```
> 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
  #Lazy
  alpha[1]:=0.5:beta[1]:=2:gamm[1]:=0.5:

  alpha[2]:=1:beta[2]:=3:gamm[2]:=0.0:#
  alpha[3]:=0.8:beta[3]:=4:gamm[3]:=0.2:  #
  #alpha[4]:=0.4:beta[4]:=5:gamm[4]:=0.4:
  #alpha[5]:=1.0: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]:=0.3:                gamm[N]:=0.7:
  i:='i':beta[N]:=alpha[N]/(1-sum(alpha[i]/beta[i],i=1..N-1));
  if beta[N] < 0 then print(`ERROR - make beta's larger `) fi;

  print(`alpha =`,alpha);
  print(`beta =`,beta);
  print(`gamma =`,gamm);
  i:='i':
  beta_const:=sum(alpha[i],i=1..N);
  i:='i':
  #for j from 1 to N do
  #beta[j]:=beta_const;
  #od:


  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);


>
>


>
   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);
   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- >bet a[ ui nt _of _x( x) ] * x- a[ ui nt _of _x( x) ] ;
T: =x- >bet a[ i nt _of _x( x) ] * x- a[ i nt _of _x( x) ] ;
Tc: =vect or ( Kc+2, [ ] ) :
f or  j  f r om 1  t o  Kc  do
i f  si dec[ j ] =0  t hen    Tc[ j ] : =T( c[ j ] ) ;
                el se   Tc[ j ] : =uT( c[ j ] ) f i ;
od:

pr i nt ( ` Tc  =  ` ,  Tc) ;

#pl ot ( [ ' uT( x) ' , x, 0, 1, Tc[ 1] , Tc[ 2] , Tc[ 3] ] , x=0. . 1, t hi ckness=[ 2, 1, 1,
1, 1, 1, 1] ) ;
pl ot ( [ ' 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] ) ;
```

$$N := 4$$

$$\beta_4 := 1.3846153846$$

$$alpha =, \left[ \begin{array}{cccc} 0.5000000000 & 1 & 0.8000000000 & 0.3000000000 \end{array} \right]$$

$$beta =, \left[ \begin{array}{cccc} 2 & 3 & 4 & 1.3846153846 \end{array} \right]$$

$$gamma =, \left[ \begin{array}{cccc} 0.5000000000 & 0.0000000000 & 0.2000000000 & 0.7000000000 \end{array} \right]$$

$$\beta const := 2.6000000000$$

$$b =, \left[ \begin{array}{ccccc} 0 & 0.2500000000 & 0.5833333333 & 0.7833333333 & 1 \end{array} \right]$$

$$ag =, \left[ \begin{array}{cccc} 0 & 0.7500000000 & 2.3333333333 & 1.0846153846 \end{array} \right]$$

$$al =, \left[ \begin{array}{cccc} -0.5000000000 & 0.7500000000 & 2.1333333333 & 0.3846153846 \end{array} \right]$$

$$a =, \left[ \begin{array}{cccc} -0.5000000000 & 0.7500000000 & 2.1333333333 & 0.3846153846 \end{array} \right]$$

$$gamma =, \left[ \begin{array}{cccc} 0.5000000000 & 0.0000000000 & 0.2000000000 & 0.7000000000 \end{array} \right]$$

$$Kc =, 3$$

$$c =, \left[ \begin{array}{ccccccccc} 0 & 0.5833333333 & 0.7833333333 & c_4 & c_5 & c_6 & c_7 & c_8 \end{array} \right]$$

$$sidec =, \left[ \begin{array}{ccccccccc} 1 & 1 & 1 & sidec_4 & sidec_5 & sidec_6 & sidec_7 & sidec_8 \end{array} \right]$$

$$j\_of\_c =, \left[ \begin{array}{ccccccccc} 1 & 3 & 4 & j\_of\_c_4 & j\_of\_c_5 & j\_of\_c_6 & j\_of\_c_7 & j\_of\_c_8 \end{array} \right]$$

$$uint\_of\_x := x \rightarrow piecewise\left( x < b_2, 1, x < b_3, 2, x < b_4, 3, x < b_5, 4, x < b_6, 5, x < b_7, 6, x \right.$$

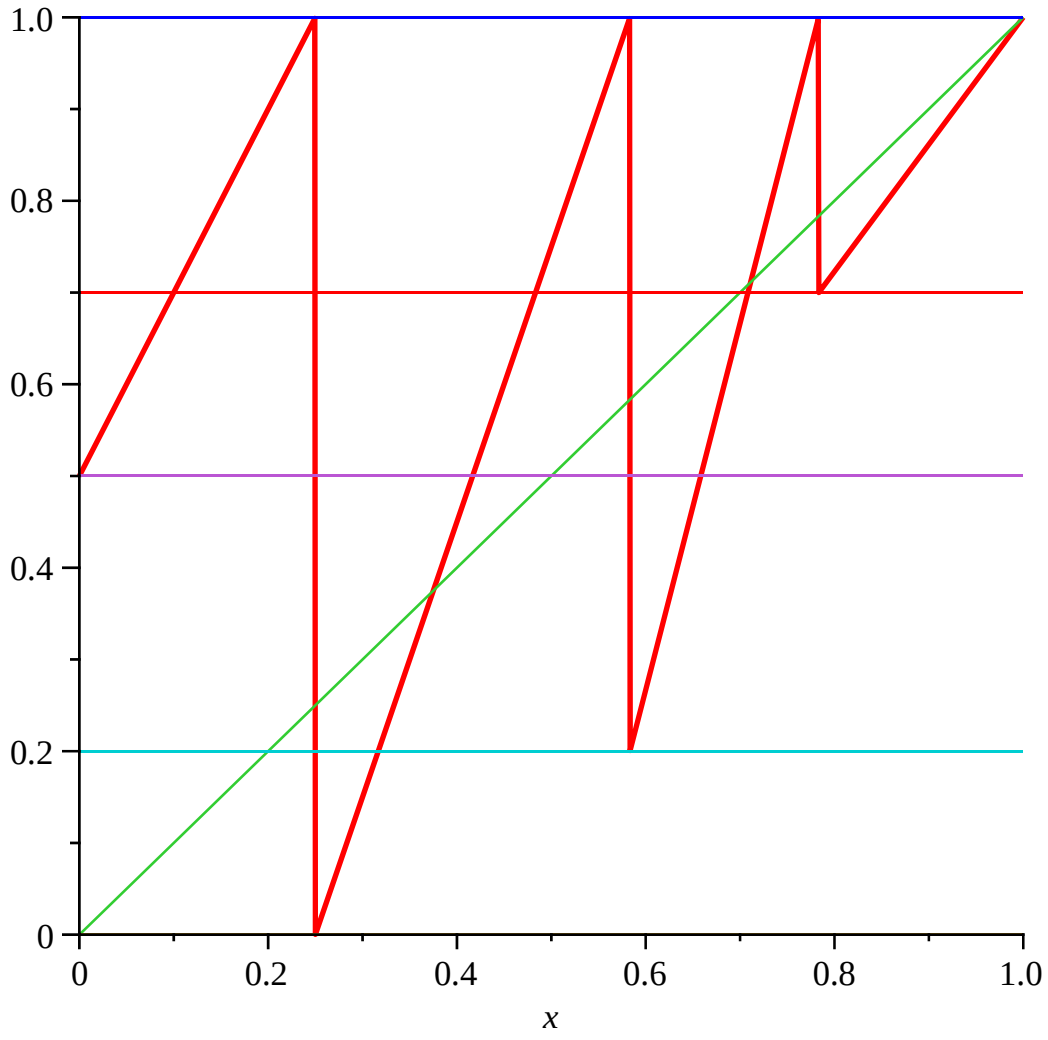
$< b_8, 7, x < b_9, 8, 9)$

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

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

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

$$Tc = , \left[ \begin{array}{ccccc} 0.5000000000 & 0.2000000000 & 0.7000000000 & Tc_4 & Tc_5 \end{array} \right]$$




```
> ### Conjugated greedy
  tbb:=vector(N+1,[]):#for printing only = b[]
  tbeta:=vector(N,[]):

  talpha:=vector(N,[]):
  tgamm:=vector(N,[]):#heiths of lower ends of hanging branches
                      # alpha[i]+gamma[i]<1   !!!!!!!
                      #if gamma[i]>0
  for i from 1 to N do
  tbeta[i]:=beta[N-i+1];
  talpha[i]:=alpha[N-i+1];
  tgamm[i]:=0;
  od:
```

```
print(`talpha =`,talpha);
print(`tbeta =`,tbeta);
print(`tgamma =`,tgamm);
tb[1]:=0:
for j from 1 to N do
tb[j+1]:=tb[j]+talpha[j]/tbeta[j]:
 od: i:='i':
#tb[N+1]:=1:
tag:=vector(N,[]):
tal:=vector(N,[]):
ta:=vector(N,[]):
tc:=vector(N,[]):
for j from 1 to N do
tbb[j]:=tb[j];
tag[j]:=tbeta[j]*tb[j];
tal[j]:=-1+tbeta[j]*tb[j+1];
od:
tbb[N+1]:=1:
for j from 1 to N do
ta[j]:=tag[j]-tgamm[j];
 od:
print(`b =`,bb);
print(`ag =`,ag);
print(`al =`,al);
print(`a =`,a);
print(`gamma =`,gamm);
print(`tb =`,tbb);
for i from 1 to N+1 do
print(1-b[N-i+2]);
od:
print(`tag =`,tag);
print(`tal =`,tal);
print(`ta =`,ta);
for i from 1 to N do
print(beta[N-i+1]-1-a[N-i+1]);
od:
print(`tgamma =`,tgamm);
tuint_of_x:=x->piecewise(x<tb[2],1,# This function needs
additions by hand for
                                    # N>9 . Automathic procedure
causes plotting problems
                                    # but is used in other
programs
                          x<tb[3],2,
                          x<tb[4],3,
                          x<tb[5],4,
                          x<tb[6],5,
```

```
                                x<tb[7],6,
                                x<tb[8],7,
                                x<tb[9],8,
                                      9);
tint_of_x:=x->piecewise(x<=tb[2],1,# This function needs
additions by hand for
                                              # N>9 . Automathic procedure
causes plotting problems
                                              # but is used in other
programs
                                x<=tb[3],2,
                                x<=tb[4],3,
                                x<=tb[5],4,
                                x<=tb[6],5,
                                x<=tb[7],6,
                                x<=tb[8],7,
                                x<=tb[9],8,
                                      9);
x:='x':
tuT:=x->tbeta[tuint_of_x(x)]*x-ta[tuint_of_x(x)];
tT:=x->tbeta[tint_of_x(x)]*x-ta[tint_of_x(x)];
##########
tKc:=0:# new number of c points
for j from 1 to N do if talpha[j]<1 then tKc:=tKc+1 fi od:
for j from 1 to N do if (tgamm[j]>0 and talpha[j]+tgamm[j]<1)
then tKc:=tKc+1 fi od:
print(`tKc =`,tKc);
tc:=vector(2*N,[]):
tsidec:=vector(2*N,[]):# 1 left (use uT), 0 right (use T)
tj_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 (talpha[j]<1 and tgamm[j]=0) then   tc[cj]:=tb[j+1]; tsidec
[cj]:=0;tj_of_c[cj]:=j;cj:=cj+1 fi;
if (talpha[j]<1 and tgamm[j]+talpha[j]=1) then  tc[cj]:=tb[j];
tsidec[cj]:=1;tj_of_c[cj]:=j;cj:=cj+1 fi;
if (tgamm[j]>0 and tgamm[j]+talpha[j]<1) then  tc[cj]:=tb[j];
tsidec[cj]:=1;tj_of_c[cj]:=j;cj:=cj+1 ;
                        tc[cj]:=tb[j+1]; tsidec[cj]:=0;tj_of_c[cj]:=
j;cj:=cj+1 fi;
od:
print(`tc =`,tc);
print(`tsidec =`,tsidec);
print(`tj_of_c =`,tj_of_c);
print(`b =`,bb);print(`tb =`,tbb);
```

```
pl ot ( [ ' 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] ) ;
pl ot ( [ ' 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] ) ;
```

$$talpha =, \left[\begin{array}{cccc} 0.3000000000 & 0.8000000000 & 1 & 0.5000000000 \end{array}\right]$$

$$tbeta =, \left[\begin{array}{cccc} 1.3846153846 & 4 & 3 & 2 \end{array}\right]$$

$$tgamma =, \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \end{array}\right]$$

$$b =, \left[\begin{array}{ccccc} 0 & 0.2500000000 & 0.5833333333 & 0.7833333333 & 1 \end{array}\right]$$

$$ag =, \left[\begin{array}{cccc} 0 & 0.7500000000 & 2.3333333333 & 1.0846153846 \end{array}\right]$$

$$al =, \left[\begin{array}{cccc} -0.5000000000 & 0.7500000000 & 2.1333333333 & 0.3846153846 \end{array}\right]$$

$$a =, \left[\begin{array}{cccc} -0.5000000000 & 0.7500000000 & 2.1333333333 & 0.3846153846 \end{array}\right]$$

$$gamma =, \left[\begin{array}{cccc} 0.5000000000 & 0.0000000000 & 0.2000000000 & 0.7000000000 \end{array}\right]$$

$$tb =, \left[\begin{array}{ccccc} 0 & 0.2166666667 & 0.4166666667 & 0.7500000000 & 1 \end{array}\right]$$

$$\begin{array}{c} 0 \\ 0.2166666667 \\ 0.4166666667 \\ 0.7500000000 \\ 1 \end{array}$$

$$tag =, \left[\begin{array}{cccc} 0.0000000000 & 0.8666666667 & 1.2500000000 & 1.5000000000 \end{array}\right]$$

$$tal =, \left[\begin{array}{cccc} -0.7000000000 & 0.6666666667 & 1.2500000000 & 1.0000000000 \end{array}\right]$$

$$ta =, \left[\begin{array}{cccc} 0.0000000000 & 0.8666666667 & 1.2500000000 & 1.5000000000 \end{array}\right]$$

$$\begin{array}{c} 0.0000000000 \\ 0.8666666667 \\ 1.2500000000 \\ 1.5000000000 \end{array}$$

$$tgamma =, \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \end{array}\right]$$

$$tuint\_of\_x := x \rightarrow piecewise\left(x < tb_2, 1, x < tb_3, 2, x < tb_4, 3, x < tb_5, 4, x < tb_6, 5, x < tb_7, 6, x < tb_8, 7, x < tb_9, 8, 9\right)$$

$$tint\_of\_x := x \rightarrow piecewise\left(x \leq tb_2, 1, x \leq tb_3, 2, x \leq tb_4, 3, x \leq tb_5, 4, x \leq tb_6, 5, x \leq tb_7, 6, x\right.$$

