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


> N:=5;

  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]:=3:gamm[1]:=0.0:

  alpha[2]:=0.35:beta[2]:=3:gamm[2]:=0.2:#
  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]:=0.3:gamm[N]:=0.7:
  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;left c[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;left c[cj]:=1;

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

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

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

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


>
>
```

```
>
   uint_of_x:=proc(x) local i, ui;
                              ui:=1;
                             for i from 2 to N do
                             if x>=b[i] then ui:=i fi;
                             od;
                             return ui;
                  end proc;
   #uint_of_x(0.1);uint_of_x(0.3);uint_of_x(0.55);uint_of_x(0.8);


   int_of_x:=proc(x) local i, ui;
                              ui:=1;
                             for i from 2 to N do
                             if x>b[i] then ui:=i fi;
                             od;
                             return ui;
                  end proc;

   #int_of_x(0.1);int_of_x(0.3);int_of_x(0.56);int_of_x(0.87);




   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[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],Tc[3],Tc[4]],x=0..1,thickness=[2,
   1,1,1,1,1,1,1],numpoints=1000);
```

$$N := 5$$

$$\beta_5 := -2.0000000000$$

$$alpha =, \left[\begin{array}{ccccc} 1 & 0.3500000000 & 0.8000000000 & 1 & 0.3000000000 \end{array}\right]$$

$$beta =, \begin{bmatrix} 3 & 3 & -4 & -5 & -2.0000000000 \end{bmatrix}$$

$$gamma =, \begin{bmatrix} 0.0000000000 & 0.2000000000 & 0.1000000000 & 0.0000000000 & 0.7000000000 \end{bmatrix}$$

$$\beta const := 3.4500000000$$

$$b =, \begin{bmatrix} 0 & \frac{1}{3} & 0.4500000000 & 0.6500000000 & 0.8500000000 & 1 \end{bmatrix}$$

$$ag =, \begin{bmatrix} 0 & 1 & -1.8000000000 & -3.2500000000 & -1.7000000000 \end{bmatrix}$$

$$al =, \begin{bmatrix} 0 & 0.3500000000 & -3.6000000000 & -5.2500000000 & -3.0000000000 \end{bmatrix}$$

$$a =, \begin{bmatrix} 0.0000000000 & 0.8000000000 & -2.7000000000 & -4.2500000000 & -2.7000000000 \end{bmatrix}$$

$$gamma =, \begin{bmatrix} 0.0000000000 & 0.2000000000 & 0.1000000000 & 0.0000000000 & 0.7000000000 \end{bmatrix}$$

$$Kc =, 5$$

$$c =, \begin{bmatrix} \frac{1}{3} & 0.4500000000 & 0.4500000000 & 0.6500000000 & 1 & c_6 & c_7 & c_8 & c_9 & c_{10} \end{bmatrix}$$

$$sidec =, \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & sidec_6 & sidec_7 & sidec_8 & sidec_9 & sidec_{10} \end{bmatrix}$$

$$leftc =, \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & leftc_6 & leftc_7 & leftc_8 & leftc_9 & leftc_{10} \end{bmatrix}$$

$$j\_of\_c =, \begin{bmatrix} 2 & 2 & 3 & 3 & 5 & j\_of\_c_6 & j\_of\_c_7 & j\_of\_c_8 & j\_of\_c_9 & j\_of\_c_{10} \end{bmatrix}$$

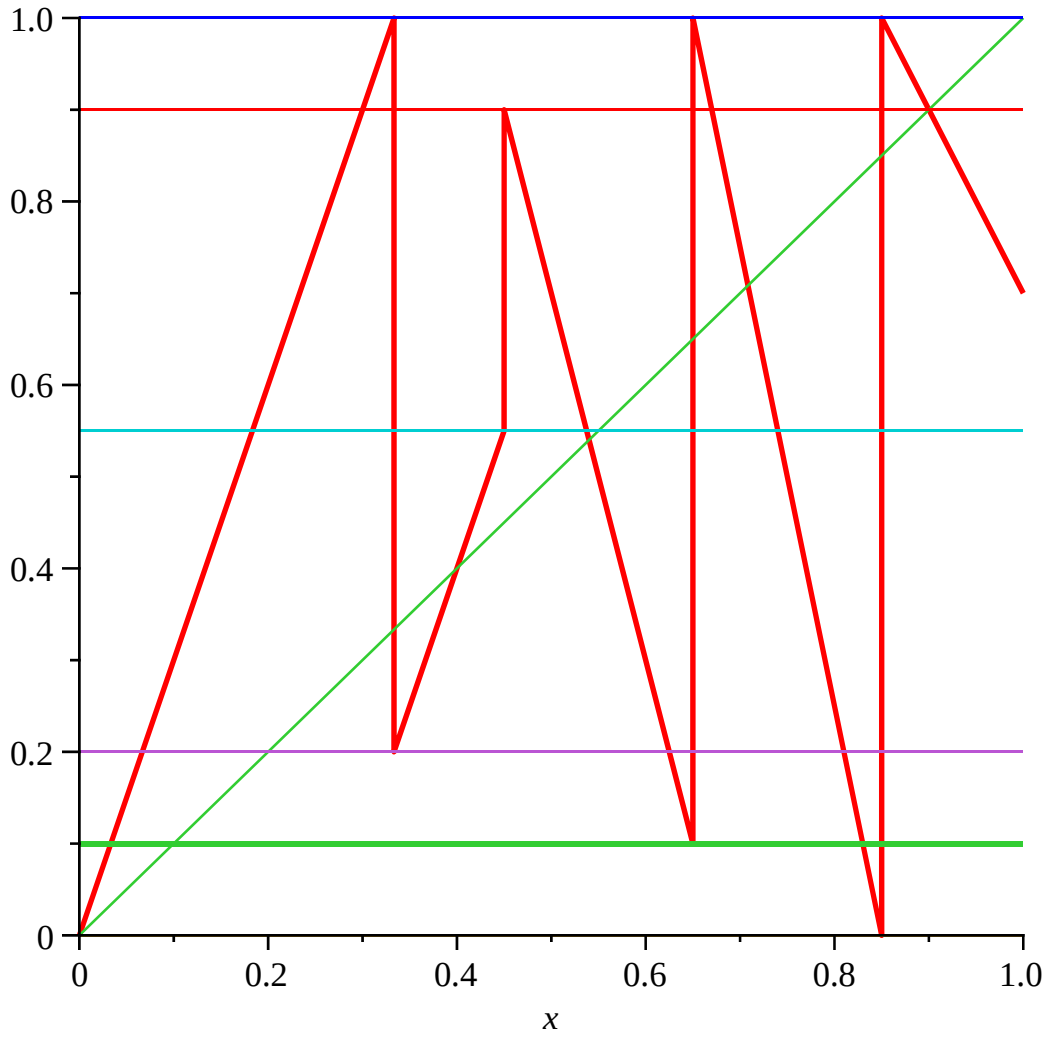
```
uint_of_x := proc(x)
    local i, ui;
    ui := 1; for i from 2 to N do if b[i] <= x then ui := i end if end do; return ui
end proc
int_of_x := proc(x)
    local i, ui;
    ui := 1; for i from 2 to N do if b[i] < x then ui := i end if end do; return ui
end proc
```

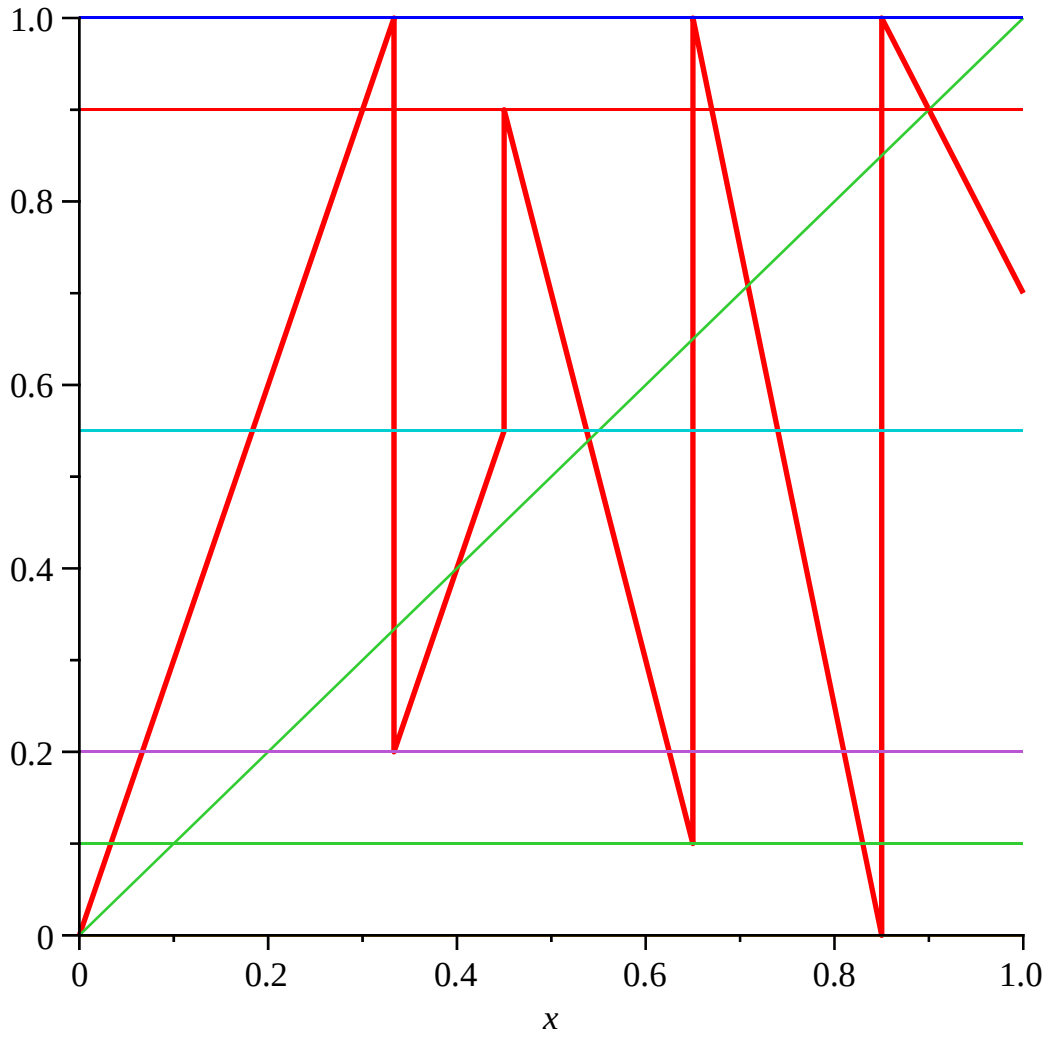
$$uT := x \rightarrow \beta_{uint\_of\_x(x)}\, x - a_{uint\_of\_x(x)}$$

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

$$Tc =,$$

$$\begin{bmatrix} 0.2000000000 & 0.5500000000 & 0.9000000000 & 0.1000000000 & 0.7000000000 & Tc_6 & Tc_7 \end{bmatrix}$$


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




