000000000, −0.0000000000, −0.0000000000, −0.2661844323, −0.2845212912,
−0.2961033701]]

1/beta =, 0.4166666667

eigenvalues of -S =, −0.4166666667, −0.1253905953, −0.1253905953, −0.4166666667,
−0.0000000000, −0.0000000000

$5.756223156\ 10^{-42}$

[[0.1205632966, −0.2845212912, −0.2661844323, −0.0000000000, −0.0000000000,
−0.0000000000],
[−0.0723379780, 0.1707127747, −0.1736226739, −0.0000000000, −0.0000000000,
−0.0000000000],
[−0.0000000000, −0.0000000000, 0.4166666667, −0.0000000000, −0.0000000000,
−0.0000000000],
[−0.0000000000, −0.0000000000, −0.0000000000, 0.4166666667, −0.0000000000,
−0.0000000000],
[−0.0000000000, −0.0000000000, −0.0000000000, −0.1736226739, 0.1707127747,
−0.0723379780],
[−0.0000000000, −0.0000000000, −0.0000000000, −0.2661844323, −0.2845212912,
0.1205632966]]

[0.4166666667, 0.4166666667, 0.4166666667, 0.4166666667, 0.4166666667,
0.4166666667]

1/beta(beta-1) =, 0.2976190476

DD := $\left[4.941226002\ 10^{19}, 2.093799355\ 10^{19}, 1.0000000000, 1.0000000000, 2.093799355\ 10^{19}, 4.941226002\ 10^{19}\right]$

1, 0.2976190476
2, 0.1240079365
3, 0.0516699735
4, 0.0215291556
5, 0.0089704815
6, 0.0037377006
7, 0.0015573753
8, 0.0006489064
9, 0.0002703776
10, 0.0001126574