$$\leq tb_8, 7, x \leq tb_9, 8, 9\Big)$$

$$tuT := x \rightarrow tbeta_{tuint\_of\_x(x)}\ x - ta_{tuint\_of\_x(x)}$$

$$tT := x \rightarrow tbeta_{tint\_of\_x(x)}\ x - ta_{tint\_of\_x(x)}$$

$$tKc =, 3$$

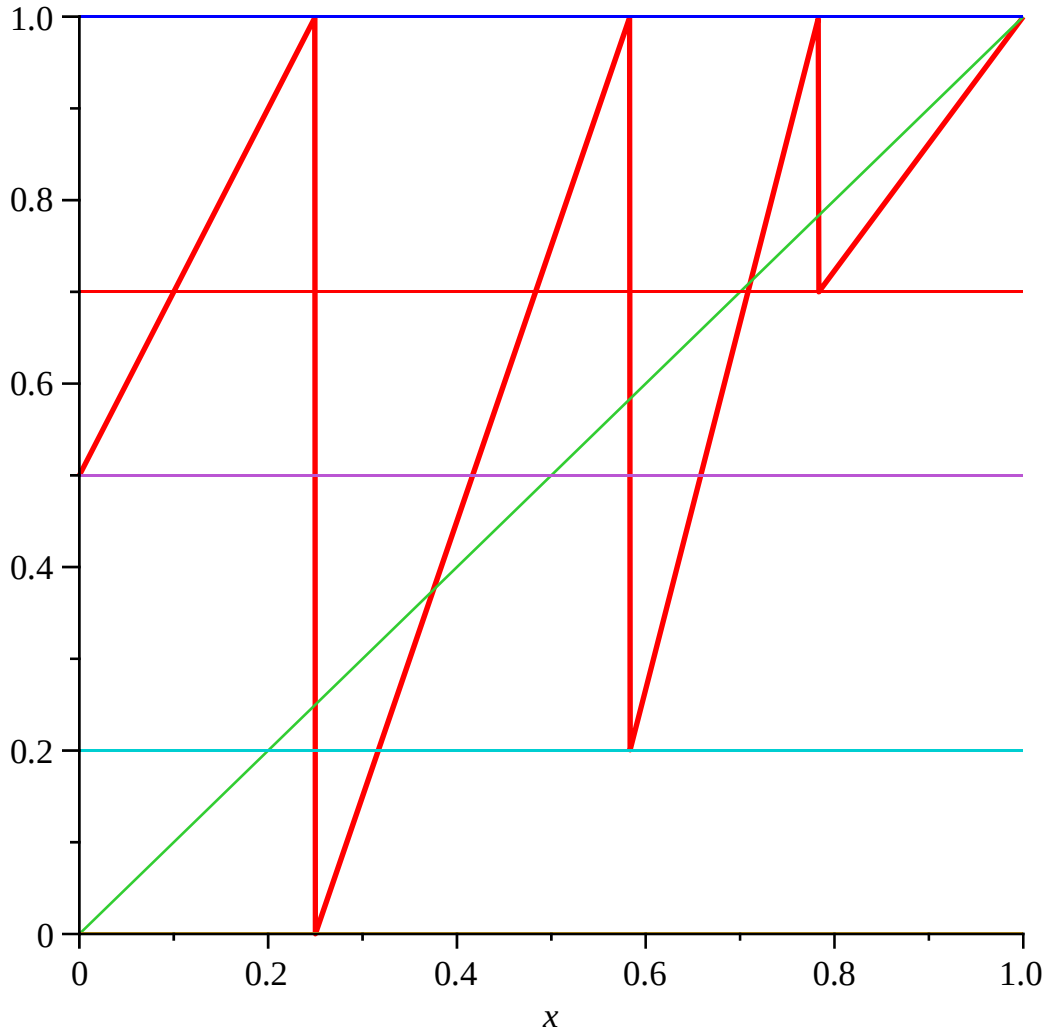
$$tc =, \left[\begin{array}{cccccccc} 0.2166666667 & 0.4166666667 & 1.0000000000 & tc_4 & tc_5 & tc_6 & tc_7 & tc_8 \end{array}\right]$$

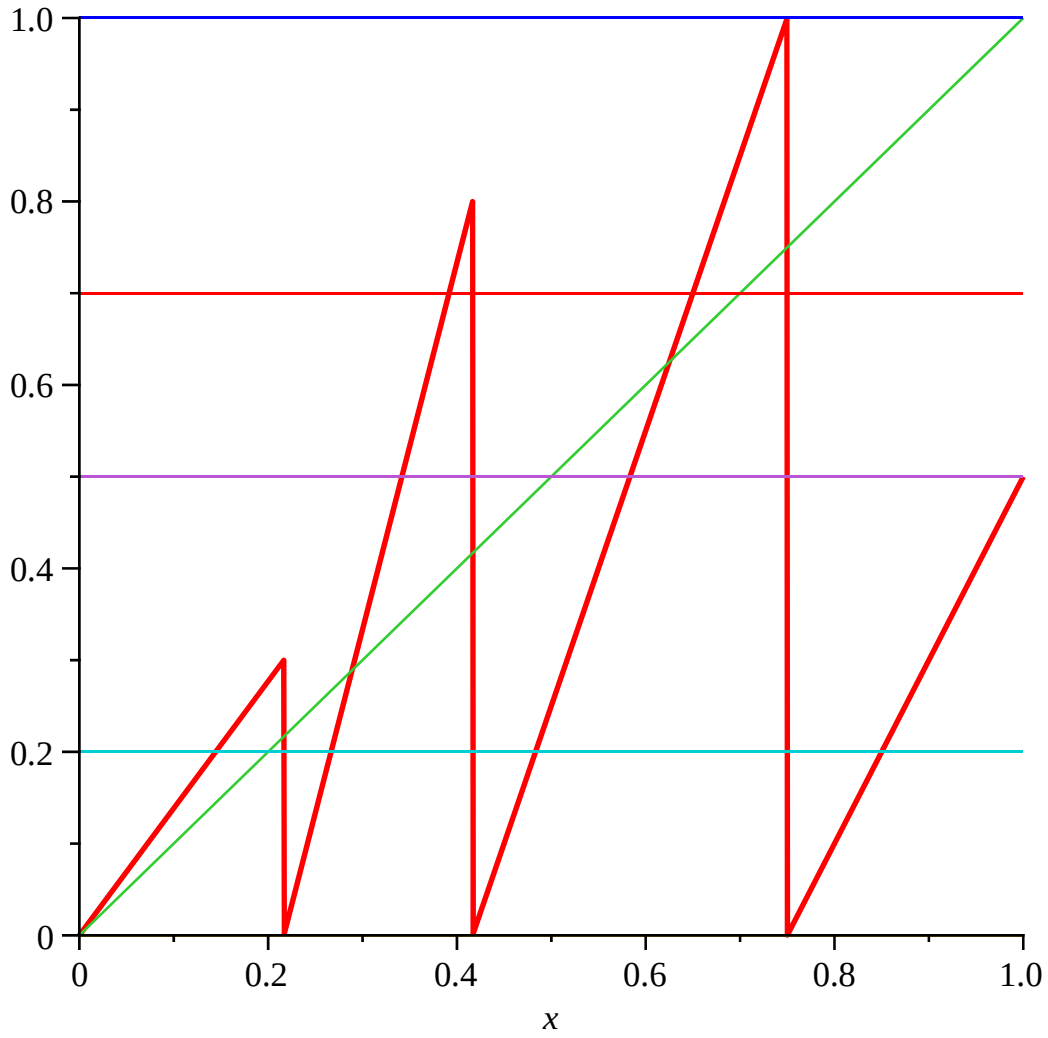
$$tsidec =, \left[\begin{array}{cccccccc} 0 & 0 & 0 & tsidec_4 & tsidec_5 & tsidec_6 & tsidec_7 & tsidec_8 \end{array}\right]$$

$$tj\_of\_c =, \left[\begin{array}{cccccccc} 1 & 2 & 4 & tj\_of\_c_4 & tj\_of\_c_5 & tj\_of\_c_6 & tj\_of\_c_7 & tj\_of\_c_8 \end{array}\right]$$

$$b =, \left[\begin{array}{ccccc} 0 & 0.2500000000 & 0.5833333333 & 0.7833333333 & 1 \end{array}\right]$$

$$tb =, \left[\begin{array}{ccccc} 0 & 0.2166666667 & 0.4166666667 & 0.7500000000 & 1 \end{array}\right]$$






```
>
> 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[uint_of_x(xxt)];
  d[i]:=a[uint_of_x(xxt)];
  db[i]:=a[uint_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;
```

$$Digits := 100$$

$$NN := 50$$

$$xx := 0.3957188605$$

$$[0.7500000000, 0.7500000000, 0.7500000000, 0.3846153846, 0.3846153846, 0.3846153846, 0.3846153846, 2.1333333333, 0.3846153846, 0.3846153846, 0.3846153846, 2.1333333333, 0.3846153846, 2.1333333333, 0.3846153846, 0.3846153846, 0.3846153846, 0.3846153846, 2.1333333333, 2.1333333333, 0.3846153846, 0.3846153846, 2.1333333333, 0.3846153846, 0.3846153846, 2.1333333333, 0.3846153846, 0.3846153846, 0.3846153846, 2.1333333333, 2.1333333333, 0.3846153846, 0.3846153846, 0.3846153846, 2.1333333333, 0.3846153846, 0.3846153846, 2.1333333333, 0.3846153846, 0.3846153846, 0.3846153846, 0.3846153846, 0.3846153846, 2.1333333333, 0.3846153846, 0.3846153846, 0.3846153846, 0.3846153846]$$

$$uIs\_it\_x := 0.3957188605$$

$$[0.7500000000, 0.7500000000, 0.7500000000, 0.3846153846, 0.3846153846, 0.3846153846, 0.3846153846, 2.1333333333, 0.3846153846, 0.3846153846, 0.3846153846, 2.1333333333, 0.3846153846, 2.1333333333, 0.3846153846, 0.3846153846, 0.3846153846, 0.3846153846, 2.1333333333, 2.1333333333, 0.3846153846, 0.3846153846, 2.1333333333, 0.3846153846, 0.3846153846, 2.1333333333, 0.3846153846, 0.3846153846, 0.3846153846, 2.1333333333, 2.1333333333, 0.3846153846, 0.3846153846, 0.3846153846, 2.1333333333, 0.3846153846, 0.3846153846, 2.1333333333, 0.3846153846, 0.3846153846, 0.3846153846, 0.3846153846, 0.3846153846, 2.1333333333, 0.3846153846, 0.3846153846, 0.3846153846, 0.3846153846]$$

$$Is\_it\_x := 0.3957188605$$

$$terr := 7.133396421\ 10^{-15}$$

$$err := 7.133396421\ 10^{-15} \qquad (1)$$

```
> ud:=vector(50):Digits:=100;NN:=50;#Expansion with variable
  slopes
  d:=vector(50):                          #using conjugated greedy map
  xx:=evalf(rand()/10^12);
  xxt:=xx:
  bet:=1:
  for i from 1 to NN do
  bet:=bet/tbeta[tuint_of_x(xxt)];
  ud[i]:=ta[tuint_of_x(xxt)];
```

```
    udb[i]:=ta[tuint_of_x(xxt)]*bet;
    xxt:=tuT(xxt);
    od:

    xxt:=xx:
    bet:=1:
    for i from 1 to NN do
    bet:=bet/tbeta[tuint_of_x(xxt)];
    d[i]:=ta[tuint_of_x(xxt)];
    db[i]:=ta[tuint_of_x(xxt)]*bet;
    xxt:=tT(xxt);
    od:
    print(ud);
    tuIs_it_x:=evalf(sum(udb[j1],j1=1..NN));
    print(d);
    tIs_it_x:=evalf(sum(db[j1],j1=1..NN));
    terr:=xx-tuIs_it_x;
    err:=xx-tIs_it_x;
>
```

$$Digits := 100$$

$$NN := 50$$

$$xx := 0.1931398164$$

[0.0000000000, 0.8666666667, 0.0000000000, 0.8666666667, 0.8666666667, 0.0000000000,
0.8666666667, 0.0000000000, 0.0000000000, 0.0000000000, 0.0000000000,
0.0000000000, 0.8666666667, 0.0000000000, 0.0000000000, 0.0000000000,
0.0000000000, 0.0000000000, 0.8666666667, 0.0000000000, 0.0000000000,
0.0000000000, 0.0000000000, 0.8666666667, 0.8666666667, 0.0000000000,
0.0000000000, 0.8666666667, 0.8666666667, 0.0000000000, 0.0000000000,
0.8666666667, 0.0000000000, 0.8666666667, 0.0000000000, 0.0000000000,
0.0000000000, 0.0000000000, 0.0000000000, 0.8666666667, 0.8666666667,
0.8666666667, 0.8666666667, 0.8666666667, 1.2500000000, 0.0000000000,
0.0000000000, 0.0000000000, 0.8666666667, 0.8666666667]

$$tuIs\_it\_x := 0.1931398164$$

[0.0000000000, 0.8666666667, 0.0000000000, 0.8666666667, 0.8666666667, 0.0000000000,
0.8666666667, 0.0000000000, 0.0000000000, 0.0000000000, 0.0000000000,
0.0000000000, 0.8666666667, 0.0000000000, 0.0000000000, 0.0000000000,
0.0000000000, 0.0000000000, 0.8666666667, 0.0000000000, 0.0000000000,
0.0000000000, 0.0000000000, 0.8666666667, 0.8666666667, 0.0000000000,
0.0000000000, 0.8666666667, 0.8666666667, 0.0000000000, 0.0000000000,
0.8666666667, 0.0000000000, 0.8666666667, 0.0000000000, 0.0000000000,
0.0000000000, 0.0000000000, 0.0000000000, 0.8666666667, 0.8666666667,
0.8666666667, 0.8666666667, 0.8666666667, 1.2500000000, 0.0000000000,
0.0000000000, 0.0000000000, 0.8666666667, 0.8666666667]

$$tIs\_it\_x := 0.1931398164$$

$$terr := 1.790009610\ 10^{-17}$$

$$err := 1.790009610\ 10^{-17} \quad (2)$$

```
> 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 LAZY MAP

  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] 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] 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 xxt:=uT(xxt) else xxt:=T(xxt)fi;
       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));
```

```
#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:
#for n from 1 to 20 do

#print(1,2,cc[1,2,n]*beta[1],valc[1,n]);
#print(3,2,cc[3,2,n]*beta[2],valc[3,n]);
#od:
```

$$NN := 50$$

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

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

$$xxt := 0$$

$$bet := 1$$

$$Is\_it\_x := -3.203707712\ 10^{-20}$$

$$xxt := 0.5833333333$$

$$bet := 1$$

$$Is\_it\_x := 0.5833333333$$

$$xxt := 0.7833333333$$

$$bet := 1$$

$$Is\_it\_x := 0.7833333333$$

$$S_1 := 0.6249261956$$

$$S_2 := 0.3680269052$$

$$S_3 := 0.7034143519 \qquad (3)$$

```
> #Expansion of c1, c2 ... and all the S's GREEDY MAP

for i from 1 to tKc do
xxt:=tc[i];
bet:=1:
      for n from 1 to NN+1 do
       if tsidec[i]=1 then intx:=tuint_of_x(xxt) else intx:=
tint_of_x(xxt) fi;
          bet_real:=bet;
          bet:=bet/tbeta[intx];
          dcb[i,n]:=ta[intx]*bet;

         if tsidec[i]=0 then
              for ii from 1 to Kc do
```

```
                    if xxt>tc[ii] 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/(tbeta[j7]),j7=1..
intx-1)*bet_real fi;
                #bc[i,n]:=tb[intx]*bet_real;
                ################
                       else
                for ii from 1 to Kc do
                    if xxt<tc[ii] 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/(tbeta[j8]),j8=
intx+1..N)*bet_real fi;
                #bc[i,n]:=tb[intx+1]*bet_real;
        fi;
        tvalc[i,n]:=xxt;
        tbetc[i,n]:=bet_real;
        if tsidec[i]=1 then xxt:=tuT(xxt) else xxt:=tT(xxt)fi;
        od:
tIs_it_x:=sum(dcb[i,j1],j1=1..NN);
od;
for i from 1 to tKc do
tS[i]:=evalf(sum(Sc[i,j2+1],j2=1..NN));
#sum_b[i]:=evalf(sum(bc[i,j2+1],j2=1..NN));
od;
for i from 1 to tKc do
for j from 1 to tKc do
tSS[i,j]:=evalf(sum(cc[i,j,j1+1],j1=1..NN));

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

#print(1,2,cc[1,2,n]*beta[1],valc[1,n]);
#print(3,2,cc[3,2,n]*beta[2],valc[3,n]);
#od:
```

$$xxt := 0.2166666667$$

$$bet := 1$$

$$tIs\_it\_x := 0.2166666667$$

$$xxt := 0.4166666667$$

$$bet := 1$$

$$tIs\_it\_x := 0.4166666667$$

$$xxt := 1.0000000000$$

$$bet := 1$$

$$tIs\_it\_x := 1.0000000000$$

$$tS_1 := 0.7034143519$$

$$tS_2 := 0.3680269052$$

$$tS_3 := 0.6249261956 \quad \textbf{(4)}$$

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

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



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;