```
> delta1:=(xw,yw)->piecewise(xw<=yw,0,1):
  delta2:=(xw,yw)->piecewise(xw<yw,0,1):
  # symbolic versions of alpha, beta
  alps:=vector(N,[]):
  bets:=vector(N,[]):
  # Procedure producing coefficients in the non simplified
  equations
  Dc:=proc(k,x) local val,ii4;

  if (sidec[k]=0 and k<= Kc) then
              val:=S[k];
             val:=val-sum(delta1(x,Tc[ii4])*SS[k,ii4]*delta1(0.5,
  sidec[ii4])/bets[j_of_c[ii4]],ii4=1..Kc);
             val:=val-sum(delta1(Tc[ii4],x)*SS[k,ii4]*delta1(sidec
  [ii4],0.5)/bets[j_of_c[ii4]],ii4=1..Kc);
             val:=val-delta2(Tc[k],x)/bets[j_of_c[k]];
  end if;
  if (sidec[k]=1 and k<= Kc) then
              val:=S[k];
             val:=val-sum(delta1(x,Tc[ii4])*SS[k,ii4]*delta1(0.5,
  sidec[ii4])/bets[j_of_c[ii4]],ii4=1..Kc);
             val:=val-sum(delta1(Tc[ii4],x)*SS[k,ii4]*delta1(sidec
```

```
[ii4], 0.5) / bets[j_of_c[ii4]], ii4=1..Kc);
            val:=val - delta2(x, Tc[k]) / bets[j_of_c[k]];
    end if;
    if (k = Kc+1) then
             val:=1- sum(1/ bets[ii4], ii4=1..N);
            val:=val +sum(delta1(x, Tc[ii4]) * delta1(0.5, sidec[ii4])
    / bets[j_of_c[ii4]], ii4=1..Kc);
            val:=val +sum(delta1(Tc[ii4], x) * delta1(sidec[ii4], 0.5)
    / bets[j_of_c[ii4]], ii4=1..Kc);

    end if;
            return val;
        end proc;
    for k from 1 to Kc+1 do
    Dc(k, 0.6);
    od;
    #
```

```
Dc := proc(k, x)
    local val, ii4;
    if sidec[k] = 0 and k <= Kc then
        val := S[k];
        val := val - (sum(delta1(x, Tc[ii4]) * SS[k, ii4] * delta1(0.5000000000, sidec[ii4])
        / bets[j_of_c[ii4]], ii4 = 1..Kc));
        val := val - (sum(delta1(Tc[ii4], x) * SS[k, ii4] * delta1(sidec[ii4], 0.5000000000)
        / bets[j_of_c[ii4]], ii4 = 1..Kc));
        val := val - delta2(Tc[k], x) / bets[j_of_c[k]]
    end if;
    if sidec[k] = 1 and k <= Kc then
        val := S[k];
        val := val - (sum(delta1(x, Tc[ii4]) * SS[k, ii4] * delta1(0.5000000000, sidec[ii4])
        / bets[j_of_c[ii4]], ii4 = 1..Kc));
        val := val - (sum(delta1(Tc[ii4], x) * SS[k, ii4] * delta1(sidec[ii4], 0.5000000000)
        / bets[j_of_c[ii4]], ii4 = 1..Kc));
        val := val - delta2(x, Tc[k]) / bets[j_of_c[k]]
    end if;
    if k = Kc + 1 then
        val := 1 - (sum(1 / bets[ii4], ii4 = 1..N));
        val := val + sum(delta1(x, Tc[ii4]) * delta1(0.5000000000, sidec[ii4]) / bets[j_of_c[ii4
        ]], ii4 = 1..Kc);
        val := val + sum(delta1(Tc[ii4], x) * delta1(sidec[ii4], 0.5000000000) / bets[j_of_c[ii4
        ]], ii4 = 1..Kc)
    end if;
```

```
    return val
end proc
```

$$S_1 - \frac{SS_{1,2}}{bets_2} - \frac{SS_{1,5}}{bets_5} - \frac{1}{bets_2}$$

$$S_2 - \frac{SS_{2,2}}{bets_2} - \frac{SS_{2,5}}{bets_5}$$

$$S_3 - \frac{SS_{3,2}}{bets_2} - \frac{SS_{3,5}}{bets_5} - \frac{1}{bets_3}$$

$$S_4 - \frac{SS_{4,2}}{bets_2} - \frac{SS_{4,5}}{bets_5} - \frac{1}{bets_3}$$

$$S_5 - \frac{SS_{5,2}}{bets_2} - \frac{SS_{5,5}}{bets_5}$$

$$1 - \frac{1}{bets_1} - \frac{1}{bets_3} - \frac{1}{bets_4} \qquad \textbf{(1)}$$

```
> # Chosing x's and actually producing the coefficients in the non
  simplified equations
  CofD:=matrix(Kc+1,Kc+1,[]);
  xc:=vector(Kc+1,[]):
  Tc[Kc+1]:=0;
  Tc[Kc+2]:=1;
  Tcl:=convert(Tc,list):
  Tc_s:=sort(Tcl,`<`):
  print(Tc_s);
  for i from 1 to Kc+1 do
  x:=(Tc_s[i]+Tc_s[i+1])/2;
  xc[i]:=x;
  for k from 1 to Kc+1 do
  CofD[i,k]:=Dc(k,x);
  od:
  od:
  print(`xc = `,xc);

  print((CofD));
```

$$CofD := array(1..6, 1..6, [\,])$$

$$Tc_6 := 0$$

$$Tc_7 := 1$$

$$[0, 0.1000000000, 0.2000000000, 0.5500000000, 0.7000000000, 0.9000000000, 1]$$

$$xc = ,$$

$$[0.0500000000, 0.1500000000, 0.3750000000, 0.6250000000, 0.8000000000, 0.9500000000]$$

$$\left[\left[S_1-\frac{SS_{1,1}}{bets_2}-\frac{SS_{1,4}}{bets_3}-\frac{SS_{1,5}}{bets_5},S_2-\frac{SS_{2,1}}{bets_2}-\frac{SS_{2,4}}{bets_3}-\frac{SS_{2,5}}{bets_5}-\frac{1}{bets_2},S_3-\frac{SS_{3,1}}{bets_2}\right.\right. \tag{2}$$

$$-\frac{SS_{3,4}}{bets_3}-\frac{SS_{3,5}}{bets_5}-\frac{1}{bets_3},S_4-\frac{SS_{4,1}}{bets_2}-\frac{SS_{4,4}}{bets_3}-\frac{SS_{4,5}}{bets_5},S_5-\frac{SS_{5,1}}{bets_2}-\frac{SS_{5,4}}{bets_3}$$

$$\left.-\frac{SS_{5,5}}{bets_5},1.0000000000-\frac{1}{bets_1}-\frac{1}{bets_4}\right],$$

$$\left[S_1-\frac{SS_{1,1}}{bets_2}-\frac{SS_{1,5}}{bets_5},S_2-\frac{SS_{2,1}}{bets_2}-\frac{SS_{2,5}}{bets_5}-\frac{1}{bets_2},S_3-\frac{SS_{3,1}}{bets_2}-\frac{SS_{3,5}}{bets_5}-\frac{1}{bets_3},S_4\right.$$

$$-\frac{SS_{4,1}}{bets_2}-\frac{SS_{4,5}}{bets_5}-\frac{1}{bets_3},S_5-\frac{SS_{5,1}}{bets_2}-\frac{SS_{5,5}}{bets_5},1.0000000000-\frac{1}{bets_1}-\frac{1}{bets_3}$$

$$\left.-\frac{1}{bets_4}\right],$$

$$\left[S_1-\frac{SS_{1,5}}{bets_5}-\frac{1}{bets_2},S_2-\frac{SS_{2,5}}{bets_5}-\frac{1}{bets_2},S_3-\frac{SS_{3,5}}{bets_5}-\frac{1}{bets_3},S_4-\frac{SS_{4,5}}{bets_5}\right.$$

$$\left.-\frac{1}{bets_3},S_5-\frac{SS_{5,5}}{bets_5},1.0000000000-\frac{1}{bets_1}-\frac{1}{bets_2}-\frac{1}{bets_3}-\frac{1}{bets_4}\right],$$

$$\left[S_1-\frac{SS_{1,2}}{bets_2}-\frac{SS_{1,5}}{bets_5}-\frac{1}{bets_2},S_2-\frac{SS_{2,2}}{bets_2}-\frac{SS_{2,5}}{bets_5},S_3-\frac{SS_{3,2}}{bets_2}-\frac{SS_{3,5}}{bets_5}-\frac{1}{bets_3},S_4\right.$$

$$-\frac{SS_{4,2}}{bets_2}-\frac{SS_{4,5}}{bets_5}-\frac{1}{bets_3},S_5-\frac{SS_{5,2}}{bets_2}-\frac{SS_{5,5}}{bets_5},1-\frac{1}{bets_1}-\frac{1}{bets_3}-\frac{1}{bets_4}\left.\right],$$

$$\left[S_1-\frac{SS_{1,2}}{bets_2}-\frac{1}{bets_2},S_2-\frac{SS_{2,2}}{bets_2},S_3-\frac{SS_{3,2}}{bets_2}-\frac{1}{bets_3},S_4-\frac{SS_{4,2}}{bets_2}-\frac{1}{bets_3},S_5\right.$$

$$-\frac{SS_{5,2}}{bets_2}-\frac{1}{bets_5},1.0000000000-\frac{1}{bets_1}-\frac{1}{bets_3}-\frac{1}{bets_4}-\frac{1}{bets_5}\left.\right],$$

$$\left[S_1-\frac{SS_{1,2}}{bets_2}-\frac{SS_{1,3}}{bets_3}-\frac{1}{bets_2},S_2-\frac{SS_{2,2}}{bets_2}-\frac{SS_{2,3}}{bets_3},S_3-\frac{SS_{3,2}}{bets_2}-\frac{SS_{3,3}}{bets_3},S_4\right.$$

$$-\frac{SS_{4,2}}{bets_2}-\frac{SS_{4,3}}{bets_3}-\frac{1}{bets_3},S_5-\frac{SS_{5,2}}{bets_2}-\frac{SS_{5,3}}{bets_3}-\frac{1}{bets_5},1.0000000000-\frac{1}{bets_1}$$

$$\left.\left.-\frac{1}{bets_4}-\frac{1}{bets_5}\right]\right]$$

