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


> alphad:=0.46;#between 0 and 1
  betad:=6.2;#  bigger than 1 if it is larger than 7 some
  correction to the
              # function int_of_x may be necessary
  N:=floor(1-alphad+betad)+1;
  EE:=vector(N,[]);
  for i from 1 to N do
  if rand()/10^(12)>0.5 then EE[i]:=1
                            else EE[i]:=-1 fi;
  od:
  print(`EE = `,EE);



  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

  if EE[1]>0 then alpha[1]:=1-alphad;  gamm[1]:=1-alpha[1];
             else alpha[1]:=1-alphad;gamm[1]:=0;  fi;
  if EE[N]>0 then alpha[N]:=frac(1-alphad+betad);  gamm[N]:=0;
             else alpha[N]:=frac(1-alphad+betad);gamm[N]:=1-alpha
  [N];  fi;

  for j from 2 to N-1 do
  alpha[j]:=1;gamm[j]:=0;
  od:



  i:='i':
  beta_const:=sum(alpha[i],i=1..N);
  i:='i':

  for j from 1 to N do
  beta[j]:=beta_const*EE[j];
  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]/abs(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
   if beta[j]>0 then a[j]:=ag[j]-gamm[j] else a[j]:=ag[j]-gamm[j]-
   alpha[j] fi;
    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 lower, 0 upper
leftc:=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 beta[j]>0 then
if (alpha[j]<1 and gamm[j]+alpha[j]=1) then  c[cj]:=b[j]; sidec
[cj]:=1;leftc[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;leftc[cj]:=1;

  j_of_c[cj]:=j;cj:=cj+1 ;
                    c[cj]:=b[j+1]; sidec[cj]:=0;leftc[cj]:=0;

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

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

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]:=0;leftc[cj]:=1;

     j_of_c[cj]:=j;cj:=cj+1 ;
                    c[cj]:=b[j+1]; sidec[cj]:=1;leftc[cj]:=0;

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

```
   leftc[cj]:=1;

    j_of_c[cj]:=j;cj:=cj+1 fi;
   end if;

   od:
   print(`c =`,c);
   print(`sidec =`,sidec);
   print(`leftc =`,leftc);
   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 left c[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],Tc[3],Tc[4]],x=0..1,thickness=
[2,1,1,1,1,1,1],numpoints=1000);
plot([`T(x)`,x,0,1,Tc[1],Tc[2]],x=0..1,thickness=[3,1,1,1,1,1,1,
1],color=[red,black,black,black,green,green],numpoints=1000);
```

$$alphad := 0.4600000000$$

$$betad := 6.2000000000$$

$$N := 7$$

$$EE := array(1..7, [\,])$$

$$EE = , \left[\begin{array}{ccccccc} 1 & -1 & 1 & 1 & -1 & -1 & 1 \end{array}\right]$$

$$\alpha_1 := 0.5400000000$$

$$gamm_1 := 0.4600000000$$

$$\alpha_7 := 0.7400000000$$

$$gamm_7 := 0$$

$$\beta const := 6.2800000000$$

$$alpha = , \left[\begin{array}{ccccccc} 0.5400000000 & 1 & 1 & 1 & 1 & 1 & 0.7400000000 \end{array}\right]$$

$$beta = , [6.2800000000, -6.2800000000, 6.2800000000, 6.2800000000, -6.2800000000, -6.2800000000, 6.2800000000]$$

$$gamma = , \left[\begin{array}{ccccccc} 0.4600000000 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

$$b = , [0, 0.0859872611, 0.2452229299, 0.4044585987, 0.5636942675, 0.7229299363, 0.8821656051, 1]$$

$$ag = , [0.0000000000, -0.5400000000, 1.5400000000, 2.5400000000, -3.5400000000, -4.5400000000, 5.5400000000]$$

$$al = , [-0.4600000000, -2.5400000000, 1.5400000000, 2.5400000000, -5.5400000000, -6.5400000000, 5.2800000000]$$

$$a = , [-0.4600000000, -1.5400000000, 1.5400000000, 2.5400000000, -4.5400000000, -5.5400000000, 5.5400000000]$$

$$gamma = , \left[\begin{array}{ccccccc} 0.4600000000 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

$$Kc = , 2$$

$$c = , \left[\begin{array}{cccccccccccccc} 0 & 1 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & c_{11} & c_{12} & c_{13} & c_{14} \end{array}\right]$$

$sidec =, \left[1, 0, sidec_3, sidec_4, sidec_5, sidec_6, sidec_7, sidec_8, sidec_9, sidec_{10}, sidec_{11}, sidec_{12}, sidec_{13}, sidec_{14}\right]$

$leftc =, \left[1, 0, leftc_3, leftc_4, leftc_5, leftc_6, leftc_7, leftc_8, leftc_9, leftc_{10}, leftc_{11}, leftc_{12}, leftc_{13}, leftc_{14}\right]$

$j\_of\_c =, \left[1, 7, j\_of\_c_3, j\_of\_c_4, j\_of\_c_5, j\_of\_c_6, j\_of\_c_7, j\_of\_c_8, j\_of\_c_9, j\_of\_c_{10}, j\_of\_c_{11}, j\_of\_c_{12}, j\_of\_c_{13}, j\_of\_c_{14}\right]$

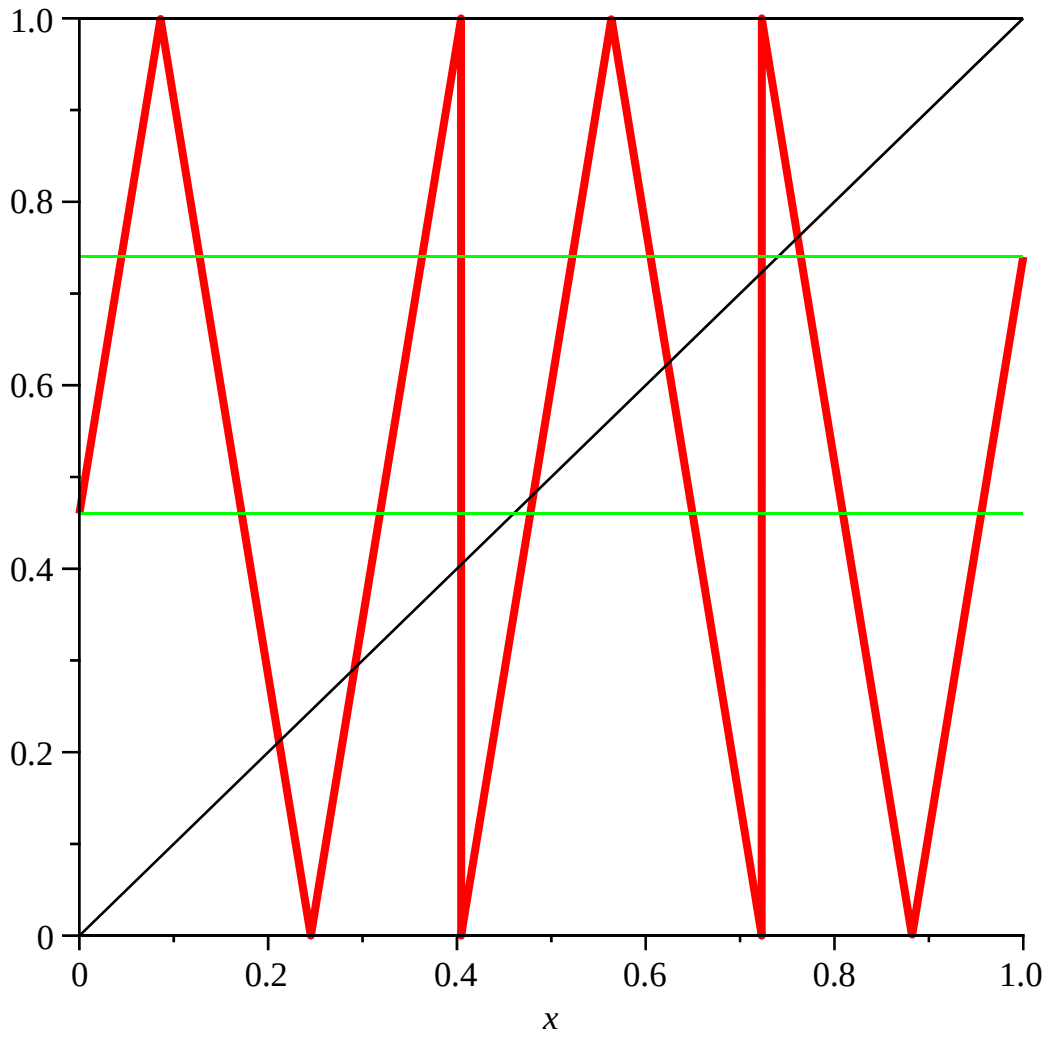
$uint\_of\_x := x \rightarrow piecewise\left(x < b_2, 1, x < b_3, 2, x < b_4, 3, x < b_5, 4, x < b_6, 5, x < b_7, 6, x < b_8, 7, x < b_9, 8, 9\right)$