MM[i,i]:=MM[i,i]+1.0;
od:

print(MM);
print(`det MM = `,det(MM));
print(ve);



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

$$MM = , \begin{bmatrix} -0.0000000000 & -0.0000000000 & -0.0000000000 \\ -0.5001771304 & -0.2500239864 & -0.0451388889 \\ -0.6921441872 & -0.3076332644 & -0.9583333333 \end{bmatrix}$$

$$eigenvalues\ MM =, -0.9774235613, -0.2309337584, -0.0000000000$$

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

$$\begin{bmatrix} 1.0000000000 & -0.0000000000 & -0.0000000000 \\ -0.5001771304 & 0.7499760136 & -0.0451388889 \\ -0.6921441872 & -0.3076332644 & 0.0416666667 \end{bmatrix}$$

$$det\ MM = , 0.0173627768$$

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

$$DD := \begin{bmatrix} 1.0000000000 & 7.9992325099 & 99.6713807157 \end{bmatrix}$$

$$-2.475363054\ 10^{-14} \qquad \textbf{(5)}$$

```
> tMM:=matrix(tKc,tKc,[]):#GREEDY MAP
  for i from 1 to Kc do
  for j from 1 to Kc do

  tMM[i,j]:=-tSS[j,i];
  od;od;
  print(`tMM = `,tMM);



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

  tMM[i,i]:=tMM[i,i]+1.0;
  od:

  print(tMM);
  print(`det tMM = `,det(tMM));
  print(ve);



  tDD:=linsolve(tMM,ve);
  sum((tS[ii7]-1/tbeta[tj_of_c[ii7]])*tDD[ii7],ii7=1..tKc)-(1-sum
  (1/tbeta[i8],i8=1..N));
```

$$tMM = , \begin{bmatrix} -0.9583333333 & -0.3076332644 & -0.6921441872 \\ -0.0451388889 & -0.2500239864 & -0.5001771304 \\ -0.0000000000 & -0.0000000000 & -0.0000000000 \end{bmatrix}$$

$$eigenvalues\ tMM =, -0.9774235613, -0.2309337584, -0.0000000000$$

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

$$\begin{bmatrix} 0.0416666667 & -0.3076332644 & -0.6921441872 \\ -0.0451388889 & 0.7499760136 & -0.5001771304 \\ -0.0000000000 & -0.0000000000 & 1.0000000000 \end{bmatrix}$$

$$det\ tMM = , 0.0173627768$$

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

$$tDD := \begin{bmatrix} 99.6713807157 & 7.9992325099 & 1.0000000000 \end{bmatrix}$$

$$-2.475363054\ 10^{-14} \qquad \textbf{(6)}$$

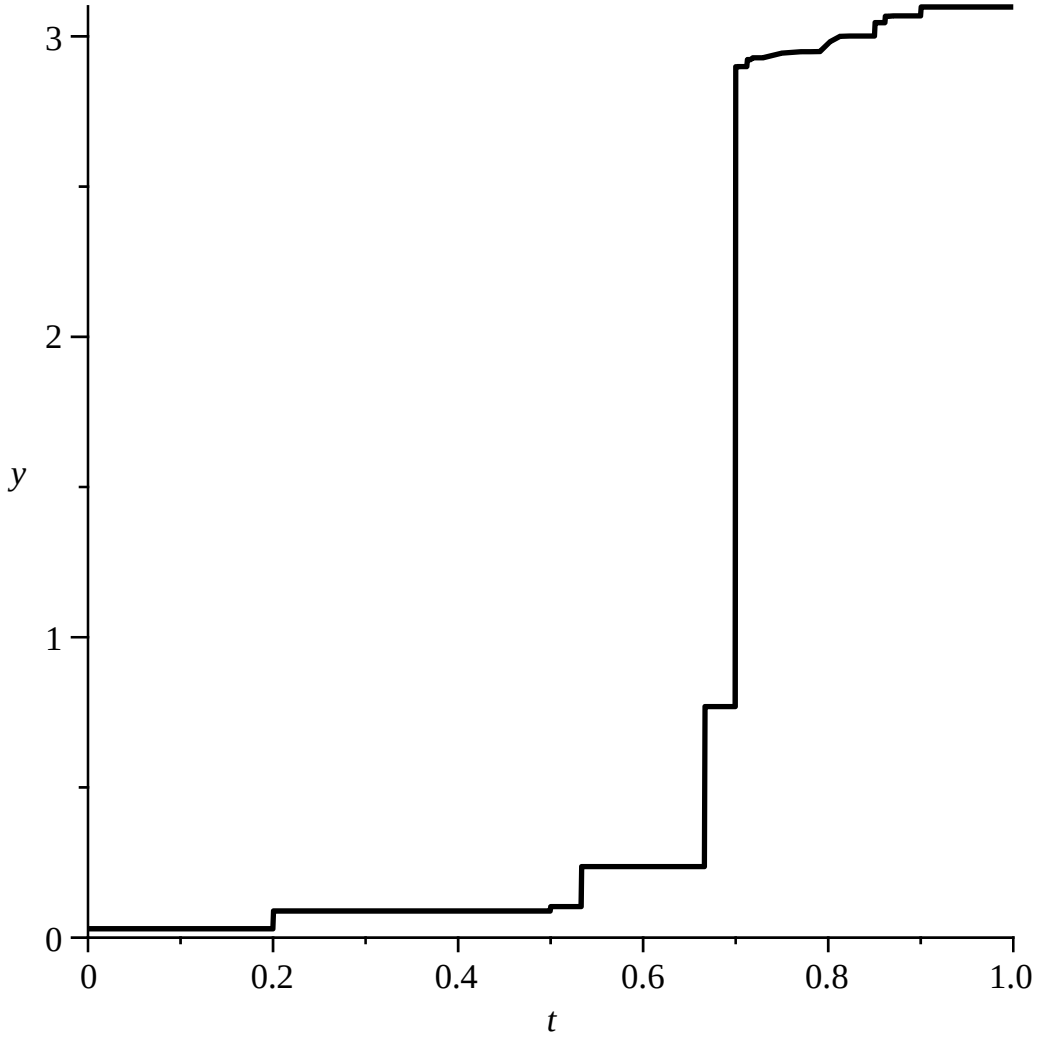
```
>
> density:=proc(t) local j,i, den;# LAZY MAP
                    i:='i':

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

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


                    od;
                   return den;
            end proc;
    tdensity:=proc(t) local j,i, den;    ### GREEDY  MAP
                    i:='i':

                   den:=1:
                   for j from 1 to Kc do
                   if tsidec[j]=0 then
                     den:=den+ tDD[j]*sum((chi(0,tvalc[j,i1+1],t))*
tbetc[j,i1+1],i1=1..50) fi;

                    if tsidec[j]=1 then
                     den:=den+ tDD[j]*sum((uchi(tvalc[j,i1+1],1,t))*
tbetc[j,i1+1],i1=1..50) fi;


                    od;
                   return den;
            end proc;
#Normalizing factor
```

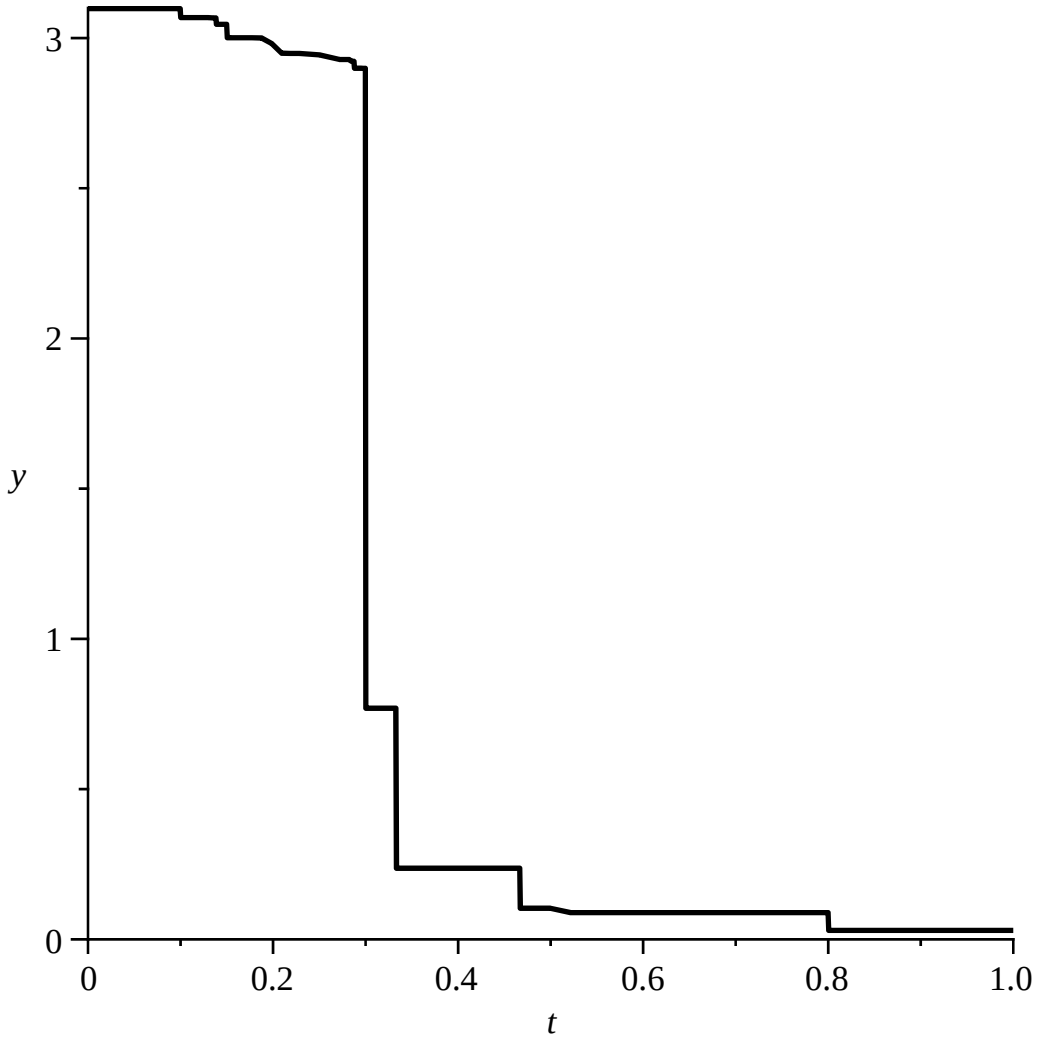
```
NC:=1:
for j from 1 to Kc do
if sidec[j]=0 then
 NC:=NC+DD[j]*sum((valc[j,i1+1])*betc[j,i1+1],i1=1..50) fi;
if sidec[j]=1 then
NC:=NC+DD[j]*sum((1-valc[j,i1+1])*betc[j,i1+1],i1=1..50) fi;

 od:

print(`NC = `,NC);

plot([(1/NC)*density(t)],t=0..1-0.000001,y=0..3.1,color=black,
thickness=2);
plot([(1/NC)*tdensity(t)],t=0..1-0.000001,y=0..3.1,color=black,
thickness=2);
```

```
density := proc(t)
    local j, i, den;
    i := 'i';
    den := 1;
    for j to Kc do
        if sidec[j] = 0 then
            den := den + DD[j] * (sum(chi(0, valc[j, i1 + 1], t) * betc[j, i1 + 1], i1 = 1 ..50))
        end if;
        if sidec[j] = 1 then
            den := den + DD[j] * (sum(uchi(valc[j, i1 + 1], 1, t) * betc[j, i1 + 1], i1 = 1 ..50))
        end if
    end do;
    return den
end proc
tdensity := proc(t)
    local j, i, den;
    i := 'i';
    den := 1;
    for j to Kc do
        if tsidec[j] = 0 then
            den := den + tDD[j] * (sum(chi(0, tvalc[j, i1 + 1], t) * tbetc[j, i1 + 1], i1 = 1
            ..50))
        end if;
        if tsidec[j] = 1 then
            den := den + tDD[j] * (sum(uchi(tvalc[j, i1 + 1], 1, t) * tbetc[j, i1 + 1], i1 = 1
            ..50))
        end if
```

```
    end do;
    return den
end proc
```