```
> ixc:=vector(Kc,[]):
  for i from 1 to Kc do
  level:=Tc[i];
  for j from 1 to Kc+1 do
  if (sidec[i]=1 and xc[j]<level) then ixc[i]:=j fi;
  if (sidec[i]=0 and xc[j]<=level) then ixc[i]:=j fi;
  od:
  od:
```

```
print(`ixc = `,ixc);

for i from 1 to Kc+1 do
vD[i]:=row(CofD,i);
od:
for i from 1 to Kc do
if sidec[i]=0 then nvD[i]:=evalm(vD[ixc[i]+1]-vD[ixc[i]]) fi;
if sidec[i]=1 then nvD[i]:=evalm(vD[ixc[i]]-vD[ixc[i]+1]) fi;
od:



nvD[Kc+1]:=evalm(vD[3]-nvD[5]);# This to BE ADJUSTED BY HAND
CofD_n:=matrix(Kc+1,Kc+1,[]):
for i from 1 to Kc+1 do
for j from 1 to Kc+1 do
CofD_n[i,j]:=nvD[i][j];
od;od;
evalm(CofD_n);
```

$$ixc = , \left[\begin{array}{ccccc} 2 & 3 & 5 & 1 & 4 \end{array}\right]$$

$$nvD_6 := \left[S_1 - \frac{1}{bets_2}, S_2 - \frac{1}{bets_2}, S_3 - \frac{1}{bets_3}, S_4 - \frac{1}{bets_3}, S_5 - \frac{1}{bets_5}, 1.0000000000 - \frac{1}{bets_1} - \frac{1}{bets_2} - \frac{1}{bets_3} - \frac{1}{bets_4} - \frac{1}{bets_5}\right]$$

$$\left[\left[-\frac{SS_{1,1}}{bets_2} + \frac{1}{bets_2}, -\frac{SS_{2,1}}{bets_2}, -\frac{SS_{3,1}}{bets_2}, -\frac{SS_{4,1}}{bets_2}, -\frac{SS_{5,1}}{bets_2}, \frac{1}{bets_2}\right], \right. \tag{3}$$

$$\left[-\frac{SS_{1,2}}{bets_2}, -\frac{SS_{2,2}}{bets_2} + \frac{1}{bets_2}, -\frac{SS_{3,2}}{bets_2}, -\frac{SS_{4,2}}{bets_2}, -\frac{SS_{5,2}}{bets_2}, \frac{1}{bets_2}\right],$$

$$\left[-\frac{SS_{1,3}}{bets_3}, -\frac{SS_{2,3}}{bets_3}, -\frac{SS_{3,3}}{bets_3} + \frac{1}{bets_3}, -\frac{SS_{4,3}}{bets_3}, -\frac{SS_{5,3}}{bets_3}, \frac{1}{bets_3}\right],$$

$$\left[-\frac{SS_{1,4}}{bets_3}, -\frac{SS_{2,4}}{bets_3}, -\frac{SS_{3,4}}{bets_3}, -\frac{SS_{4,4}}{bets_3} + \frac{1}{bets_3}, -\frac{SS_{5,4}}{bets_3}, \frac{1}{bets_3}\right],$$

$$\left[-\frac{SS_{1,5}}{bets_5}, -\frac{SS_{2,5}}{bets_5}, -\frac{SS_{3,5}}{bets_5}, -\frac{SS_{4,5}}{bets_5}, -\frac{SS_{5,5}}{bets_5} + \frac{1}{bets_5}, \frac{1}{bets_5}\right],$$

$$\left.\left[S_1 - \frac{1}{bets_2}, S_2 - \frac{1}{bets_2}, S_3 - \frac{1}{bets_3}, S_4 - \frac{1}{bets_3}, S_5 - \frac{1}{bets_5}, 1.0000000000 - \frac{1}{bets_1} - \frac{1}{bets_2} - \frac{1}{bets_3} - \frac{1}{bets_4} - \frac{1}{bets_5}\right]\right]$$

```
>
> for i from 1 to Kc+1 do # this part works only after runing the
program to the end and returning
```

```
vD[i]:=row(CofD_n,i);          # Then, RUN IT TWICE
od;                          # This checks if the last equation of
EQS is dependent of the others
for i from 1 to Kc do
if sidec[i]=0 then
                eta[i]:=1-alpha[j_of_c[i]]-gamm[j_of_c[i]];
                    else
                eta[i]:=gamm[j_of_c[i]];
end if;
print(i,eta[i]);
od;
for i from 1 to Kc+1 do
sD[i]:=sum(eta[i56]*vD[i56][i],i56=1..Kc)+vD[Kc+1][i];
od;
for i from 1 to N do
bets[i]:=abs(beta[i]);
alps[i]:=alpha[i];
od:

det(CofD_n);
```

$$vD_1 := \left[-\frac{SS_{1,1}}{bets_2} + \frac{1}{bets_2} \quad -\frac{SS_{2,1}}{bets_2} \quad -\frac{SS_{3,1}}{bets_2} \quad -\frac{SS_{4,1}}{bets_2} \quad -\frac{SS_{5,1}}{bets_2} \quad \frac{1}{bets_2} \right]$$

$$vD_2 := \left[-\frac{SS_{1,2}}{bets_2} \quad -\frac{SS_{2,2}}{bets_2} + \frac{1}{bets_2} \quad -\frac{SS_{3,2}}{bets_2} \quad -\frac{SS_{4,2}}{bets_2} \quad -\frac{SS_{5,2}}{bets_2} \quad \frac{1}{bets_2} \right]$$

$$vD_3 := \left[-\frac{SS_{1,3}}{bets_3} \quad -\frac{SS_{2,3}}{bets_3} \quad -\frac{SS_{3,3}}{bets_3} + \frac{1}{bets_3} \quad -\frac{SS_{4,3}}{bets_3} \quad -\frac{SS_{5,3}}{bets_3} \quad \frac{1}{bets_3} \right]$$

$$vD_4 := \left[-\frac{SS_{1,4}}{bets_3} \quad -\frac{SS_{2,4}}{bets_3} \quad -\frac{SS_{3,4}}{bets_3} \quad -\frac{SS_{4,4}}{bets_3} + \frac{1}{bets_3} \quad -\frac{SS_{5,4}}{bets_3} \quad \frac{1}{bets_3} \right]$$

$$vD_5 := \left[-\frac{SS_{1,5}}{bets_5} \quad -\frac{SS_{2,5}}{bets_5} \quad -\frac{SS_{3,5}}{bets_5} \quad -\frac{SS_{4,5}}{bets_5} \quad -\frac{SS_{5,5}}{bets_5} + \frac{1}{bets_5} \quad \frac{1}{bets_5} \right]$$

$$vD_6 := \left[S_1 - \frac{1}{bets_2}, S_2 - \frac{1}{bets_2}, S_3 - \frac{1}{bets_3}, S_4 - \frac{1}{bets_3}, S_5 - \frac{1}{bets_5}, 1.0000000000 - \frac{1}{bets_1} - \frac{1}{bets_2} - \frac{1}{bets_3} - \frac{1}{bets_4} - \frac{1}{bets_5} \right]$$

1, 0.2000000000

2, 0.4500000000

3, 0.1000000000

4, 0.1000000000

5, 0.7000000000

$$sD_1 := -5.189994312\ 10^{-48}$$

$$sD_2 := -3.400000000\ 10^{-100}$$

$$sD_3 := -1.576460772\ 10^{-46}$$

$$sD_4 := -7.784991468\ 10^{-48}$$

$$sD_5 := -2.\ 10^{-100}$$

$$sD_6 := 1.\ 10^{-100}$$

$$3.440210981\ 10^{-49} \quad \textbf{(4)}$$