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

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

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

$$Tc =, \left[\begin{array}{cccc} 0.4600000000 & 0.7400000000 & Tc_3 & Tc_4 \end{array}\right]$$



>

```
>
> ud:=vector(50):Digits:=100;NN:=50;#Expansion with variable
  slopes
  d:=vector(50):
  xx:=evalf(rand()/10^12);
  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;
```

$$Digits := 100$$

$$NN := 50$$

$$xx := 0.5041800682$$

$$[2.5400000000, -4.5400000000, -4.5400000000, -5.5400000000, 5.5400000000, -4.5400000000, 5.5400000000, 1.5400000000, 2.5400000000, 2.5400000000, -1.5400000000, 2.5400000000, 2.5400000000, 1.5400000000, -5.5400000000, 2.5400000000, -0.4600000000, -4.5400000000, -5.5400000000, 5.5400000000, -1.5400000000, -5.5400000000, 1.5400000000, -0.4600000000, -4.5400000000, -4.5400000000, 2.5400000000, 1.5400000000, 2.5400000000, 1.5400000000, 1.5400000000, 2.5400000000, -1.5400000000, -0.4600000000, -5.5400000000, -4.5400000000, 1.5400000000, -4.5400000000, 5.5400000000, -0.4600000000, -5.5400000000, -5.5400000000, -5.5400000000, -4.5400000000, 2.5400000000, -4.5400000000, -5.5400000000, -4.5400000000, -4.5400000000, -1.5400000000]$$

$$uIs\_it\_x := 0.5041800682$$

$$[2.5400000000, -4.5400000000, -4.5400000000, -5.5400000000, 5.5400000000, -4.5400000000, 5.5400000000, 1.5400000000, 2.5400000000, 2.5400000000,$$

-1.5400000000, 2.5400000000, 2.5400000000, 1.5400000000, -5.5400000000,
2.5400000000, -0.4600000000, -4.5400000000, -5.5400000000, 5.5400000000,
-1.5400000000, -5.5400000000, 1.5400000000, -0.4600000000, -4.5400000000,
-4.5400000000, 2.5400000000, 1.5400000000, 2.5400000000, 1.5400000000,
1.5400000000, 2.5400000000, -1.5400000000, -0.4600000000, -5.5400000000,
-4.5400000000, 1.5400000000, -4.5400000000, 5.5400000000, -0.4600000000,
-5.5400000000, -5.5400000000, -5.5400000000, -4.5400000000, 2.5400000000,
-4.5400000000, -5.5400000000, -4.5400000000, -4.5400000000, -1.5400000000]

$$Is\_it\_x := 0.5041800682$$

$$terr := -1.351648018\ 10^{-41}$$

$$err := -1.351648018\ 10^{-41} \quad (1)$$

>
>

>

```
> 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:
  varleftc:=leftc[i];
       for n from 1 to NN+1 do

        if varleftc>0   then intx:=uint_of_x(xxt) else intx:=
  int_of_x(xxt) fi;

           varleftc:=varleftc*sign(beta[intx]);
           bet_real:=bet;
           beta_one[i,n]:=beta[intx];
           bet:=bet/beta[intx];
           dcb[i,n]:=a[intx]*bet;

          if leftc[i]=0  then
              if bet_real>0 then
               for ii from 1 to Kc do
                    if xxt>c[ii]+10^(-20) 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/abs(beta[j7]),j7=1..
```

```
intx-1)*abs(bet_real) fi;
                              else
             for ii from 1 to Kc do
                  if xxt<c[ii]-10^(-20) 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/abs(beta[j8]),j8=
intx+1..N)*abs(bet_real) fi;
            end if;


             ################
                    else
            if bet_real>0 then
                    for ii from 1 to Kc do
                            if xxt<c[ii]-10^(-20) 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/abs(beta[j8]),j8=
intx+1..N)*abs(bet_real) fi;
                           else
                     for ii from 1 to Kc do
                            if xxt>c[ii]+10^(-20) 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/abs(beta[j7]),j7=1..
intx-1)*abs(bet_real) fi;
             end if;



      fi;
      valc[i,n]:=xxt;
      betc[i,n]:=bet_real;
if bet_real>0 then
      if leftc[i]=1 then
                     Rounding:=infinity;
                     xxt:=uT(xxt) :if xxt>1 then xxt:=1.00 fi;
                     else
                     Rounding:=0;
                        xxt:=T(xxt): if xxt<0 then xxt:=0.00 fi;
      fi;
               else
if leftc[i]=0 then
```

```
                    Rounding:=infinity;
                    xxt:=uT(xxt) :if xxt>1 then xxt:=1.00 fi;
                    else
                    Rounding:=0;
                        xxt:=T(xxt):if xxt<0 then xxt:=0.00 fi;
      fi;
end if;
      Rounding:=nearest;#print(xxt);
      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,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(abs(cc[i,j,j1+1]),j1=1..NN));

#print(`SS[`,i,j,`] =`,SS[i,j]):
od;od:
for i from 1 to 20 do
#print(valc[2,i],valc[3,i]);
od;
```

$$NN := 150$$

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

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

$$xxt := 0$$

$$bet := 1$$

$$varleftc := 1$$

$$Is\_it\_x := 2.284651978\ 10^{-102}$$

$$xxt := 1$$

$$bet := 1$$

$$varleftc := 0$$

$$Is\_it\_x := 1.0000000000$$

$$S_1 := 0.0938527644$$

$$S_2 := 0.1307505249 \qquad \textbf{(2)}$$

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

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


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

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

MMM[i,i]:=MMM[i,i]+1;
od:

print(MMM);
print(ve);