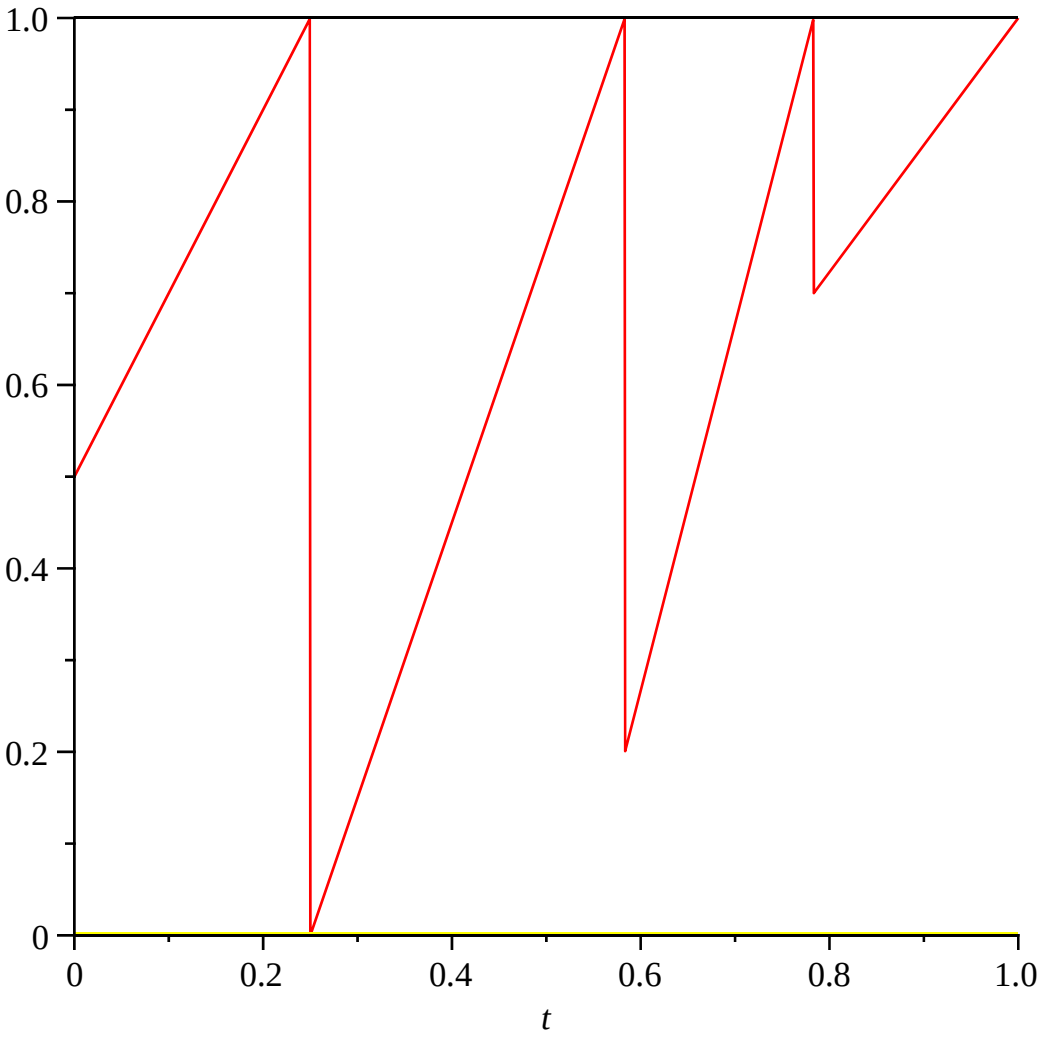
$NC = , 33.7995685749$






```
> 
> 
#check density   LAZY
#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;
```