```
>
> 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);
  uls_it_x:=evalf(sum(udb[j1],j1=1..NN));
  print(d);
  ls_it_x:=evalf(sum(db[j1],j1=1..NN));
  terr:=xx-uls_it_x;
  err:=xx-ls_it_x;
```

$$Digits := 100$$

$$NN := 50$$

$$xx := 0.3957188605$$

[0.8000000000, 0.8000000000, 0.8000000000, 0.0000000000, −2.7000000000, −2.7000000000, −4.2500000000, −4.2500000000, −4.2500000000, 0.0000000000, 0.8000000000, 0.8000000000, −2.7000000000, −4.2500000000, 0.0000000000,

-4.2500000000, 0.0000000000, -2.7000000000, 0.0000000000, -2.7000000000,
-4.2500000000, 0.8000000000, 0.8000000000, -2.7000000000, -2.7000000000,
-2.7000000000, -2.7000000000, -2.7000000000, -4.2500000000, 0.0000000000,
0.0000000000, -4.2500000000, -2.7000000000, -4.2500000000, 0.0000000000,
0.0000000000, 0.0000000000, -2.7000000000, 0.0000000000, -2.7000000000,
-4.2500000000, -2.7000000000, -4.2500000000, 0.0000000000, -2.7000000000,
-2.7000000000, -2.7000000000, -2.7000000000, -2.7000000000, -2.7000000000]

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

[0.8000000000, 0.8000000000, 0.8000000000, 0.0000000000, -2.7000000000,
-2.7000000000, -4.2500000000, -4.2500000000, -4.2500000000, 0.0000000000,
0.8000000000, 0.8000000000, -2.7000000000, -4.2500000000, 0.0000000000,
-4.2500000000, 0.0000000000, -2.7000000000, 0.0000000000, -2.7000000000,
-4.2500000000, 0.8000000000, 0.8000000000, -2.7000000000, -2.7000000000,
-2.7000000000, -2.7000000000, -2.7000000000, -4.2500000000, 0.0000000000,
0.0000000000, -4.2500000000, -2.7000000000, -4.2500000000, 0.0000000000,
0.0000000000, 0.0000000000, -2.7000000000, 0.0000000000, -2.7000000000,
-4.2500000000, -2.7000000000, -4.2500000000, 0.0000000000, 0.8000000000,
-2.7000000000, -2.7000000000, -4.2500000000, -4.2500000000, -2.7000000000]

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

$$terr := -2.363136101\ 10^{-25}$$

$$err := 4.901319320\ 10^{-27} \quad \textbf{(5)}$$

```
>
>

>
```

```
> 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]-0.5;
        for  n from 1  to NN+1  do


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

```
          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;
#Iteration:
                              xxt:=evalf(beta[intx]*xxt-a[intx]):

varleftc:=varleftc*sign(beta[intx]);
if NN<40 then print(xxt,varleftc) fi;
      od:
Is_it_x:=sum(dcb[i,j1],j1=1..NN);print(`error =`,Is_it_x-c[i]);
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;
```

$$NN := 150$$

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

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

$$xxt := \frac{1}{3}$$

$$bet := 1$$

$$varleftc := 0.5000000000$$

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

$$error =, 9.341989762\ 10^{-47}$$

$$xxt := 0.4500000000$$

$$bet := 1$$

$$varleftc := -0.5000000000$$

$$Is\_it\_x := 0.4500000000$$

$$error =, 2.\ 10^{-100}$$

$$xxt := 0.4500000000$$

$$bet := 1$$

$$varleftc := 0.5000000000$$

$$Is\_it\_x := 0.4500000000$$

$$error =, -3.152921545\ 10^{-46}$$

$$xxt := 0.6500000000$$

$$bet := 1$$

$$varleftc := -0.5000000000$$

$$Is\_it\_x := 0.6500000000$$

$$error =, -1.401298464\ 10^{-46}$$

$$xxt := 1$$

$$bet := 1$$

$$varleftc := -0.5000000000$$

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

$$error =, -1.\ 10^{-100}$$

$$S_1 := 0.5193415638$$

$$S_2 := 0.3024420498$$

$$S_3 := 0.3722222222$$

$$S_4 := 0.4484567901$$

$$S_5 := 0.3609194253 \qquad \textbf{(6)}$$

```
>
>
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(MM,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));
```

$MM =$,

$$\begin{bmatrix} -0.3456790123 & -0.3550438377 & -0.3333333333 & -0.3518518519 & -0.1052105211 \\ -0.3456790123 & -0.3550438377 & -0.3333333333 & -0.3518518519 & -0.1052105211 \\ -0.3456790123 & -0.3550438377 & -0.3333333333 & -0.3518518519 & -0.1052105211 \\ -0.4567901235 & -0.1042187552 & -0.3333333333 & -0.3518518519 & -0.1250125013 \\ -0.4506172840 & -0.0877171050 & -0.1666666667 & -0.3703703704 & -0.5210521052 \end{bmatrix}$$

$eigenvalues\ MM =, -1.4732750507, -6.383782392\ 10^{-16}, -0.0293761540, -0.4043089358,$
0.0000000000

$$\begin{bmatrix} 0.6543209877 & -0.3550438377 & -0.3333333333 & -0.3518518519 & -0.1052105211 \\ -0.3456790123 & 0.6449561623 & -0.3333333333 & -0.3518518519 & -0.1052105211 \\ -0.3456790123 & -0.3550438377 & 0.6666666667 & -0.3518518519 & -0.1052105211 \\ -0.4567901235 & -0.1042187552 & -0.3333333333 & 0.6481481481 & -0.1250125013 \\ -0.4506172840 & -0.0877171050 & -0.1666666667 & -0.3703703704 & 0.4789478948 \end{bmatrix}$$

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

$DD :=$

$$\begin{bmatrix} -2.1333999110 & -2.1333999110 & -2.1333999110 & -1.8850392848 & -2.5101099734 \end{bmatrix}$$

$$3.620694753\ 10^{-46} \qquad (7)$$

```
>
>
```

```
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]);
```

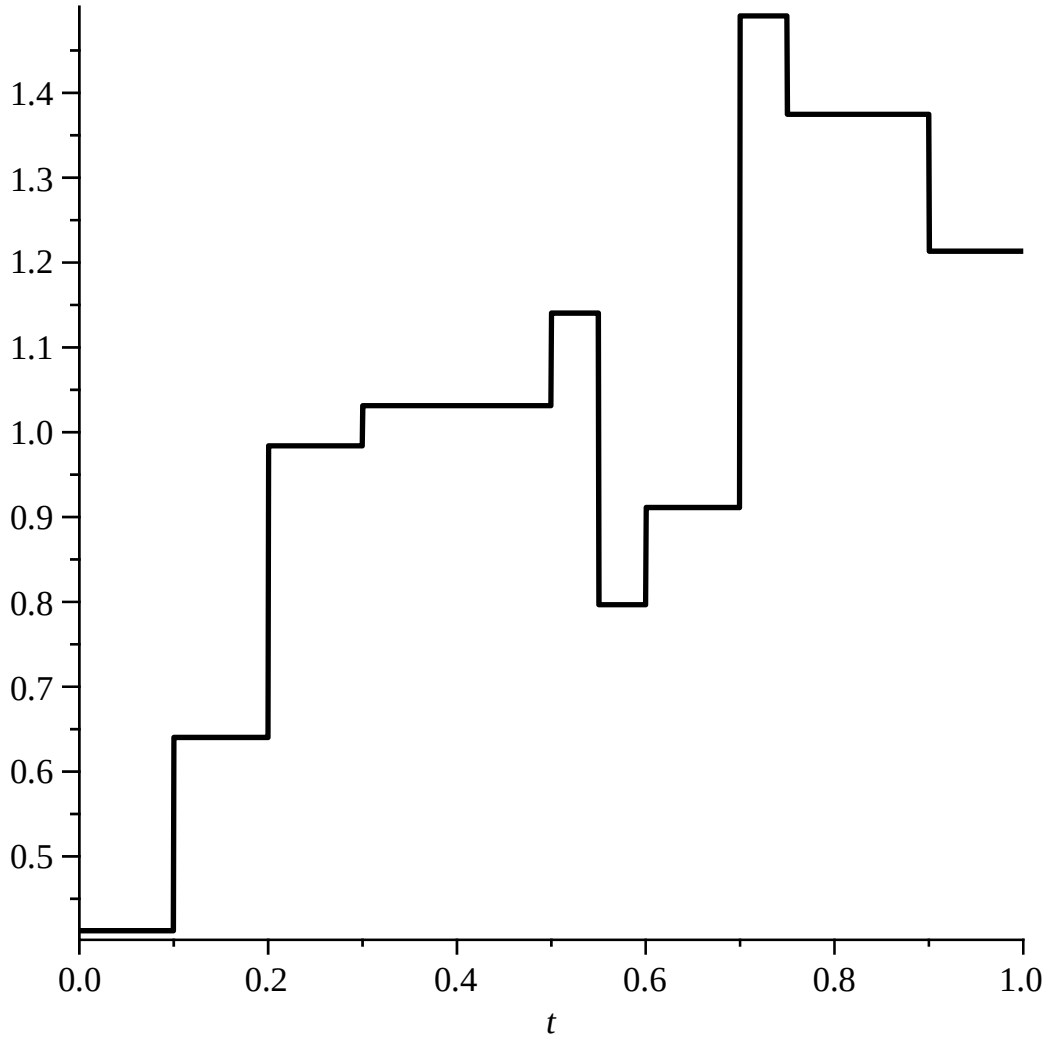
```
                    fi;
                od;
                fi;

                 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,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
```

