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


>
  N:=4;

  bb:=vector(N+1,[]):#for printing only = b[]
  beta:=vector(N,[]):

  alpha:=vector(N,[]):
  gamm=vector(N,[]):#heiths of lower ends of hanging branches
                        # alpha[i]+gamma[i]<1   !!!!!!!
                        #if gamma[i]>0

  alpha[1]:=2/3:beta[1]:=1:gamm[1]:=1/3:

  alpha[2]:=1/6:beta[2]:=2:gamm[2]:=0.0:#
  alpha[3]:=2/6:beta[3]:=2.0:gamm[3]:=0:  #
  alpha[4]:=1/6:beta[4]:=2:gamm[4]:=5/6:
  #alpha[5]:=0.5:beta[5]:=9:gamm[5]:=0:
  #alpha[6]:=0.6:beta[6]:=7:gamm[6]:=0.2:#
  #alpha[7]:=1.0:beta[7]:=6:gamm[7]:=0.0:  #
  #alpha[N]:=0.25:  beta[N]:=2.4:              gamm[N]:=0.75:
  i:='i':
  beta_const:=sum(alpha[i],i=1..N);
  i:='i':
  #for j from 1 to N do
  #beta[j]:=beta_const;
  #od:

  print(`alpha =`,alpha);
  print(`beta =`,beta);
  print(`gamma =`,gamm);



  b[1]:=0:
  for j from 1 to N do
  b[j+1]:=b[j]+alpha[j]/beta[j]:
   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[j]*b[j];
   al[j]:=-1+beta[j]*b[j+1];
   od:
   bb[N+1]:=1:
   for j from 1 to N do
   a[j]:=ag[j]-gamm[j];
    od:

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



   # 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:=0:# new number of c points
   for j from 1 to N do if alpha[j]<1 then Kc:=Kc+1 fi od:
   for j from 1 to N do if (gamm[j]>0 and alpha[j]+gamm[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)
   j_of_c:=vector(2*N,[]):# shows the index of the interval
   associated with c

   cj:=1:# this is the new index for c points
   for j from 1 to N do
   if (alpha[j]<1 and gamm[j]=0) then   c[cj]:=b[j+1]; sidec[cj]:=
```

```
   0;j_of_c[cj]:=j;cj:=cj+1 fi;
   if (alpha[j]<1 and gamm[j]+alpha[j]=1) then  c[cj]:=b[j]; sidec
   [cj]:=1;j_of_c[cj]:=j;cj:=cj+1 fi;
   if (gamm[j]>0 and gamm[j]+alpha[j]<1) then  c[cj]:=b[j]; sidec
   [cj]:=1;j_of_c[cj]:=j;cj:=cj+1 ;
                          c[cj]:=b[j+1]; sidec[cj]:=0;j_of_c[cj]:=j;
   cj:=cj+1 fi;




   od:
   print(`c =`,c);
   print(`sidec =`,sidec);
   print(`j_of_c =`,j_of_c);


>
>


>
   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[uint_of_x(x)]*x-a[uint_of_x(x)];
T:=x->beta[int_of_x(x)]*x-a[int_of_x(x)];
Tc:=vector(Kc+2,[]):
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, Tc[1], Tc[2]], x=0..1, thickness=[2, 1, 1, 1, 1, 1, 1]);
plot([`T(x)`, x, 0, 1, Tc[1], Tc[2]], x=0..1, thickness=[2, 1, 1, 1, 1, 1, 1]);
```

$$N := 4$$

$$\beta const := \frac{4}{3}$$

$$alpha =, \left[\frac{2}{3} \quad \frac{1}{6} \quad \frac{1}{3} \quad \frac{1}{6} \right]$$

$$beta =, \left[1 \quad 2 \quad 2.0000000000 \quad 2 \right]$$

$$gamma =, \left[\frac{1}{3} \quad 0.0000000000 \quad 0 \quad \frac{5}{6} \right]$$

$$b =, \left[0 \quad \frac{2}{3} \quad \frac{3}{4} \quad 0.9166666667 \quad 1 \right]$$

$$ag =, \left[0 \quad \frac{4}{3} \quad 1.5000000000 \quad 1.8333333333 \right]$$

$$al =, \left[-\frac{1}{3} \quad \frac{1}{2} \quad 0.8333333333 \quad 1 \right]$$

$$a =, \left[-\frac{1}{3} \quad 1.3333333333 \quad 1.5000000000 \quad 1.0000000000 \right]$$

$$gamma =, \left[\frac{1}{3} \quad 0.0000000000 \quad 0 \quad \frac{5}{6} \right]$$

$$Kc =, 4$$

$$c =, \left[0 \quad \frac{3}{4} \quad 0.9166666667 \quad 0.9166666667 \quad c_5 \quad c_6 \quad c_7 \quad c_8 \right]$$

$$sidec =, \left[\begin{array}{cccccccc} 1 & 0 & 0 & 1 & sidec_5 & sidec_6 & sidec_7 & sidec_8 \end{array}\right]$$

$$j\_of\_c =, \left[\begin{array}{cccccccc} 1 & 2 & 3 & 4 & j\_of\_c_5 & j\_of\_c_6 & j\_of\_c_7 & j\_of\_c_8 \end{array}\right]$$

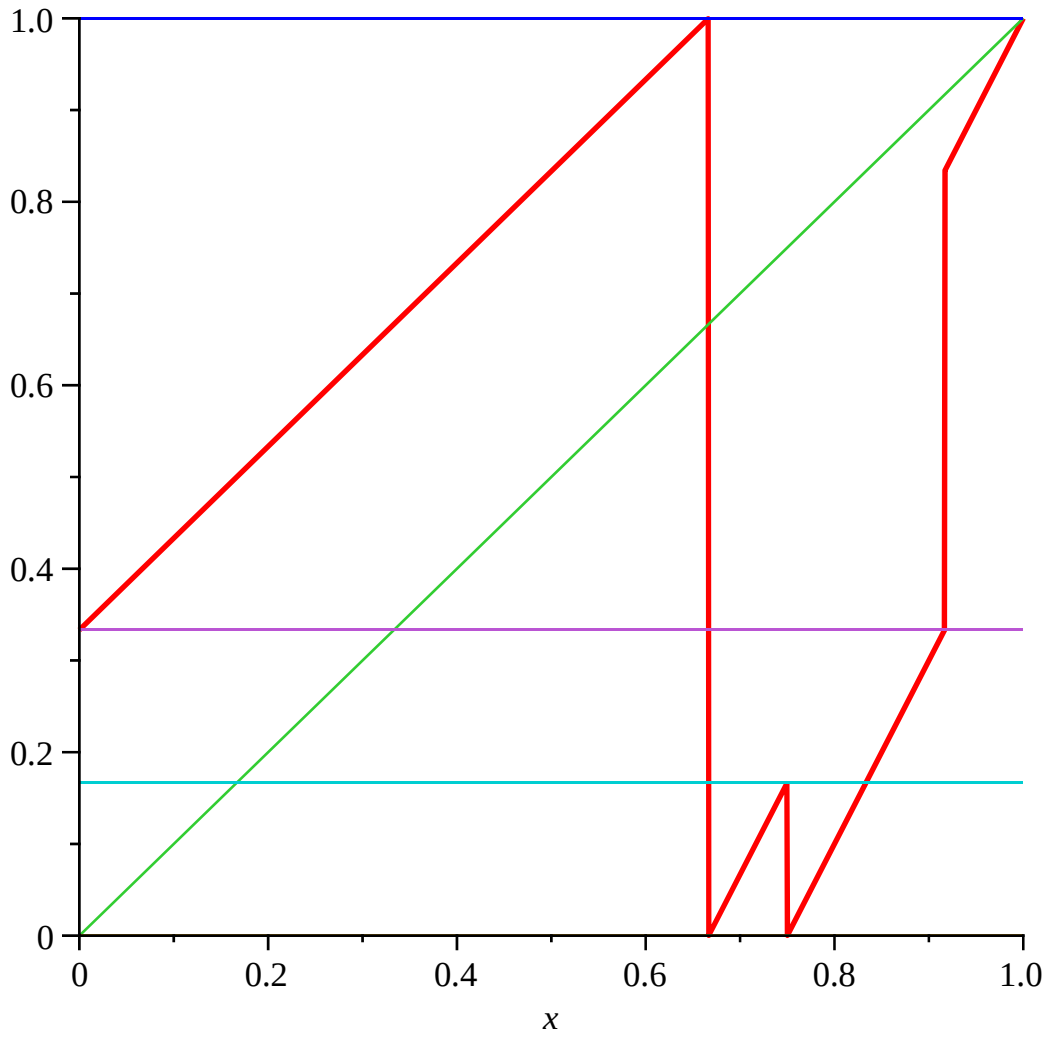
$$uint\_of\_x := x \rightarrow piecewise\left(x < b_2, 1, x < b_3, 2, x < b_4, 3, x < b_5, 4, x < b_6, 5, x < b_7, 6, x < b_8, 7, x < b_9, 8, 9\right)$$

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

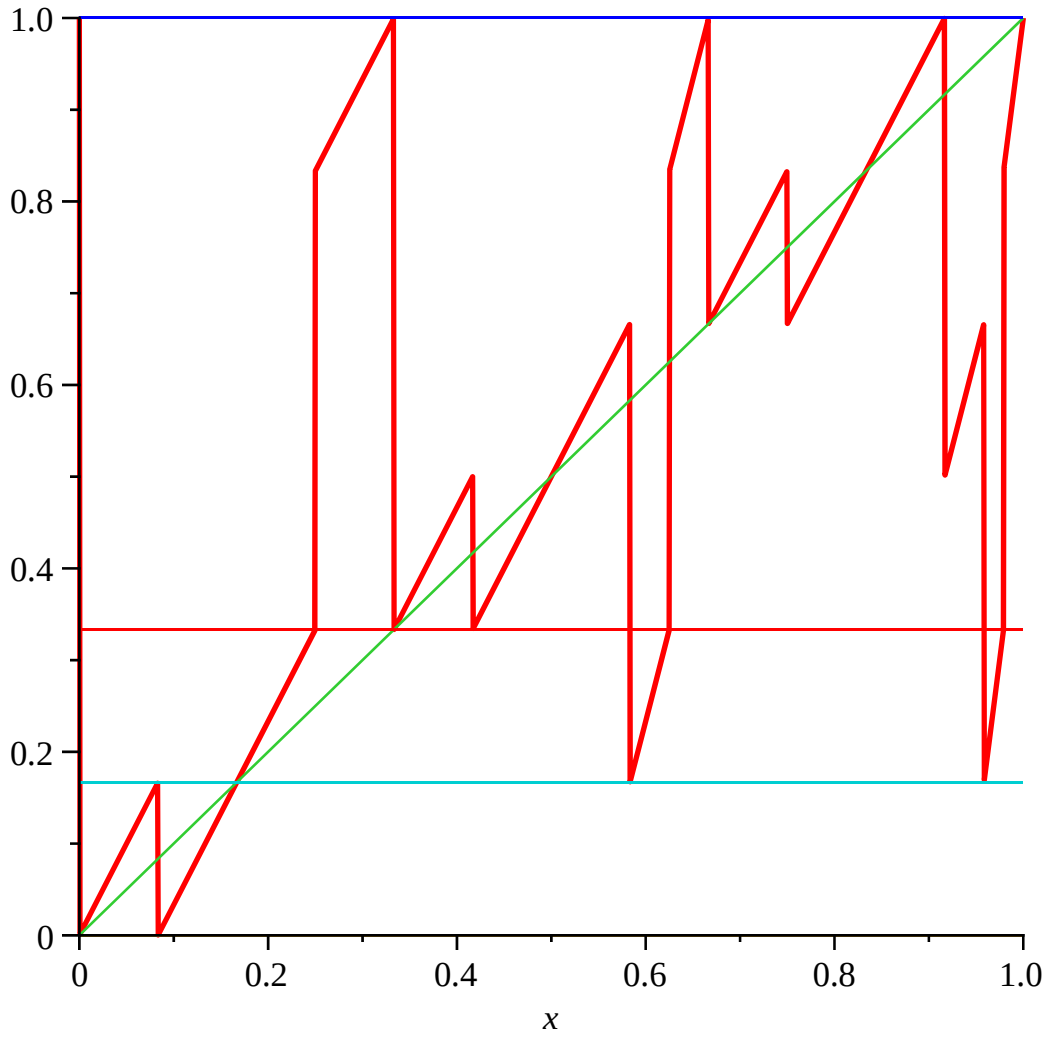
$$uT := x \rightarrow \beta_{uint\_of\_x(x)}\, x - a_{uint\_of\_x(x)}$$

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

$$Tc = , \left[\begin{array}{cccccc} \frac{1}{3} & 0.1666666667 & 0.3333333333 & 0.8333333333 & Tc_5 & Tc_6 \end{array}\right]$$