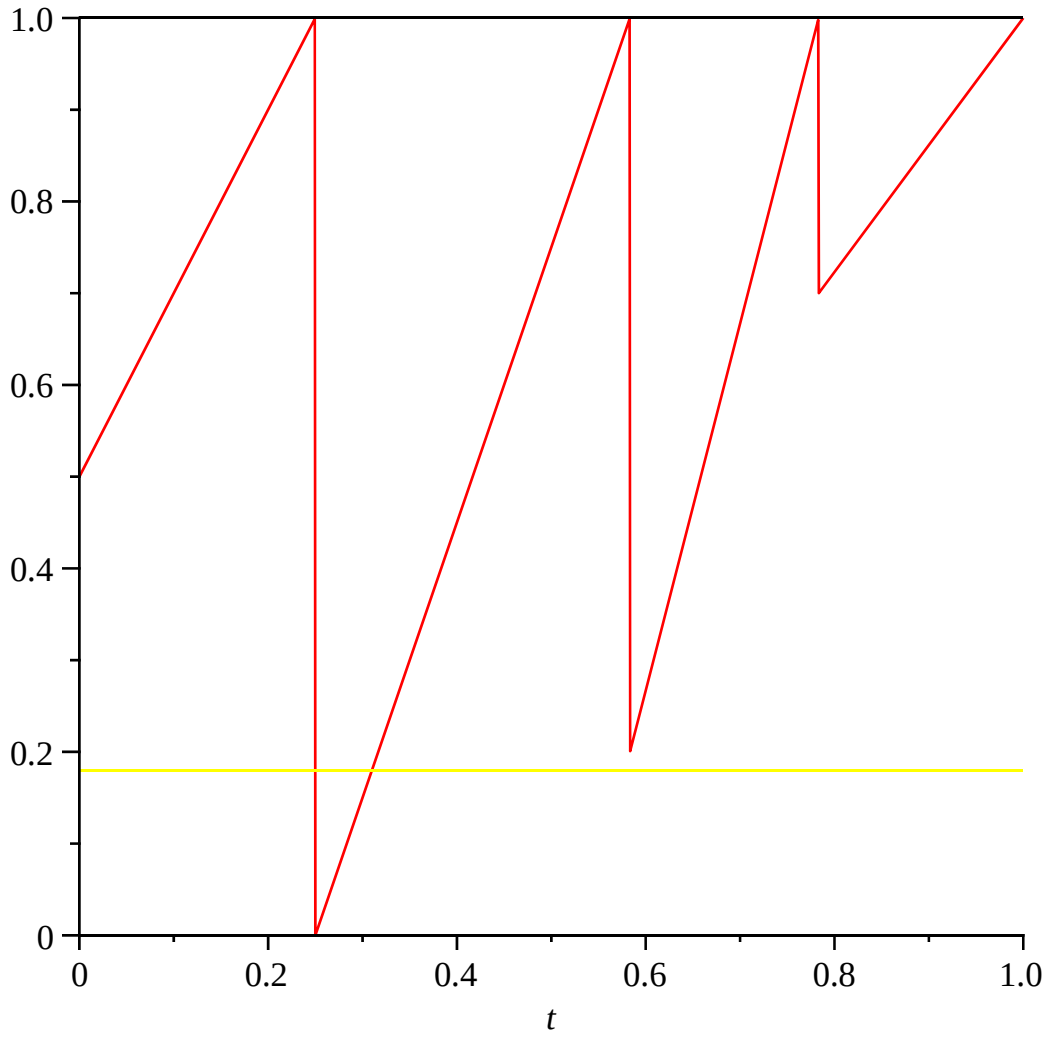
```
for j6 from 0 to 9 do
print(`y =`,y[j6]);
print(`err[`,j6,`]=`,err[j6]);
od;
#check density  GREEDY
#preimages
for j6 from 0 to 9 do
y[j6]:=j6/10+(0.1)*rand()/10^12;
od:
for j6 from 0 to 9 do
for i3 from 1 to N do
pre[i3]:=(y[j6]+ta[i3])/tbeta[i3];
od;
plot([tT(t),0,1,y[j6]],t=0..1,
color=[red,black,black,yellow]);
su:=0:
for i3 from 1 to N do
if (pre[i3]>=tb[i3] and pre[i3]<=tb[i3+1]) then
 su:=su+evalf(tdensity(pre[i3])/tbeta[i3]);
print(i3);
fi;
od;
err[j6]:=evalf(tdensity(y[j6])-su);
od;

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




$su := 0$

2

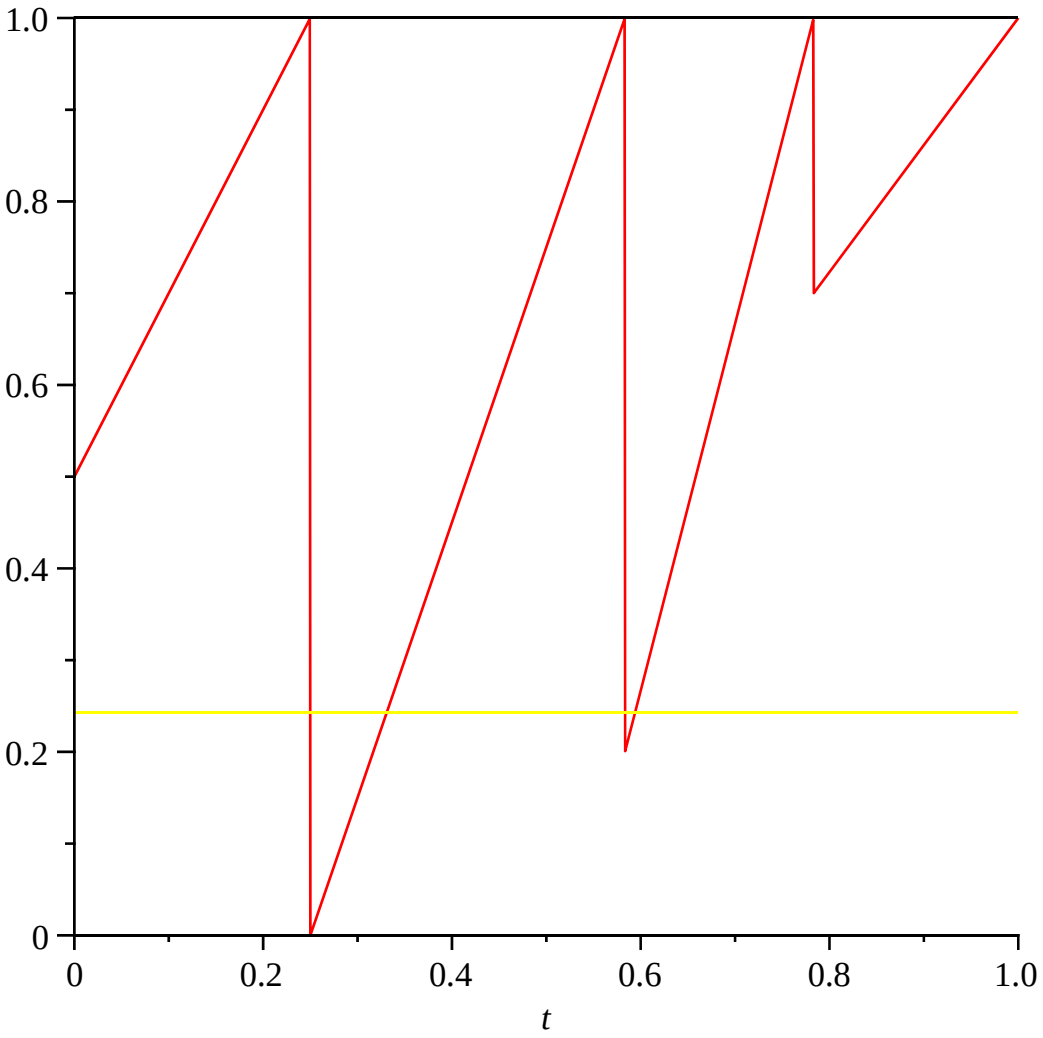
$err_0 := 2.475363054\ 10^{-14}$




$su := 0$

2

$err_1 := 2.475363054\ 10^{-14}$

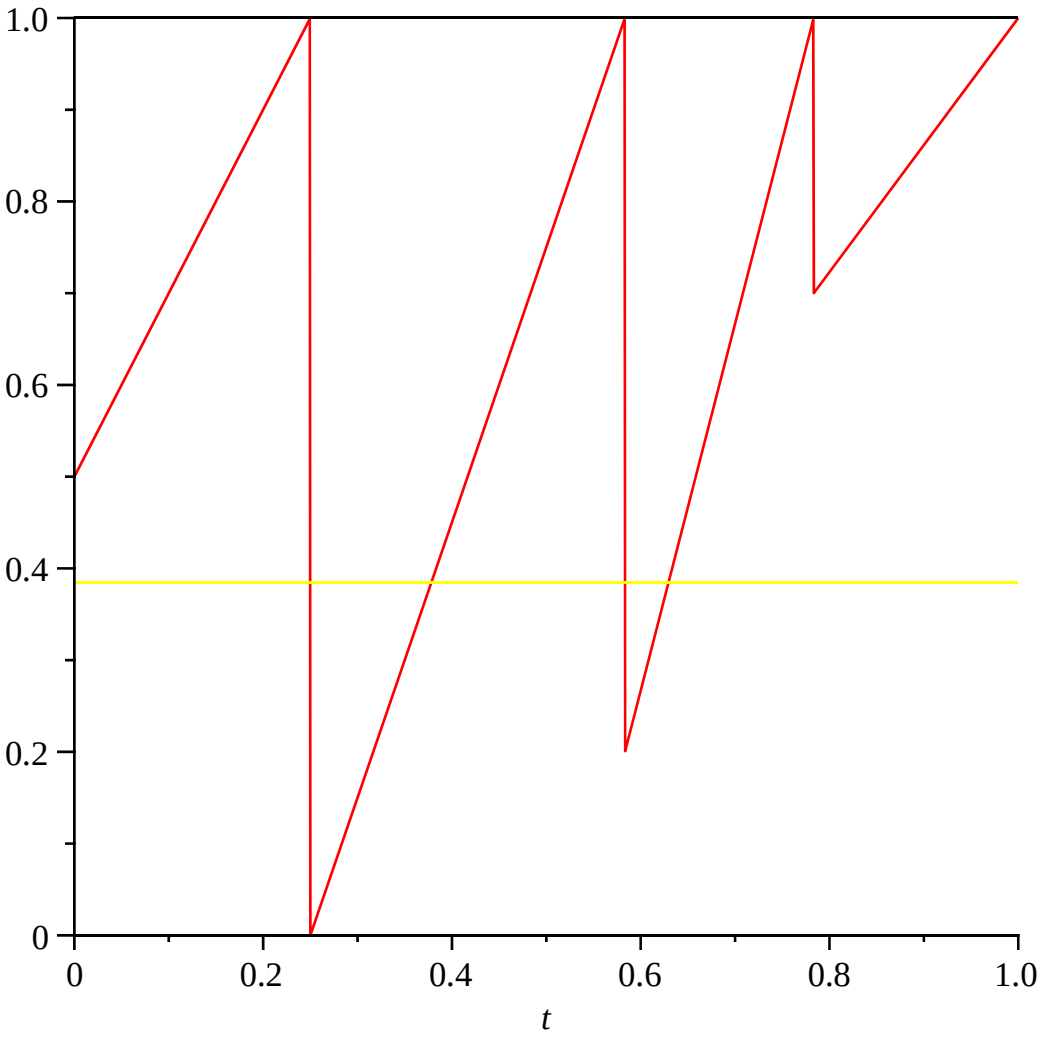



$su := 0$

2

3

$err_2 := 2.475363054\ 10^{-14}$

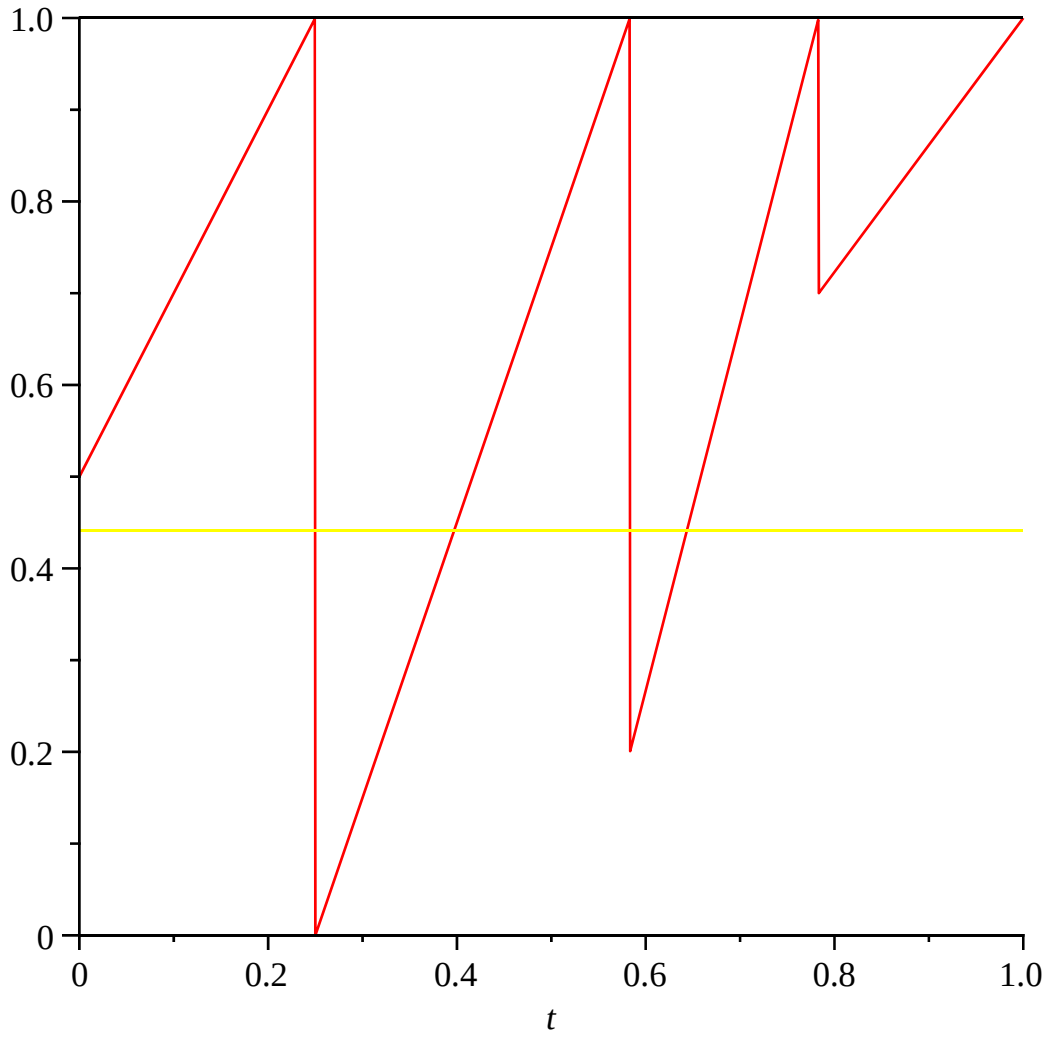



$su := 0$

2

3