1.1568074366 **(2)**

>

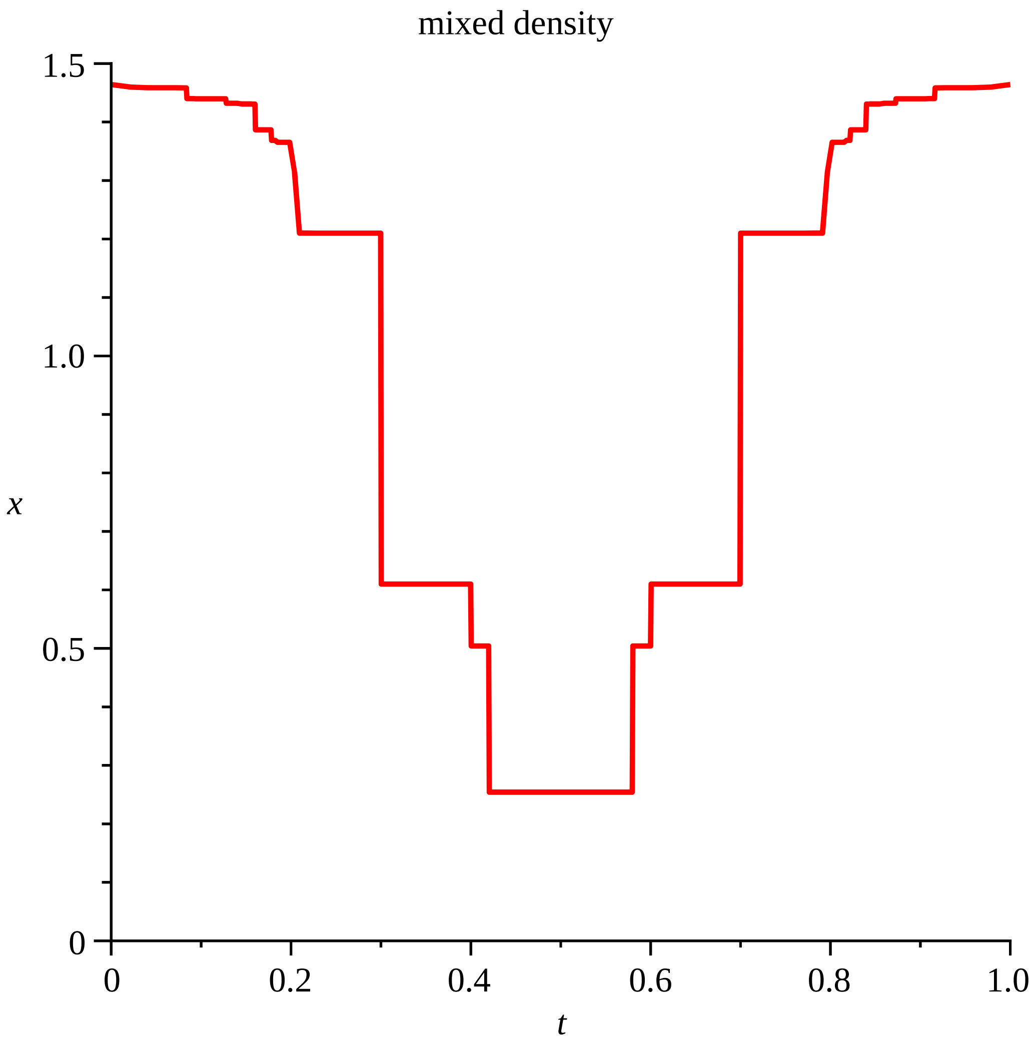
```
        density:=proc(t) local j, den;
                   den:=1/beta;
                 for j from 1 to KK do
                 if U[K[j]]=0 then
                   den:=den+ DD[j]*sum((chi(0,valc[j,i1+1],t))
  /beta^(i1+1),i1=1..50)
                             else
                   den:=den+ DD[j]*sum((uchi(valc[j,i1+1],1,t))
  /beta^(i1+1),i1=1..50)
                   fi;
                   od;
                 return den;
             end proc;
  #Normalizing factor
  tNC:=1/beta:
  for j from 1 to KK do
   tNC:=tNC+tDD[j]*sum((1-tvalc[j,i1+1])/beta^(i1+1),i1=1..50)
   od:
  NC:=1/beta:
  for j from 1 to KK do
  if U[K[j]]=0 then
   NC:=NC+DD[j]*sum((valc[j,i1+1])/beta^(i1+1),i1=1..50)
  else
  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,x=0..1.5,title=
  "mixed density",thickness=2);
```

```
density := proc(t)
   local j, den;
   den := 1 / beta;
   for j to KK do
      if U[K[j]] = 0 then
         den := den + DD[j] * (sum(chi(0, valc[j, i1 + 1], t) / beta^(i1 + 1), i1 = 1 ..50))
      else
         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
```

$$NC = , 1.430031519\ 10^{19}$$



```
>
>
#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;
```

$$y_6 := 0.9974920499$$

$$su := 0$$

$$1$$

$$2$$

$$3$$

$$err_0 := -0.7753466252$$

$$su := 0$$

$$2$$

$$3$$

$$err_1 := 0.4166666667$$

$$su := 0$$

$$2$$

$$err_2 := 0.4166666667$$

$$su := 0$$

$$2$$

$$3$$

$$err_3 := 0.4166666667$$

$$su := 0$$

$$2$$

$$err_4 := 0.4166666667$$

$$su := 0$$

$$2$$

$$3$$

$$err_5 := 0.4166666667$$

$$su := 0$$

$$4$$

$$5$$

$$6$$

$$err_6 := -0.7753466252$$

$$y\ =,\ 0.1236858858$$

$$err[,\ 0,\ ]=,\ -0.7753466252$$

$$y\ =,\ 0.3386279633$$

```
err[, 1, ]=, 0.4166666667
y =, 0.4977575830
err[, 2, ]=, 0.4166666667
y =, 0.4000000000
err[, 3, ]=, 0.4166666667
y =, 0.4427552057
err[, 4, ]=, 0.4166666667
y =, 0.3314754631
err[, 5, ]=, 0.4166666667
y =, 0.9974920499
err[, 6, ]=, -0.7753466252
```

```
>
#check density greedy
#preimages
y:=0.500000000000;
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;
```