```
> x: =' x' :
  pl ot ( [ ' T( T( T( x) ) ) ' , x, 0, 1, Tc[ 1] , Tc[ 2] , Tc[ 3] ] , x=0. . 1, t hi ckness=[ 2,
  1, 1, 1, 1, 1, 1, 1] ) ;
  T( T( c[ 2] ) ) ;
```

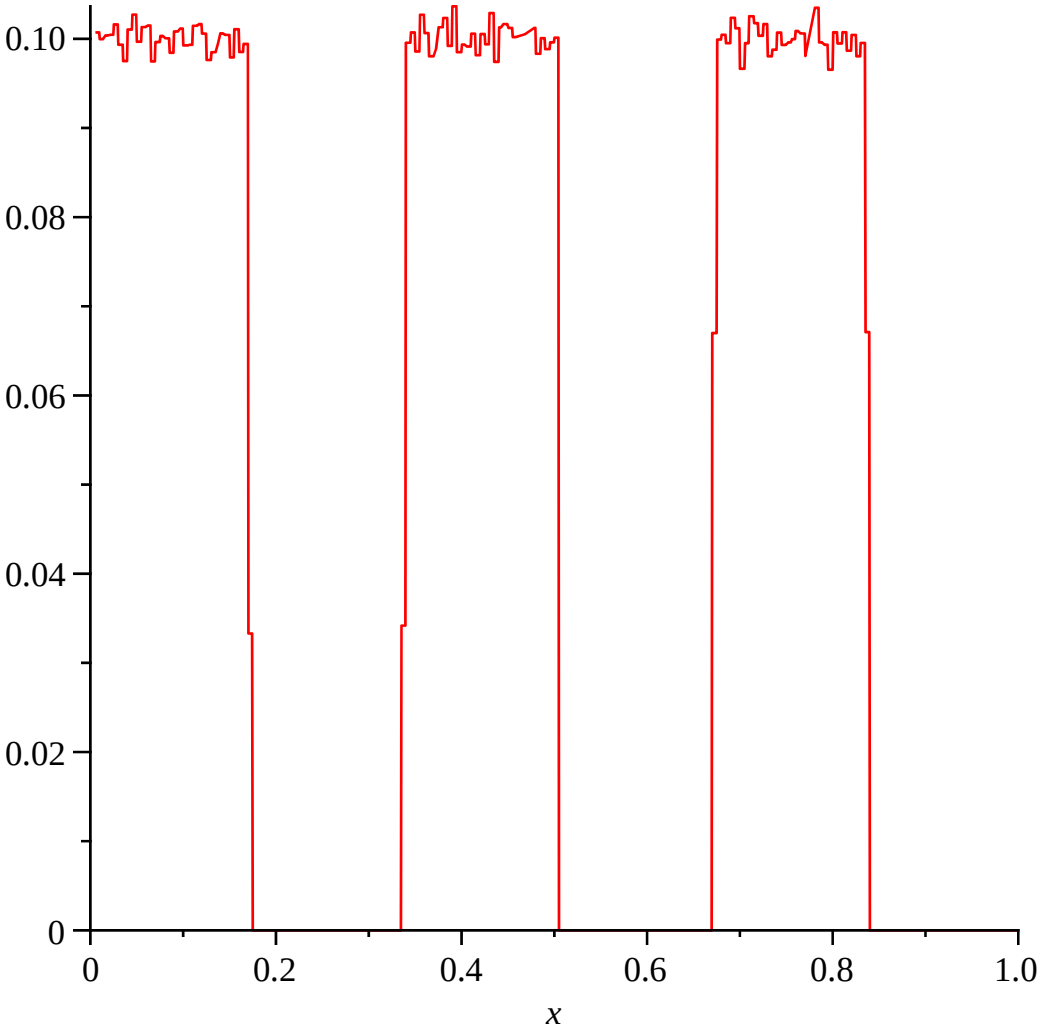


0.5000000000 **(1)**

```
> niter:=500000; Nb:=200: v := array(1..Nb):
  AA:=0: BB:=1:
  for i from 1 to Nb do v[i]:=0; od:
  y:=0.102347777777333333337:
  for j from 1 to niter do
       y:=T(y):
       iy:=floor(Nb*(y-AA)/(BB-AA))+1;
       if iy>Nb then iy:=Nb fi;
       v[iy]:=v[iy]+1
    od:
  x:='x': g:=x->10*v[floor(Nb*x)]/niter;
  plot(g(x),x=0..1);
```

$$niter := 500000$$

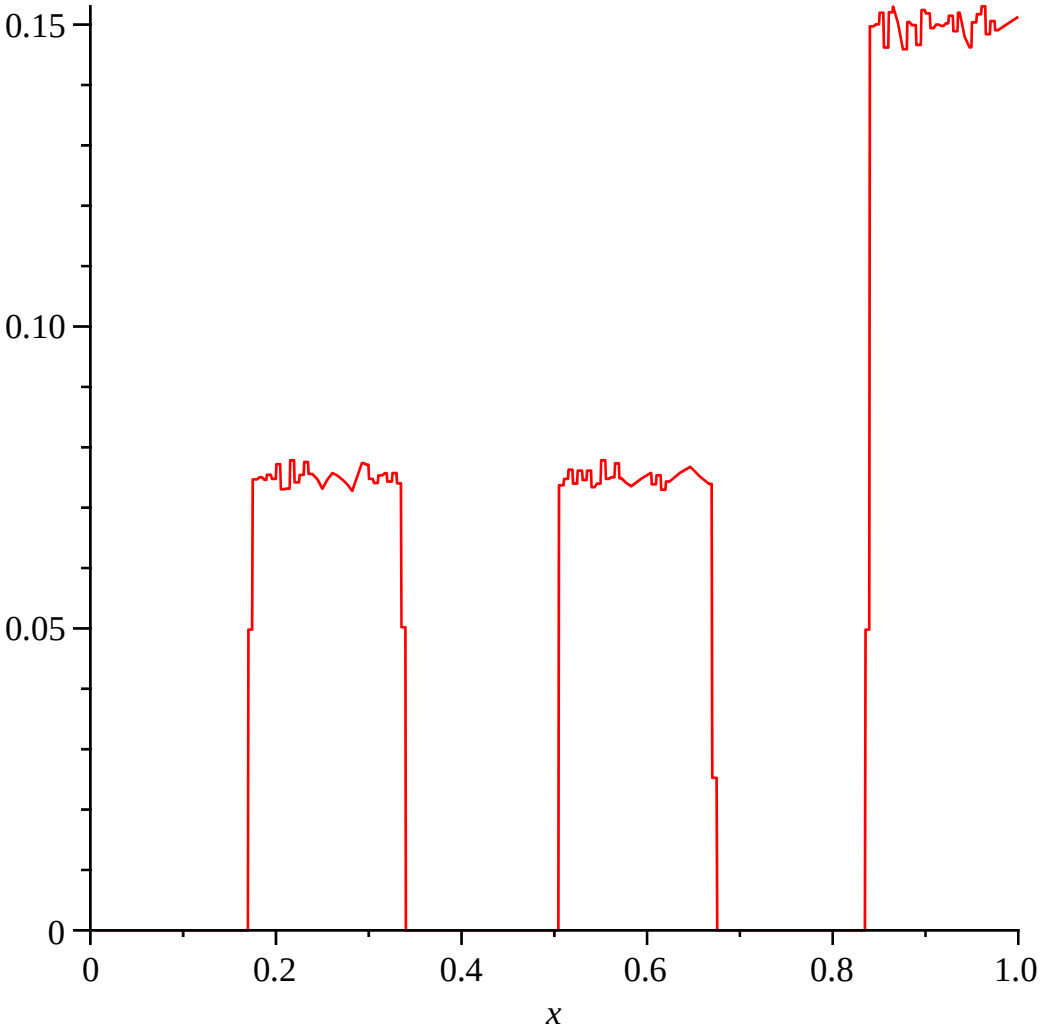
$$g := x \rightarrow \frac{10\, v_{\mathrm{floor}(Nb\, x)}}{niter}$$