$err_3 := 2.475363054\ 10^{-14}$

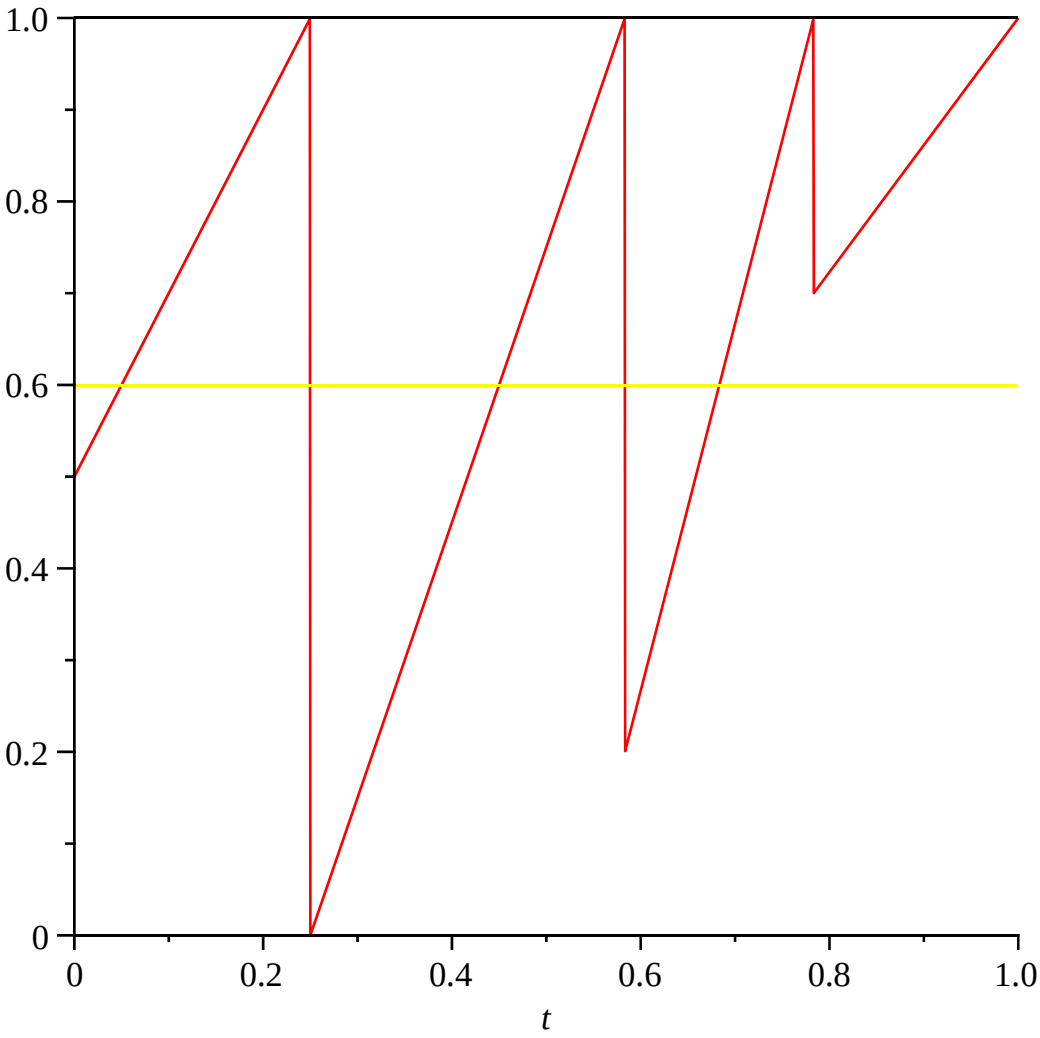



$su := 0$

2

3

$err_4 := 2.475363054\ 10^{-14}$



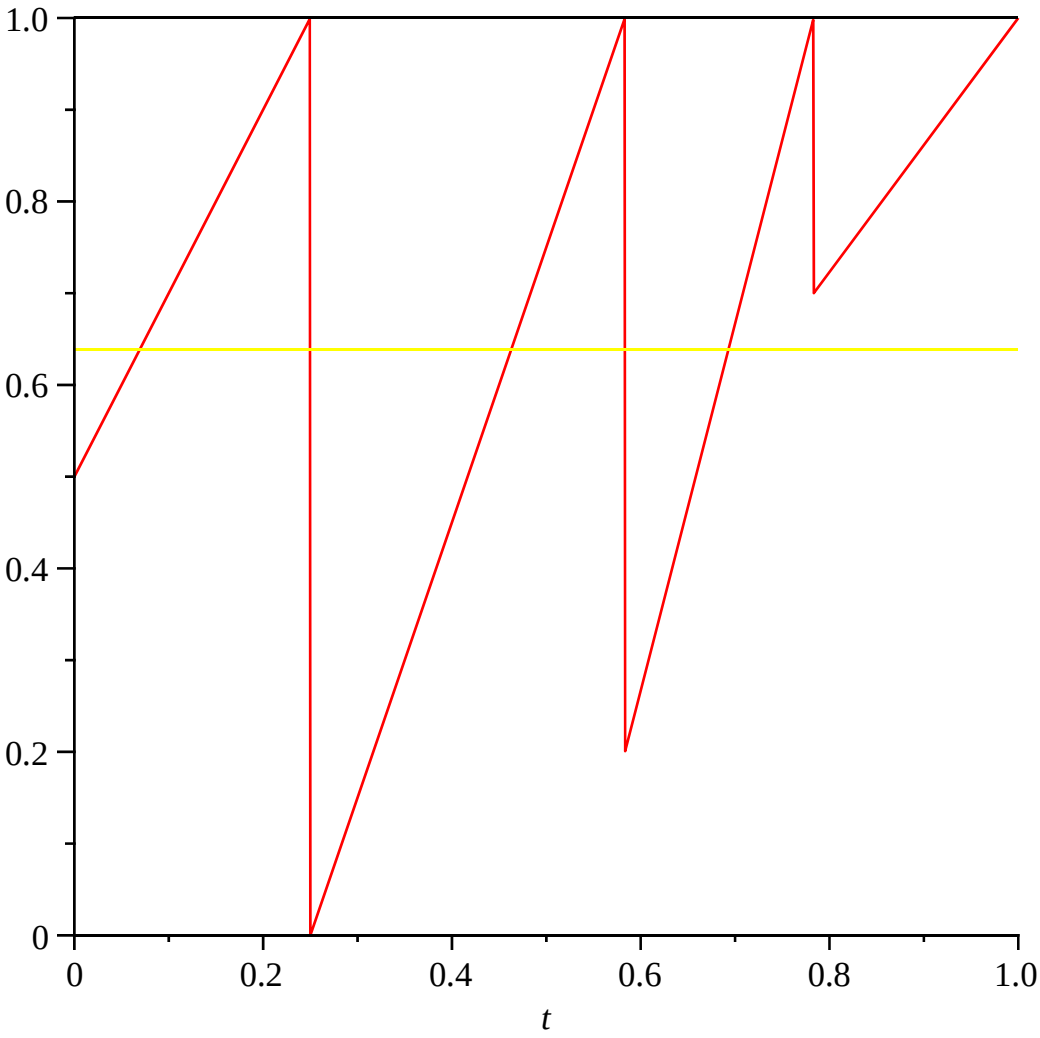

$su := 0$

1

2

3

$err_5 := 2.475363054\ 10^{-14}$



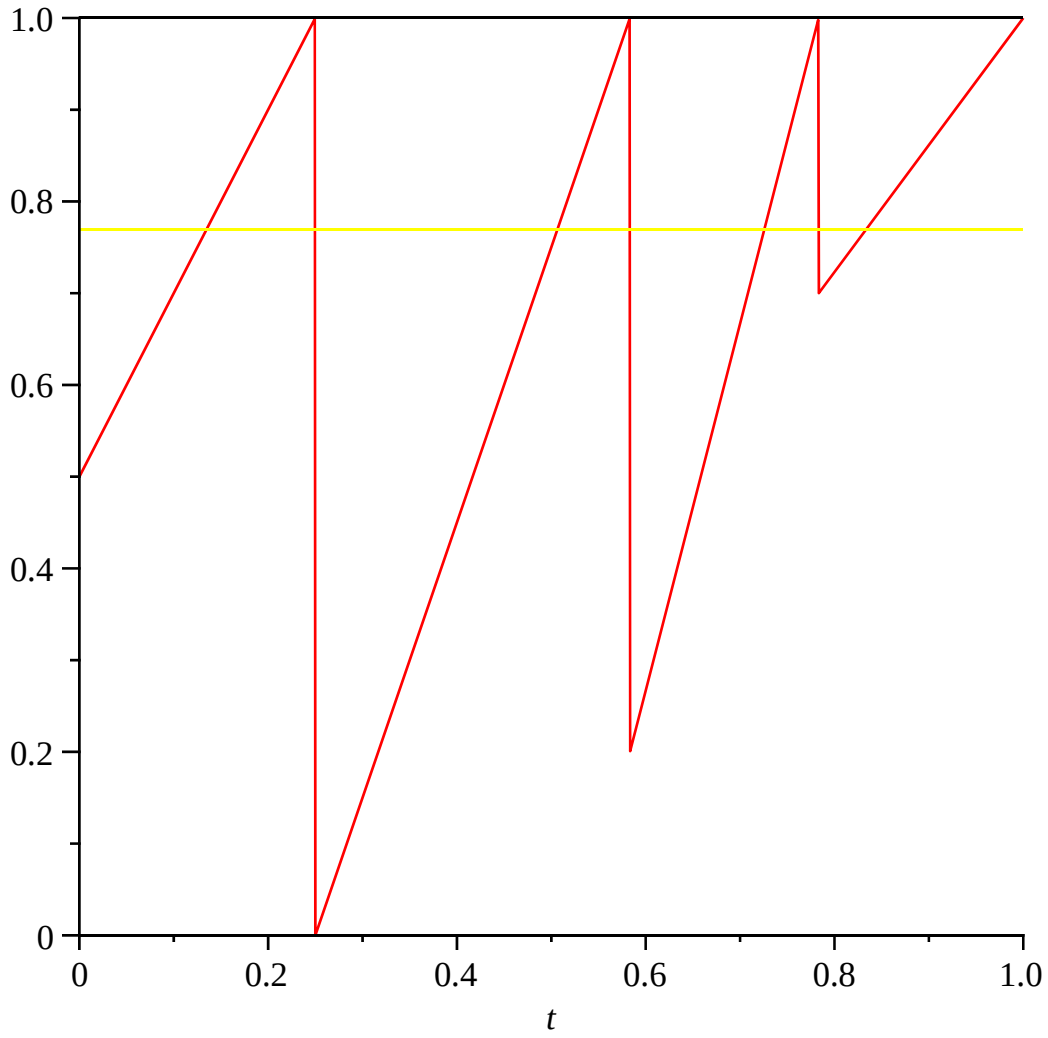

$su := 0$

1

2

3

$err_6 := 2.475363054\ 10^{-14}$




$su := 0$

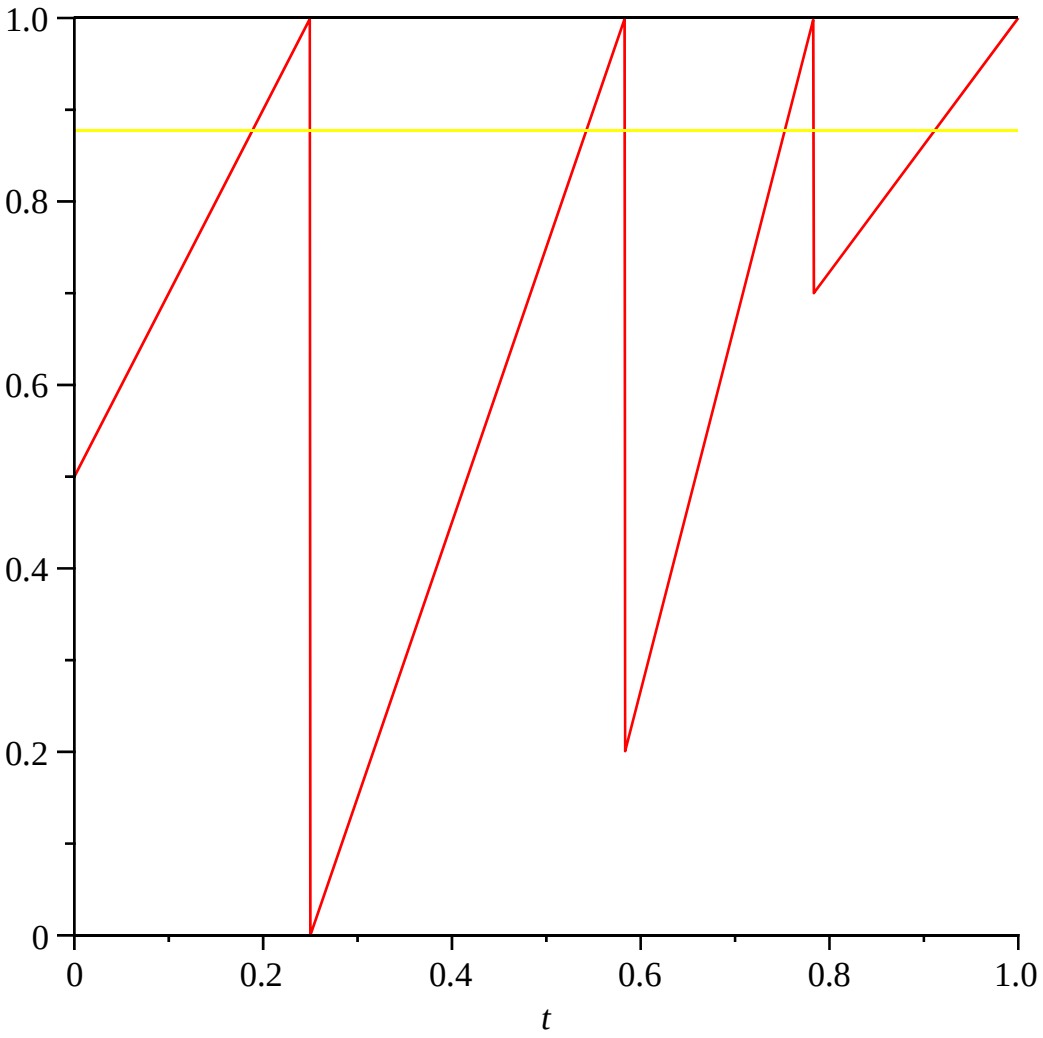
1

2

3

4

$err_7 := 1.023798059\ 10^{-14}$




$su := 0$

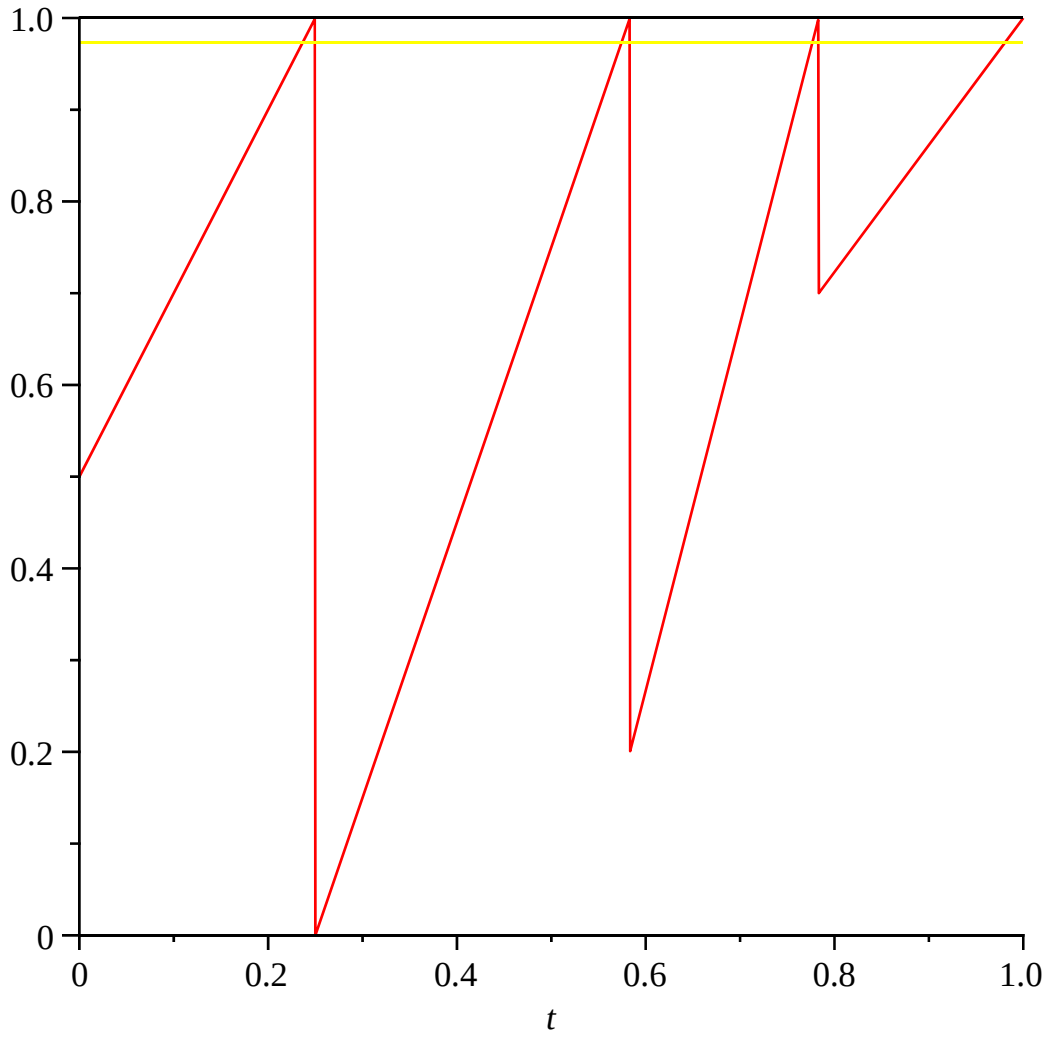
1

2

3

4

$err_8 := 1.023795377\ 10^{-14}$




$su := 0$

1

2

3

4

$err_9 := -2.331835501\ 10^{-13}$

$y =, 0.0022424170$

$err[, 0, ]=, 2.475363054\ 10^{-14}$

$y =, 0.1800187484$

$err[, 1, ]=, 2.475363054\ 10^{-14}$

$y =, 0.2427552057$

$err[, 2, ]=, 2.475363054\ 10^{-14}$

$y =, 0.3842622684$

$err[, 3, ]=, 2.475363054\ 10^{-14}$

$y =, 0.4412286286$

$err[, 4, ]=, 2.475363054\ 10^{-14}$

$y =, 0.5996417214$

$err[, 5, ]=, 2.475363054\ 10^{-14}$

