http://www.mapleprimes.com/questions/143775-How-To-Find-The-Integral

```
> restart; interface(version);
```

Classic Worksheet Interface, Maple 16.02, Windows, Nov 18 2012, Build ID 788210

Task

```
> sinc:= t -> sin(t)/t; # non-normalized sinc
  #Sinc:= t -> 1/t/Pi*sin(t*Pi); # normalized sinc
  a:= k -> 2^(-k);
  #'sinc(x*a(k))': '%'= combine(%);
```

$$\mathrm{sinc} := t \to \frac{\sin(t)}{t}$$

$$a := k \to 2^{(-k)}$$

```
> `Assertion: `;
  Pi = 'Int(Product(sinc(x*a(k)), k = 0 .. n), x = -infinity .. infinity)',
  n in IN[0];
```

Assertion:

$$\pi = \int_{-\infty}^{\infty} \prod_{k=0}^{n} \mathrm{sinc}(x\, a(k))\, dx,\ n \in \mathrm{IN}_0$$

```
> `some first check: `;
  N:='N':
  'Int(Product(sinc(x*a(k)), k = 0 .. n), x = -infinity .. infinity)':
  combine(%):
  ['seq'( eval(%, n=N), N = 0 .. 6)];
  ``=value(%); N:='N':
```

some first check:

$$\left[\mathrm{seq}\left(\int_{-\infty}^{\infty} \prod_{k=0}^{N} \frac{\sin(x\, 2^{(-k)})\, 2^{k}}{x}\, dx,\ N = 0 .. 6\right)\right]$$

$$= [\pi, \pi, \pi, \pi, \pi, \pi, \pi]$$

```
> assume(k::nonnegint); getassumptions(k);
```

$$\{ k{::}\mathrm{AndProp}(\mathrm{integer}, \mathrm{RealRange}(0, \infty)) \}$$

Idea: use Fourier methods, as in Borwein, David; Borwein, Jonathan M. (2001), "Some remarkable properties of sinc and related integrals", The Ramanujan Journal 5 (1): 73–89, Thm 2 (ii) at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.8186 (for a download to avoid those expensive commercial downloads).

```
>
```

Recall Fourier transformation

```
> with(inttrans);
  NumericEventHandler(invalid_operation =
  `Heaviside/EventHandler`(value_at_zero = 1)):
  'Heaviside(0)': '%'=%;
```

[addtable, fourier, fouriercos, fouriersin, hankel, hilbert, invfourier, invhilbert, invlaplace, invmellin, laplace, mellin, savetable]

$$\mathrm{Heaviside}(0) = 1$$

```
> # notation
  Phi = `Fourier transform`, (Phi@@(-1)) = `inverse Fourier transform`;
  ``;
  "non-unitary convention";
  fourier( f(x),x,t): '%'= convert(%, Int);
```

Φ = Fourier transform, $\Phi^{(-1)}$ = inverse Fourier transform

"non-unitary convention"

$$\mathrm{fourier}(f(x), x, t) = \int_{-\infty}^{\infty} f(x)\, \mathbf{e}^{(-I\, x\, t)}\, dx$$

```
> Phi^2 = -2*Pi*id;
  'fourier(fourier(f(x),x,t),t,x) = 2*Pi*f(-x)'; is(%);
  #'fourier(f(t),t,x) = 2*Pi*invfourier(f(t),t,-x)';
  #convert(%, Int): is(%);
```

$$\Phi^2 = -2\, \pi\, \mathrm{id}$$

$$\mathrm{fourier}(\mathrm{fourier}(f(x), x, t), t, x) = 2\, \pi\, f(-x)$$

true

Rectangles and sinc

Rectangles and sinc functions are Fourier transform pairs

```
> piecewise(a <=t and t <= b, 1, 0):
  rect:= unapply(%, a,b, t);
  ``=rect(a,b, t),
  `        assuming a < b   !!`;
```

$$\mathrm{rect} := (a, b, t) \to \mathrm{piecewise}(a \le t \textbf{ and } t \le b, 1, 0)$$

$$= \begin{cases} 1 & a \le t \textbf{ and } t \le b \\ 0 & \text{otherwise} \end{cases},\quad \text{assuming } a < b\ \ !!$$

```
> 'sinc(x)' ='fourier(rect(-1,1,z), z,x)'/2; is(%);
```

$$\mathrm{sinc}(x) = \frac{1}{2}\,\mathrm{fourier}(\mathrm{rect}(-1, 1, z), z, x)$$