```
> niter:=500000; Nb:=200: v := array(1..Nb):
  AA:=0: BB:=1:
  for i from 1 to Nb do v[i]:=0; od:
  y:=0.2347777777333333337:
  for j from 1 to niter do
       y:=T(y):
       iy:=floor(Nb*(y-AA)/(BB-AA))+1;
       if iy>Nb then iy:=Nb fi;
       v[iy]:=v[iy]+1
    od:
  x:='x': g:=x->10*v[floor(Nb*x)]/niter;
  plot(g(x),x=0..1);
```

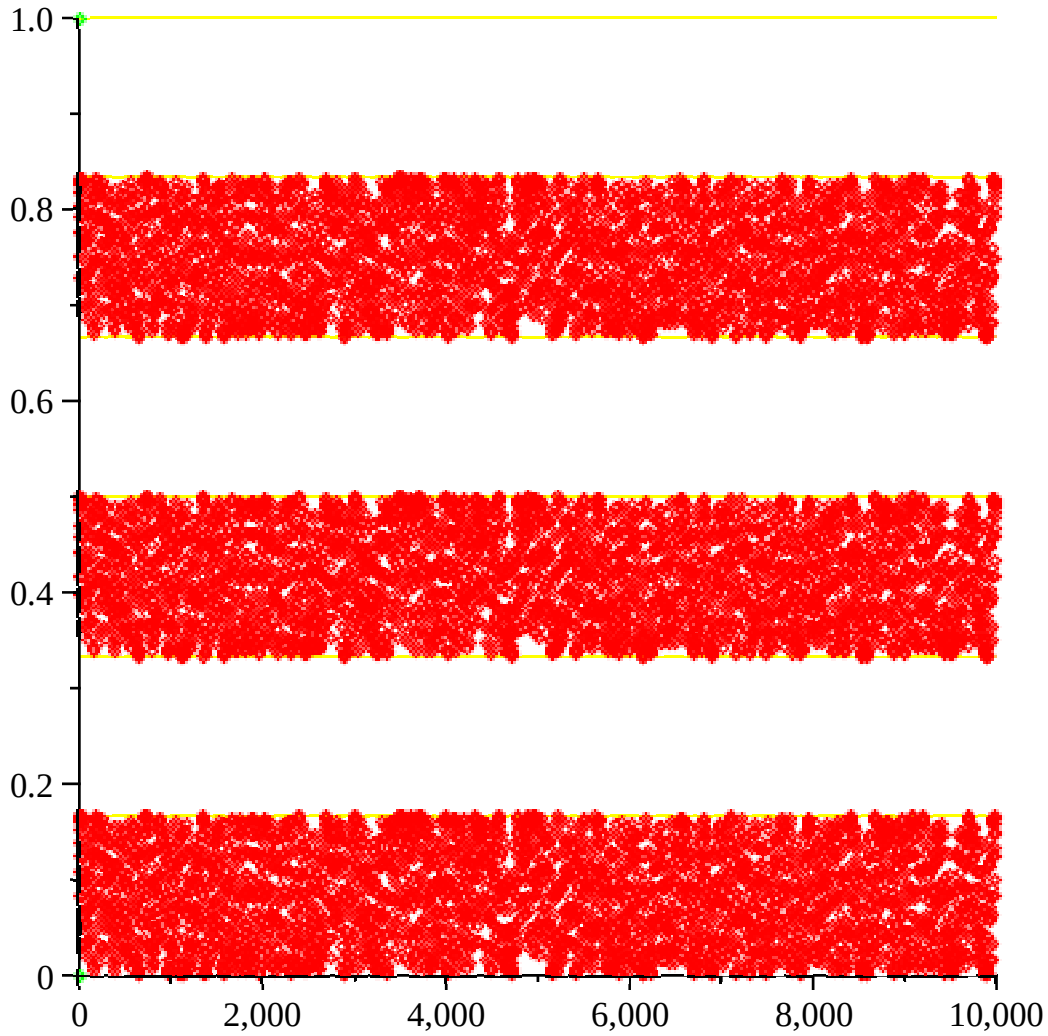
$$niter := 500000$$

$$g := x \rightarrow \frac{10\, v_{\mathrm{floor}(Nb\, x)}}{niter}$$




```
> x:=evalf(rand()/10^(12));
  Np:=10000:
  for i from 1 to Np do
  px[i]:=x:
  x:=T(x);
  od:
  y:='y':
  pl1:=plot([1/6,2/6,3/6,4/6,5/6,6/6],0..Np,color=yellow):
  display(pl1,POINTS(seq([i,px[i]],i=1..Np),COLOR(RGB,1,0,0)),
  POINTS([0.0,0.0],[1.0,1.0],COLOR(RGB,0,1,0)));
```

$$x := 0.3957188605$$




```
>
> ud:=vector(50):Digits:=100;NN:=50;#Expansion with variable
  slopes
  d:=vector(50):
  xx:=0.23777337737373737;
  xxt:=xx:
  bet:=1:
  for i from 1 to NN do
  bet:=bet/beta[uint_of_x(xxt)];
  ud[i]:=a[uint_of_x(xxt)];
  udb[i]:=a[uint_of_x(xxt)]*bet;
  xxt:=uT(xxt);
  od:

  xxt:=xx:
  bet:=1:
  for i from 1 to NN do
  bet:=bet/beta[int_of_x(xxt)];
  d[i]:=a[int_of_x(xxt)];
  db[i]:=a[int_of_x(xxt)]*bet;
  xxt:=T(xxt);
  od:
```

```
print(ud);
uIs_it_x:=evalf(sum(udb[j1],j1=1..NN));
print(d);
Is_it_x:=evalf(sum(db[j1],j1=1..NN));
terr:=xx-uIs_it_x;
err:=xx-Is_it_x;
```

$$Digits := 100$$

$$NN := 50$$

$$xx := 0.2377733774$$

$$\left[-\frac{1}{3}, -\frac{1}{3}, 1.5000000000, -\frac{1}{3}, -\frac{1}{3}, 1.0000000000, 1.0000000000, 1.5000000000, -\frac{1}{3},\right.$$
$$-\frac{1}{3}, 1.0000000000, 1.0000000000, 1.5000000000, -\frac{1}{3}, -\frac{1}{3}, 1.0000000000,$$
$$1.5000000000, -\frac{1}{3}, -\frac{1}{3}, 1.5000000000, -\frac{1}{3}, -\frac{1}{3}, 1.0000000000, 1.0000000000,$$
$$1.0000000000, 1.5000000000, -\frac{1}{3}, -\frac{1}{3}, 1.5000000000, -\frac{1}{3}, -\frac{1}{3}, 1.5000000000, -\frac{1}{3},$$
$$-\frac{1}{3}, 1.5000000000, -\frac{1}{3}, -\frac{1}{3}, 1.0000000000, 1.5000000000, -\frac{1}{3}, -\frac{1}{3}, 1.5000000000,$$
$$\left.-\frac{1}{3}, -\frac{1}{3}, 1.0000000000, 1.5000000000, -\frac{1}{3}, -\frac{1}{3}, 1.0000000000, 1.0000000000\right]$$

$$uIs\_it\_x := 0.2377733191$$

$$\left[-\frac{1}{3}, -\frac{1}{3}, 1.5000000000, -\frac{1}{3}, -\frac{1}{3}, 1.0000000000, 1.0000000000, 1.5000000000, -\frac{1}{3},\right.$$
$$-\frac{1}{3}, 1.0000000000, 1.0000000000, 1.5000000000, -\frac{1}{3}, -\frac{1}{3}, 1.0000000000,$$
$$1.5000000000, -\frac{1}{3}, -\frac{1}{3}, 1.5000000000, -\frac{1}{3}, -\frac{1}{3}, 1.0000000000, 1.0000000000,$$
$$1.0000000000, 1.5000000000, -\frac{1}{3}, -\frac{1}{3}, 1.5000000000, -\frac{1}{3}, -\frac{1}{3}, 1.5000000000, -\frac{1}{3},$$
$$-\frac{1}{3}, 1.5000000000, -\frac{1}{3}, -\frac{1}{3}, 1.0000000000, 1.5000000000, -\frac{1}{3}, -\frac{1}{3}, 1.5000000000,$$
$$\left.-\frac{1}{3}, -\frac{1}{3}, 1.0000000000, 1.5000000000, -\frac{1}{3}, -\frac{1}{3}, 1.0000000000, 1.0000000000\right]$$

$$Is\_it\_x := 0.2377733191$$

$$terr := 5.828829832\ 10^{-8}$$

$$err := 5.828829832\ 10^{-8} \qquad \textbf{(2)}$$

