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

> N:=2;

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]:=1:beta[1]:=1.8:gamm[1]:=0.0:

alpha[2]:=1:beta[2]:=3:gamm[2]:=0.0:#
#alpha[3]:=0.8:beta[3]:=-4:gamm[3]:=0.1:  #
#alpha[4]:=1:beta[4]:=-5:gamm[4]:=0.0:
#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]:=0.8:gamm[N]:=0.2:
i:='i':beta[N]:=-alpha[N]/(1-sum(alpha[i]/abs(beta[i]),i=1..N-1)
);


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]/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 leftc[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]],x=0..1,thickness=[2,1,1,1,1,1,1],
  numpoints=1000);
  plot(['T(x)',x,0,1,Tc[1]],x=0..1,thickness=[2,1,1,1,1,1,1,1],
  numpoints=1000);
```

$$N := 2$$

$$\beta_{2} := -1.8000000000$$

$$\mathit{alpha} =, \left[\begin{array}{cc} 1 & 0.8000000000 \end{array}\right]$$

$$\mathit{beta} =, \left[\begin{array}{cc} 1.8000000000 & -1.8000000000 \end{array}\right]$$

$$\mathit{gamma} =, \left[\begin{array}{cc} 0.0000000000 & 0.2000000000 \end{array}\right]$$

$$\beta\mathit{const} := 1.8000000000$$

$$b =, \left[\begin{array}{ccc} 0 & 0.5555555556 & 1 \end{array}\right]$$

$$\mathit{ag} =, \left[\begin{array}{cc} 0.0000000000 & -1.0000000000 \end{array}\right]$$

$$\mathit{al} =, \left[\begin{array}{cc} 0.0000000000 & -2.8000000000 \end{array}\right]$$

$$a =, \left[\begin{array}{cc} 0.0000000000 & -2.0000000000 \end{array}\right]$$

$$\mathit{gamma} =, \left[\begin{array}{cc} 0.0000000000 & 0.2000000000 \end{array}\right]$$

$$\mathit{Kc} =, 1$$

$$c =, \left[\begin{array}{cccc} 1 & c_{2} & c_{3} & c_{4} \end{array}\right]$$

$$\mathit{sidec} =, \left[\begin{array}{cccc} 1 & \mathit{sidec}_{2} & \mathit{sidec}_{3} & \mathit{sidec}_{4} \end{array}\right]$$

$$\mathit{leftc} =, \left[\begin{array}{cccc} 0 & \mathit{leftc}_{2} & \mathit{leftc}_{3} & \mathit{leftc}_{4} \end{array}\right]$$

$$\mathit{j\_of\_c} =, \left[\begin{array}{cccc} 2 & \mathit{j\_of\_c}_{2} & \mathit{j\_of\_c}_{3} & \mathit{j\_of\_c}_{4} \end{array}\right]$$

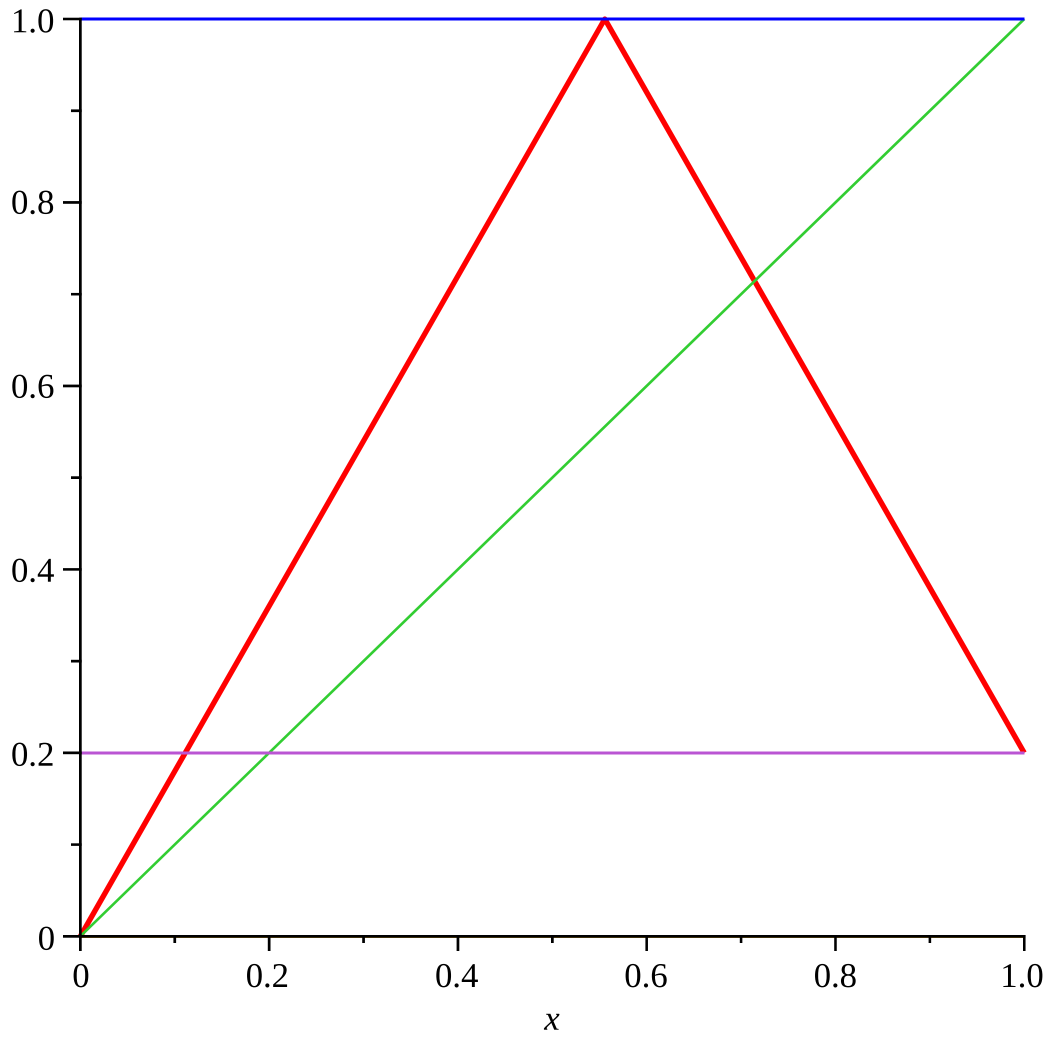
$$\mathit{uint\_of\_x} := x \rightarrow \mathit{piecewise}\left(x < b_{2}, 1, x < b_{3}, 2, x < b_{4}, 3, x < b_{5}, 4, x < b_{6}, 5, x < b_{7}, 6, x < b_{8}, 7, x < b_{9}, 8, 9\right)$$

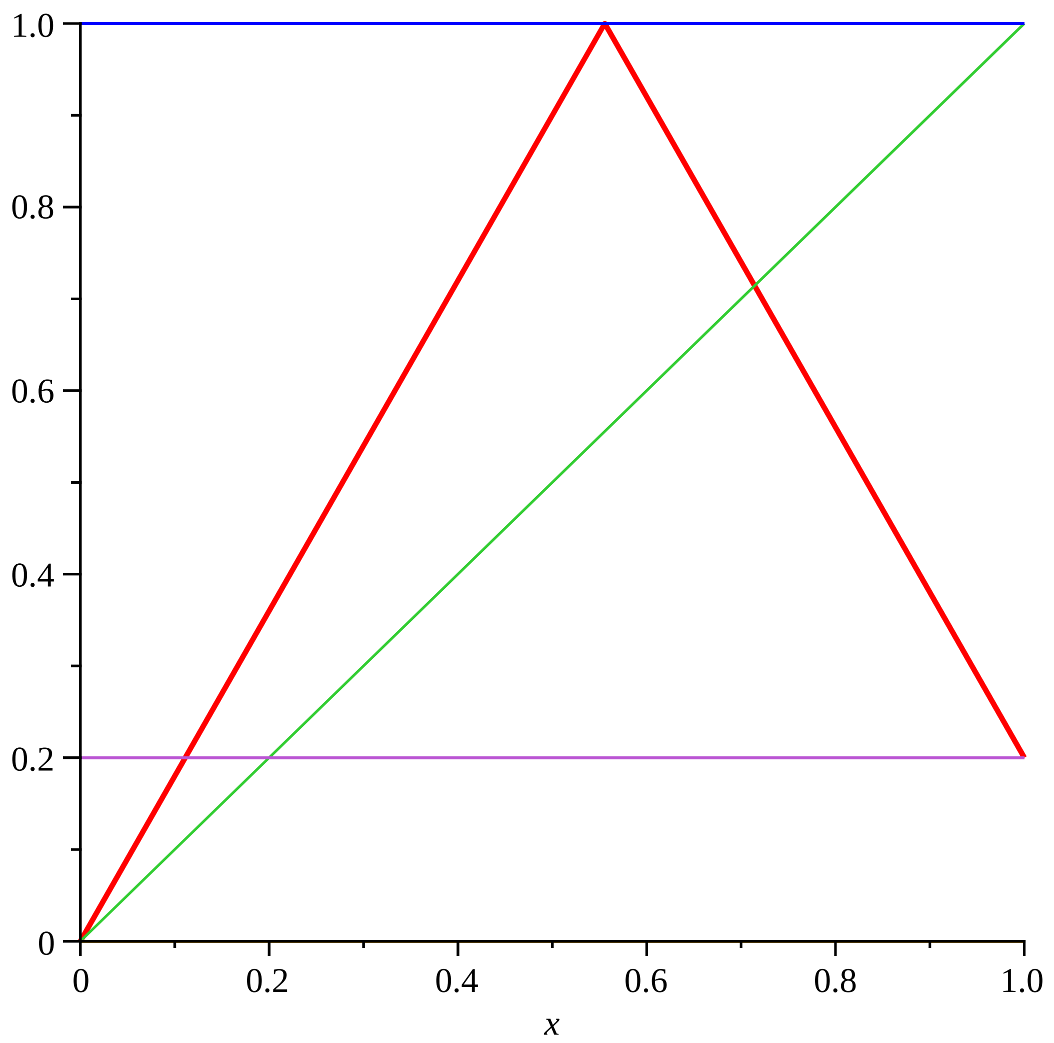
$$\mathit{int\_of\_x} := x \rightarrow \mathit{piecewise}\left(x \leq b_{2}, 1, x \leq b_{3}, 2, x \leq b_{4}, 3, x \leq b_{5}, 4, x \leq b_{6}, 5, x \leq b_{7}, 6, x \leq b_{8}, 7, x \leq b_{9}, 8, 9\right)$$

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

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

$$\mathit{Tc} = , \left[\begin{array}{ccc} 0.2000000000 & \mathit{Tc}_{2} & \mathit{Tc}_{3} \end{array}\right]$$


1.0
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1.0
x