true

```
> s:= (k,z) -> 2^(-1+k)*rect(-2^(-k), 2^(-k), z);
  ``;
  'sinc(x*a(k))' = fourier('s'(k,z), z,x); is(%);
```

$$s := (k, z) \to 2^{(k-1)}\, \mathrm{rect}(-2^{(-k)}, 2^{(-k)}, z)$$

$$\mathrm{sinc}(x\, a(k)) = \mathrm{fourier}(s(k, z), z, x)$$

true

```
>
```

Convolution

```
> # take some care for variables if using the following in a simple way ...

  Conv:=(f,g)-> tau -> Int(f(tau-y)*g(y), y = -infinity .. infinity);
  #Conv(f,g);
```

$$\mathrm{Conv} := (f, g) \to \tau \to \int_{-\infty}^{\infty} f(\tau - y)\, g(y)\, dy$$

That gives a commutative algebra structure with multiplication = convolution. Or in a less educated language: one can manipulate like usual multiplication (a unity is missing)

Some notation (since Maple has no 'big star' free for a usual notation:

```
> 'Conv(f,g)' = `f#g`;
```

$$\mathrm{Conv}(f, g) = f\#g$$

Remember the Convolution Theorem, there are 2 versions: a general one and one for the Fourier transform (statements may vary by a constant factor depending on conventions used for the Fourier transform):

```
> # Convolution Theorem, general version
  #`Reference ???`;
  http://web.eecs.utk.edu/~roberts/WebAppendices/D-ConvProperties.pdf
  "area of the convolution = product of the areas";
  'Int( Conv(f,g)(t), t = -infinity .. infinity) =
    Int(f(t), t = -infinity .. infinity) * Int( g(t), t = -infinity ..
  infinity)';

  'Int( `(f#g)`(t), t = -infinity .. infinity) =
    Int(f(t), t = -infinity .. infinity) * Int( g(t), t = -infinity ..
  infinity)';
```

"area of the convolution = product of the areas"

$$\int_{-\infty}^{\infty} \mathrm{Conv}(f, g)(t)\, dt = \int_{-\infty}^{\infty} f(t)\, dt \int_{-\infty}^{\infty} g(t)\, dt$$

$$\int_{-\infty}^{\infty} (f\#g)(t)\, dt = \int_{-\infty}^{\infty} f(t)\, dt \int_{-\infty}^{\infty} g(t)\, dt$$

```
> # Convolution Theorem, version for Fourier transforms

  'Phi( Conv(f,g ) ) = Phi(f)* Phi(g)';
  Phi(`f#g`) = Phi(f)*Phi(g);
```

$$\Phi(\mathrm{Conv}(f, g)) = \Phi(f)\,\Phi(g)$$

$$\Phi(f\#g) = \Phi(f)\,\Phi(g)$$

As corollaries the convolution can be expressed by the Fourier transform and any convolution product commutes with the Fourier transform

```
> # Corollary: convolution computed via Fourier transforms
  `f#g` = (Phi@@(-1))( Phi(f)*Phi(g) );
  #`` = (Phi@@(-1))( F*G ); ``;  F=Phi(f), G=Phi(g), Phi = `Fourier
  transform`;
```

$$f\#g = (\Phi^{(-1)})(\Phi(f)\,\Phi(g))$$

```
> # Corollary: iterated convolution products
  Phi(`f#g#h ...`) =  Phi(f)*Phi(g)*Phi(h)*` ...`;
  #Phi(f) = 'sinc(x*a(k))', `...`;
```

$$\Phi(f\#g\#h\ \ldots) = \Phi(f)\,\Phi(g)\,\Phi(h)\ \ldots$$

Proofs

```
> # Maple knows the identity:
  fourier(f(t),t,z)*fourier(g(t),t,z):
  'invfourier'(%,z,t);
  ``=subs(int=Int, %): subs(_U1 = eta, %):
  combine(%);
  Change(%, eta=-y, y);
  ``='Conv'(g,f )(t);
  #``= `(g#f)(t)`;
```

$$\mathrm{invfourier}(\mathrm{fourier}(f(t), t, z)\,\mathrm{fourier}(g(t), t, z), z, t)$$

$$= \int_{-\infty}^{\infty} f(-\eta)\, g(t+\eta)\, d\eta$$

$$= \int_{-\infty}^{\infty} f(y)\, g(t-y)\, dy$$

$$= \mathrm{Conv}(g, f)(t)$$

```
> `(f#g) # h`;
  ``= (Phi@@(-1))(  Phi(`f#g`) * Phi(h) );
  ``= (Phi@@(-1))( (Phi(f)*Phi(g)) * Phi(h) );
```