```
>
>
> NN:=150; 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];
bet:=1:
      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;
         bet_real:=bet;

         bet:=bet/beta[intx];
         dcb[i,n]:=a[intx]*bet;

        if sidec[i]=0 then
            for ii from 1 to Kc do
                 if xxt>c[ii]+10^(-15) then cc[i,ii,n]:=1*
bet_real else cc[i,ii,n]:=0 fi;
            od;
            if intx=1 then Sc[i,n]:= 0
                        else Sc[i,n]:=sum(1/(beta[j7]),j7=1..
intx-1)*bet_real fi;
            #bc[i,n]:=b[intx]*bet_real;
            ################
                   else
            for ii from 1 to Kc do
                 if xxt<c[ii]-10^(-15) then cc[i,ii,n]:=1*
bet_real else cc[i,ii,n]:=0 fi;
            od;
             if intx=N then Sc[i,n]:= 0
                        else Sc[i,n]:=sum(1/(beta[j8]),j8=
intx+1..N)*bet_real fi;
            #bc[i,n]:=b[intx+1]*bet_real;
      fi;
      valc[i,n]:=xxt;
      betc[i,n]:=bet_real;
      if sidec[i]=1 then
                       Rounding:=infinity;
                        xxt:=uT(xxt)
                         else
                        Rounding:=0;
                        xxt:=T(xxt)
fi;
Rounding:=nearest;

      od:
Is_it_x:=sum(dcb[i,j1],j1=1..NN);
od;

for i from 1 to Kc do
```

```
S[i]:=evalf(sum(Sc[i,j21+1],j21=1..NN));
#sum_b[i]:=evalf(sum(bc[i,j2+1],j2=1..NN));
od;
for i from 1 to Kc do
for j from 1 to Kc do
SS[i,j]:=evalf(sum(cc[i,j,j1+1],j1=1..NN));

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

$$NN := 150$$

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

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

$$xxt := 0$$

$$bet := 1$$

$$Is\_it\_x := -5.523000043\ 10^{-100}$$

$$xxt := \frac{3}{4}$$

$$bet := 1$$

$$Is\_it\_x := 0.7500000000$$

$$xxt := 0.9166666667$$

$$bet := 1$$

$$Is\_it\_x := 0.9166666667$$

$$xxt := 0.9166666667$$

$$bet := 1$$

$$Is\_it\_x := 0.9166666667$$

$$S_1 := 6.5000000000$$

$$S_2 := 1.5000000000$$

$$S_3 := 2.0000000000$$

$$S_4 := 2.0000000000 \qquad \textbf{(3)}$$

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

MM[i,j]:=-SS[j,i];
MMM[i,j]:=-SS[j,i];
od;od;
```

```
print(`MM = `,MM);
print(`det  MM = `,det(MM));
print(1/beta[j_of_c[5]]);


print(`eigenvalues  MM =`,eigenvalues(MM));
print(`1/  average  beta  = `,1/ave_beta);
ve:=vector(Kc,[]):
for  i  from 1  to  Kc  do
ve[i]:=1;

MMM[i,i]:=MMM[i,i]+1.0;
od:
MMM[2,2]:=0; MMM[3,3]:=0;
print(MMM);
print(`det  MMM = `,det(MMM));
print(ve);_t:='_t';



DD:=linsolve(MMM,ve);
sum((S[ii7]-1/beta[j_of_c[ii7]])*DD[ii7],ii7=1..Kc)-(1-sum
(1/beta[i8],i8=1..N));
eigenvectors(MM);
```

$$MM = , \begin{bmatrix} -0.0000000000 & -3.0000000000 & -2.0000000000 & -0.0000000000 \\ -5.0000000000 & -1.0000000000 & -1.0000000000 & -1.0000000000 \\ -5.0000000000 & -0.0000000000 & -1.0000000000 & -2.0000000000 \\ -5.0000000000 & -0.0000000000 & -1.0000000000 & -2.0000000000 \end{bmatrix}$$

$$det\ MM = , 0.0000000000$$

$$\frac{1}{\beta_{j\_of\_c_5}}$$

$$eigenvalues\ MM =, 3.7201532545, -6.7201532545, 0.0000000000, -1.0000000000$$

$$1/\ average\ beta = , \frac{1}{ave\_beta}$$

$$MMM_{2,2} := 0$$

$$MMM_{3,3} := 0$$

$$\begin{bmatrix} 1.0000000000 & -3.0000000000 & -2.0000000000 & -0.0000000000 \\ -5.0000000000 & 0 & -1.0000000000 & -1.0000000000 \\ -5.0000000000 & -0.0000000000 & 0 & -2.0000000000 \\ -5.0000000000 & -0.0000000000 & -1.0000000000 & -1.0000000000 \end{bmatrix}$$

$$det\ MMM = , -1.332267630\ 10^{-14}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\_t := \_t$$

$$DD := \begin{bmatrix} -0.2000000000 & -0.4000000000 & -0.0000000000 & -0.0000000000 \end{bmatrix}$$

$$2.960594732\ 10^{-16}$$

$$\left[3.7201532545, 1, \left\{\begin{bmatrix} 0.8022929283 & -0.5969305296 & -0.5969305296 & -0.5969305296 \end{bmatrix}\right\}\right], \quad \textbf{(4)}$$

$$\left[0.0000000000, 1, \left\{\begin{bmatrix} -0.0565685425 & 0.5656854249 & -0.8485281374 & 0.5656854249 \end{bmatrix}\right\}\right], \left[-6.7201532545, 1, \left\{\begin{bmatrix} 0.5969305296 & 0.8022929283 & 0.8022929283 & 0.8022929283 \end{bmatrix}\right\}\right], \left[-1.0000000000, 1, \left\{\begin{bmatrix} 0.1571348403 & 0.3142696805 & -0.3928371007 & -0.3928371007 \end{bmatrix}\right\}\right]$$

```
> DDv:=vector(4,[0.1571348403,0.3142696805,-0.3928371007,
  -0.3928371007]);
```

$$DDv := \begin{bmatrix} 0.1571348403 & 0.3142696805 & -0.3928371007 & -0.3928371007 \end{bmatrix} \quad \textbf{(5)}$$

```
> MMM:=matrix(4,4,[[1,-3,-2,0],[-5,0,-1,-1],[-5,0,0,-2],[-5,0,-1,
  -1]]);
  vi:=vector(4,[0,0,0,0]);
  linsolve(MMM,vi);#for more general form of invariant density
```

$$MMMi := \begin{bmatrix} 1 & -3 & -2 & 0 \\ -5 & 0 & -1 & -1 \\ -5 & 0 & 0 & -2 \\ -5 & 0 & -1 & -1 \end{bmatrix}$$

$$vi := \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \_t_1 & 2\,\_t_1 & -\frac{5}{2}\,\_t_1 & -\frac{5}{2}\,\_t_1 \end{bmatrix} \quad \textbf{(6)}$$

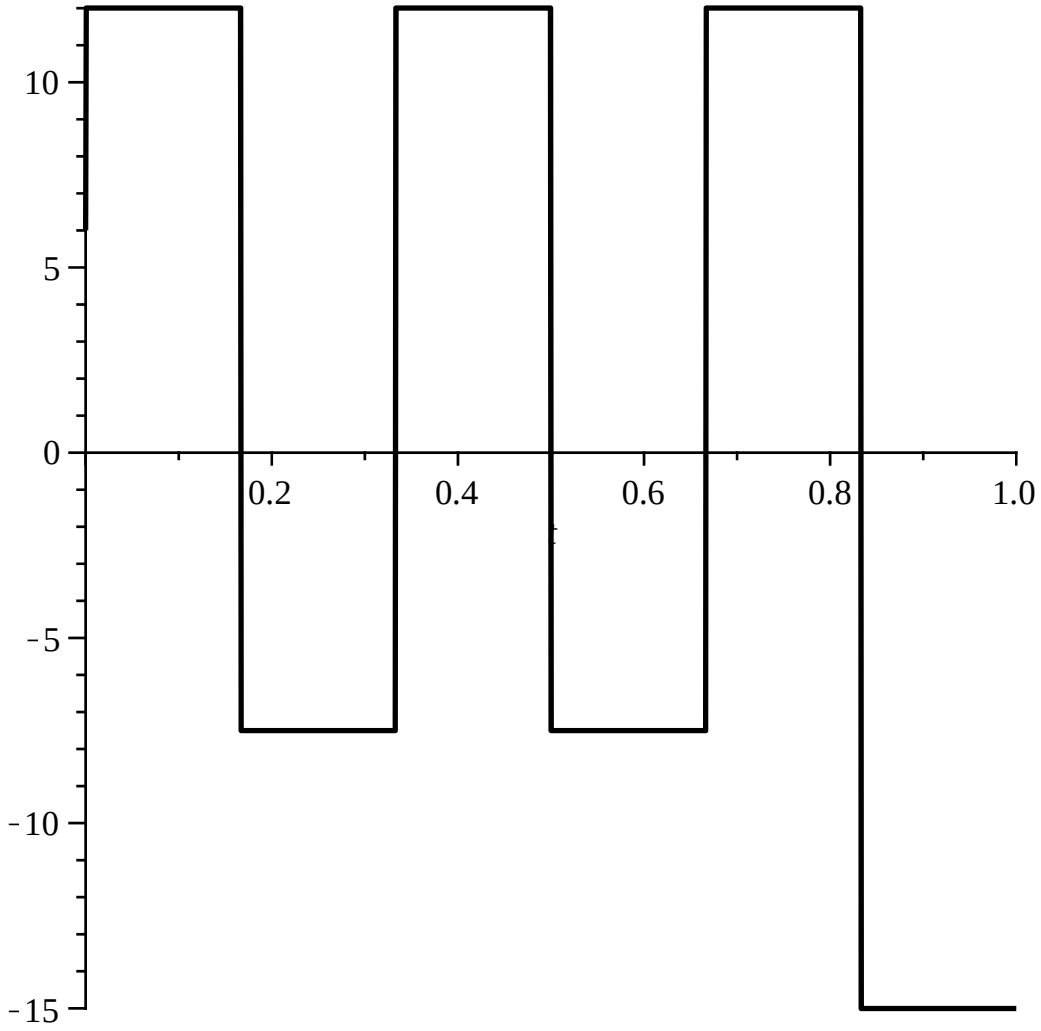
```
>
  density:=proc(t) local j,i, den;
                   i:='i':
                   #den:=sum(chi(b[i],b[i+1],t)/beta[i],i=1..N);
                  den:=0:
```