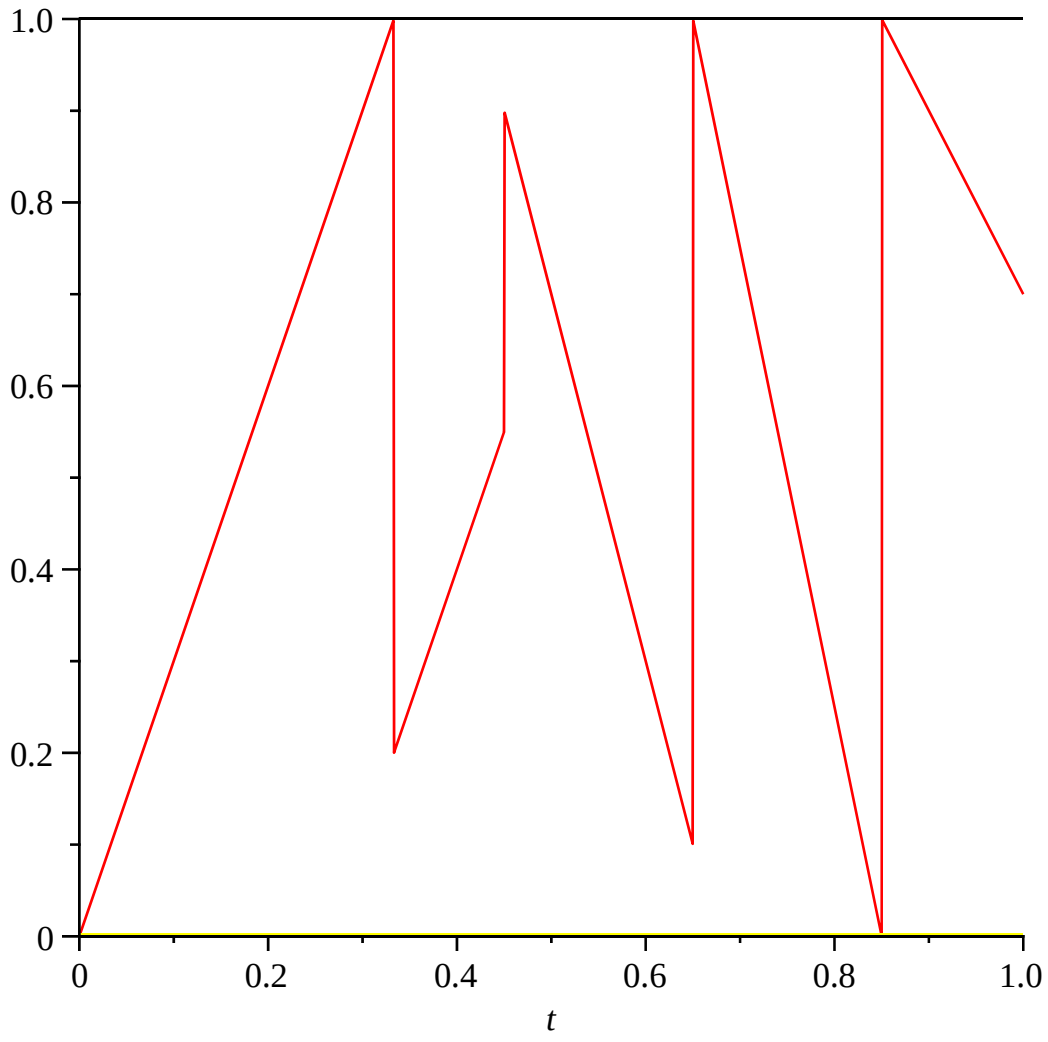
$$NC := -2.0686444935$$

$$NC = , -2.0686444935$$




```
>
>
    #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;
```