$$(f\#g)\ \#\ h$$

$$= (\Phi^{(-1)})(\Phi(f\#g)\,\Phi(h))$$

$$= (\Phi^{(-1)})(\Phi(f)\,\Phi(g)\,\Phi(h))$$

```
>
```

Convolutions for Rectangles

The convolution of 2 rectangles can be described completely

```
> # k = 0 and 1
  'Trapez' = `( s(0, . ) # s(1, . ) )`(t);
  exampleData:={a1=-1, b1=1, h1=1/2,    # s(0,t);
                a2=-1/2,b2=1/2,h2=1};  # s(1,t);
```

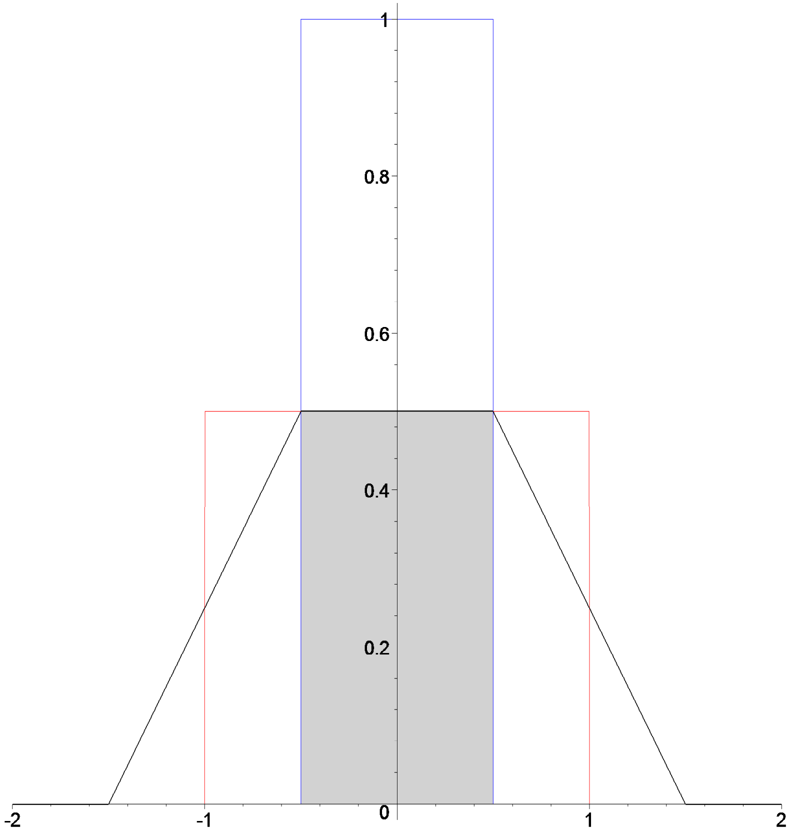
$$\mathrm{Trapez} = (\ s(0, .\ ) \# s(1, .\ )\ )(t)$$

$$\mathrm{exampleData} := \left\{ a1 = -1, a2 = \frac{-1}{2}, b1 = 1, b2 = \frac{1}{2}, h1 = \frac{1}{2}, h2 = 1 \right\}$$

```
> Trapez:='Conv'( t -> s(0,t), t -> s(1, t) )(t);
  Trapez:=value(%);
  plot(Trapez, t=-2 .. 2, thickness=2, color=black):
  plot([s(0,t), s(1,t)], t=-2 .. 2, color=[red, blue]):
  plottools:-rectangle([-1/2,0], [1/2,1/2], color="LightGrey"):
  plots:-display(%%%, %%,%);
```

$$\mathrm{Trapez} := \mathrm{Conv}(t \to s(0, t), t \to s(1, t))(t)$$

$$\mathrm{Trapez} := \begin{cases} 0 & t < \frac{-3}{2} \\ \frac{3}{4} + \frac{t}{2} & t < \frac{-1}{2} \\ \frac{1}{2} & t < \frac{1}{2} \\ \frac{3}{4} - \frac{t}{2} & t < \frac{3}{2} \\ 0 & \frac{3}{2} \le t \end{cases}$$




```
> `Convolution = Trapez = Wing + Torso and extending by Zero`;
  `Torso : = constant values within the support, Wings : = the difference`;
```