```
>
>
> 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;
```

$$Digits := 100$$

$$NN := 50$$

$$xx := 0.7301565454$$

```
[-2.0000000000, -2.0000000000, -2.0000000000, -2.0000000000, -2.0000000000,
    0.0000000000, -2.0000000000, -2.0000000000, -2.0000000000, 0.0000000000,
    -2.0000000000, 0.0000000000, 0.0000000000, -2.0000000000, 0.0000000000,
    -2.0000000000, -2.0000000000, 0.0000000000, -2.0000000000, -2.0000000000,
    -2.0000000000, 0.0000000000, -2.0000000000, -2.0000000000, -2.0000000000,
    -2.0000000000, -2.0000000000, -2.0000000000, 0.0000000000, 0.0000000000,
    -2.0000000000, -2.0000000000, -2.0000000000, -2.0000000000, -2.0000000000,
    0.0000000000, -2.0000000000, 0.0000000000, 0.0000000000, -2.0000000000,
    0.0000000000, -2.0000000000, -2.0000000000, -2.0000000000, -2.0000000000,
    0.0000000000, -2.0000000000, -2.0000000000, -2.0000000000, -2.0000000000]
```

$$uIs\_it\_x := 0.7301565454$$

```
[-2.0000000000, -2.0000000000, -2.0000000000, -2.0000000000, -2.0000000000,
    0.0000000000, -2.0000000000, -2.0000000000, -2.0000000000, 0.0000000000,
    -2.0000000000, 0.0000000000, 0.0000000000, -2.0000000000, 0.0000000000,
    -2.0000000000, -2.0000000000, 0.0000000000, -2.0000000000, -2.0000000000,
    -2.0000000000, 0.0000000000, -2.0000000000, -2.0000000000, -2.0000000000,
    -2.0000000000, -2.0000000000, -2.0000000000, 0.0000000000, 0.0000000000,
    -2.0000000000, -2.0000000000, -2.0000000000, -2.0000000000, -2.0000000000,
    0.0000000000, -2.0000000000, 0.0000000000, 0.0000000000, -2.0000000000,
    0.0000000000, -2.0000000000, -2.0000000000, -2.0000000000, -2.0000000000,
    0.0000000000, -2.0000000000, -2.0000000000, -2.0000000000, -2.0000000000]
```

$$Is\_it\_x := 0.7301565454$$

$$terr := 1.199917121\ 10^{-13}$$

$$err := 1.199917121\ 10^{-13} \qquad (1)$$

```
>
>

>



> 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:
ls_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;
```

$$NN := 150$$

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

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

$$xxt := 1$$

$$bet := 1$$

$$varleftc := 0$$

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

$$S_1 := 0.5694444444 \qquad (2)$$

```
>
>
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 = `,MMM);
print(ve);
1/beta[2];