$y$ =, 0.6386408307

$err[, 6, ]$=, $2.475363054\ 10^{-14}$

$y$ =, 0.7694607189

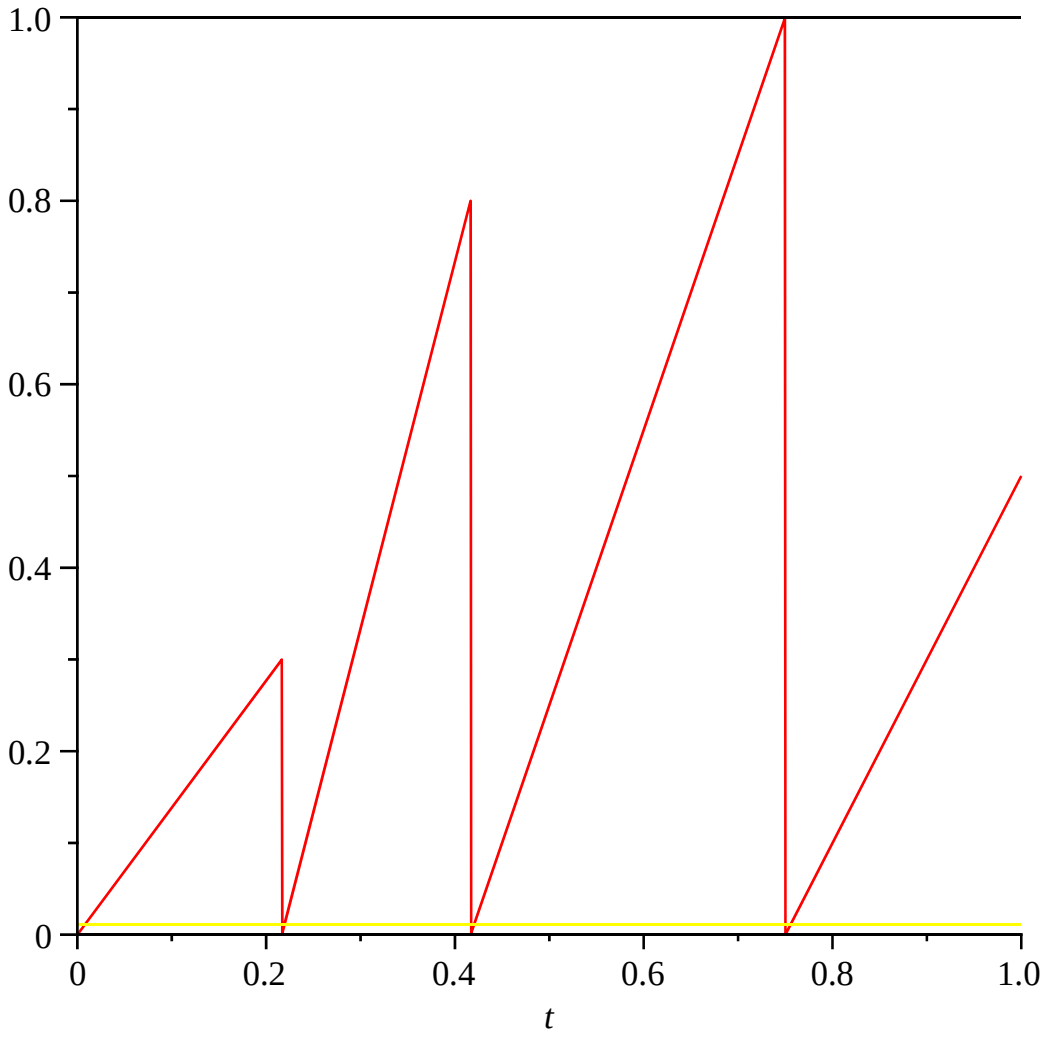
$err[, 7, ]$=, $1.023798059\ 10^{-14}$

$y$ =, 0.8773012980

$err[, 8, ]$=, $1.023795377\ 10^{-14}$

$y$ =, 0.9730616293

$err[, 9, ]$=, $-2.331835501\ 10^{-13}$




$su := 0$

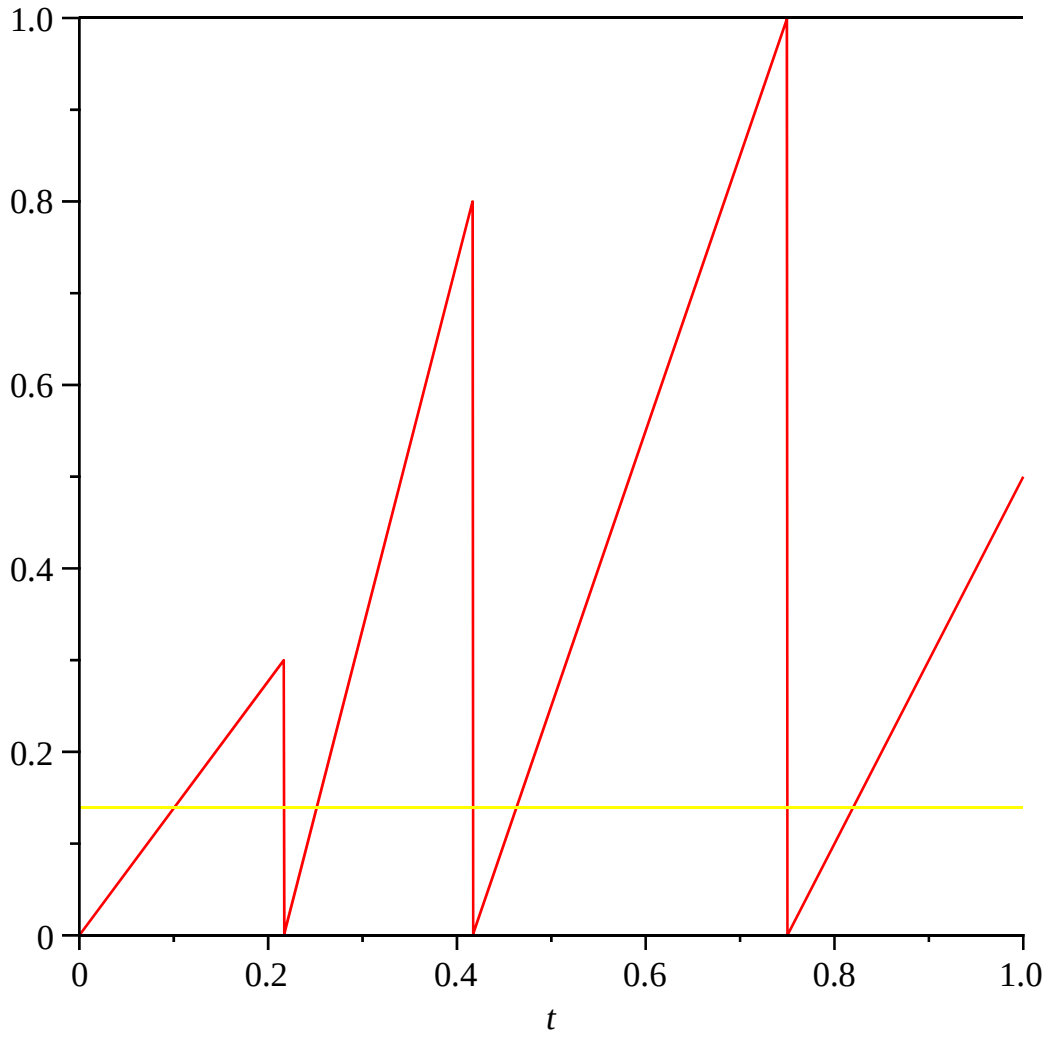
1

2

3

4

$err_0 := -2.331835501\ 10^{-13}$




$su := 0$

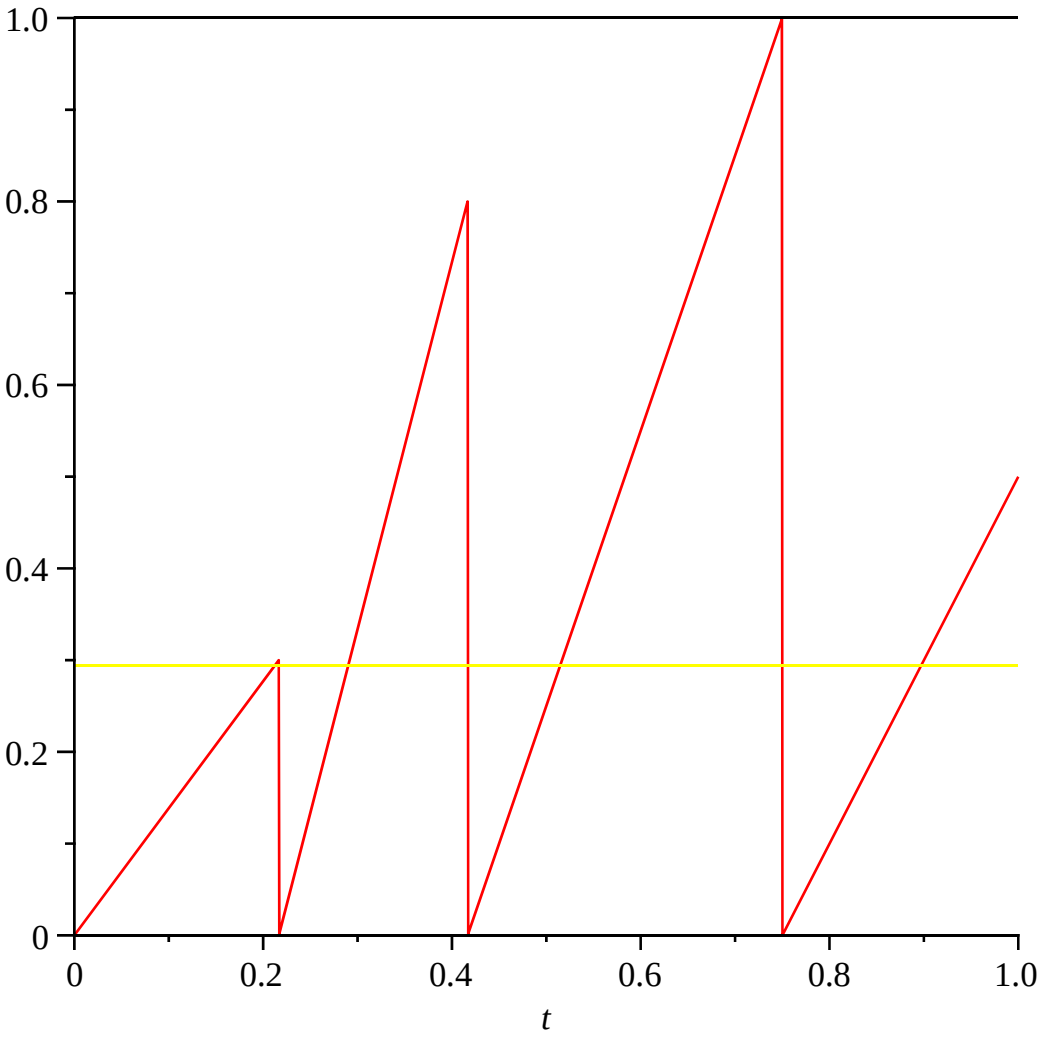
1

2

3

4

$err_1 := 1.023795377\ 10^{-14}$




$su := 0$

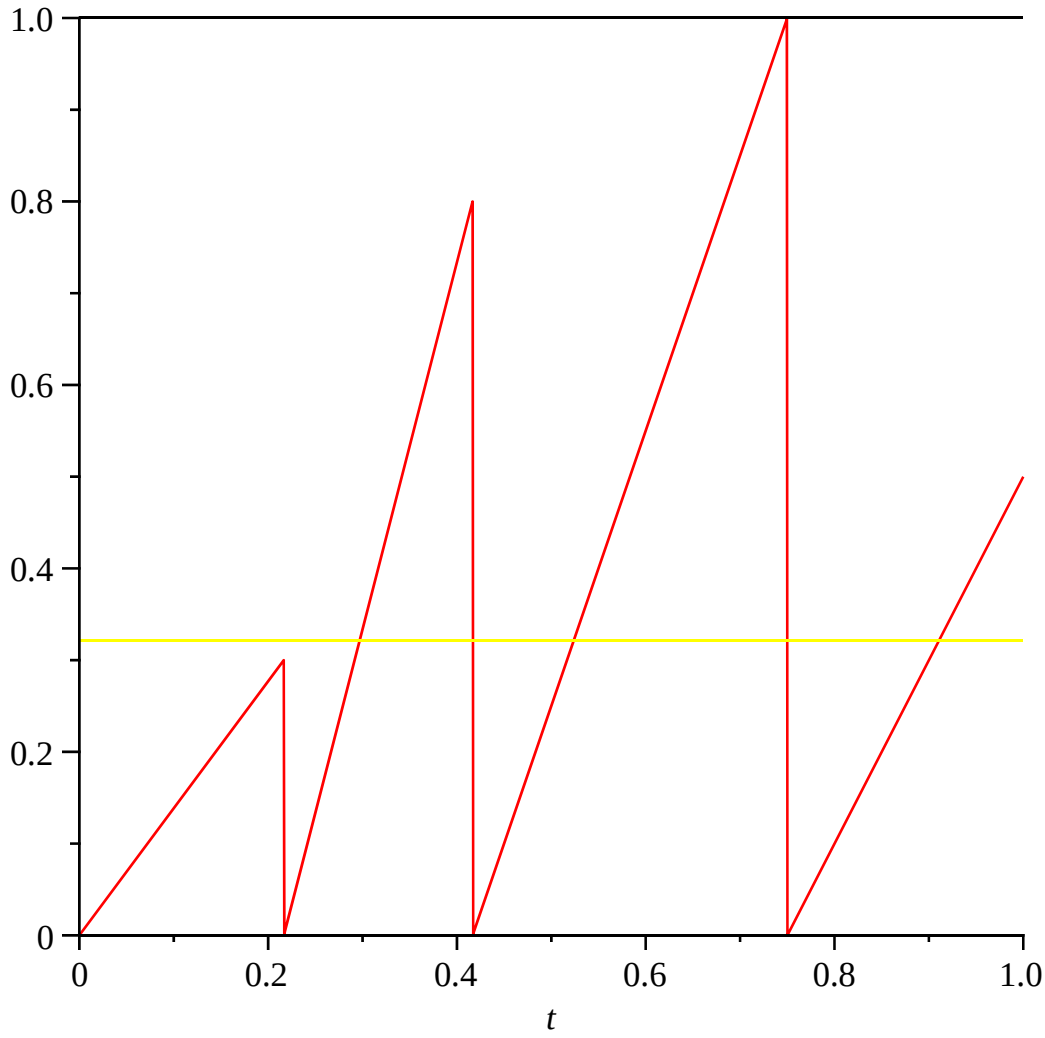
1

2

3

4

$err_2 := 2.475363054\ 10^{-14}$



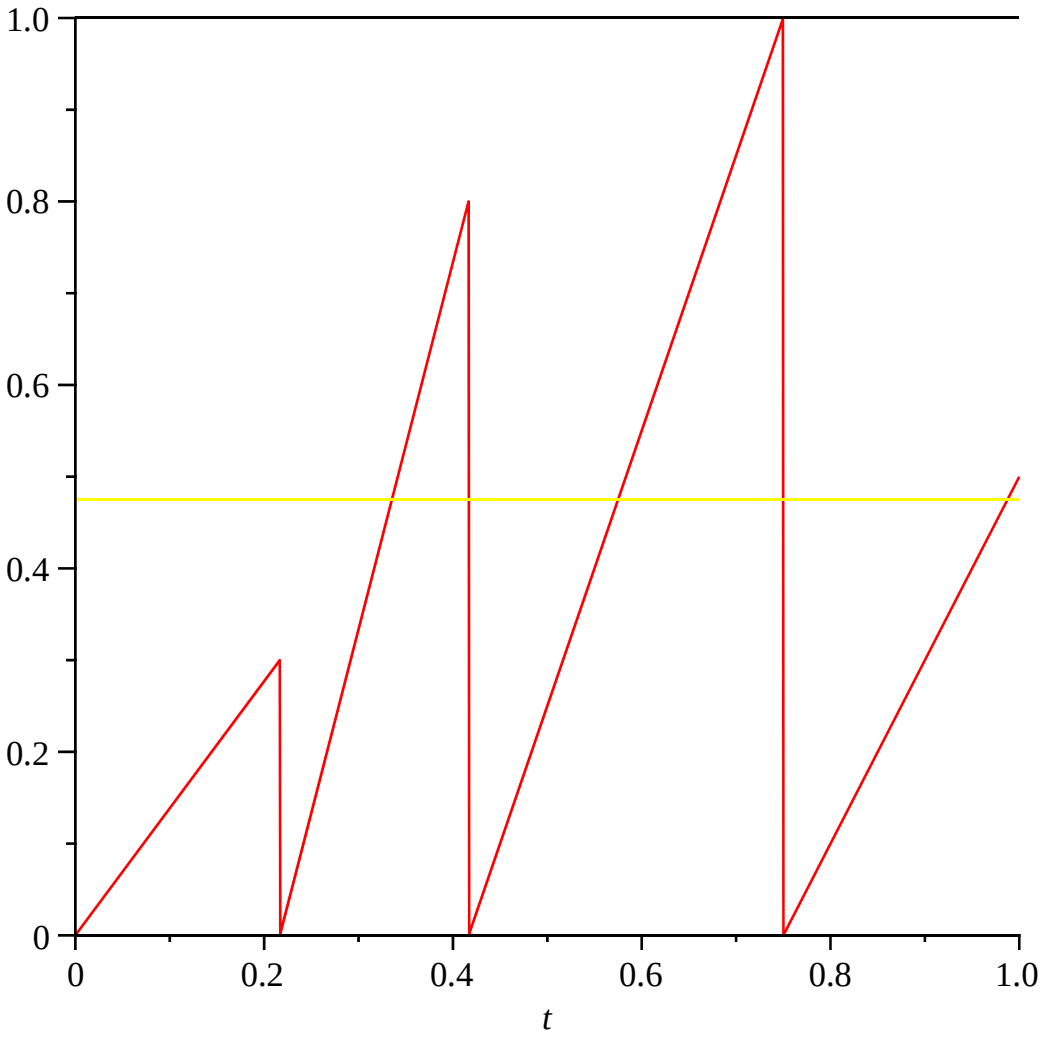

$su := 0$

2

3

4

$err_3 := 2.475363054\ 10^{-14}$



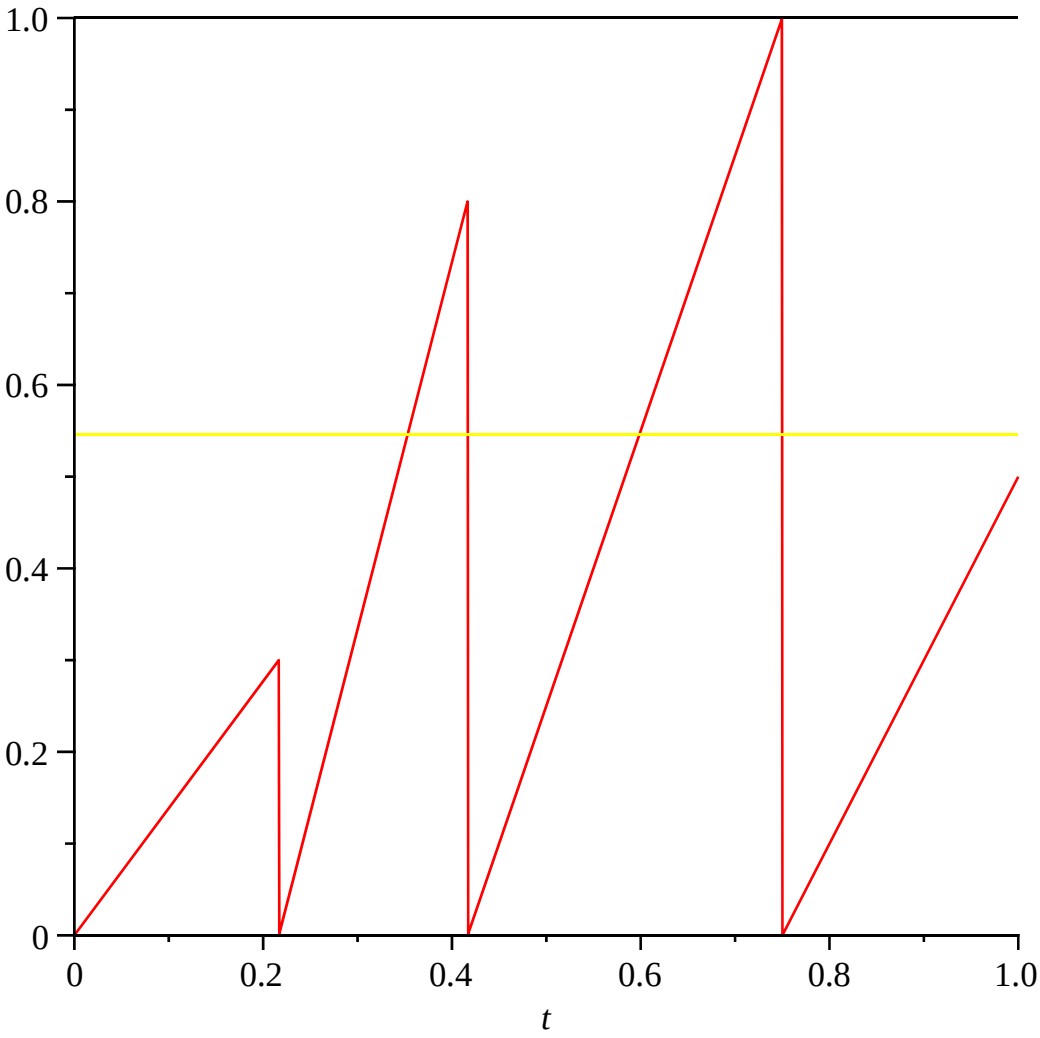

$su := 0$

2

3

4

$err_4 := 2.475363054\ 10^{-14}$

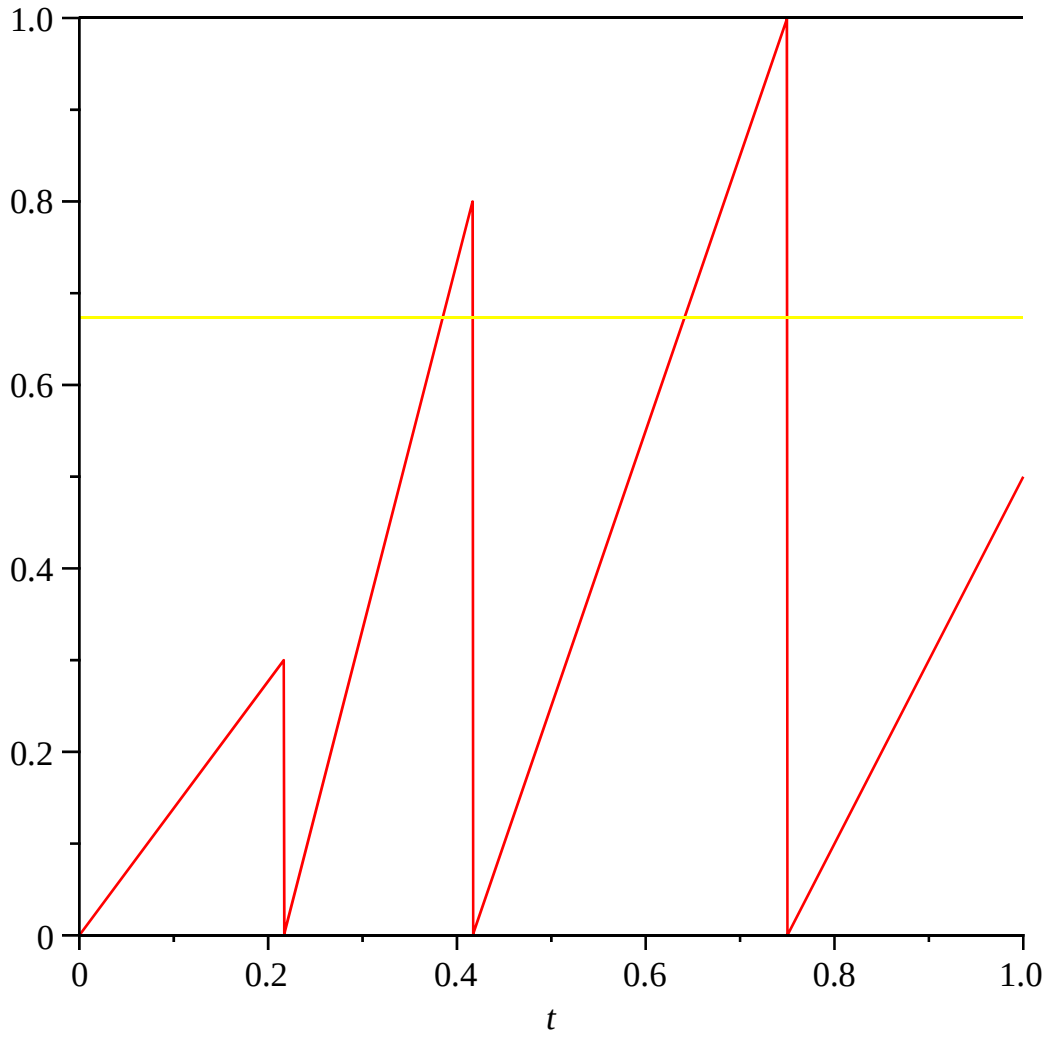



$su := 0$

2

3

$err_5 := 2.475363054\ 10^{-14}$