Convolution = Trapez = Wing + Torso and extending by Zero

Torso : = constant values within the support, Wings : = the difference

Let us view at it as something with a rectangle as a 'torso' and the remaining wings, which helps to study iterated convolution products

```
> [`Support of convolution`, `starting` =a1+a2 , `ending`=b1+b2];
  eval(%, exampleData);
```

$$[\text{Support of convolution}, \text{starting} = a1 + a2, \text{ending} = b1 + b2]$$

$$\left[\text{Support of convolution}, \text{starting} = \frac{-3}{2}, \text{ending} = \frac{3}{2}\right]$$

This holds for any convolution of functions living on intervals [a1 ... b1] and [a2... b2] (use the definition of a convolution, perhaps in any book).

The next is for rectangles only (I have no citation and what I have seen is without 'min' and 'max', which would make it false in non-symmetric situation). And in our situation of symmetric and "shrinking rectangles" the formula becomes simple:

```
> # holds for rectangles
  [Torso, `starting` = min(b1+a2, a1+b2) , `ending`=max(b1+a2, a1+b2)];
  eval(%, [a1=-b1,a2=-b2]) assuming b2 < b1;
  eval(%, exampleData);
```

$$[\text{Torso}, \text{starting} = \min(a1 + b2, b1 + a2), \text{ending} = \max(a1 + b2, b1 + a2)]$$

$$[\text{Torso}, \text{starting} = -b1 + b2, \text{ending} = b1 - b2]$$

$$\left[\text{Torso}, \text{starting} = \frac{-1}{2}, \text{ending} = \frac{1}{2}\right]$$

Thus one need the height of the torso for a complete description. That follows through the area of the Trapezoid and due to the Convolution Theorem that is the product of the areas of the 2 according rectangles.

```
> # area of trapez = height * average of bottom and top
  area = height* (`length bottom`+`length top`)/2;
  isolate(%, height): simplify(%);
  ``=rhs(%):
  eval(%, area= 1);
  eval(%, `length bottom` = 2* 3/2); # look at the result for 'Trapez'
  eval(%, `length top` = 2* 1/2);    # look at the result for 'Trapez'
```

$$\text{area} = \frac{\text{height}\,(\text{length bottom} + \text{length top})}{2}$$

$$\text{height} = \frac{2\,\text{area}}{\text{length bottom} + \text{length top}}$$

$$= \frac{2}{\text{length bottom} + \text{length top}}$$

$$= \frac{2}{3 + \text{length top}}$$

$$= \frac{1}{2}$$

```
> `height` = 2*area/(abs(-b1-b2+a1+a2)+abs(a1+b2-b1-a2));
  ``=rhs(%):
  eval(%, [a1=-b1, a2=-b2]): simplify(%); # symmetric w.r.t. 0
  eval(%, area= 1);
  eval(%, exampleData);
```

$$\text{height} = \frac{2\,\text{area}}{|-b1 - b2 + a1 + a2| + |-b1 - a2 + a1 + b2|}$$

$$= \frac{\text{area}}{|b1 + b2| + |b1 - b2|}$$

$$= \frac{1}{|b1 + b2| + |b1 - b2|}$$

$$= \frac{1}{2}$$

It was used, that in our example the areas equal 1, thus we got: the height equal 1/2. In any case.

```
> Int(`(f#g)`(t), t= -infinity .. infinity) =
    Int(f(t), t= -infinity .. infinity)*Int(g(t), t= -infinity ..
  infinity);
  ``;
  'Int(s(k,t), t= -infinity .. infinity)': '%' = simplify( value(%) );
```

$$\int_{-\infty}^{\infty} (f\#g)(t)\, dt = \int_{-\infty}^{\infty} f(t)\, dt \int_{-\infty}^{\infty} g(t)\, dt$$

$$\int_{-\infty}^{\infty} s(k, t)\, dt = 1$$

```
>
```

Iterated convolution of those rectangles

Define (by recursion) an iterated convolution product

```
> 'S'(n,t) = `#`('s'(j,t), j=0..n);
  ``;
  S:=proc(n)
  option remember;
  local fct;
  fct:= unapply(thisproc(n-1), t);
  Conv(fct, 'x -> s(n,x)')(t); #return(%);
  #subs(Int = int, %)(t);
  value(%); # assuming n::nonnegint;
  simplify(%); # assuming n::nonnegint;
  end proc;
  S(0):=s(0,t):
```