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));
```

$$MM = , \left[\begin{array}{c} -1.1250000000 \end{array} \right]$$

$$\mathit{eigenvalues\ MM} =, -1.1250000000$$

$$\mathit{MMM} = , \left[\ -0.1250000000\ \right]$$

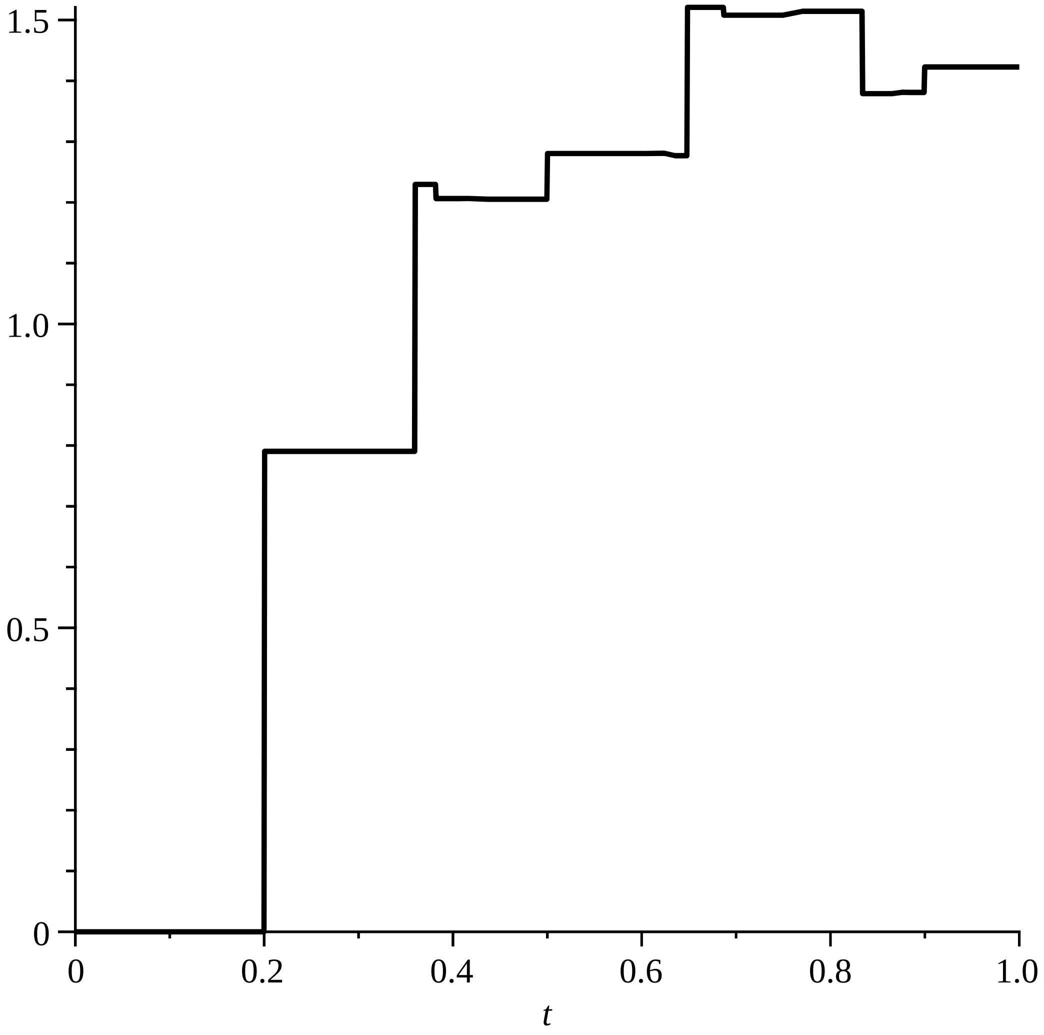
$$\left[\ 1\ \right]$$

$$-0.5555555556$$

$$\mathit{DD} := \left[\ -8.0000000000\ \right]$$

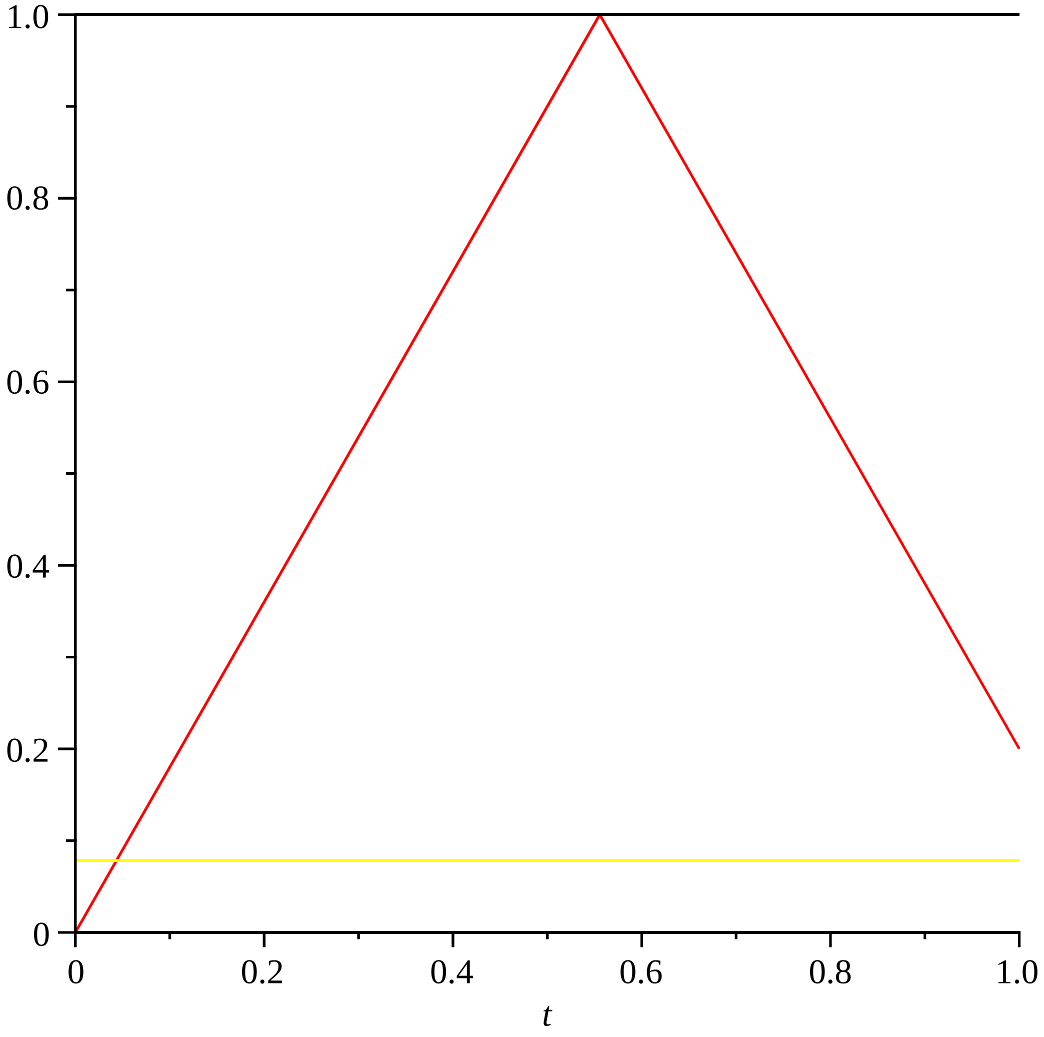
$$1.003298120\ 10^{-38} \qquad \textbf{(3)}$$