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

$$MM = , \begin{bmatrix} -0.0007646809 & -0.1593543602 \\ -0.1886292585 & -0.0300395792 \end{bmatrix}$$

$$eigenvalues\ MM =, 0.1585896793, -0.1893939394$$

$$\begin{bmatrix} 0.9992353191 & -0.1593543602 \\ -0.1886292585 & 0.9699604208 \end{bmatrix}$$

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

$$DD := \begin{bmatrix} 1.2024734895 & 1.2648162301 \end{bmatrix}$$

$$0.0000000000 \qquad \textbf{(3)}$$

```
>
>


density:=proc(t) local j,i, den ,i1;
          i1:='i1':

          den:=1:
     for j from 1 to Kc do
```

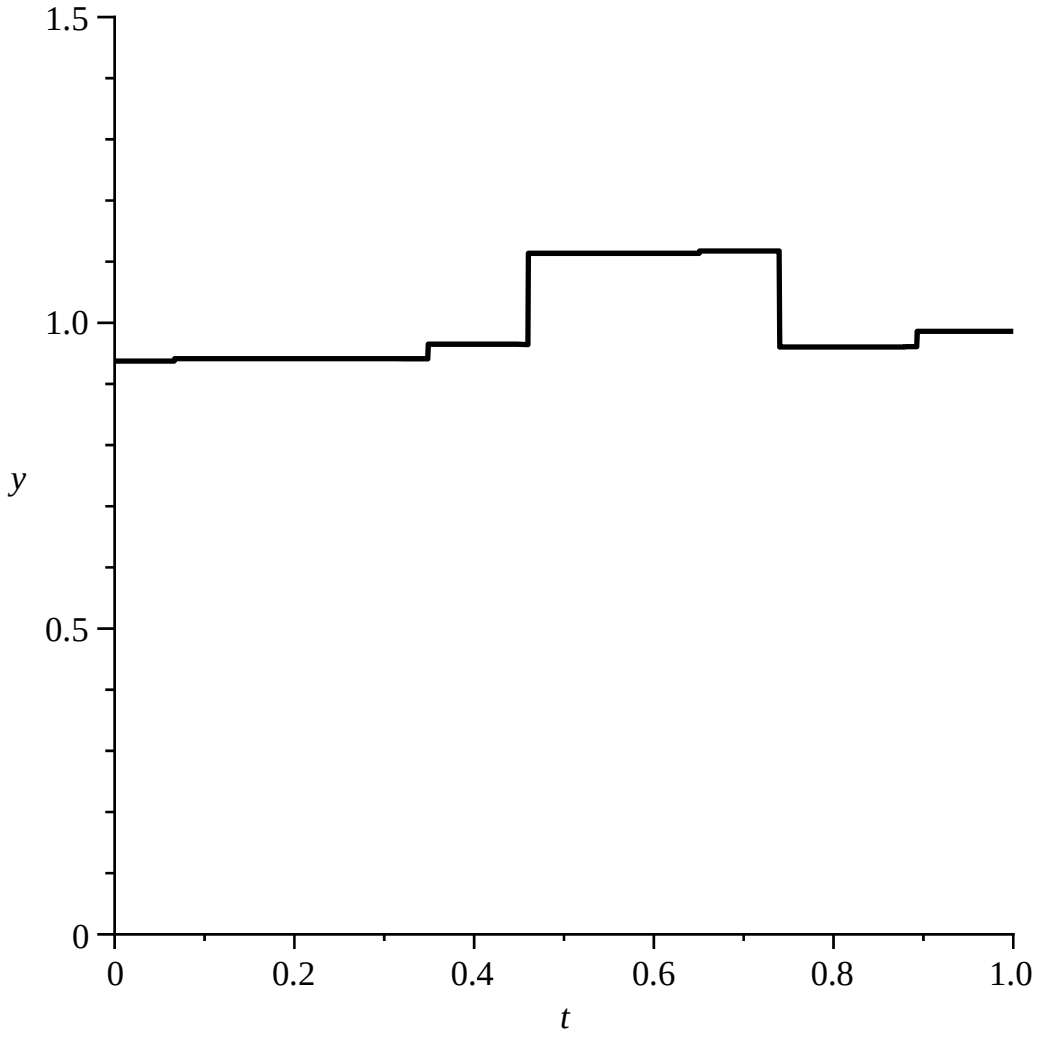
```
                    if leftc[j]=0 then
                    for i1 from 1 to 50 do
                            if betc[j,i1+1]>0 then den:=den+ DD[j]*chi
    (0,valc[j,i1+1],t)*abs(betc[j,i1+1]);
                                                    else   den:=den+ DD[j]*chi
    (valc[j,i1+1],1,t)*abs(betc[j,i1+1]);
                            fi;
                    od;
                    fi;

                     if leftc[j]=1 then
                      for i1 from 1 to 50 do
                            if betc[j,i1+1]<0 then den:=den+ DD[j]*chi
    (0,valc[j,i1+1],t)*abs(betc[j,i1+1]);
                                                    else   den:=den+ DD[j]*chi
    (valc[j,i1+1],1,t)*abs(betc[j,i1+1]);
                            fi;
                    od;
                    fi;
                od;
                    return den;
                end proc;
    #Normalizing factor

    NC:=int(density(t),t=0..1);

    print(`NC = `,NC);

    plot([(1/NC)*'density(t)'],t=0..1-0.000001,y=0..1.5,color=black,
    thickness=2);
```

```
density := proc(t)
    local j, i, den, i1;
    i1 := 'i1';
    den := 1;
    for j to Kc do
        if leftc[j] = 0 then
            for i1 to 50 do
                if 0 < betc[j, i1 + 1] then
                    den := den + DD[j] * chi(0, valc[j, i1 + 1], t) * abs(betc[j, i1 + 1])
                else
                    den := den + DD[j] * chi(valc[j, i1 + 1], 1, t) * abs(betc[j, i1 + 1])
                end if
            end do
        end if;
```

```
        if leftc[j] = 1 then
            for i1 to 50 do
                if betc[j, i1 + 1] < 0 then
                    den := den + DD[j] * chi(0, valc[j, i1 + 1], t) * abs(betc[j, i1 + 1])
                else
                    den := den + DD[j] * chi(valc[j, i1 + 1], 1, t) * abs(betc[j, i1 + 1])
                end if
            end do
        end if
    end do;
    return den
end proc
```