$$S(n, t) = \#(s(j, t), j = 0 .. n)$$

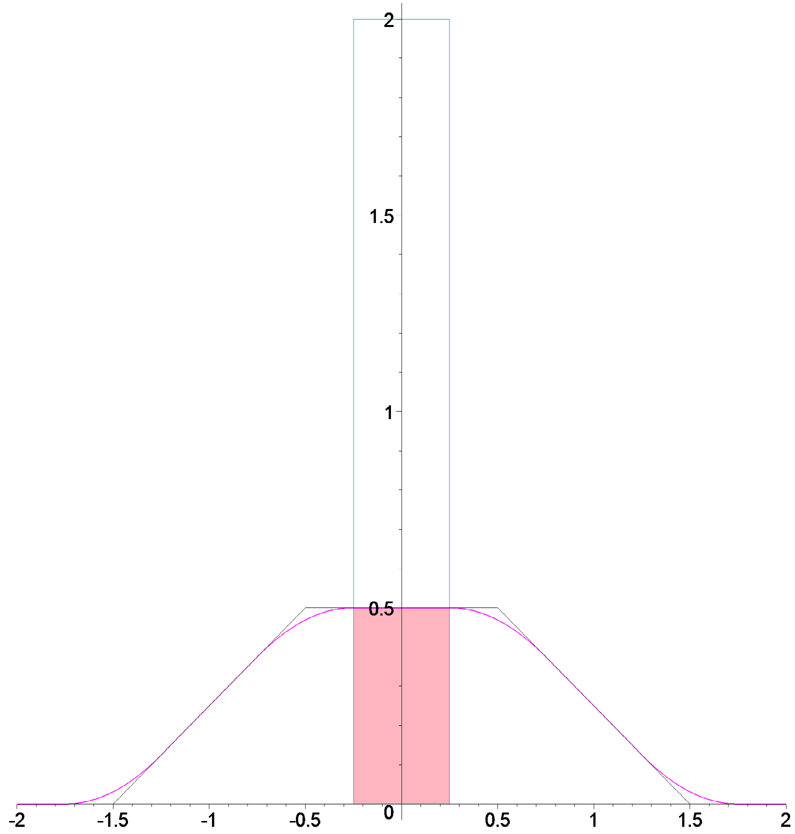
S := **proc**(n)
local fct;
option remember;
fct := unapply(thisproc(n − 1), t); Conv(fct, 'x → s(n, x)')(t); value(%); simplify(%)
end proc

Now - as in the example - decompose into wings and a torso (= the 'center', a rectangle)

```
> 'S'(n,t) = Wing(n,t) + Torso(n,t);
  `` = LeftWing(n,t) + Torso(n,t) + RightWing(n,t);
  LeftWing(n,t) =  RightWing(n,-t);
```

$$S(n, t) = \mathrm{Wing}(n, t) + \mathrm{Torso}(n, t)$$
$$= \mathrm{LeftWing}(n, t) + \mathrm{Torso}(n, t) + \mathrm{RightWing}(n, t)$$
$$\mathrm{LeftWing}(n, t) = \mathrm{RightWing}(n, -t)$$

```
> theRange:= -2 .. 2:
  #p01:=plot([s(0,t), s(1,t)], t=theRange, color=[red, blue,
  "DarkMagenta"]):
  pS1:= plot([S(1), s(2,t)], t=theRange, color=[black, "DarkCyan"],
  tickmarks=[8, 4]):
  pS2:= plot(S(2), t=theRange, thickness=2, color=magenta):
  #pT1:= plot(s(1,t)/2, t=-1/2 .. 1/2, filled=[color="Wheat",
  transparency=0.1]):
  #pT2:= plot(s(2,t)/4, t=-1/4 .. 1/4, filled=[color="LightPink",
  transparency=0.9]):
  pT2:=plottools:-rectangle([-2^(-2),0], [2^(-2),1/2], color="LightPink"):
  plots:-display( pS1,pS2,pT2);
```



The pink curve is the convolution of the trapez (= 2-fold rectangles) with the next rectangle (the small one reaching the value 2) and the torso of this 3-fold convolution is the inner rectangle, given in pink and having height = 1/2 as well.