```
                    for j from 1 to Kc do
                    if sidec[j]=0 then
                      den:=den+ DDv[j]*sum((chi(0,valc[j,i1+1],t))*
betc[j,i1+1],i1=1..150) fi;

                     if sidec[j]=1 then
                      den:=den+ DDv[j]*sum((uchi(valc[j,i1+1],1,t))*
betc[j,i1+1],i1=1..150) fi;


                     od;
                    return den;
              end proc;

#Normalizing factor

NC:=0:
for j from 1 to Kc do
if sidec[j]=0 then
 NC:=NC+DDv[j]*sum((valc[j,i1+1])*betc[j,i1+1],i1=1..50) fi;
if sidec[j]=1 then
NC:=NC+DDv[j]*sum((1-valc[j,i1+1])*betc[j,i1+1],i1=1..50) fi;

 od:

print(`NC = `,NC);

plot([(1/NC)*density(t)],t=0..1-0.000001,color=black,thickness=
2);
```

```
density := proc(t)
    local j, i, den;
    i := 'i';
    den := 0;
    for j to Kc do
        if sidec[j] = 0 then
            den := den + DDv[j]*(sum(chi(0, valc[j, i1 + 1], t)*betc[j, i1 + 1], i1 = 1
            ..150))
        end if;
        if sidec[j] = 1 then
            den := den + DDv[j]*(sum(uchi(valc[j, i1 + 1], 1, t)*betc[j, i1 + 1], i1 = 1
            ..150))
        end if
    end do;
```

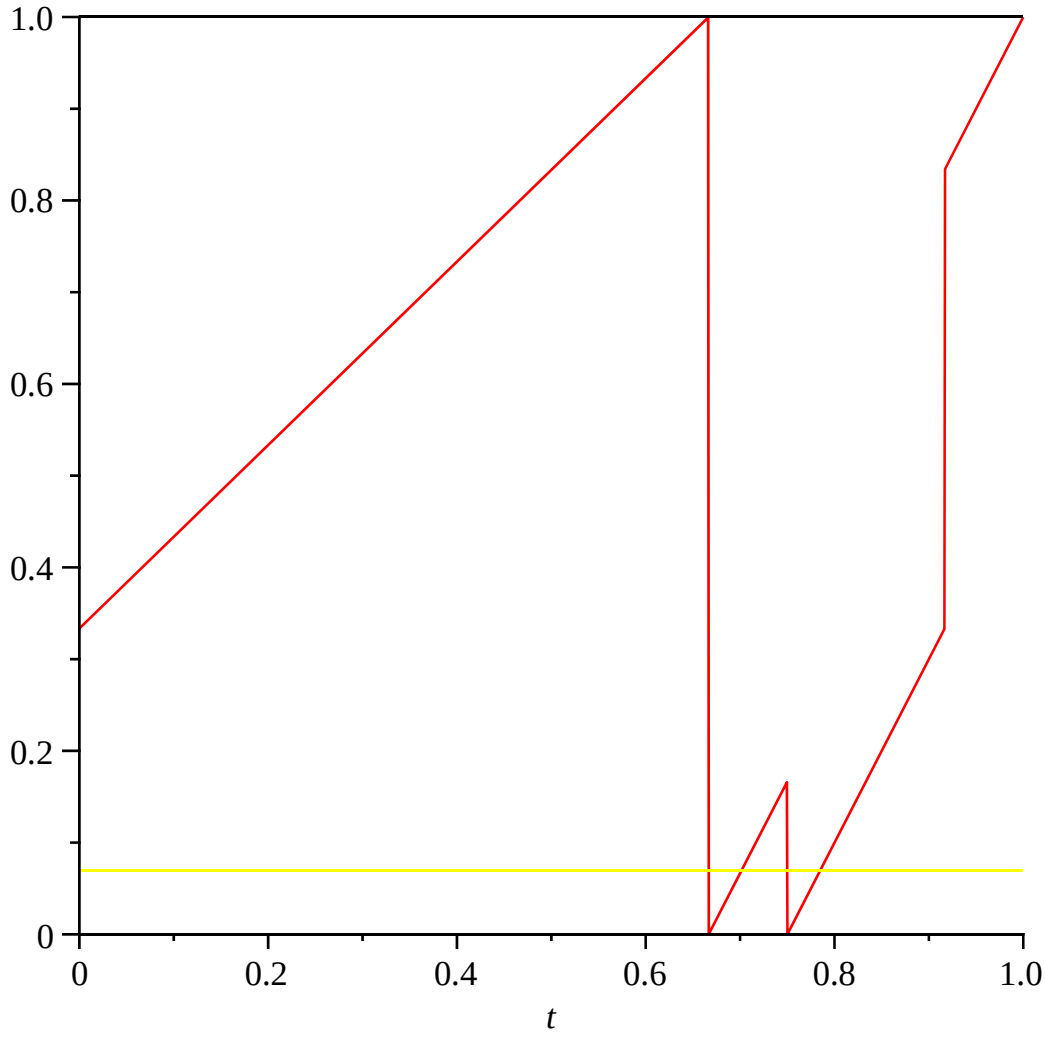
```
    return den
end proc
```

$$NC = , 0.0261819969$$




```
> 
> 
#check density
#preimages
for j6 from 0 to 9 do
y[j6]:=j6/10+(0.1)*rand()/10^12;
od:
for j6 from 0 to 9 do
for i3 from 1 to N do
pre[i3]:=(y[j6]+a[i3])/beta[i3];
od;
plot([T(t),0,1,y[j6]],t=0..1,
color=[red,black,black,yellow]);
su:=0:
for i3 from 1 to N do
if (pre[i3]>=b[i3] and pre[i3]<=b[i3+1]) then
 su:=su+evalf(density(pre[i3])/beta[i3]);
print(i3);
```

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

for j6 from 0 to 9 do
print(`y =`,y[j6]);
print(`err[`,j6,`]=`,err[j6]);
od;
```