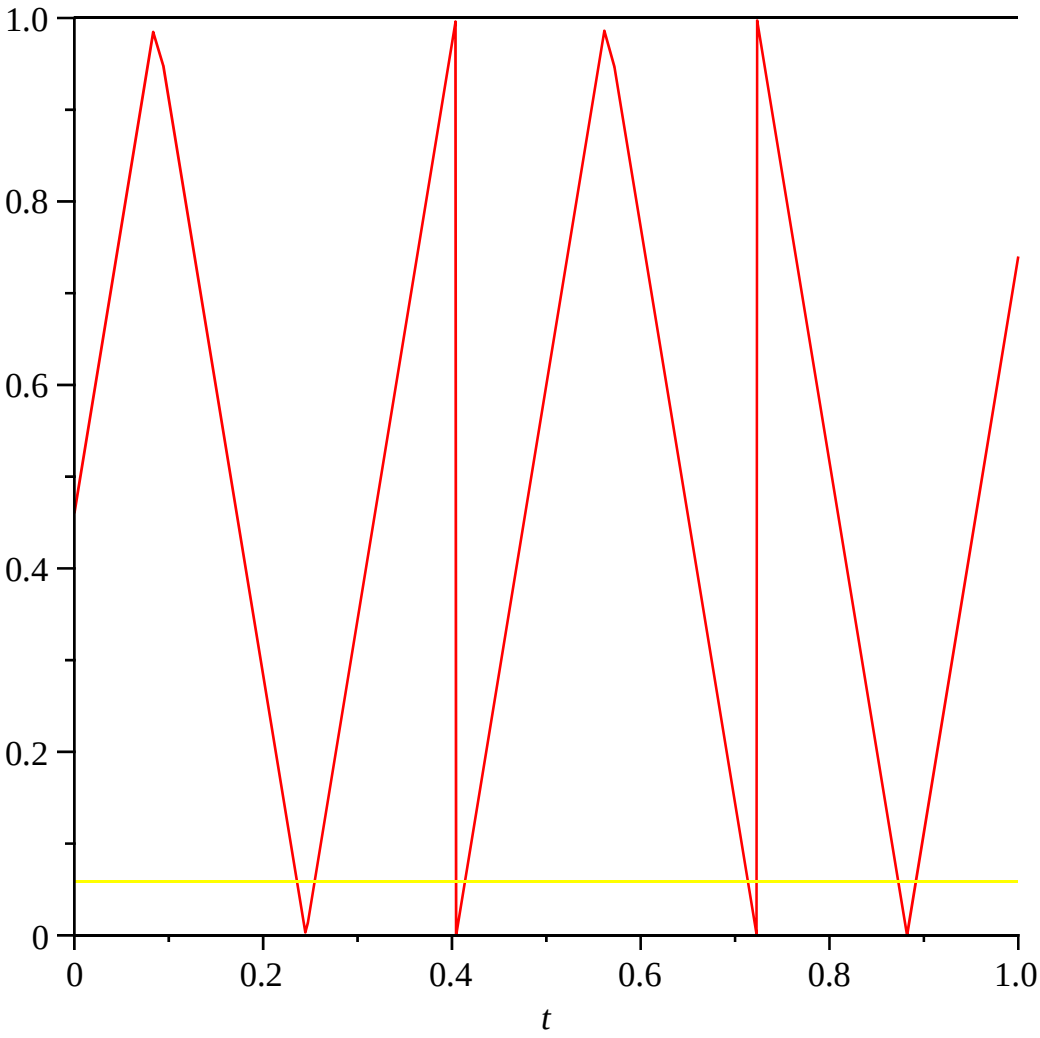
$$NC := 1.2827057271$$

$$NC = , 1.2827057271$$




```
>
>
   #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];
#print(y[j6],pre[i3],T(pre[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])/abs(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],`err[`,j6,`]=`,err[j6]);

od;
```