After studying some iterations (say 3 or 4) one can set up the following assertions about the wings and the torsi (check for k=2, the above plot)

```
> Support(RightWing(k,t)) =  2^(-k) .. Sum(2^(-j), j=0 .. k); value(%);
  eval(%, k=2), k=2;
```

$$\mathrm{Support}(\mathrm{RightWing}(k, t)) = 2^{(-k)} .. \left(\sum_{j=0}^{k} 2^{(-j)} \right)$$

$$\mathrm{Support}(\mathrm{RightWing}(k, t)) = 2^{(-k)} .. -2^{(-k)} + 2$$

$$\mathrm{Support}(\mathrm{RightWing}(2, t)) = \frac{1}{4} .. \frac{7}{4}, k = 2$$

```
> Torso(k,t) = 's(k+1,t)/2^(k+1)';
  ``=simplify(rhs(%)) assuming n::posint;
  ``= 1/2 *'rect'(2^(-k-1), 2^(-k-1), t);
  eval(%, k=2), k=2;
```

$$\mathrm{Torso}(k,t) = \frac{s(k+1,t)}{2^{(k+1)}}$$

$$= \frac{1}{2}\left(\begin{cases} 1 & -2^{(-k-1)} \le t \text{ and } t \le 2^{(-k-1)} \\ 0 & \text{otherwise} \end{cases}\right)$$

$$= \frac{1}{2}\,\mathrm{rect}(2^{(-k-1)}, 2^{(-k-1)}, t)$$

$$= \frac{1}{2}\left(\begin{cases} 1 & \frac{1}{8} \le t \text{ and } t \le \frac{1}{8} \\ 0 & \text{otherwise} \end{cases}\right), k = 2$$

Do some check at least for low values (the procedure is *very* slow)

```
> N:=4;

  'S(N) = 1/2*rect(-2^(-N), 2^(-N),t)'; %:
  simplify(%) assuming -2^(-N) <= t, t < 2^(-N):

  is(%), `assuming ( -2^(-N) <= t, t < 2^(-N) )`;
  #plot(%, t = -2^(-N+1/8) .. 2^(-N+1/8));
```

$$N := 4$$

$$S(N) = \frac{1}{2}\,\mathrm{rect}(-2^{(-N)}, 2^{(-N)}, t)$$

true, assuming (-2^(-N) <= t, t < 2^(-N))

Ingredience for a proof

One uses the formulae about the supports already used in the example above

```
> `RightWing # s(k+1,t)`;
  [`Support`(`RightWing # s(k+1,t)`), `starting` =a1+a2 ,
  `ending`=b1+b2];
  eval(%, [a1=2^(-k), b1 = 2 - 2^(-k), a2=-2^(-k-1), b2 = 2^(-k-1)]):
  simplify(%);
  #eval(%, k=N);
```

RightWing # s(k+1,t)

[Support(RightWing # s(k+1,t)), starting = a1 + a2, ending = b1 + b2]

[Support(RightWing # s(k+1,t)), starting = $2^{(-k-1)}$, ending = $-2^{(-k-1)} + 2$]

```
> `Torso # s(k+1,t)`;
  [`Support`(`Torso # s(k+1,t)`), `starting` =a1+a2 , `ending`=b1+b2];
  eval(%, [a1=-2^(-k), b1 = 2^(-k), a2=-2^(-k-1), b2 = 2^(-k-1)]):
  simplify(%);
  #eval(%, k=N);
```

Torso # s(k+1,t)

[Support(Torso # s(k+1,t)), starting = a1 + a2, ending = b1 + b2]

[Support(Torso # s(k+1,t)), starting = $-3\,2^{(-k-1)}$, ending = $3\,2^{(-k-1)}$]

```
> `Torso # s(k+1,t)`;
  [`new Torso`=`Torso`(`Torso # s(k+1,t)`), `starting` = min(b1+a2,
  a1+b2) , `ending`=max(b1+a2, a1+b2)];
  eval(%, [a1=-2^(-k), b1 = 2^(-k), a2=-2^(-k-1), b2 = 2^(-k-1)]):
  simplify(%);
  #eval(%, k=N);
  #'N': '%'=%;
```

Torso # s(k+1,t)

[new Torso = Torso(Torso # s(k+1,t)), starting = min(a1 + b2, b1 + a2),
ending = max(a1 + b2, b1 + a2)]