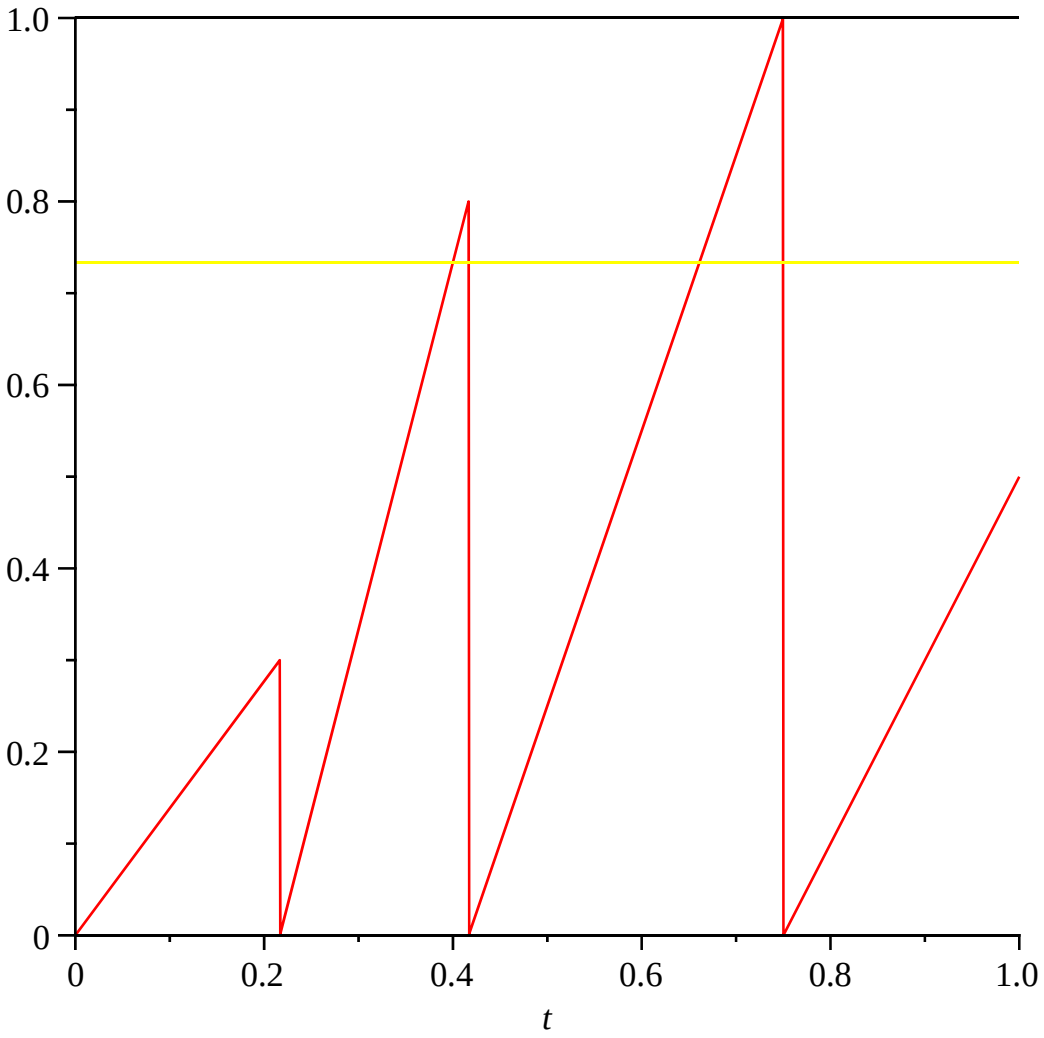



$su := 0$

2

3

$err_6 := 2.475363054\ 10^{-14}$

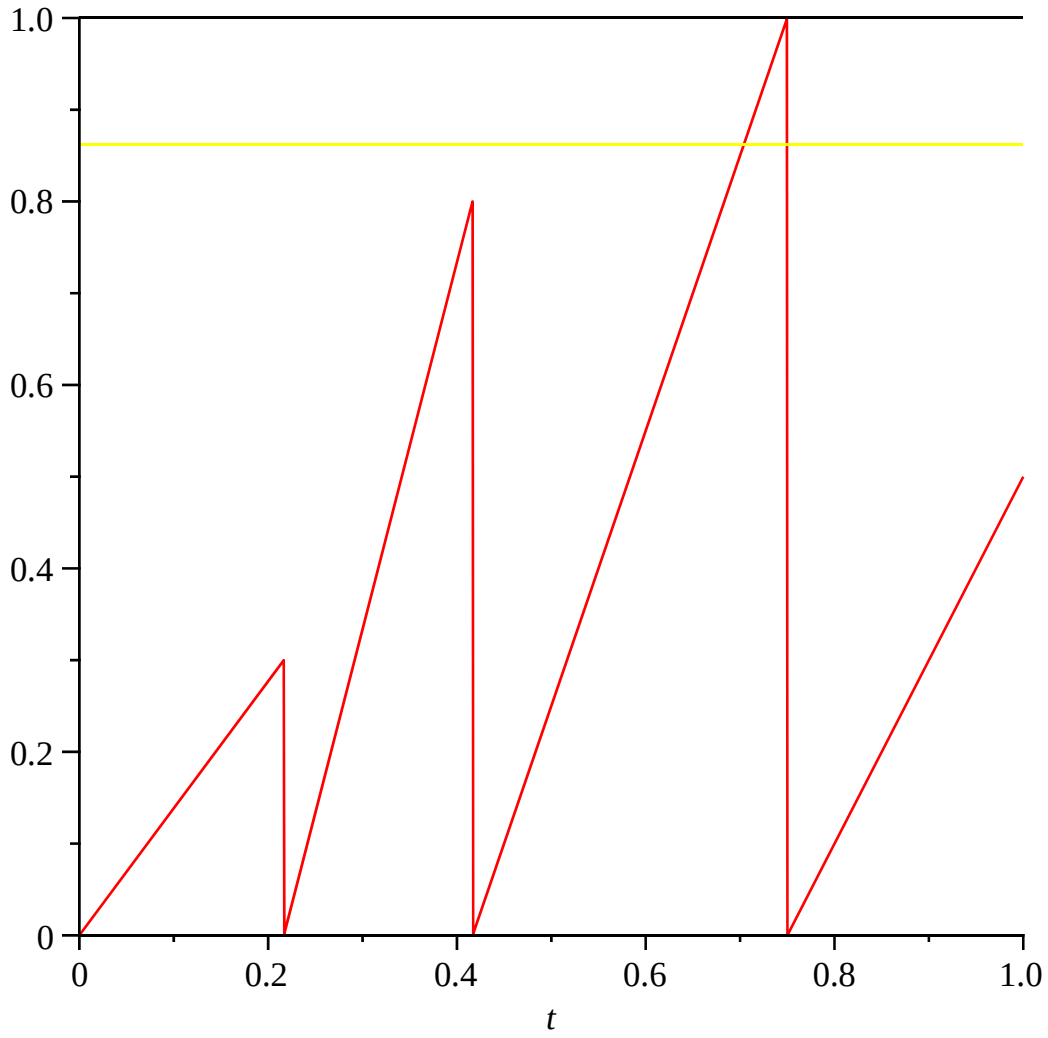



$su := 0$

2

3

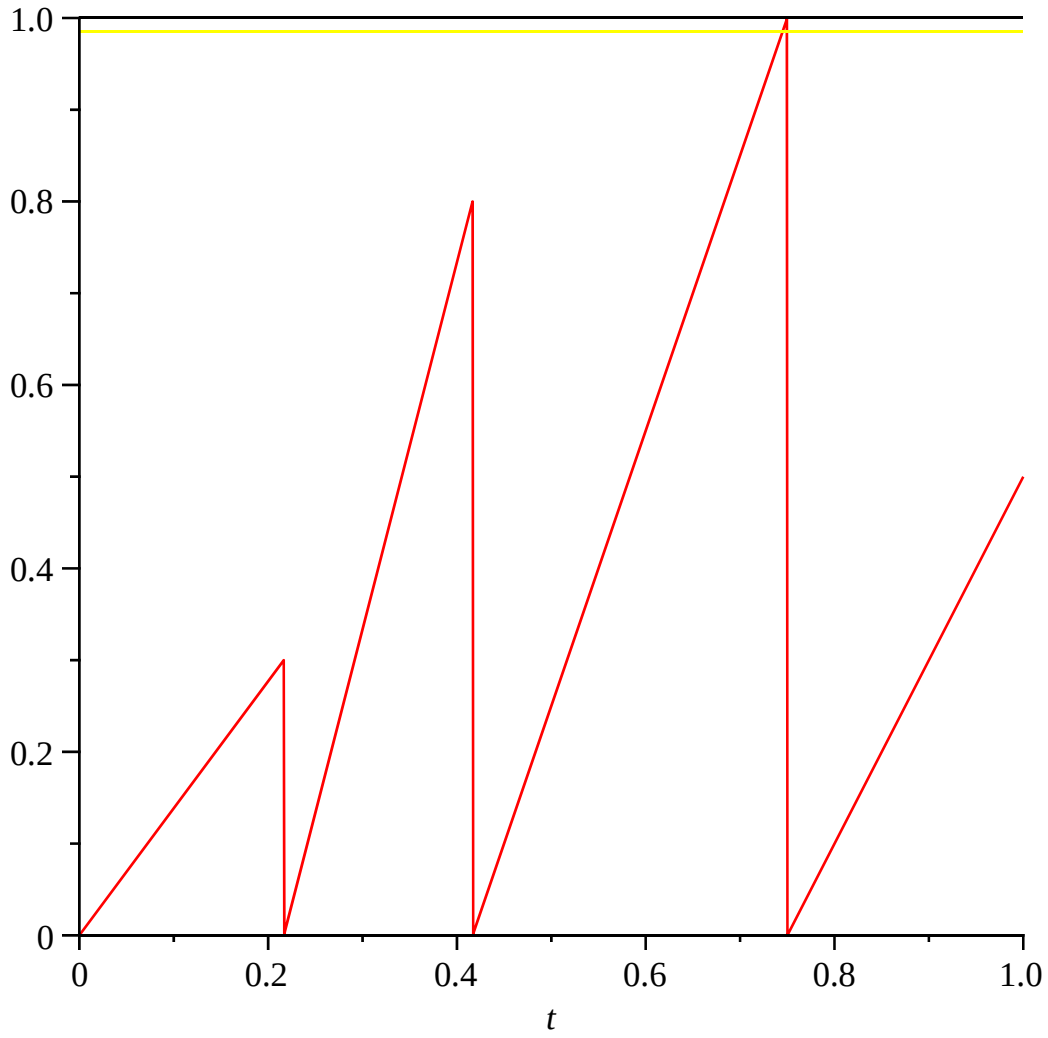
$err_7 := 2.475363054\ 10^{-14}$




$su := 0$

3

$err_8 := 2.475363054\ 10^{-14}$




$su := 0$

3

$err_9 := 2.475363054\ 10^{-14}$

$y =, 0.0106507054$

$err[, 0, ]=, -2.331835501\ 10^{-13}$

$y =, 0.1396412723$

$err[, 1, ]=, 1.023795377\ 10^{-14}$

$y =, 0.2944913350$

$err[, 2, ]=, 2.475363054\ 10^{-14}$

$y =, 0.3210936429$

$err[, 3, ]=, 2.475363054\ 10^{-14}$

$y =, 0.4750072072$

$err[, 4, ]=, 2.475363054\ 10^{-14}$

$y =, 0.5454744397$

$err[, 5, ]=, 2.475363054\ 10^{-14}$

$y =, 0.6736602622$

$err[, 6, ]=, 2.475363054\ 10^{-14}$

$y$ =, 0.7329844592

$err[, 7, ]$=, $2.475363054\ 10^{-14}$

$y$ =, 0.8615732069

$err[, 8, ]$=, $2.475363054\ 10^{-14}$

$y$ =, 0.9847018765

$err[, 9, ]$=, $2.475363054\ 10^{-14}$

>