```
>
>
Ntt:=50;

     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 Ntt 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 Ntt 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,color=black,
thickness=2);
                                Ntt := 50
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 Ntt 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 Ntt 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 := -5.6227985380
                          NC = , -5.6227985380
```



```
>
>
   #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);
```

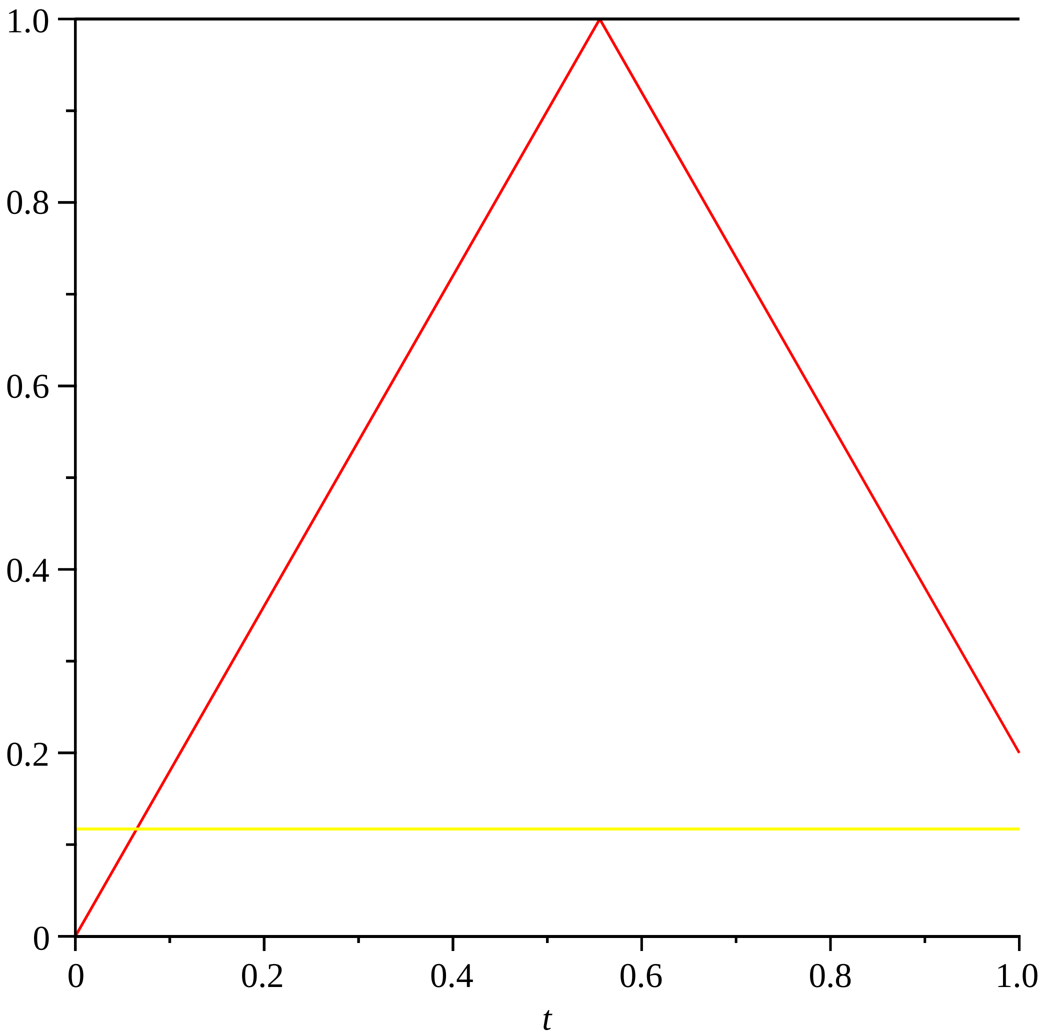
```
od;

for j6 from 0 to 9 do

print(`y =`,y[j6],`err[`,j6,`]=`,err[j6]);