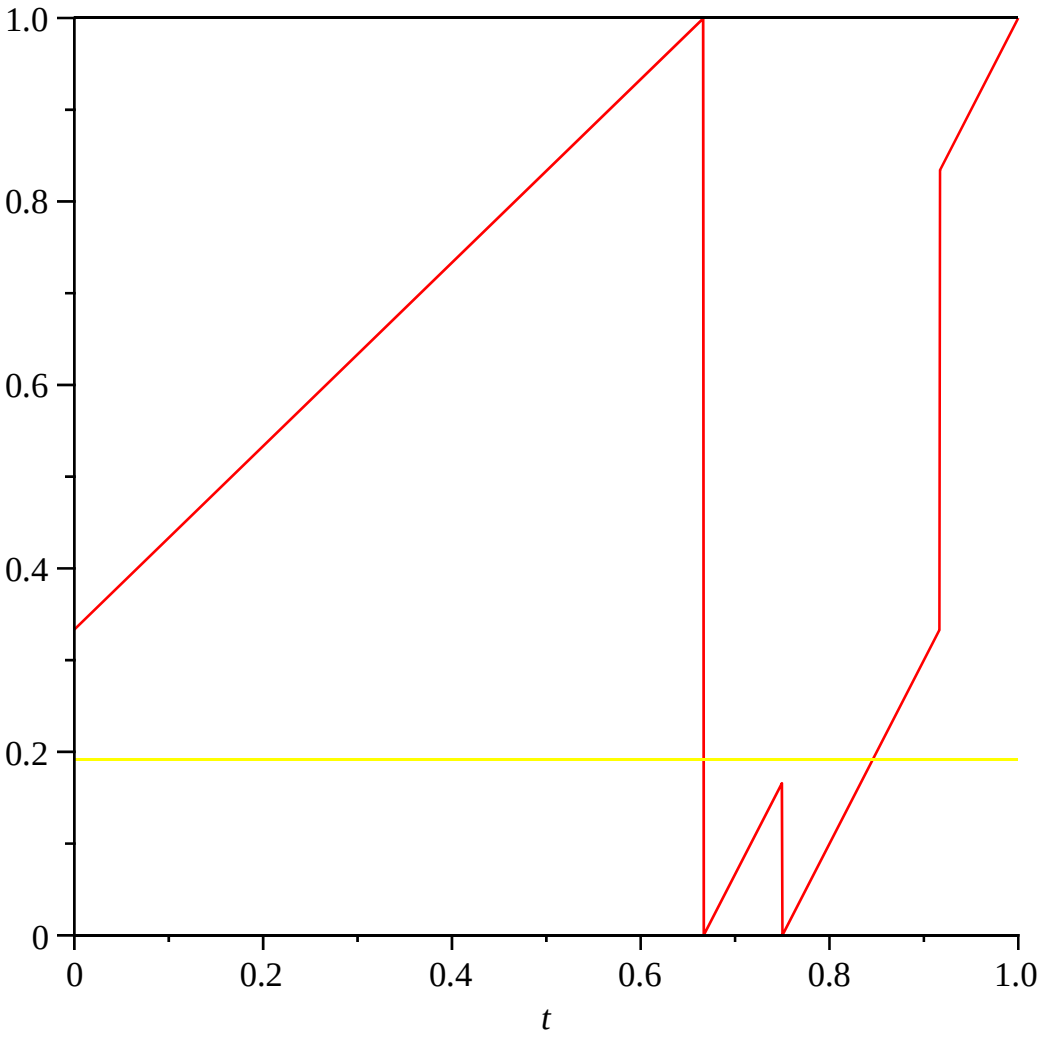


$su := 0$

2

3

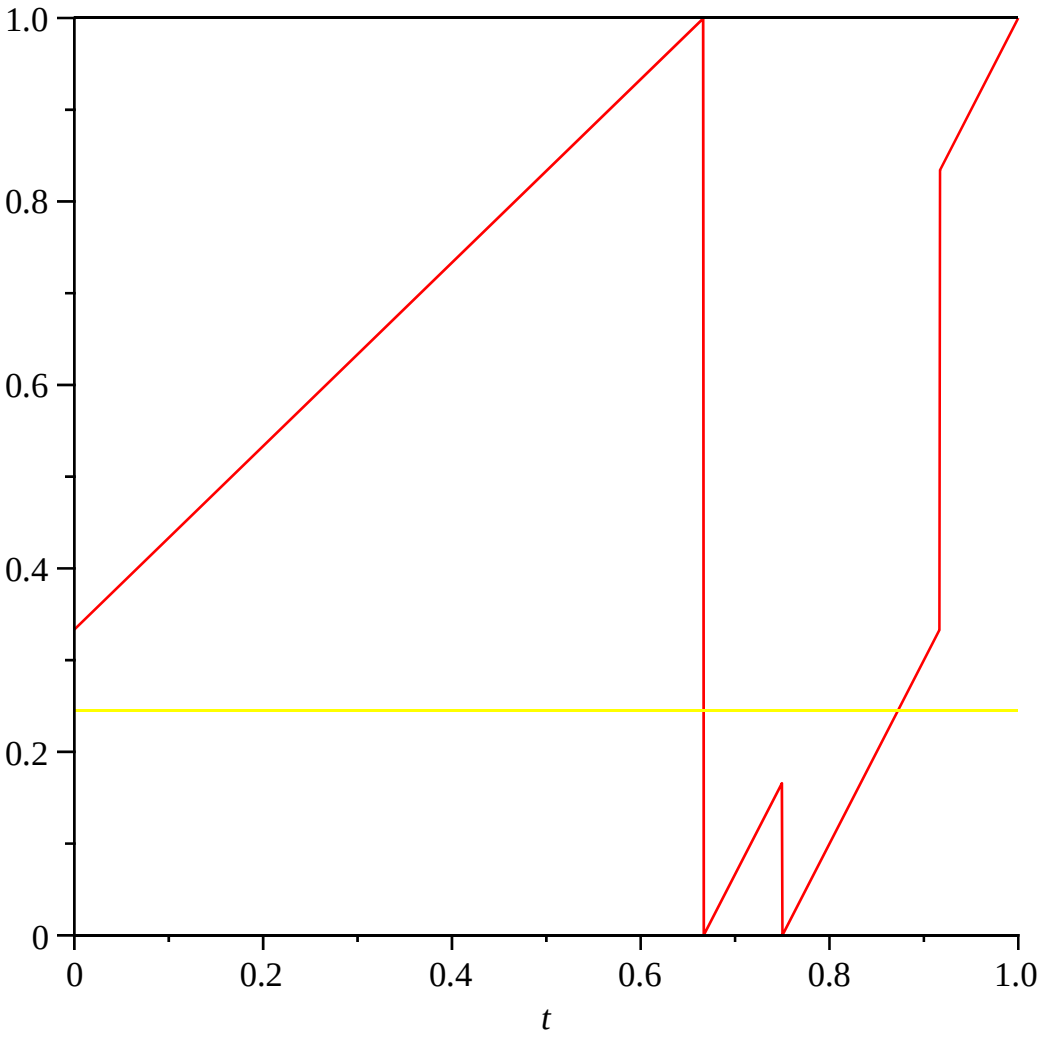
$err_0 := -2.000003489\ 10^{-10}$



$su := 0$

3

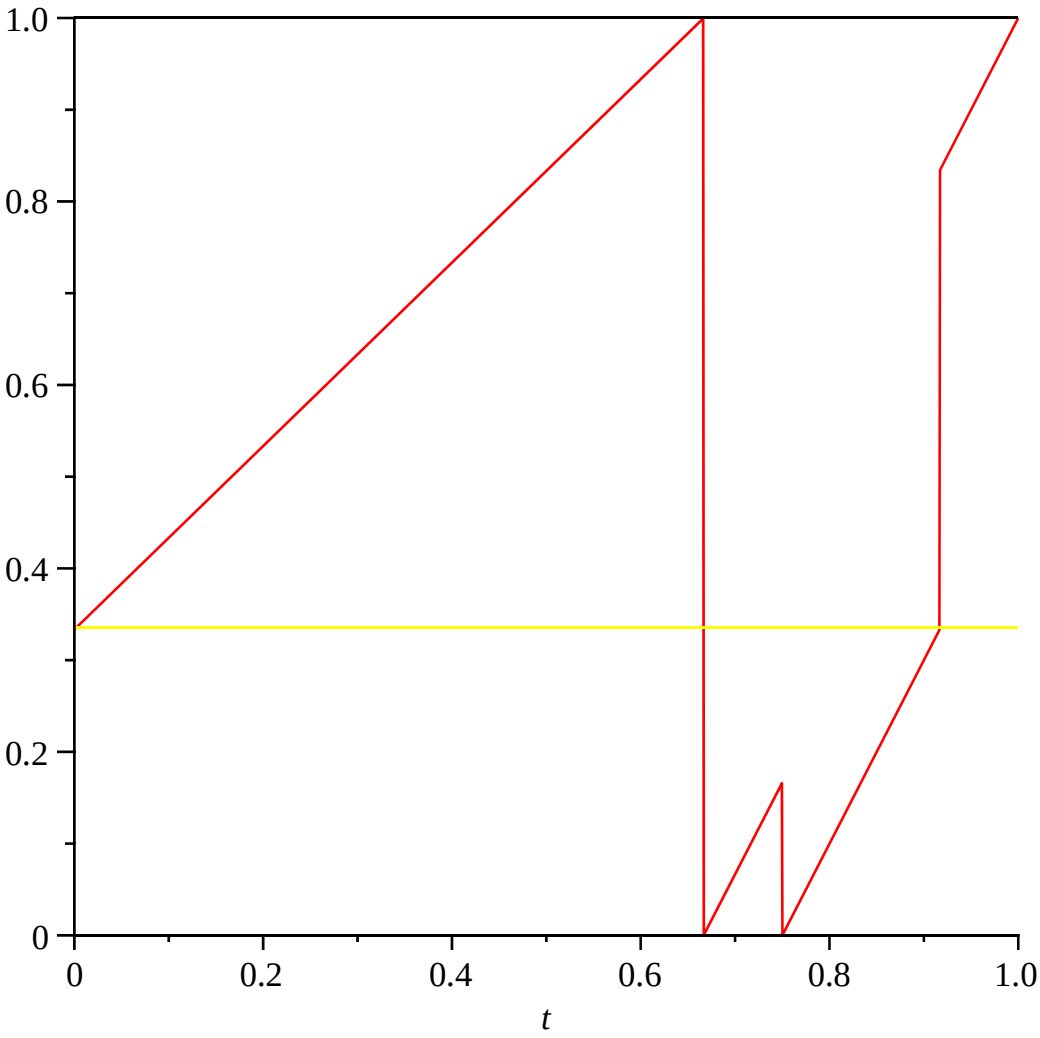
$err_1 := -1.500005234\ 10^{-10}$



$su := 0$

3

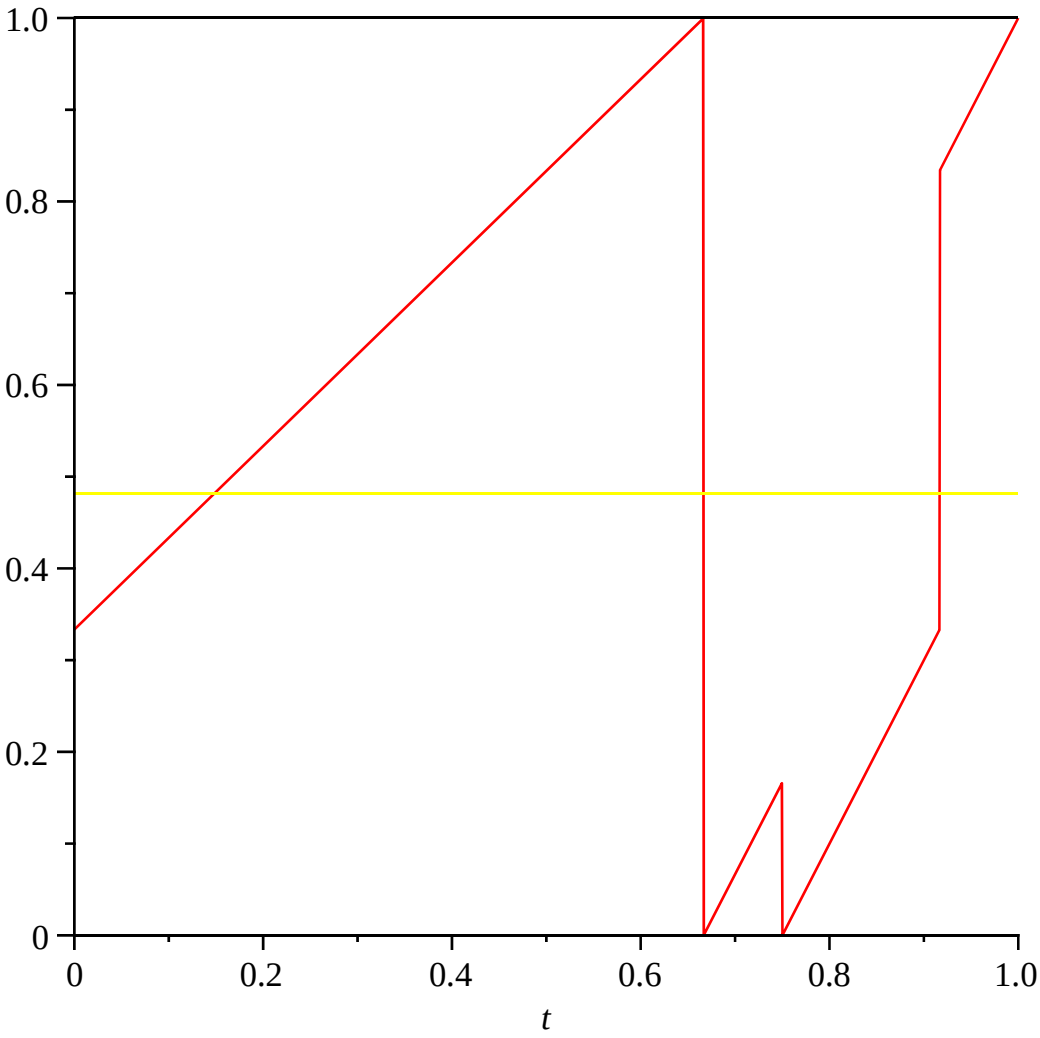
$err_2 := -1.500005234\ 10^{-10}$



$su := 0$

1

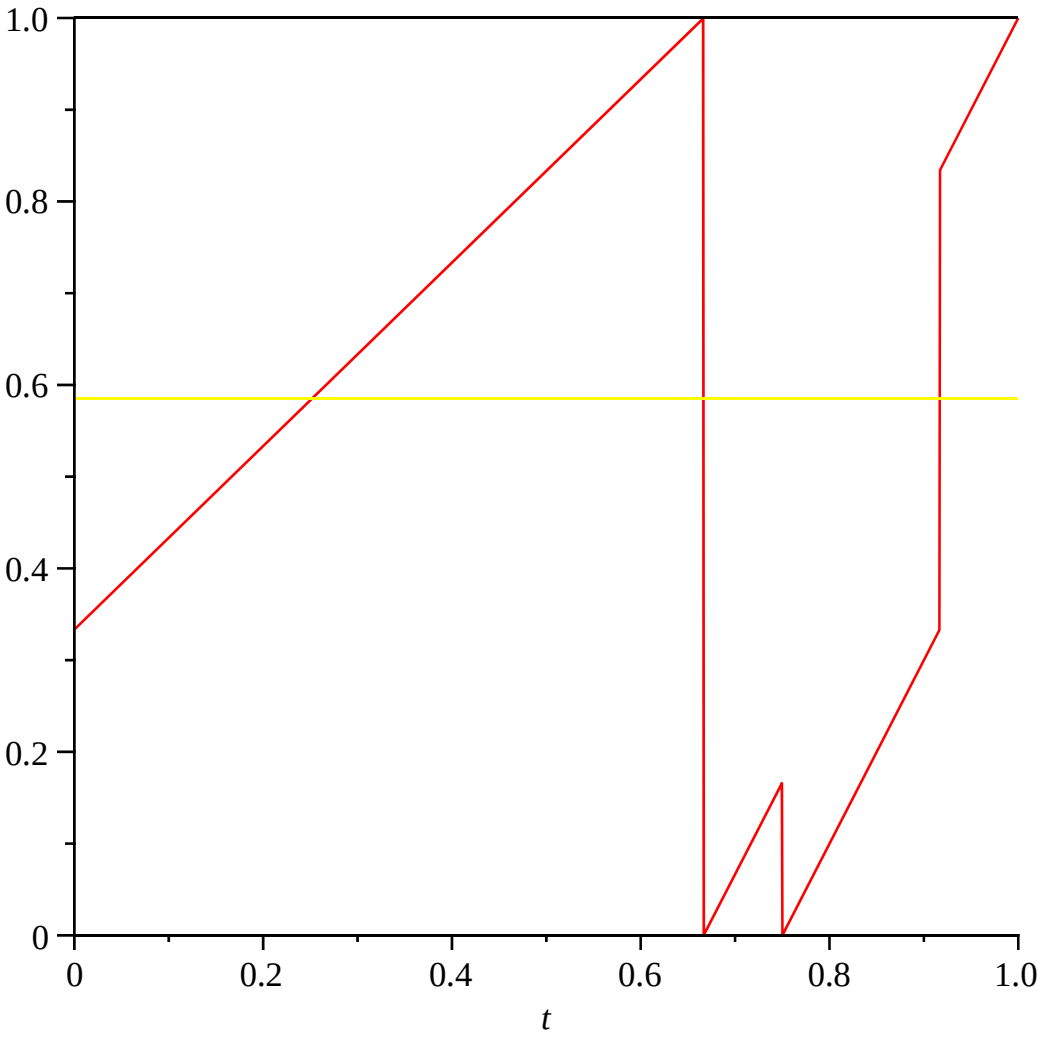
$err_3 := 1.000001745\ 10^{-10}$



$su := 0$

1

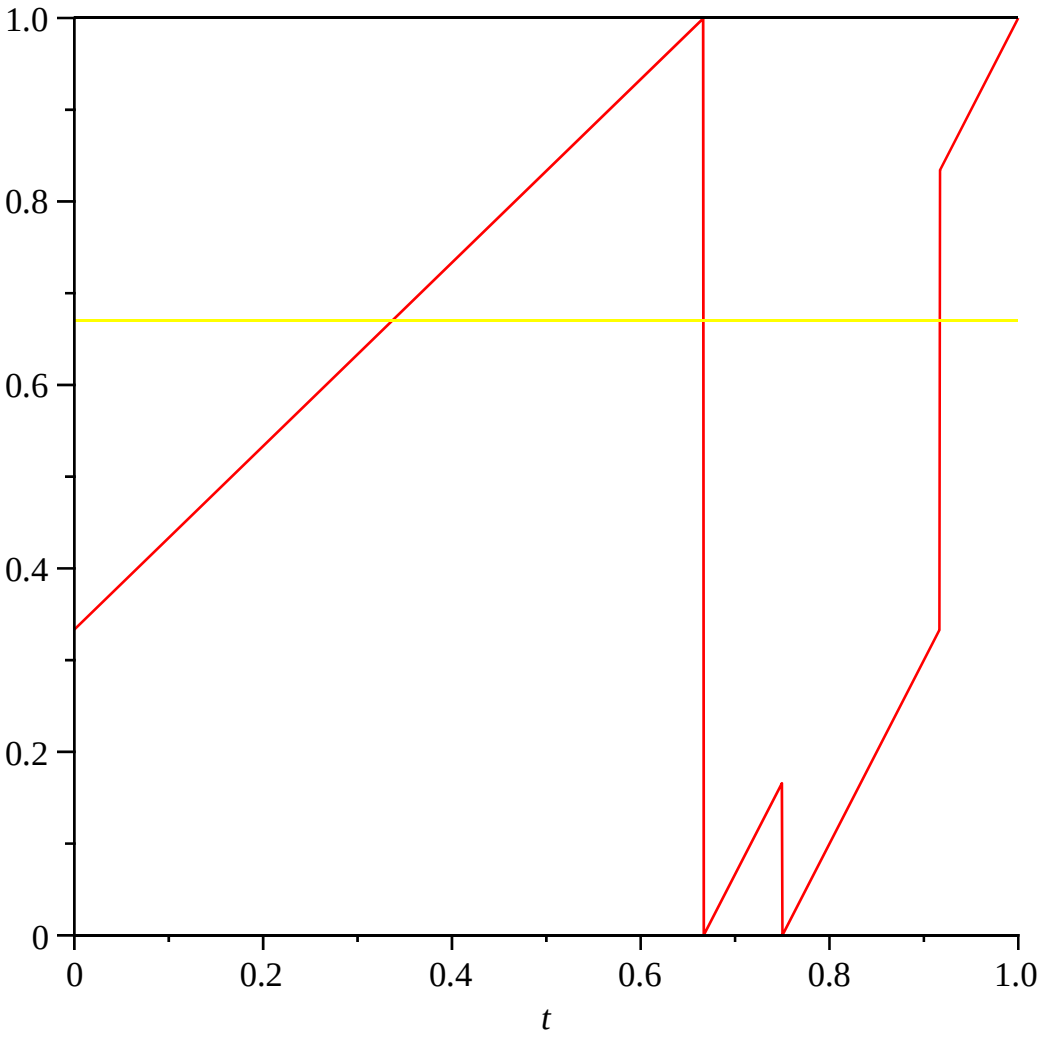
$err_4 := 1.000001745\ 10^{-10}$