[new Torso = Torso(Torso # s(k+1,t)), starting = $-2^{(-k-1)}$, ending = $2^{(-k-1)}$]

```
> `height of new Torso`;
  `area of old torso` = `width*height`, width = 2*2^(-k), height = 1/2;
  ``=2^(-k);
  `area of s(k+1,t)` = 1;
  `area of new torso` = `product of both`;
  ``=2^(-k);
  ``;
  `height` = 2*area/(abs(-b1-b2+a1+a2)+abs(a1+b2-b1-a2));
  ``=rhs(%):
  eval(%, [a1=-2^(-k), b1 = 2^(-k), a2=-2^(-k-1), b2 = 2^(-k-1)]):
  simplify(%);
  eval(%, area = 2^(-k)): simplify(%);
  #eval(%, k=N);
```

height of new Torso

area of old torso = width*height, width = $2\,2^{(-k)}$, height = $\frac{1}{2}$

$$= 2^{(-k)}$$

area of s(k+1,t) = 1

area of new torso = product of both

$$= 2^{(-k)}$$

$$\text{height} = \frac{2\ \text{area}}{|-b1 - b2 + a1 + a2| + |-b1 - a2 + a1 + b2|}$$

$$= 2^{(k-1)}\ \text{area}$$

$$= \frac{1}{2}$$

From that one "sees", that a recursive proof can be made, since the according ranges fir and do not intersect. I have to admit, that I am too lazy to write it down (better using paper + pencil the handling Maple).

Now multiplying with the next rectangle reduces to the range of that rectangle, as it it shrinked by a factor (and only the height changes).

Thus one has: $S(k)\,s(k+1,t) = \frac{1\,s(k+1,t)}{2}$

```
>
> 'S(k)*s(k+1,t) = s(k+1,t)/2';
  #N:=3; #N:=5;
  'S(N)*s(N+1,t) = s(N+1,t)/2'; simplify(%); is(%);
```

$$S(k)\,s(k+1,t) = \frac{1}{2}\,s(k+1,t)$$

$$S(N)\,s(N+1,t) = \frac{1}{2}\,s(N+1,t)$$

$$\begin{cases} 0 & t < \frac{-1}{16} \\ 4 & t \le \frac{1}{16} \\ 0 & \frac{1}{16} < t \end{cases} = \begin{cases} 0 & t < \frac{-1}{16} \\ 4 & t \le \frac{1}{16} \\ 0 & \frac{1}{16} < t \end{cases}$$

true

```
>
```

Bring it together

```
>
> u[k] = 'sinc(t*a(k))', s[k] = 's'(k,t);
  ``;
  Int( Product(u[k], k=0..n) * u[n+1] , t = IR .. ``);
  ``=Int( Product(Phi(s[k]), k=0..n) * u[n+1] , t = IR .. ``);
  ``=Int(Phi((`#`)(s[k]), k=0..n) * u[n+1] , t = IR .. ``); # by convolution
  theorem
  #Phi((`#`)(s[k]), k=0..n);
```

$$u_k = \mathrm{sinc}(t\,a(k)),\ s_k = s(k, t)$$

$$\int_{IR} u_{n+1} \left(\prod_{k=0}^{n} u_k \right) dt$$

$$= \int_{IR} \left(\prod_{k=0}^{n} \Phi(s_k) \right) u_{n+1}\, dt$$

$$= \int_{IR} \Phi(\#(s_k),\ k = 0 .. n)\, u_{n+1}\, dt$$

Now use the Plancherel-Parseval formula to continue (no conjugation need in our case)

```
> ``=Int(Phi( Phi((`#`)(s[k]), k=0..n) ) * Phi(u[n+1]) , t = IR .. ``);
  ``=Int(Phi( Phi((`#`)(s[k]), k=0..n) ) * s[n+1] , t = IR .. ``);
```

$$= \int_{IR} \Phi(\Phi(\#(s_k),\ k = 0 .. n))\, \Phi(u_{n+1})\, dt$$

$$= \int_{IR} \Phi(\Phi(\#(s_k),\ k = 0 .. n))\, s_{n+1}\, dt$$

But $\Phi^2 = -2\,\pi\,\mathrm{id}$ and the iterated product is symmetric w.r.t. zero (follows from above assertion), so the Minus sign drops out and that is

```
> ``=Int(2*Pi*(((`#`)(s[k]), k=0..n) ) * s[n+1] , t = IR .. ``);
  ``='Int(2*Pi*S(n,t) * s(n+1,t) , t = IR .. ``)';
  ``='Int(2*Pi*s(n+1,t)/2 , t = IR .. ``)';
  ``='Int(2*Pi*s(n+1,t)/2 , t = -infinity .. infinity)';
```

```
value(%) assuming n::nonnegint: combine(%);
```