$su := 0$

2

3

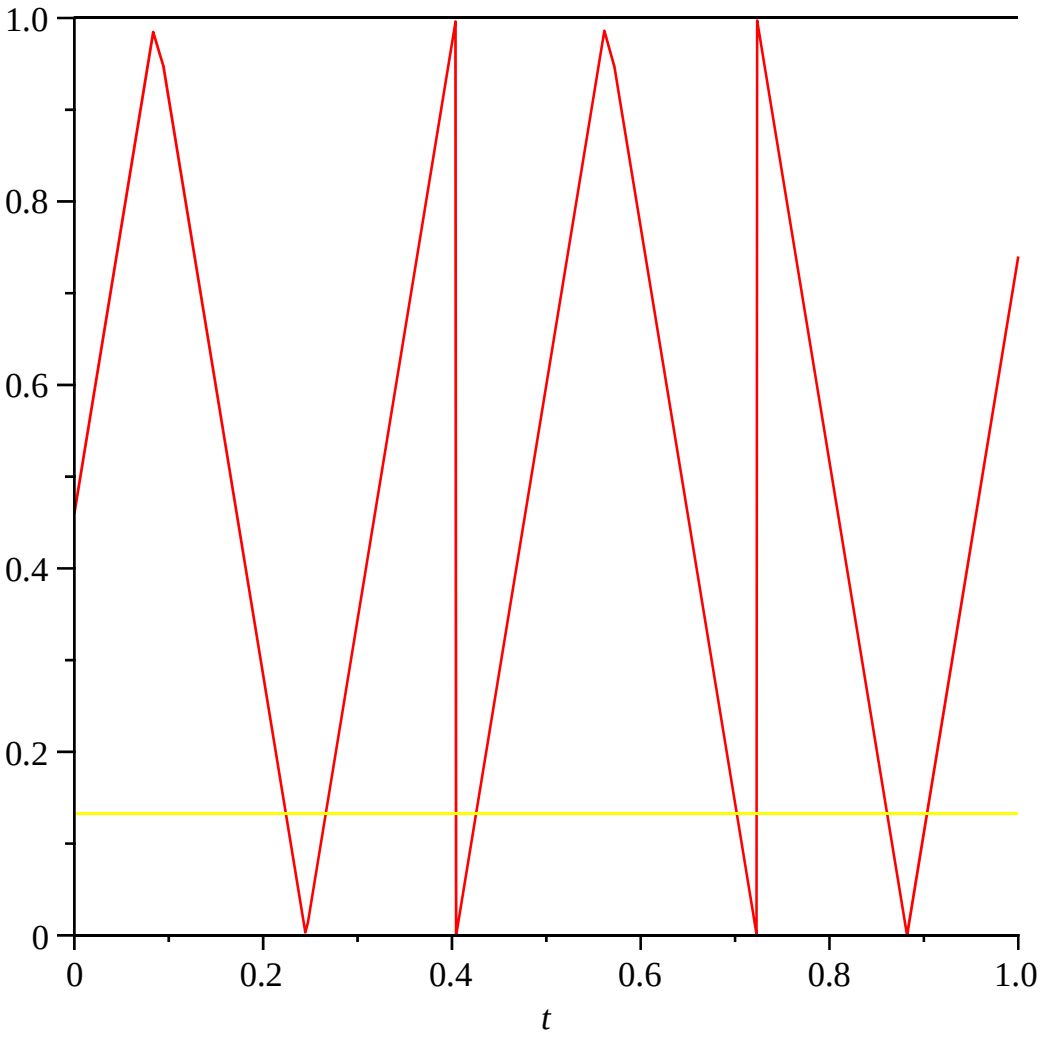
4

5

6

7

$err_0 := -1.267910517\ 10^{-41}$



$su := 0$

2

3

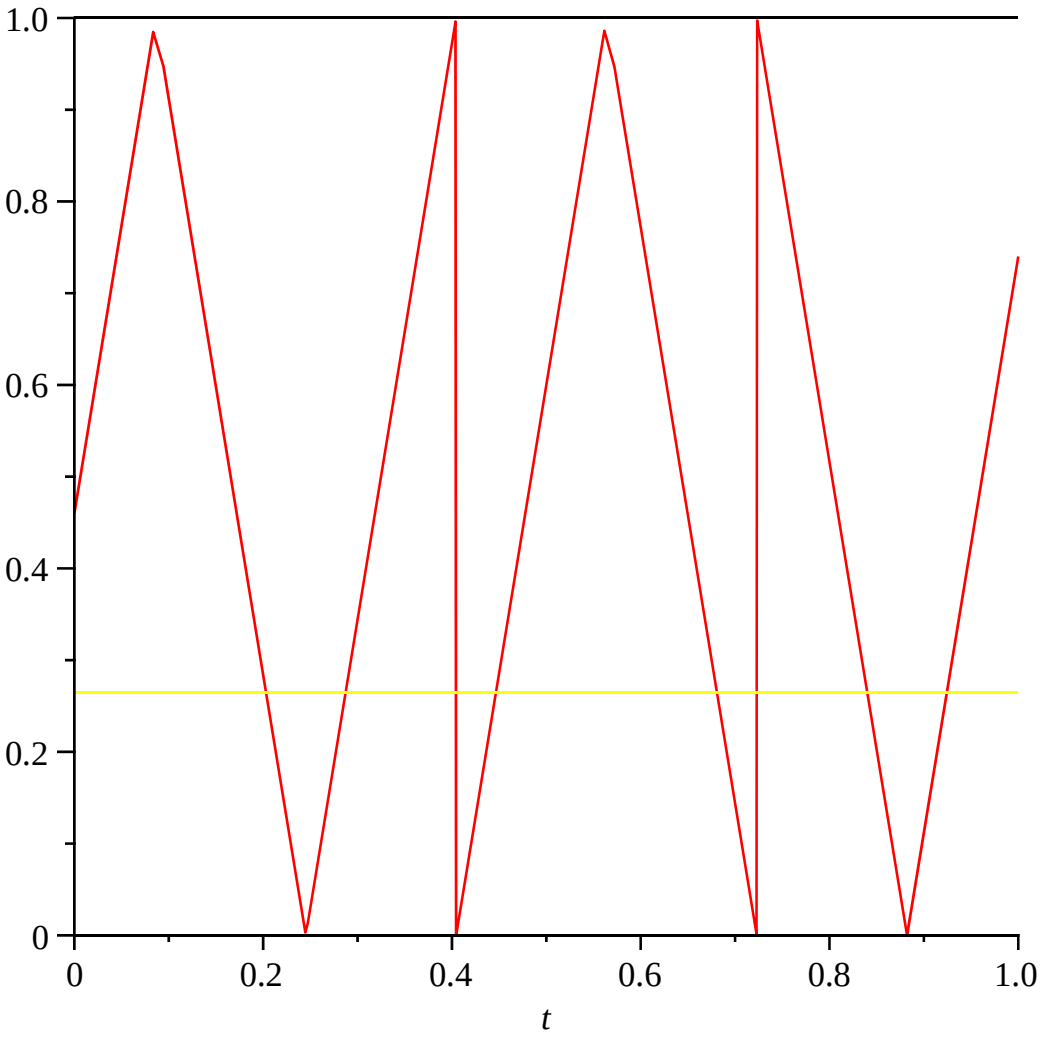
4

5

6

7

$err_1 := -1.267910517\ 10^{-41}$



$su := 0$

2

3

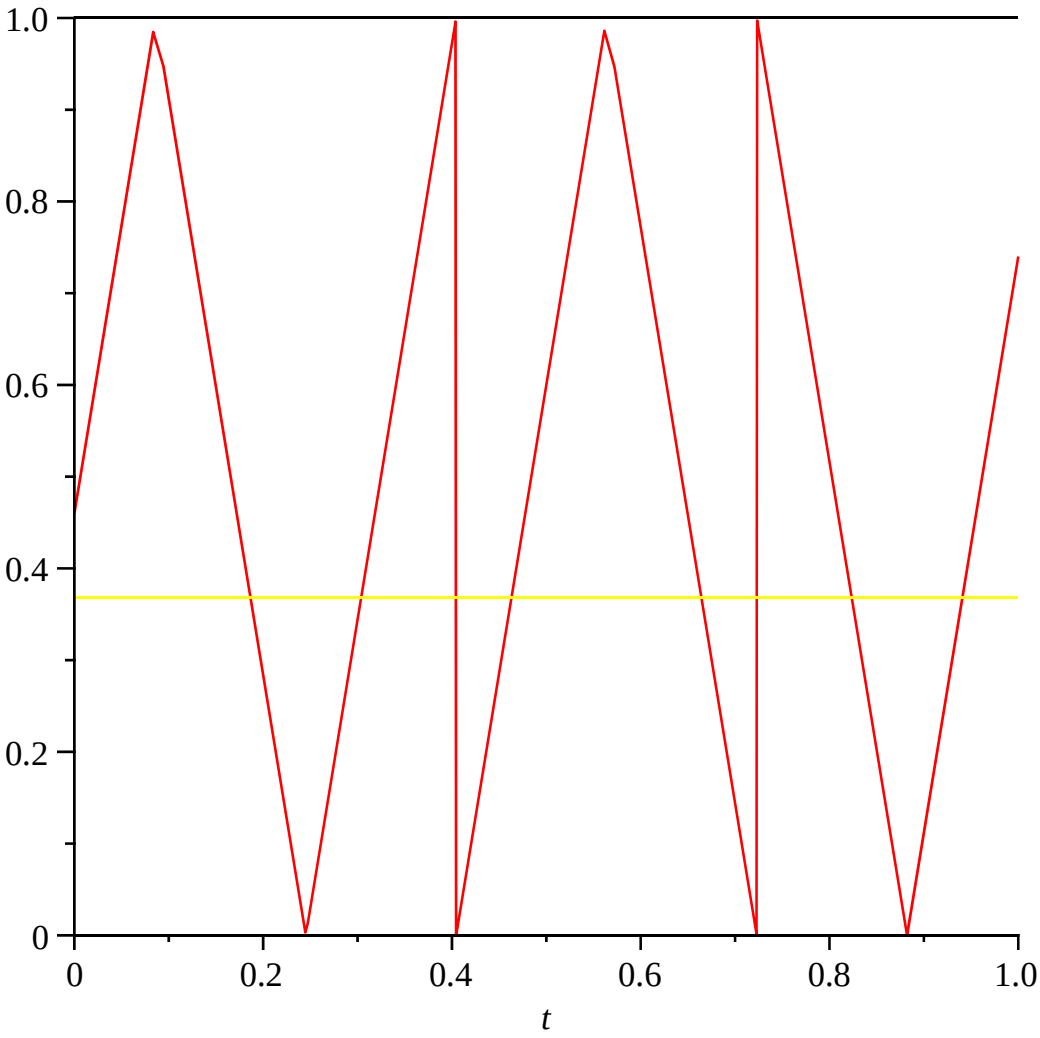
4

5

6

7

$err_2 := -1.267910517\ 10^{-41}$



$su := 0$

2

3

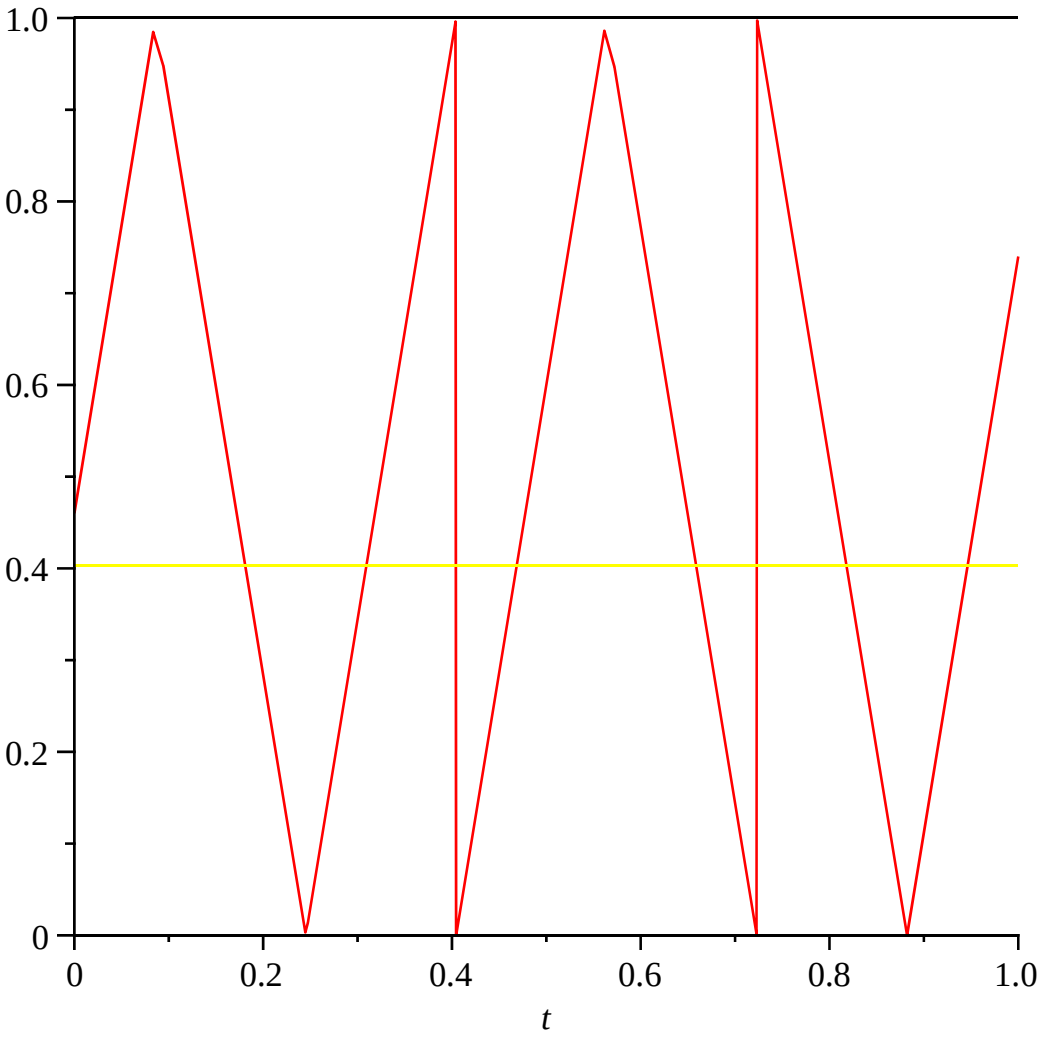
4

5

6

7