od;
```



$$su := 0$$

$$1$$

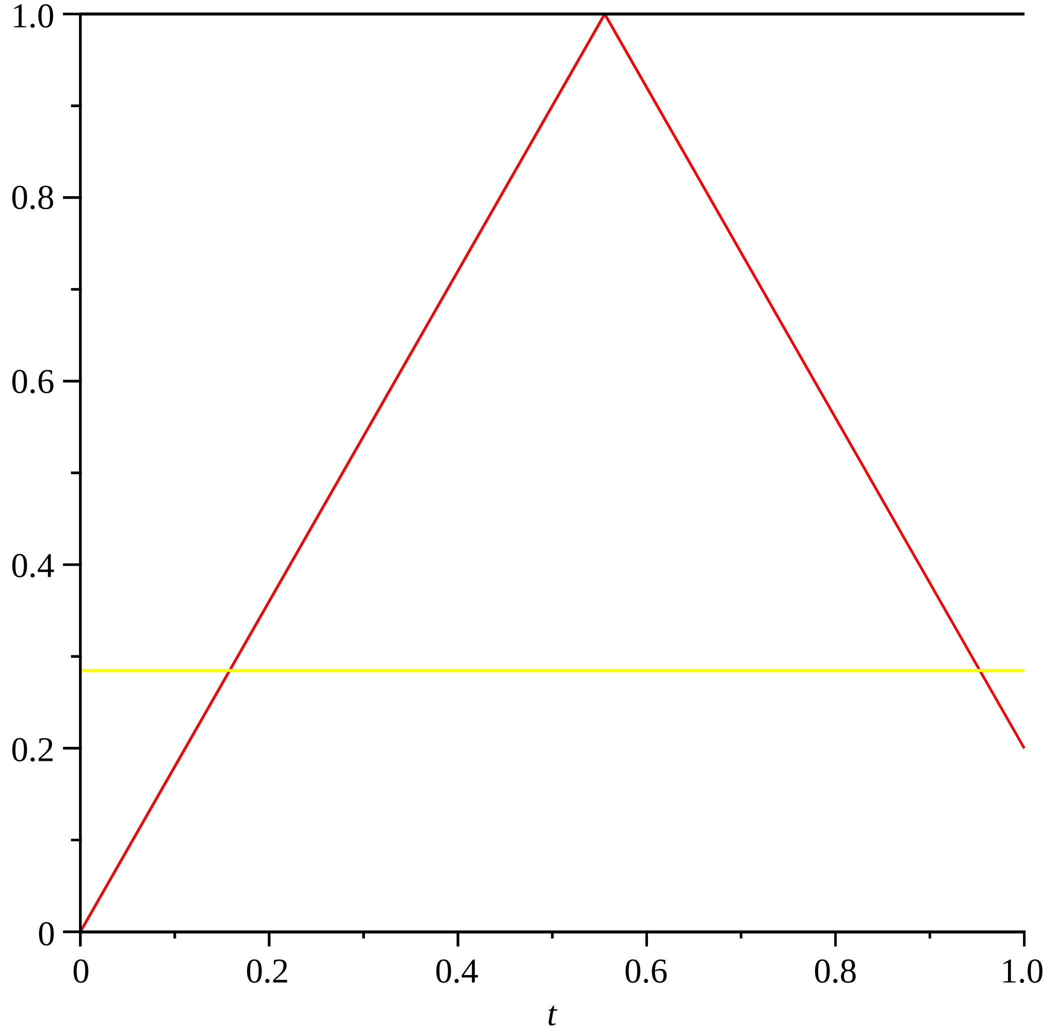
$$err_0 := 8.669617735\ 10^{-14}$$



$$su := 0$$

$$1$$

$$err_1 := 8.669617735\ 10^{-14}$$

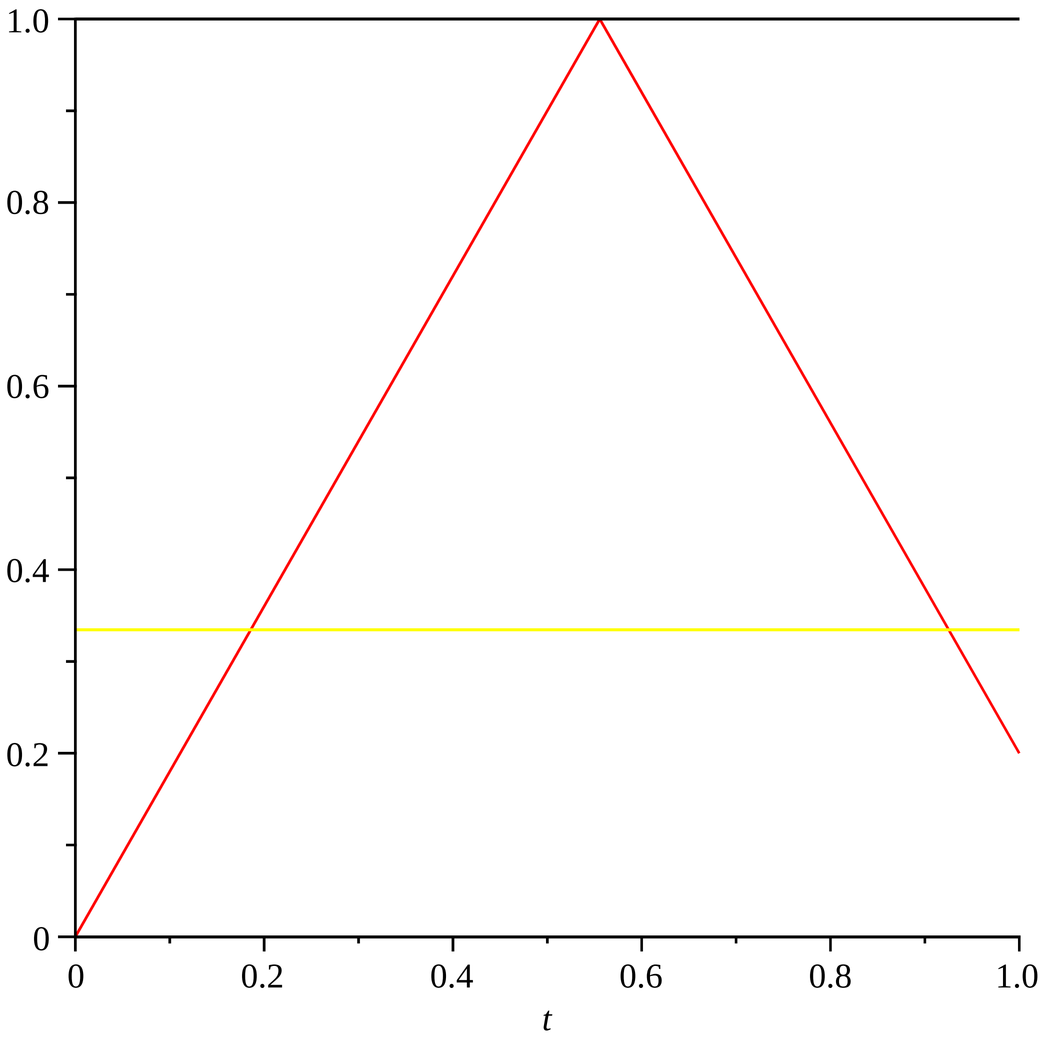


$$su := 0$$

$$1$$

$$2$$

$$err_2 := 3.582511780\ 10^{-15}$$

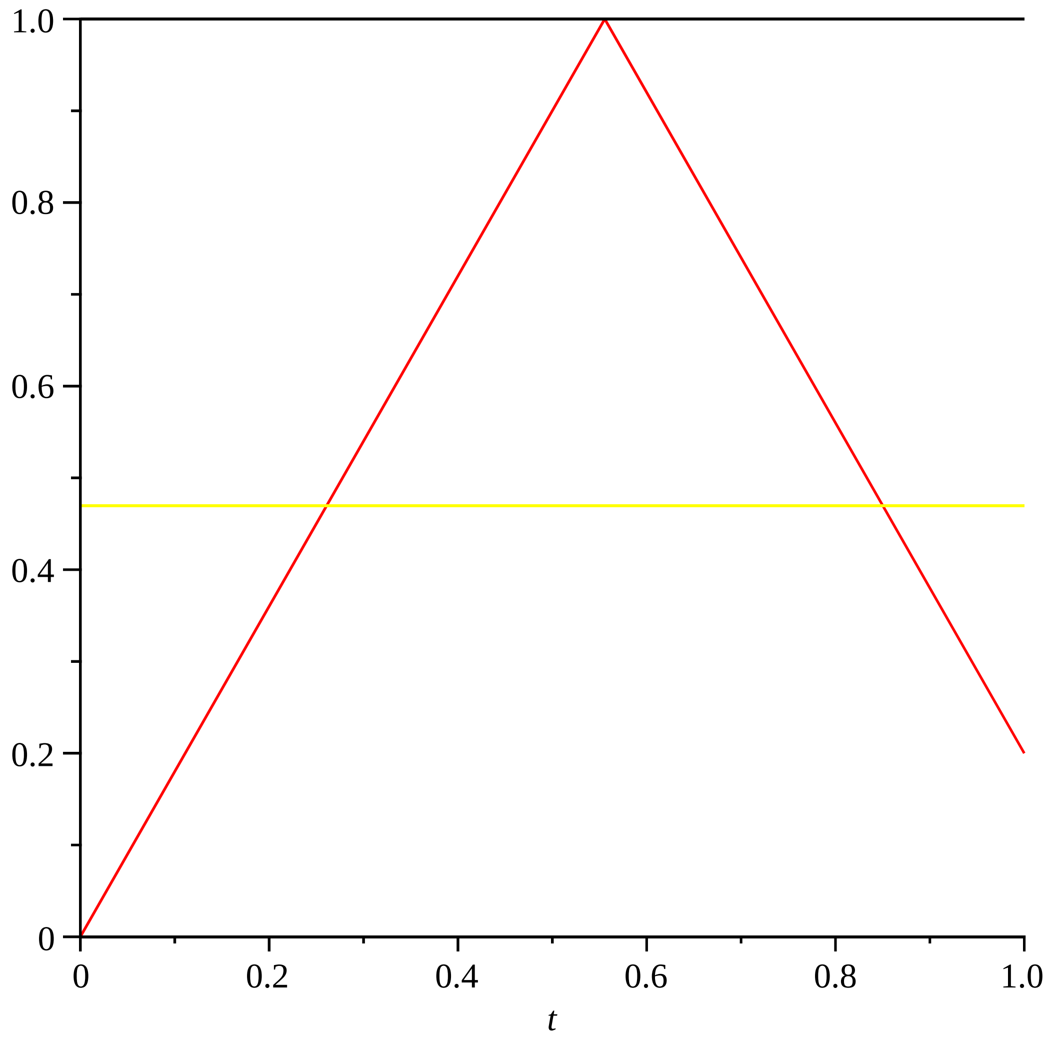


$$su := 0$$

$$1$$

$$2$$

$$err_3 := 3.582511780\ 10^{-15}$$