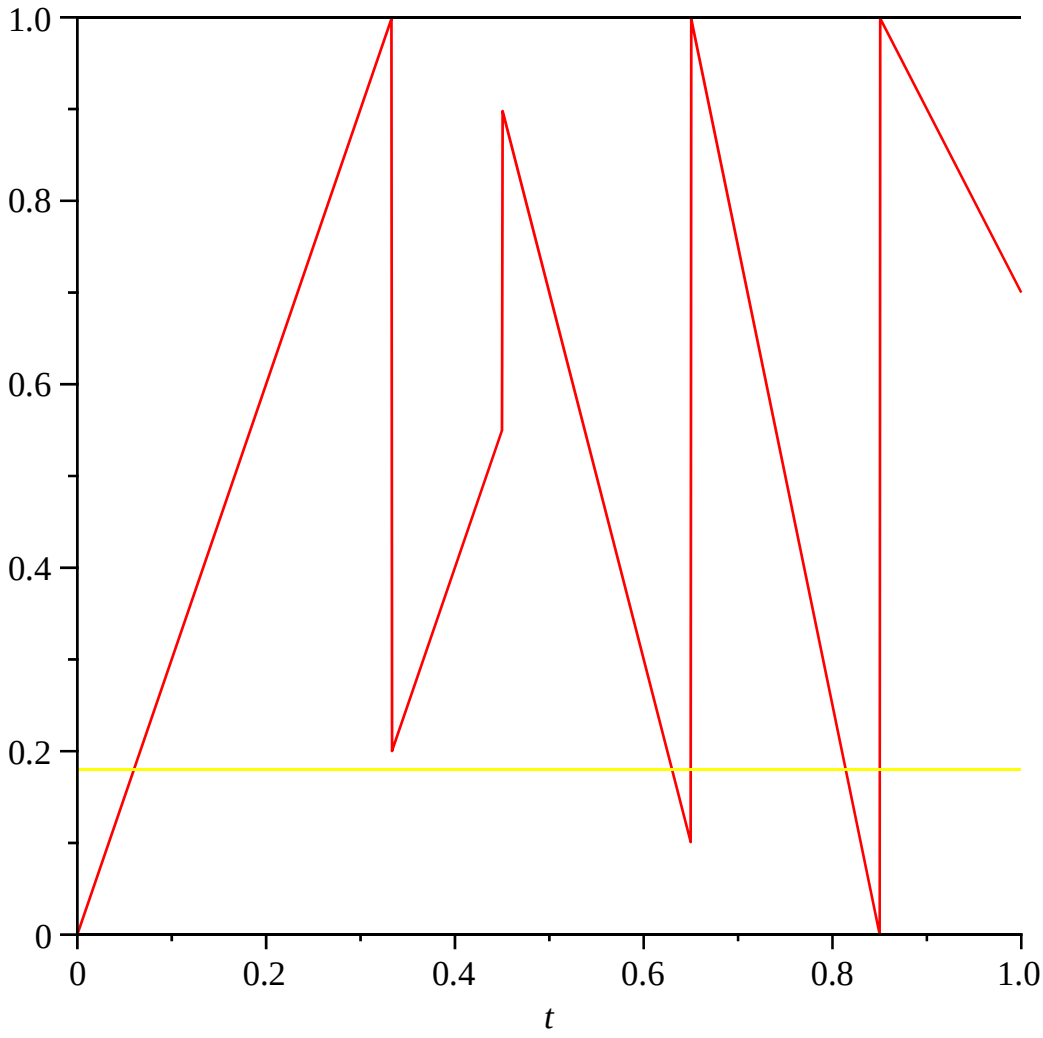


$su := 0$

1

4

$err_0 := 3.962952915\ 10^{-16}$



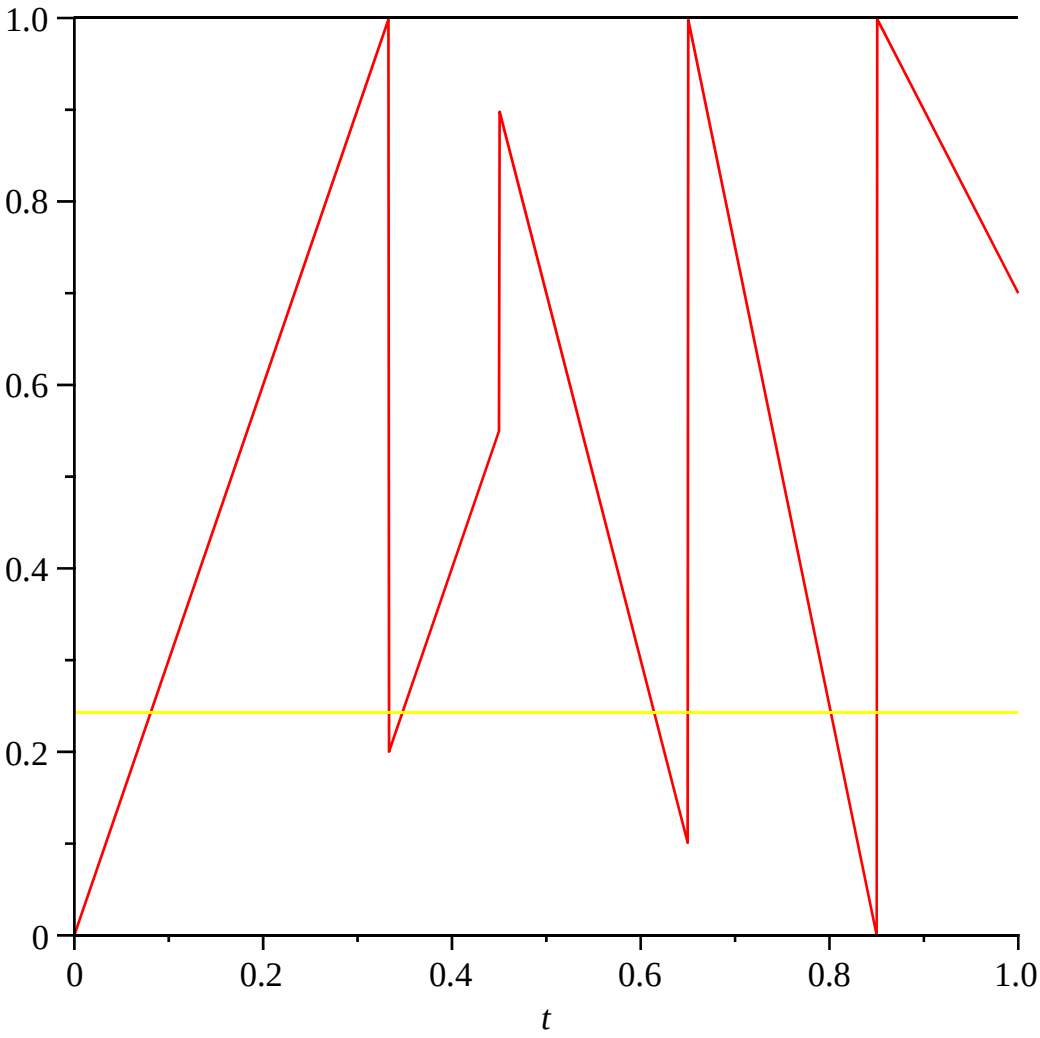
$su := 0$

1

3

4

$err_1 := 1.839942425\ 10^{-16}$



$su := 0$

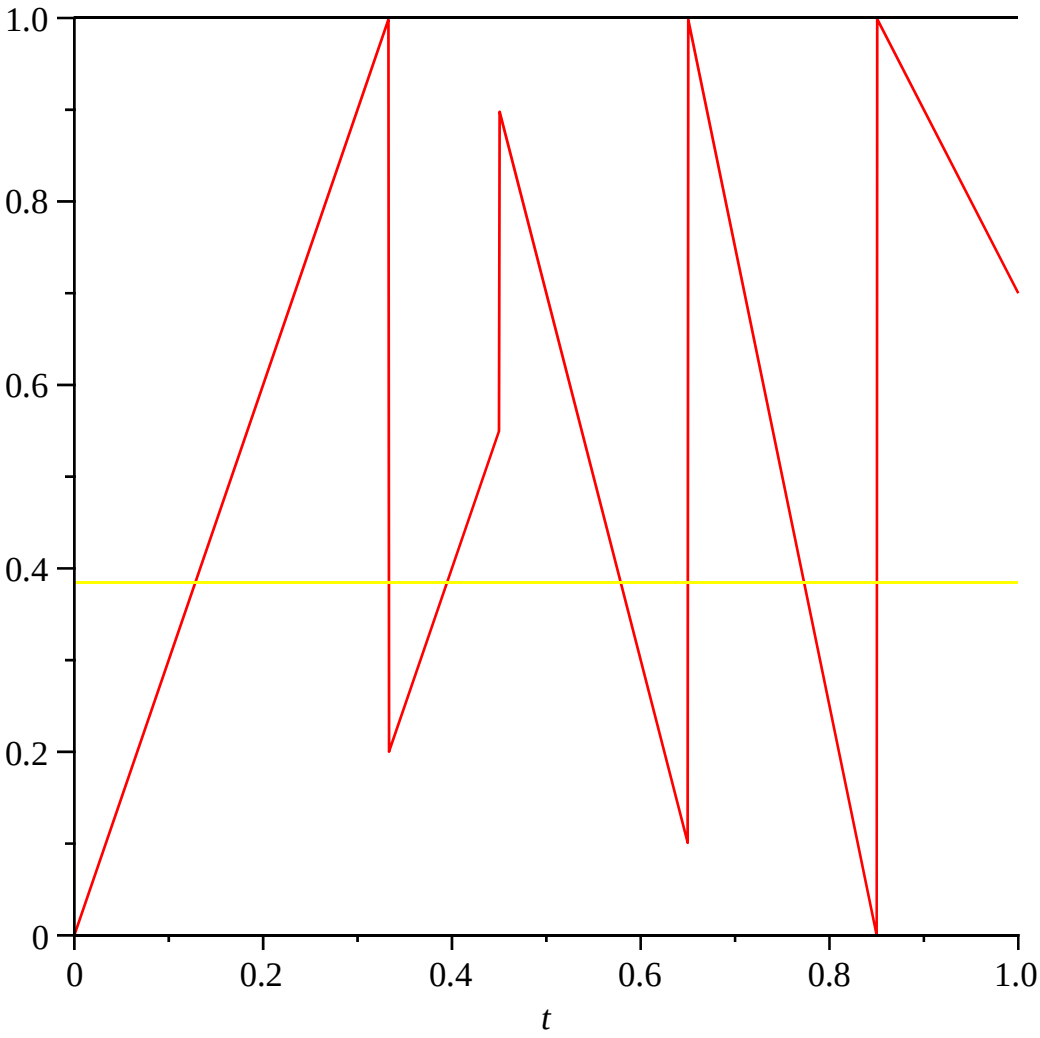
1

2

3

4

$err_2 := -9.907382288\ 10^{-17}$



$su := 0$

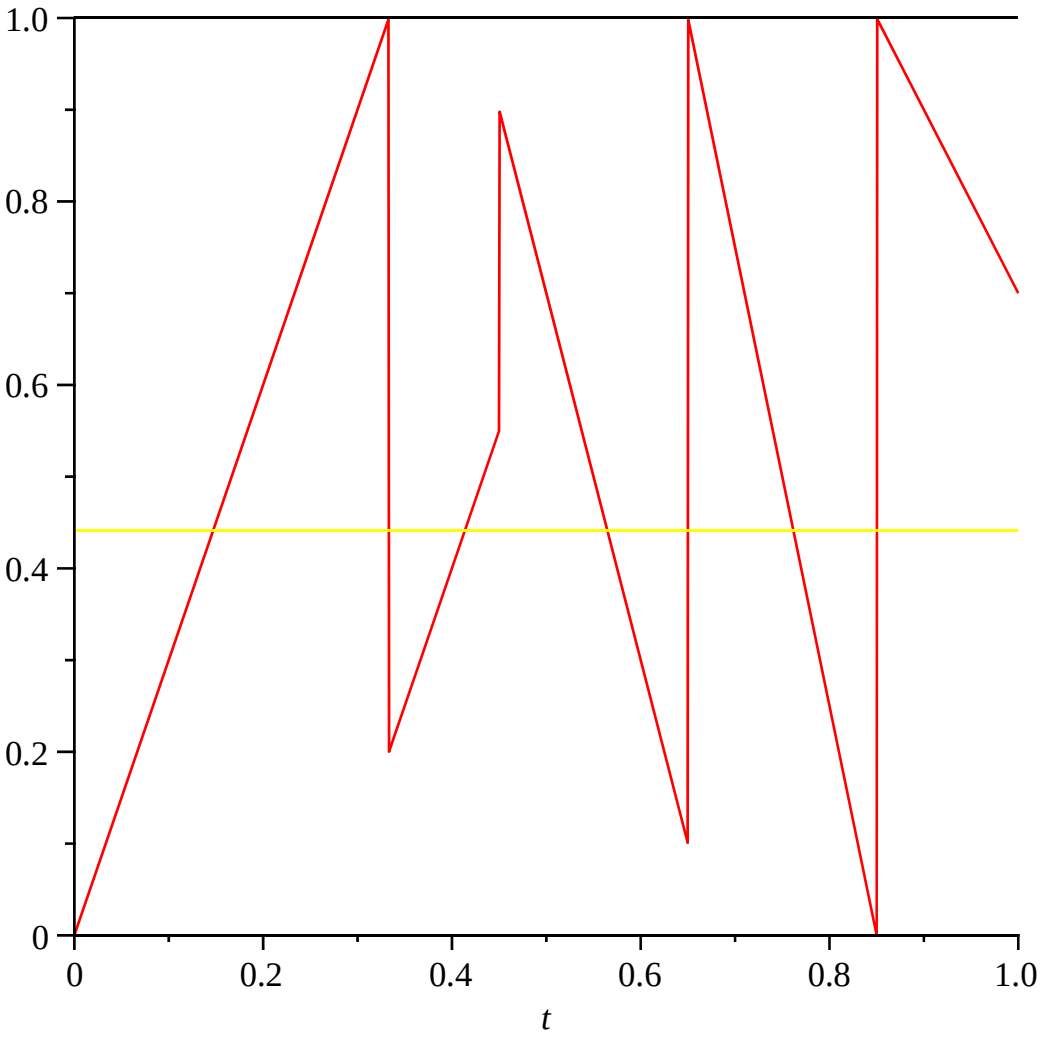
1

2

3

4

$err_3 := -9.907382288\ 10^{-17}$



$su := 0$

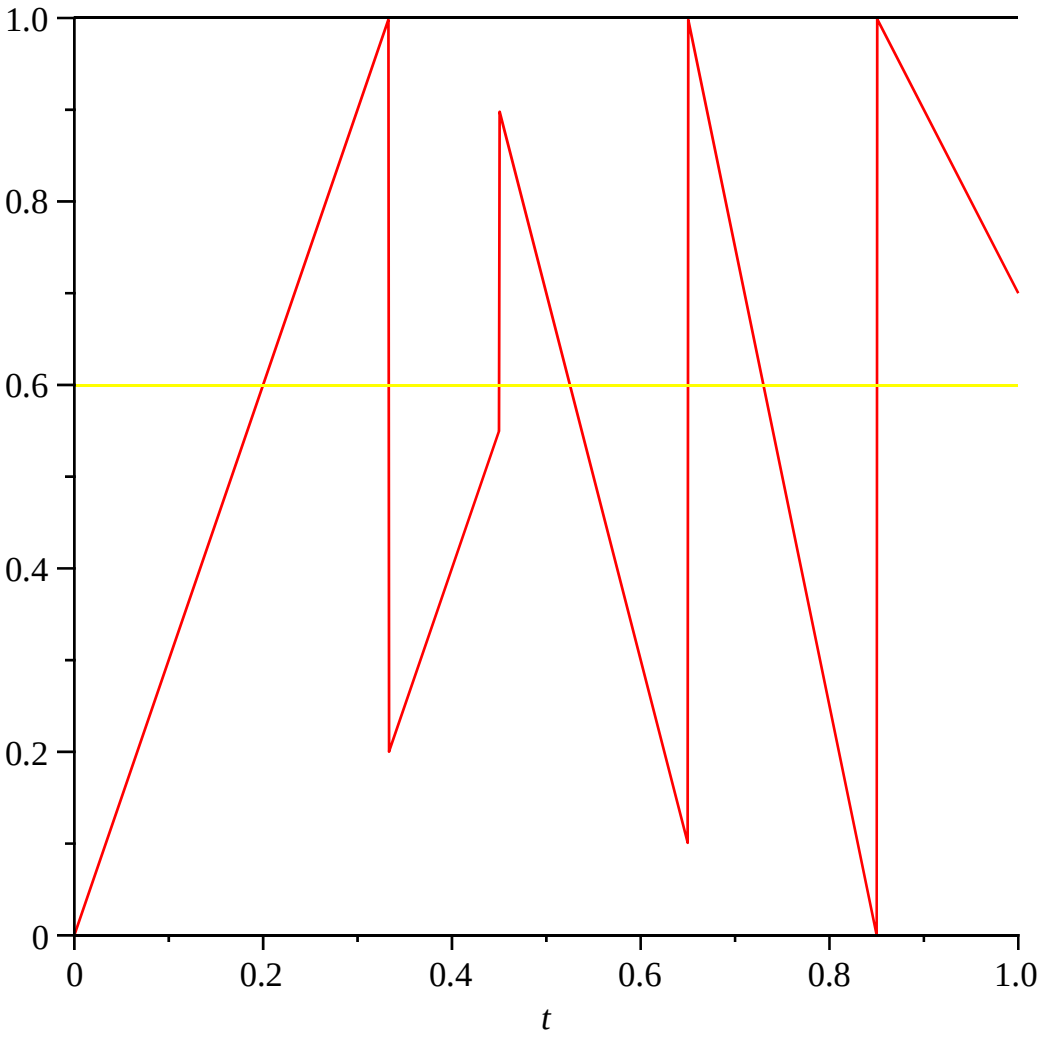
1

2

3

4

$err_4 := -9.907382288\ 10^{-17}$