$su := 0$

1

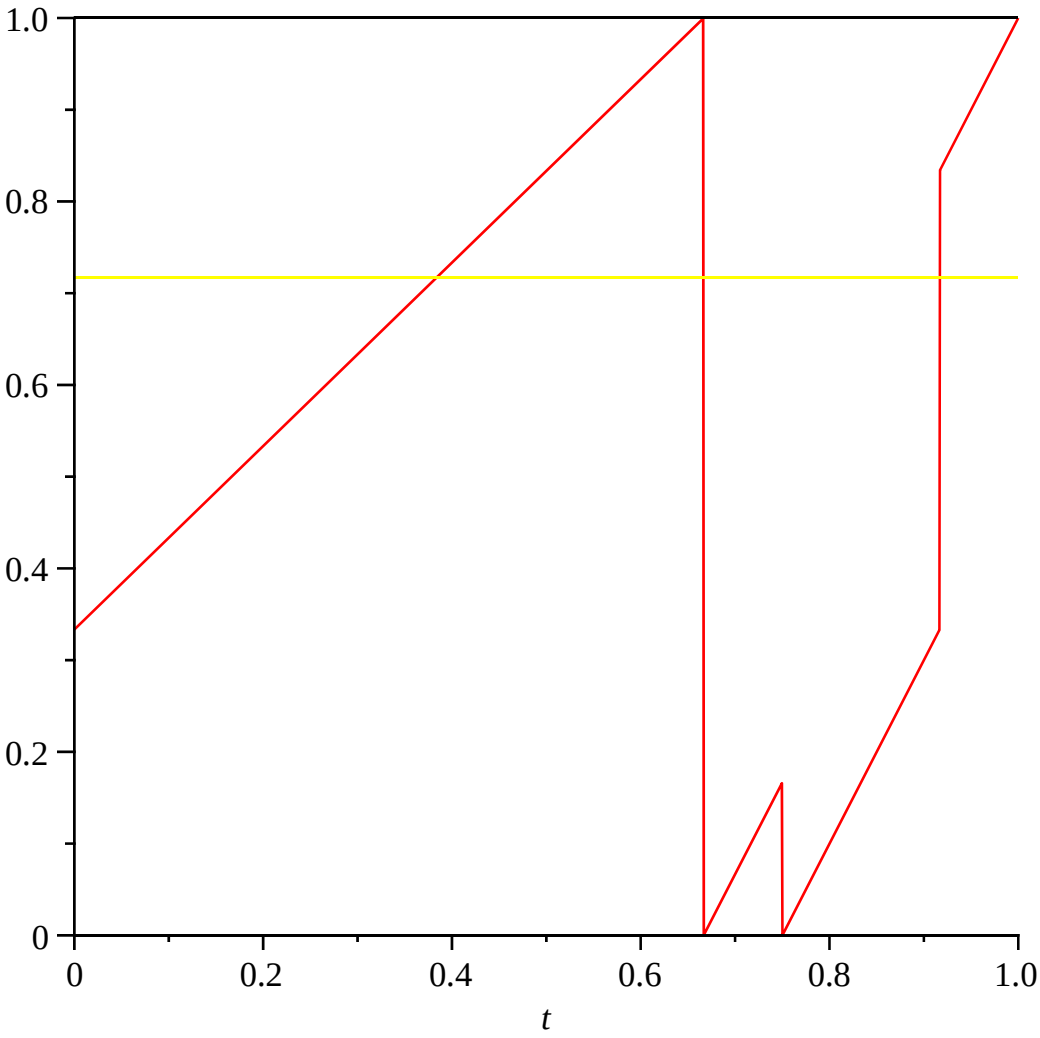
$err_5 := 1.000001745\ 10^{-10}$



$su := 0$

1

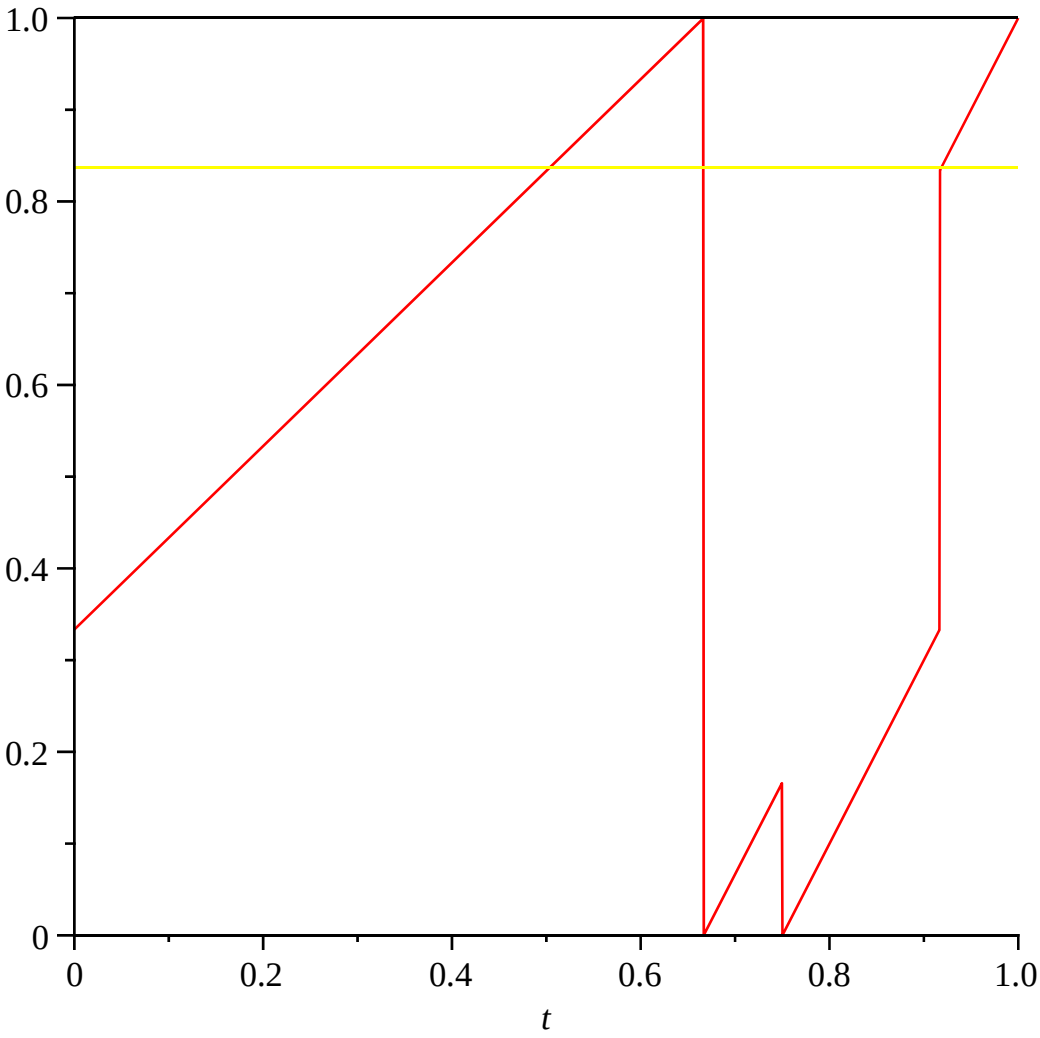
$err_6 := 1.000001745\ 10^{-10}$



$su := 0$

1

$err_7 := 1.000001745\ 10^{-10}$

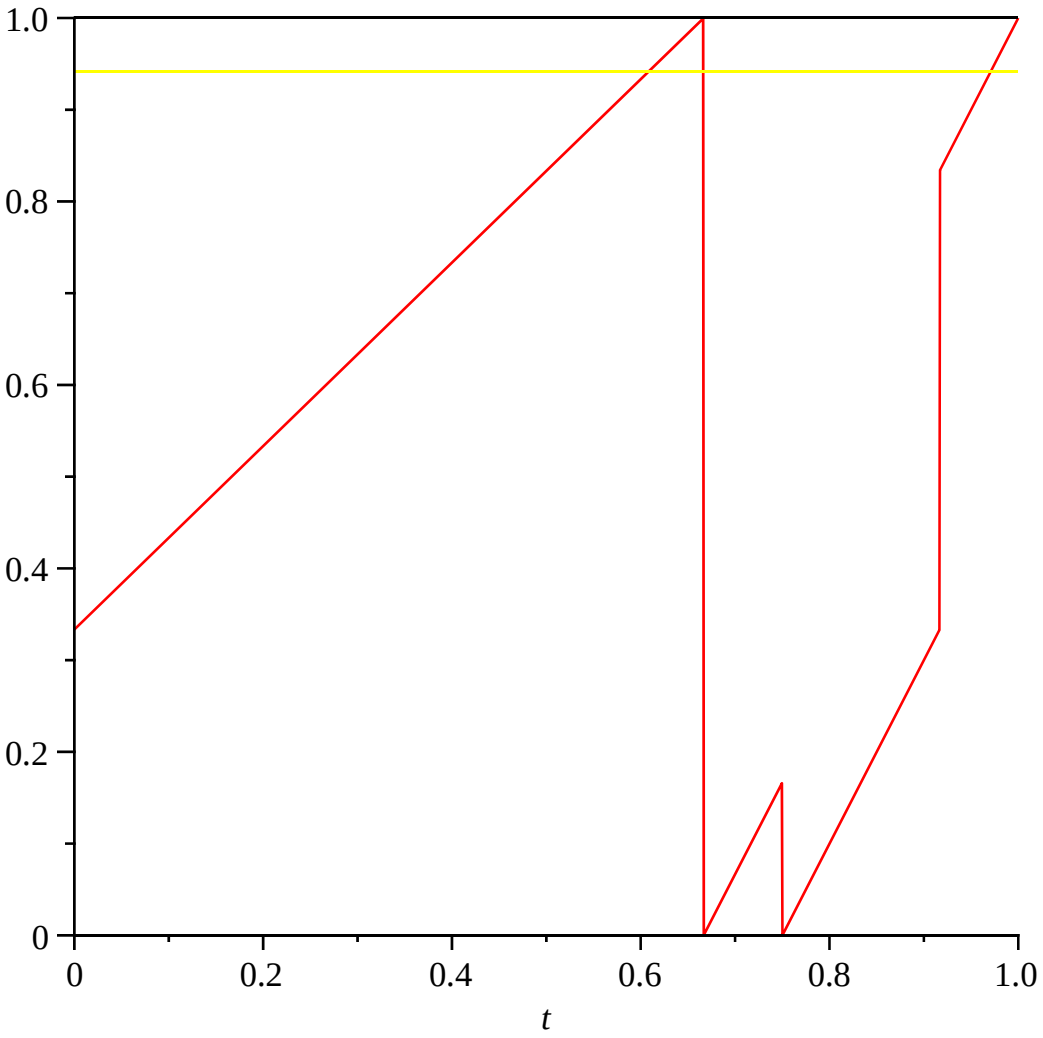


$su := 0$

1

4

$err_8 := 5.000034891\ 10^{-11}$



$su := 0$

1

4

$err_9 := 5.000034891\ 10^{-11}$

$y =, 0.0693953992$

$err[, 0,] =, -2.000003489\ 10^{-10}$

$y =, 0.1914298711$

$err[, 1,] =, -1.500005234\ 10^{-10}$

$y =, 0.2450147686$

$err[, 2,] =, -1.500005234\ 10^{-10}$

$y =, 0.3352263657$

$err[, 3,] =, 1.000001745\ 10^{-10}$

$y =, 0.4819571011$

$err[, 4,] =, 1.000001745\ 10^{-10}$

$y =, 0.5849634042$

$err[, 5,] =, 1.000001745\ 10^{-10}$

$y =, 0.6704117256$

$$err[,\ 6,\ ]=,\ 1.000001745\ 10^{-10}$$

$$y\ =,\ 0.7169276114$$

$$err[,\ 7,\ ]=,\ 1.000001745\ 10^{-10}$$

$$y\ =,\ 0.8371184668$$

$$err[,\ 8,\ ]=,\ 5.000034891\ 10^{-11}$$

$$y\ =,\ 0.9419107018$$

$$err[,\ 9,\ ]=,\ 5.000034891\ 10^{-11}$$

>