$err_3 := -1.267910517\ 10^{-41}$



$su := 0$

2

3

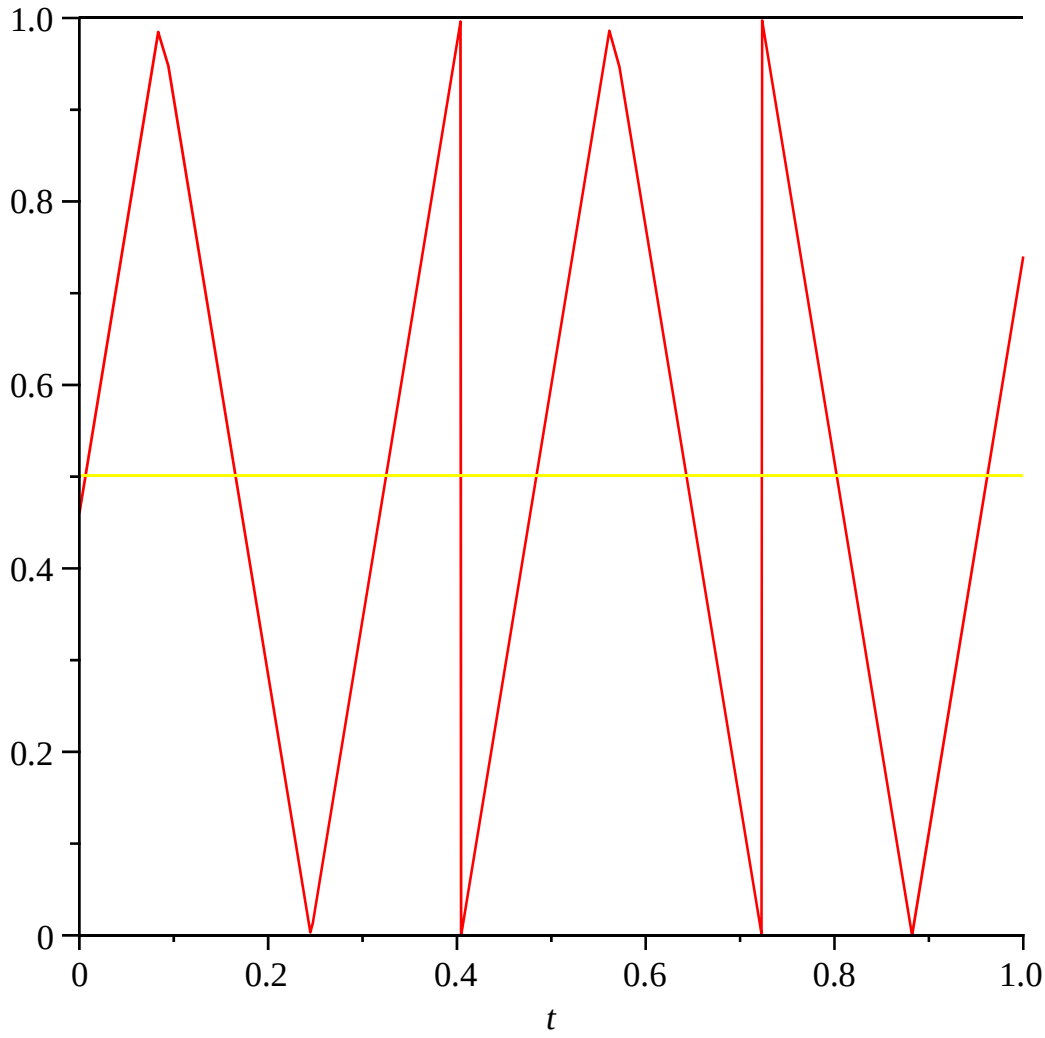
4

5

6

7

$err_4 := -1.267910517\ 10^{-41}$



$su := 0$

1

2

3

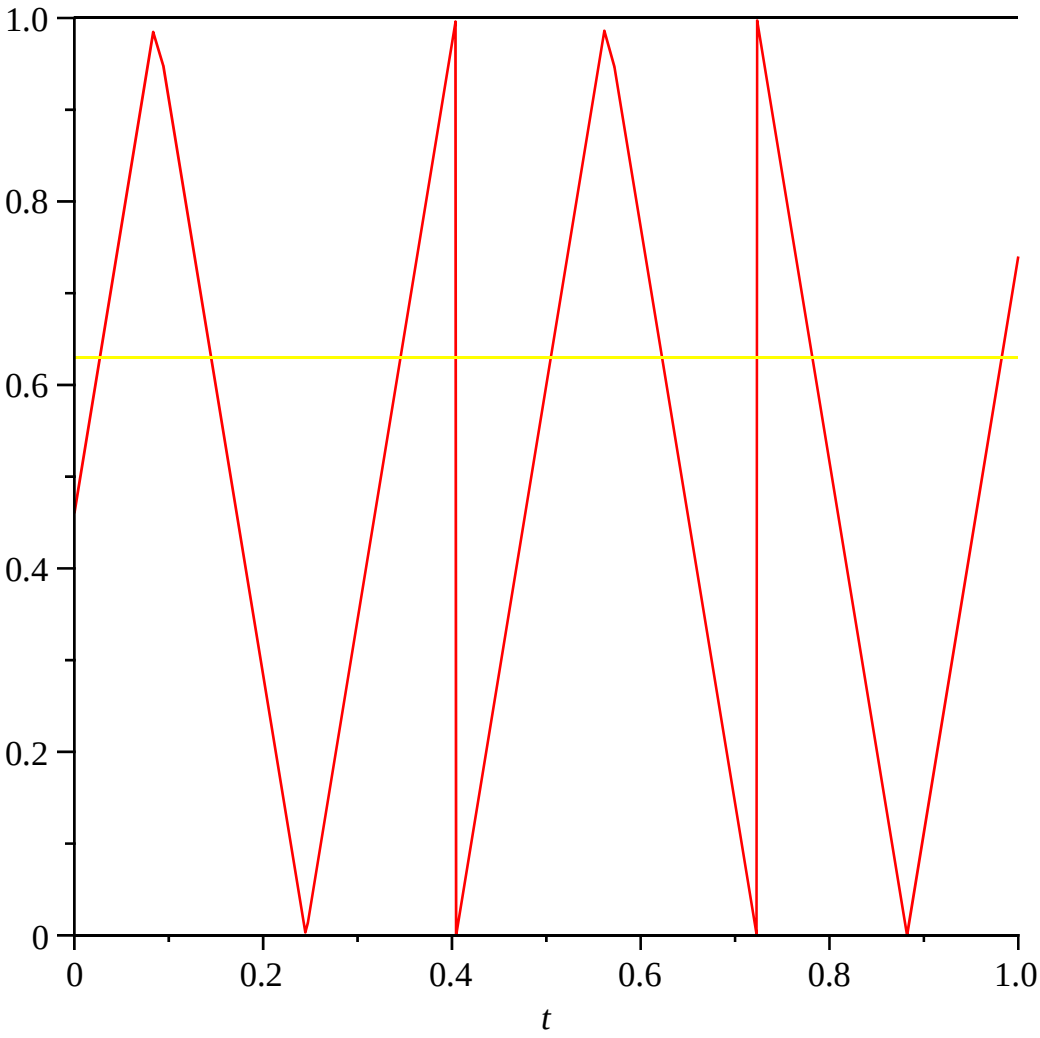
4

5

6

7

$err_5 := 1.612872487\ 10^{-41}$



$su := 0$

1

2

3

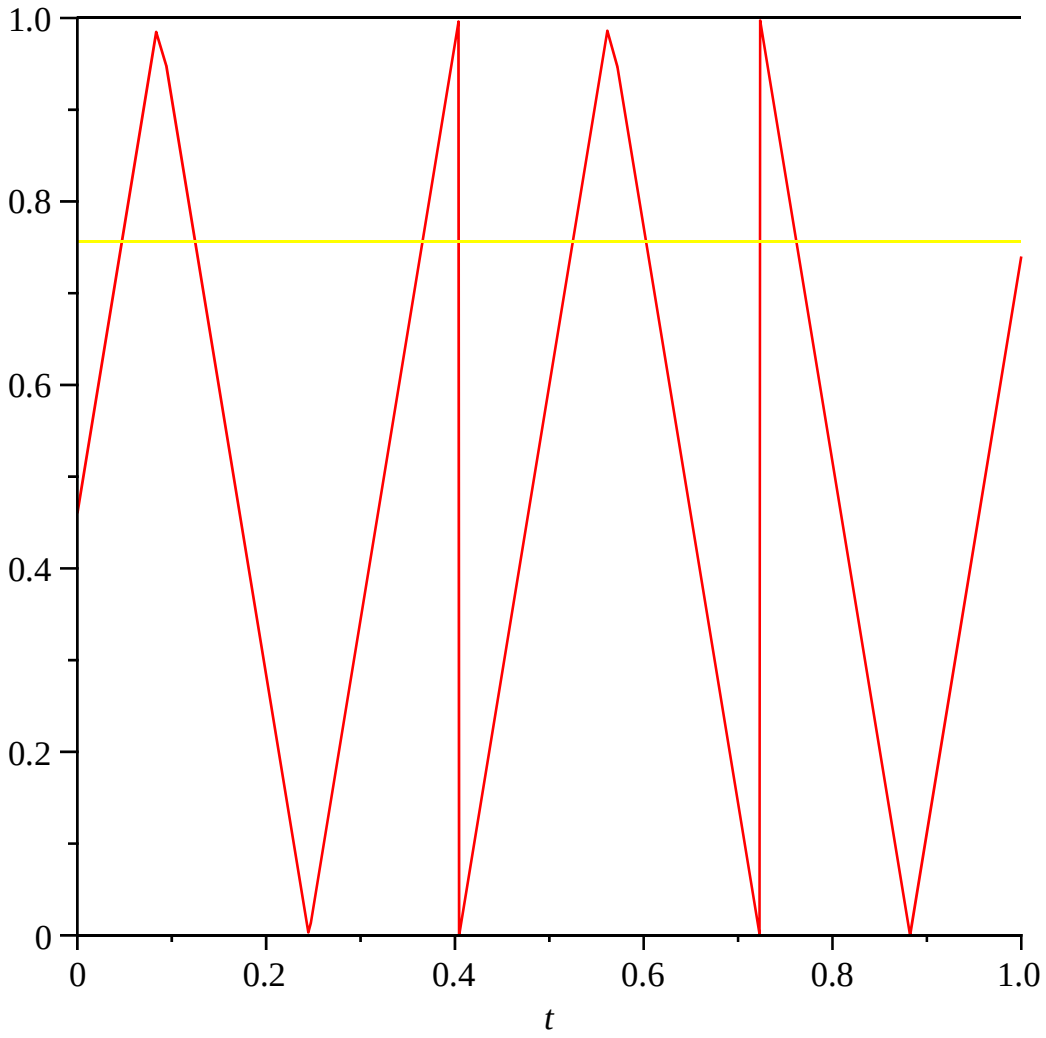
4

5

6

7

$err_6 := 1.612872487\ 10^{-41}$



$su := 0$

1

2

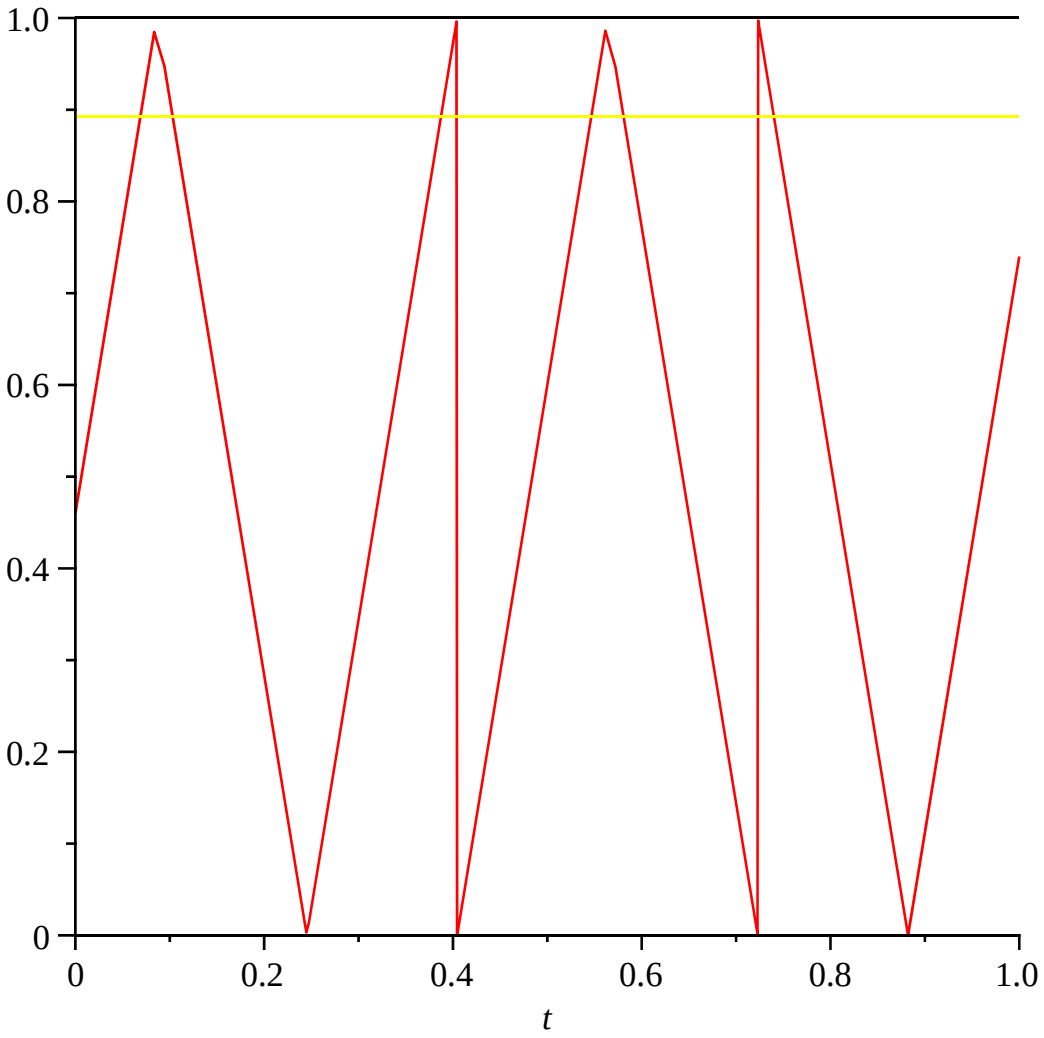