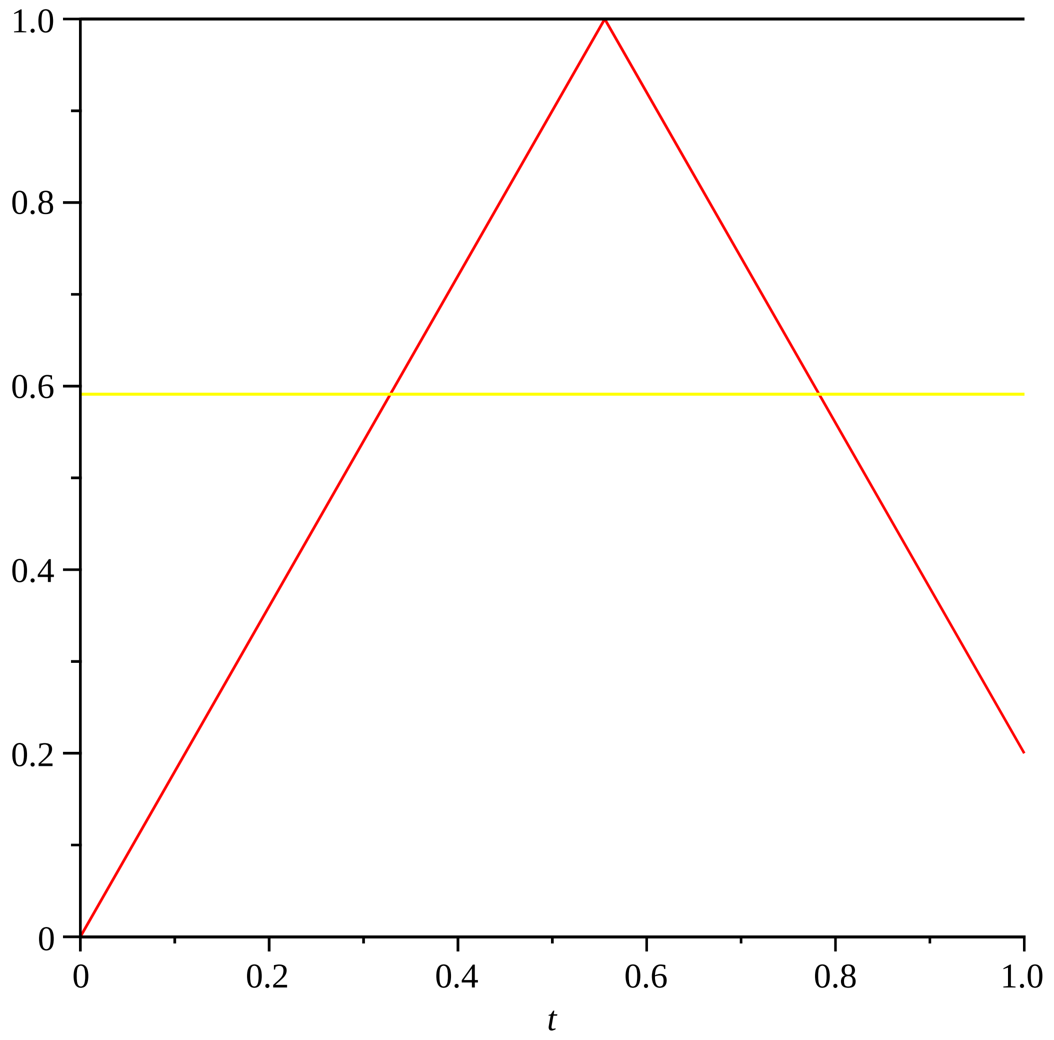


$$su := 0$$

$$1$$

$$2$$

$$err_4 := 3.582511780\ 10^{-15}$$

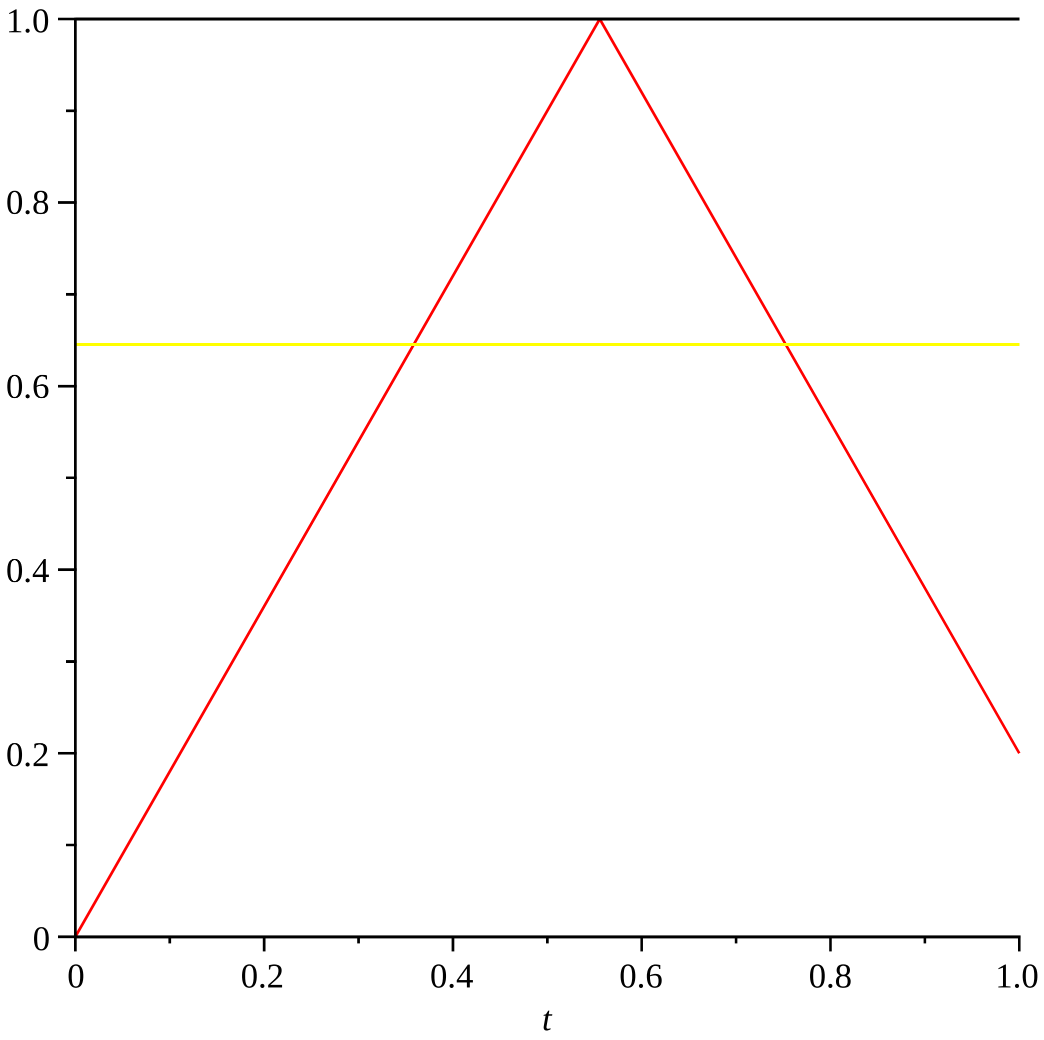


$$su := 0$$

$$1$$

$$2$$

$$err_5 := 3.582511780\ 10^{-15}$$

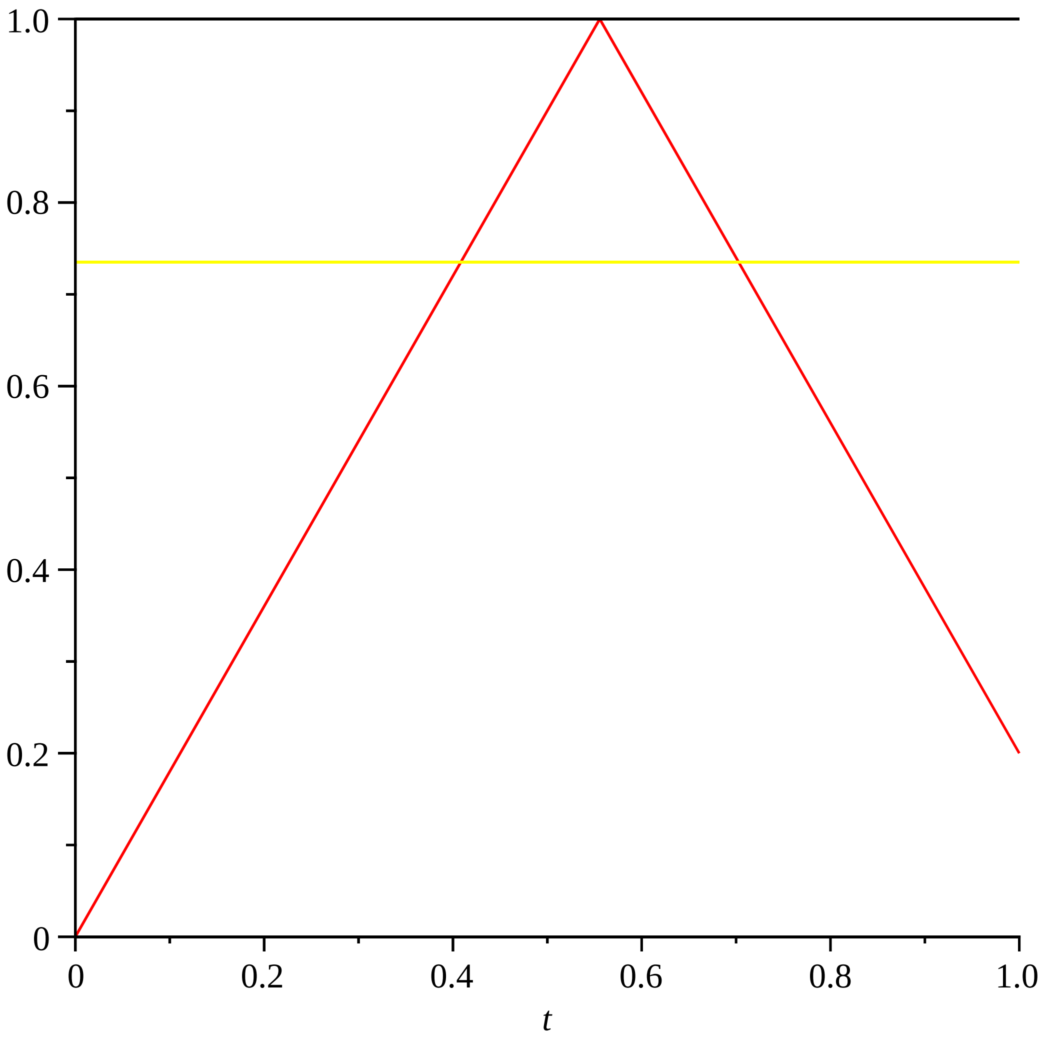


$$su := 0$$

$$1$$

$$2$$

$$err_6 := 3.582511780\ 10^{-15}$$

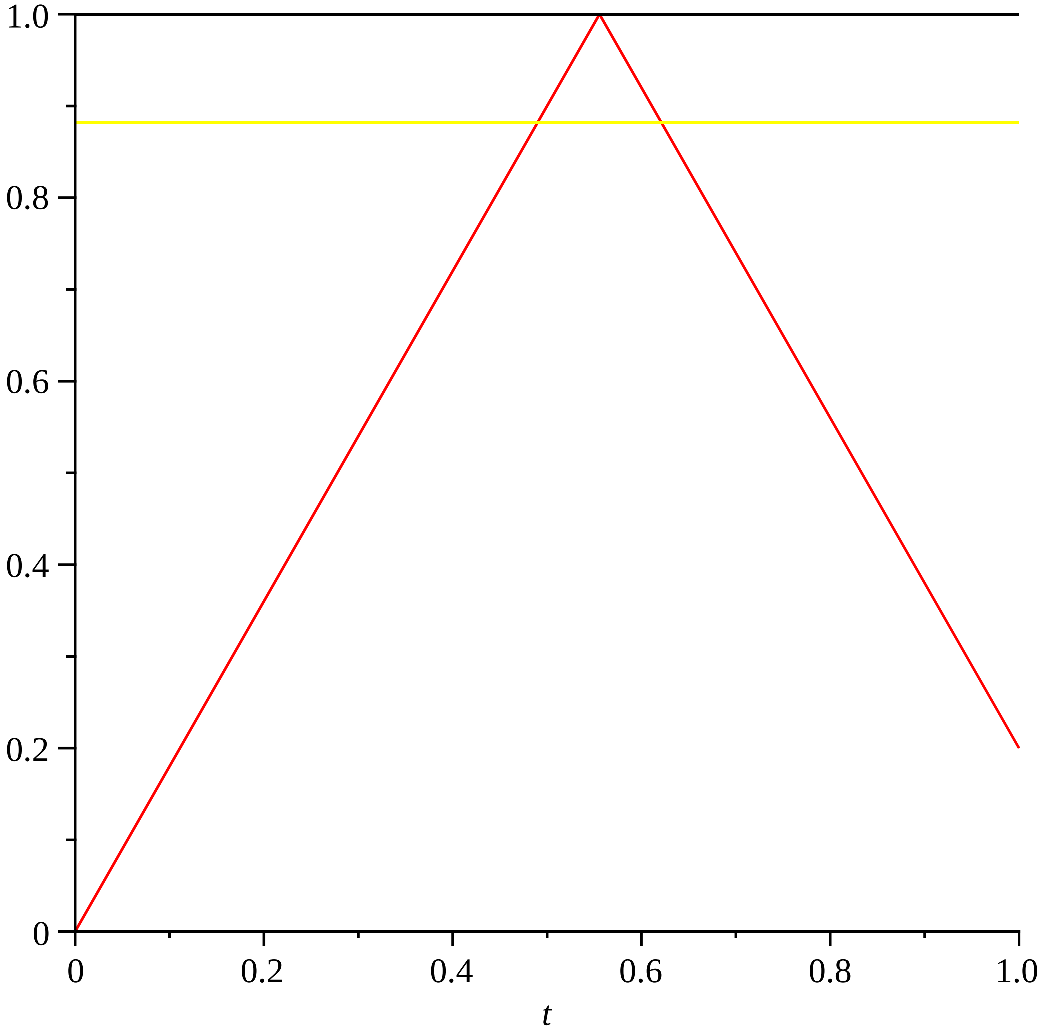


$su := 0$

1

2

$err_7 := 3.582511780\ 10^{-15}$