$$= \int_{IR} 2\,\pi\,(\#(s_k),\ k = 0 .. n)\, s_{n+1}\, dt$$

$$= \int_{IR} 2\,\pi\, S(n, t)\, s(n+1, t)\, dt$$

$$= \int_{IR} \pi\, s(n+1, t)\, dt$$

$$= \int_{-\infty}^{\infty} \pi\, s(n+1, t)\, dt$$

$$= \pi$$

```
>
>
```

Appendix: formula for the torso

Show that Torso starts at min(b1+a2, a1+b2) and ends at max(b1+a2, a1+b2) or likewise, that differentiating the convolution w.r.t. t is zero on that interval

```
> `f#g` = (Phi@@(-1))( Phi(f)*Phi(g) );
```

$$f\#g = (\Phi^{(-1)})(\Phi(f)\,\Phi(g))$$

```
> diff(invfourier(F(x), x,t), t) = 'invfourier'(F(x)*x*I, x,t); #convert(%,
  Int);
  is(%);
```

$$\frac{\partial}{\partial t}\mathrm{invfourier}(F(x), x, t) = \mathrm{invfourier}(F(x)\, x\, I, x, t)$$

true

```
> theAssumptions:=
    a1 < b1, a2 < b2,      # general assumption for rectangles
    a1+a2 < t, t < b1+b2;  # support of the convolution
```

$$theAssumptions := a1 < b1,\ a2 < b2,\ a1 + a2 < t,\ t < b1 + b2$$

```
> diffConv:=Diff( (`(f#g)`)(t), t);

  'fourier(rect(a1,b1,t),t,x)*fourier(rect(a2,b2,t),t,x)':
  ``='invfourier(%*x*I, x,t)'; # = differentiation
  %  assuming a1 < b1, a2 < b2;

  subsindets(%, 'Heaviside( anything )', 'u -> convert(u, piecewise,t)');
    # rewrite as piecewise

  simplify(%) assuming (theAssumptions);

  diffConv:=rhs(%):
```

$$diffConv := \frac{d}{dt}(f\#g)(t)$$

$$= \mathrm{invfourier}(\mathrm{fourier}(\mathrm{rect}(a1, b1, t), t, x)\, \mathrm{fourier}(\mathrm{rect}(a2, b2, t), t, x)\, x\, I, x, t)$$

$$=$$

$$-\mathrm{Heaviside}(-t + a1 + a2) + \mathrm{Heaviside}(-t + a1 + b2) + \mathrm{Heaviside}(-t + b1 + a2) - \mathrm{Heaviside}(-t + b1 + b2)$$

$$= -\left(\begin{cases} 1 & t \le a1 + a2 \\ 0 & a1 + a2 < t \end{cases}\right) + \left(\begin{cases} 1 & t \le a1 + b2 \\ 0 & a1 + b2 < t \end{cases}\right) + \left(\begin{cases} 1 & t \le b1 + a2 \\ 0 & b1 + a2 < t \end{cases}\right) - \left(\begin{cases} 1 & t \le b1 + b2 \\ 0 & b1 + b2 < t \end{cases}\right)$$

$$= -1 + \left(\begin{cases} 1 & t \le a1 + b2 \\ 0 & a1 + b2 < t \end{cases} \right) + \left(\begin{cases} 1 & t \le b1 + a2 \\ 0 & b1 + a2 < t \end{cases} \right)$$

Now help Maple ...

```
> theCondition:=[min(b1+a2, a1+b2) < t, t < max(b1+a2, a1+b2)];
  ``;
  'simplify(theCondition) assuming a1+b2 <= b1+a2'; # case 1, a1+b2 =
  minimum
    Case1:= op(%);
  'simplify(theCondition) assuming b1+a2 <= a1+b2'; # case 2, b1+a2 =
  minimum
    Case2:= op(%);
```

$$theCondition := [\min(a1 + b2, b1 + a2) < t, t < \max(a1 + b2, b1 + a2)]$$

$$(\text{simplify}(theCondition) \text{ assuming } (a1 + b2 \le b1 + a2))$$

$$Case1 := a1 + b2 < t, t < b1 + a2$$

$$(\text{simplify}(theCondition) \text{ assuming } (b1 + a2 \le a1 + b2))$$

$$Case2 := b1 + a2 < t, t < a1 + b2$$

```
> # case 1: a1+b2 <= b1+a2
  simplify(diffConv) assuming Case1;
  simplify(diffConv) assuming Case2;
```

$$0$$

$$0$$

```
>
```