3

4

5

6

$err_7 := 1.130762429\ 10^{-41}$



$su := 0$

1

2

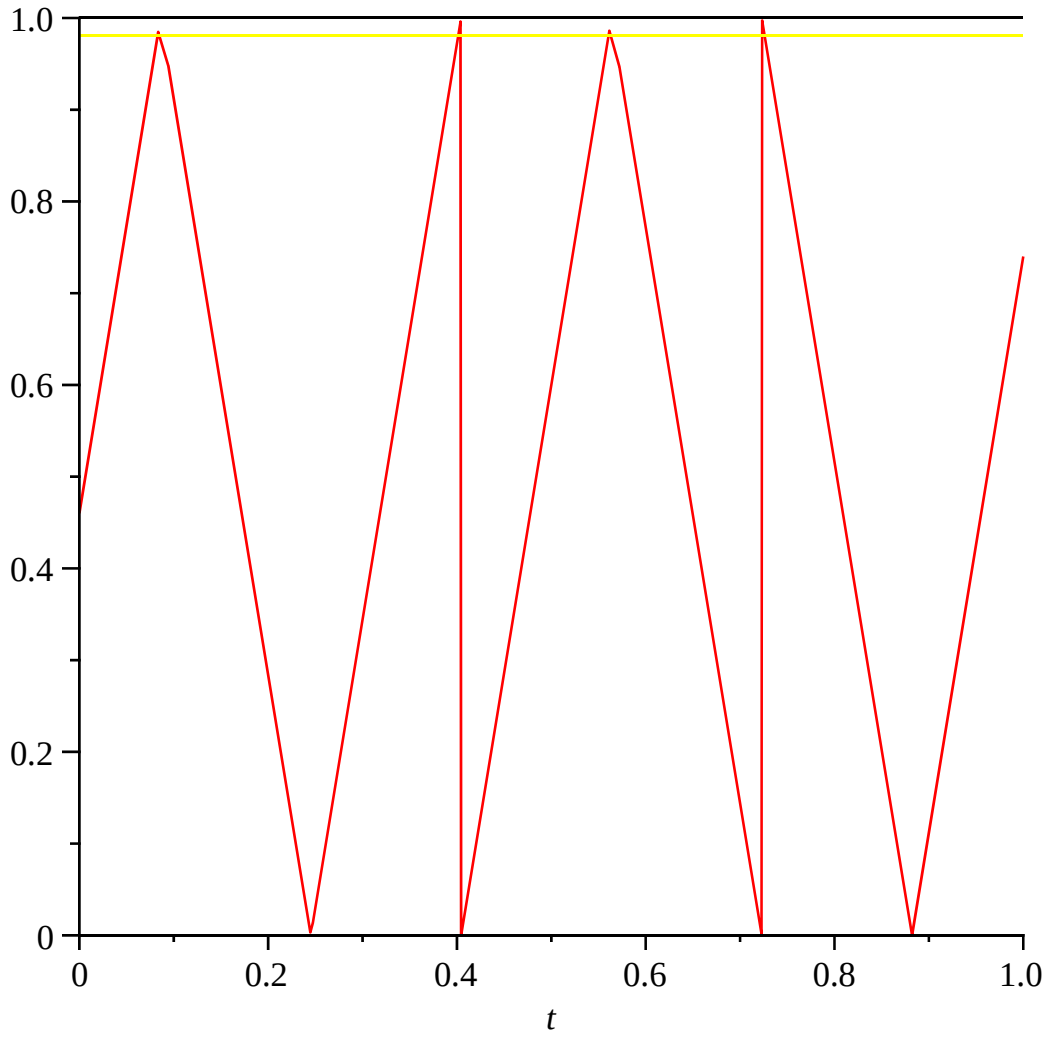
3

4

5

6

$err_8 := 1.130762429\ 10^{-41}$



$su := 0$

1

2

3

4

5

6

$err_9 := -1.416569852\ 10^{-41}$

$y =, 0.0583804136, err[, 0,]=, -1.267910517\ 10^{-41}$

$y =, 0.1332585592, err[, 1,]=, -1.267910517\ 10^{-41}$

$y =, 0.2642614553, err[, 2,]=, -1.267910517\ 10^{-41}$

$y =, 0.3683413251, err[, 3,]=, -1.267910517\ 10^{-41}$

$y =, 0.4036386777, err[, 4,]=, -1.267910517\ 10^{-41}$

$y =, 0.5010051167, err[, 5,]=, 1.612872487\ 10^{-41}$

$y =, 0.6294333430, err[, 6,]=, 1.612872487\ 10^{-41}$

$y =, 0.7561421378, err[, 7,]=, 1.130762429\ 10^{-41}$

$y =, 0.8927663026, err[, 8,]=, 1.130762429\ 10^{-41}$

y =, 0.9806253380, $err[$, 9, $]$=, $-1.416569852\ 10^{-41}$

>