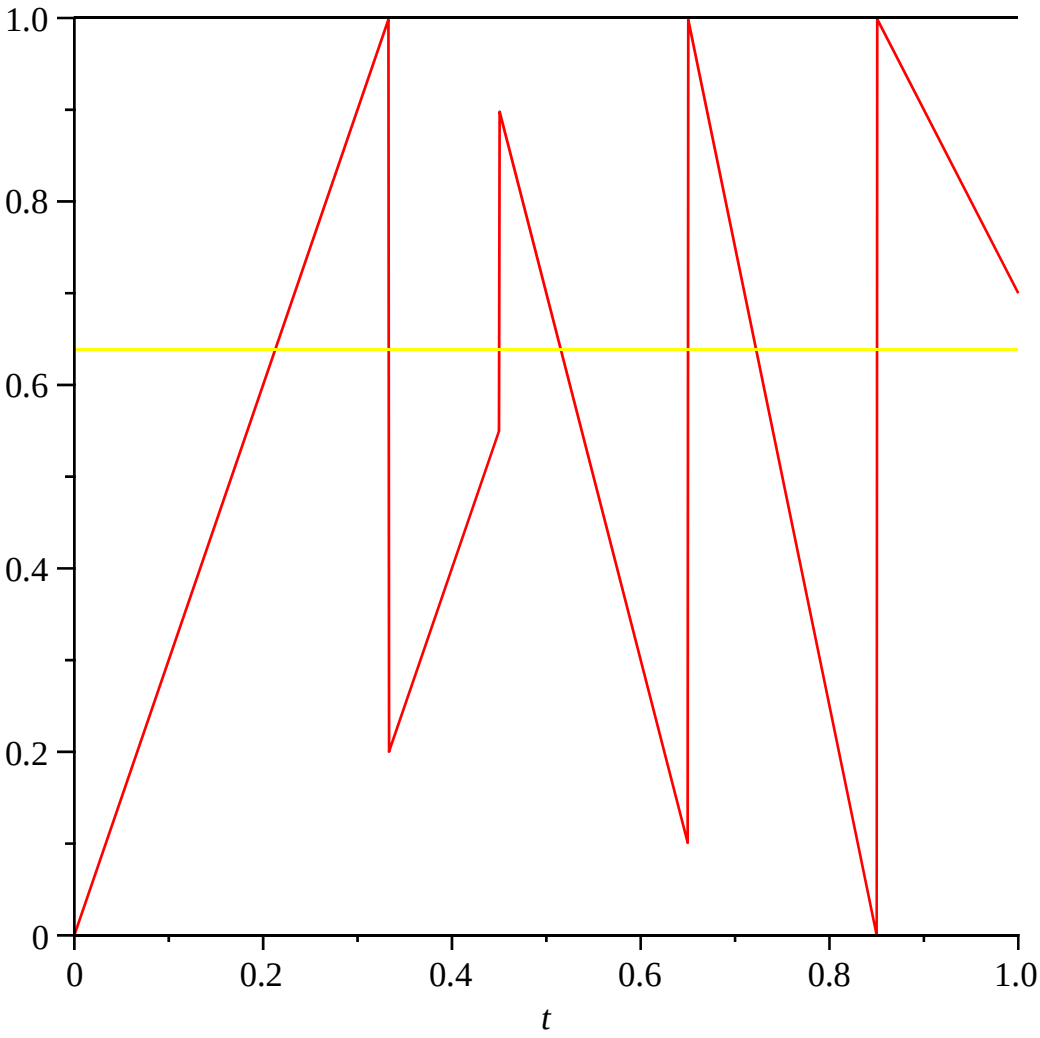
$su := 0$

1

3

4

$err_5 := 1.839942425\ 10^{-16}$



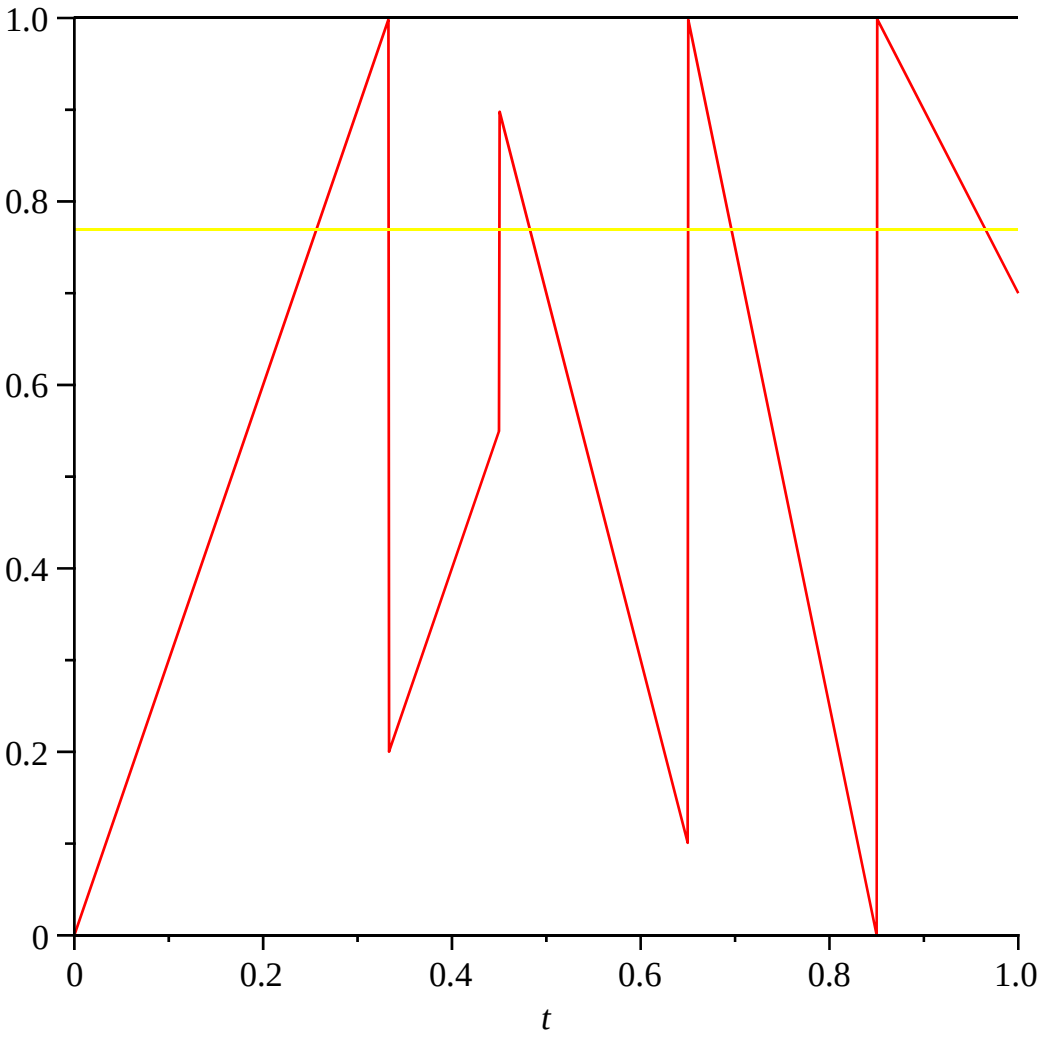
$su := 0$

1

3

4

$err_6 := 1.839942425\ 10^{-16}$



$su := 0$

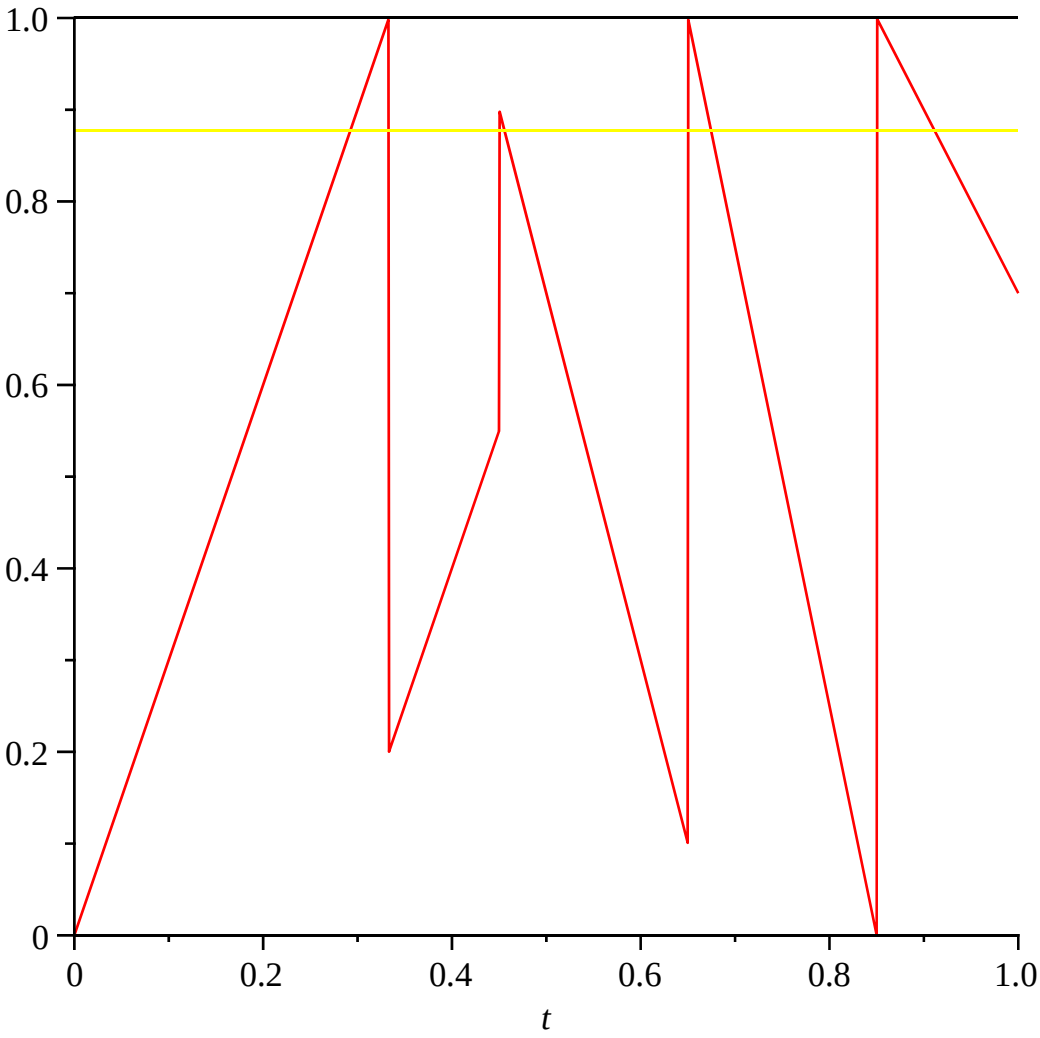
1

3

4

5

$err_7 := -1.915000132\ 10^{-16}$



$su := 0$

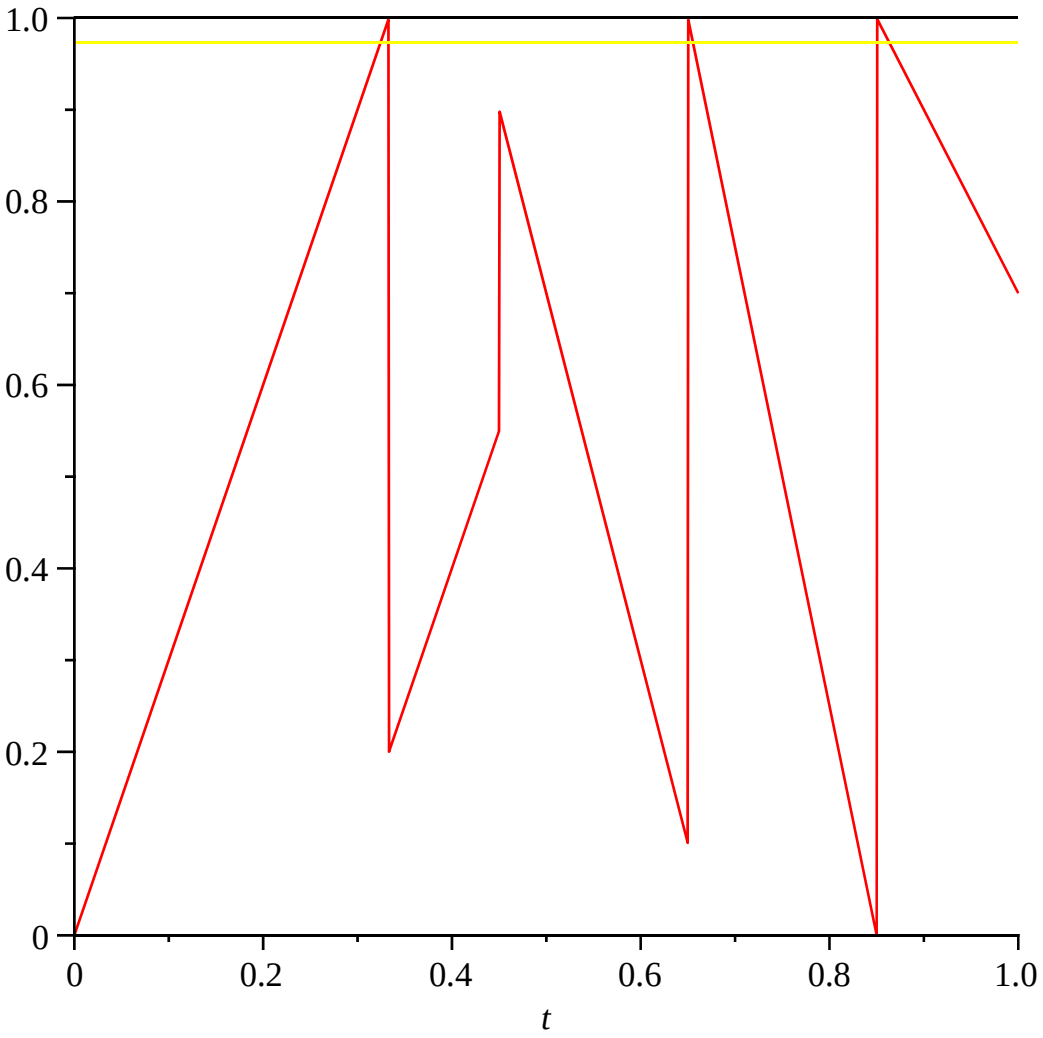
1

3

4

5

$err_8 := -1.915000132\ 10^{-16}$



$su := 0$

1

4

5

$err_9 := -1.265224912\ 10^{-16}$

$y =, 0.0022424170, err[, 0,]=, 3.962952915\ 10^{-16}$

$y =, 0.1800187484, err[, 1,]=, 1.839942425\ 10^{-16}$

$y =, 0.2427552057, err[, 2,]=, -9.907382288\ 10^{-17}$

$y =, 0.3842622684, err[, 3,]=, -9.907382288\ 10^{-17}$

$y =, 0.4412286286, err[, 4,]=, -9.907382288\ 10^{-17}$

$y =, 0.5996417214, err[, 5,]=, 1.839942425\ 10^{-16}$

$y =, 0.6386408307, err[, 6,]=, 1.839942425\ 10^{-16}$

$y =, 0.7694607189, err[, 7,]=, -1.915000132\ 10^{-16}$

$y =, 0.8773012980, err[, 8,]=, -1.915000132\ 10^{-16}$

$y =, 0.9730616293, err[, 9,]=, -1.265224912\ 10^{-16}$

>