$$y := 0.5000000000$$

$$pre_1 := 0.2083333333$$

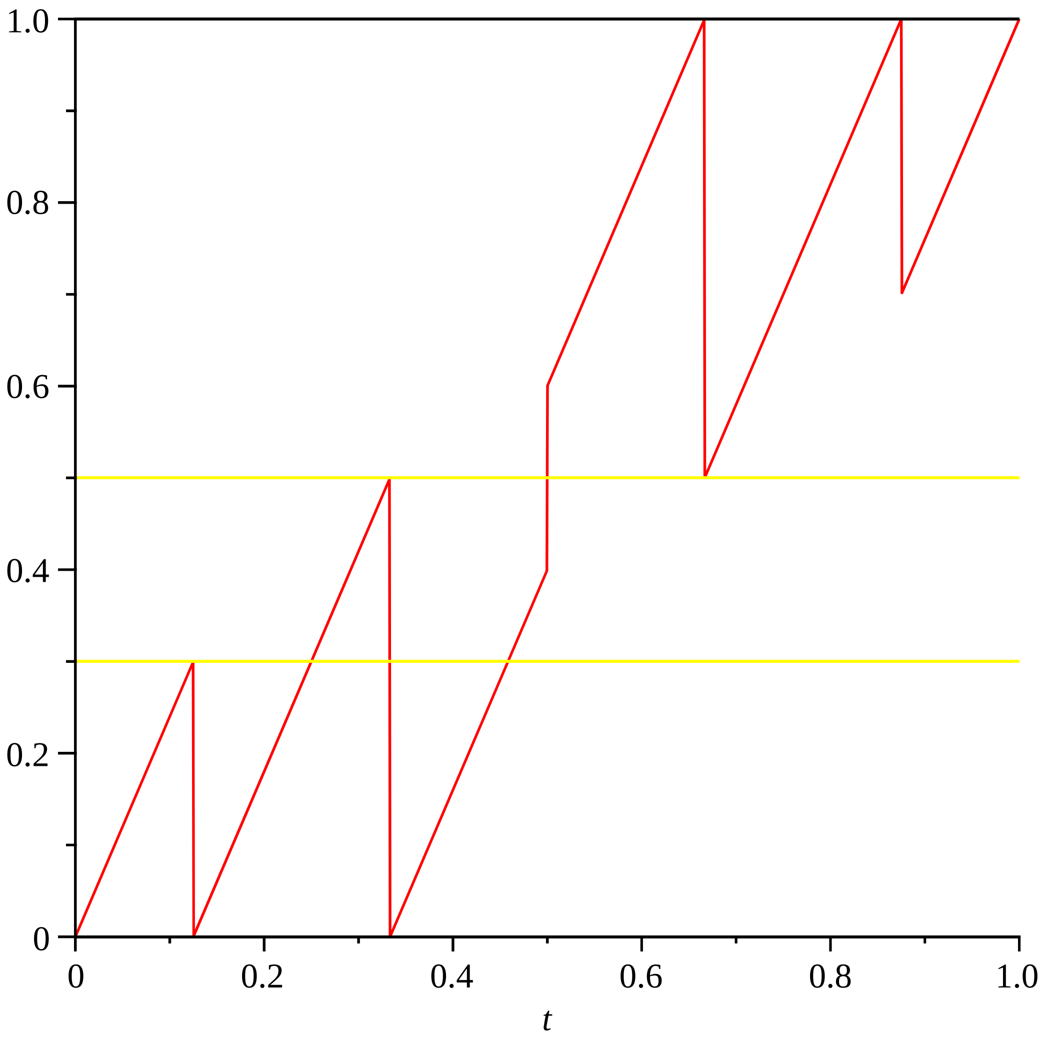
$$pre_2 := 0.3333333333$$

$$pre_3 := 0.5416666667$$

$$pre_4 := 0.4583333333$$

$$pre_5 := 0.6666666667$$

$$pre_6 := 0.7916666667$$



$$2$$

$$5$$

$$err2 := -3.635068325\ 10^{18}$$

```
>
>
```