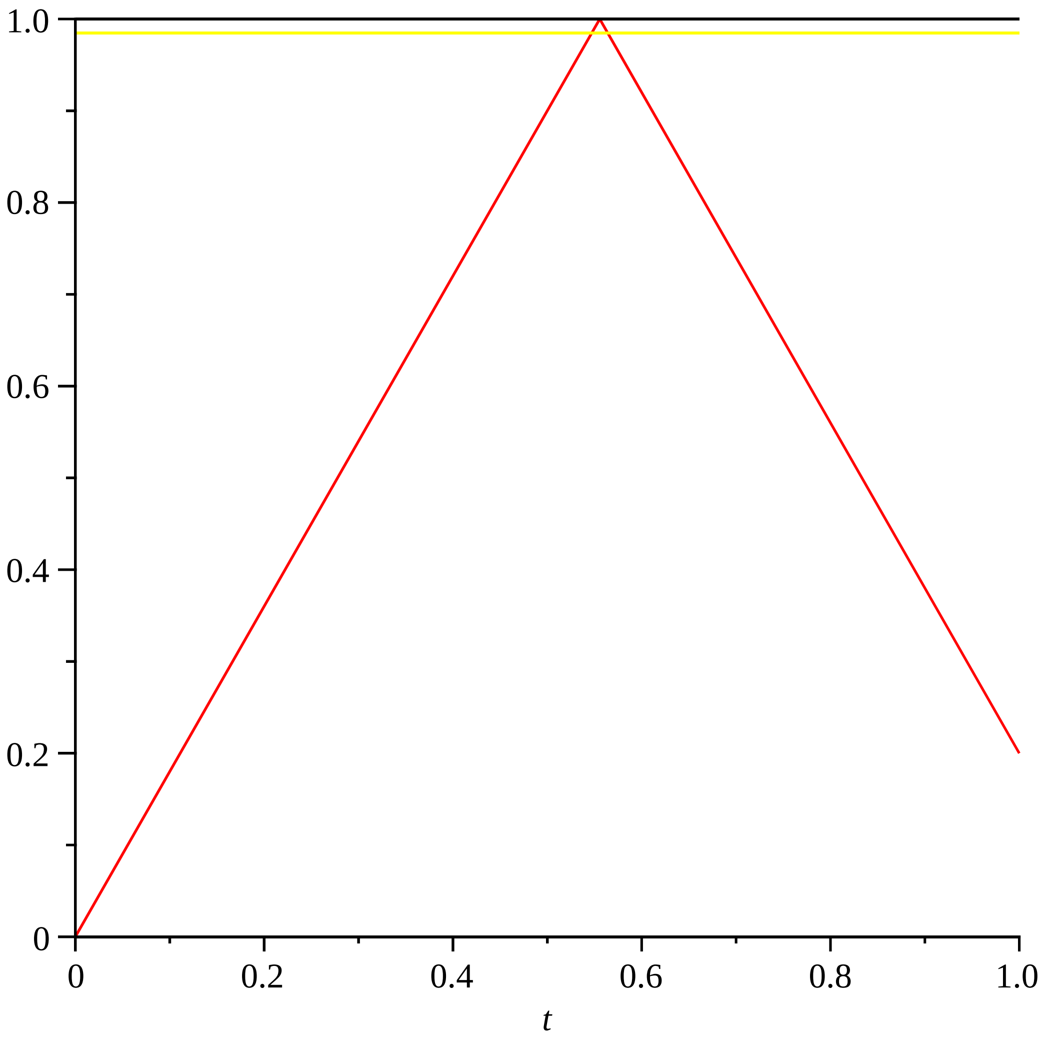


$$su := 0$$

$$1$$

$$2$$

$$err_8 := 3.582511780\ 10^{-15}$$



$$su := 0$$

$$1$$

$$2$$

$$err_9 := 3.582511780\ 10^{-15}$$

$$y =, 0.0784736498, err[, 0,]=, 8.669617735\ 10^{-14}$$

$$y =, 0.1170269516, err[, 1,]=, 8.669617735\ 10^{-14}$$

$$y =, 0.2848067203, err[, 2,]=, 3.582511780\ 10^{-15}$$

$$y =, 0.3346921332, err[, 3,]=, 3.582511780\ 10^{-15}$$

$$y =, 0.4693953992, err[, 4,]=, 3.582511780\ 10^{-15}$$

$$y =, 0.5914298711, err[, 5,]=, 3.582511780\ 10^{-15}$$

$$y =, 0.6450147686, err[, 6,]=, 3.582511780\ 10^{-15}$$

$$y =, 0.7352263657, err[, 7,]=, 3.582511780\ 10^{-15}$$

$$y =, 0.8819571011, err[, 8,]=, 3.582511780\ 10^{-15}$$

$$y =, 0.9849634042, err[, 9,]=, 3.582511780\ 10^{-15}$$

```
>
```