# Numerical Inversion of Laplace Transforms of Probability Distributions

RK

February 20, 2015

**Abstract**

The purpose of this document is to summarize main points of the paper, "Numerical Inversion of Laplace Transforms of Probability Distributions", and provide `R` code for the *Euler* method that is described in the paper. The code is used to invert the Laplace transform of some popular functions.

## Context

Laplace transform is a useful mathematical tool that one must be familiar with, while doing applied work. It is widely used in Queueing models where probability distributions are characterized in terms of transforms. Inverting a Laplace transform to get to the probability distribution is an essential task in Queueing theory. For textbook examples and simple Markovian models, one might be fortunate to find convenient forms for LT inversion. However for most of the real life situations, a practitioner needs to know a way to numerically invert LT. The paper titled,"Numerical Inversion of Laplace Transforms of Probability Distributions", written by Joseph Abate and Ward Whitt gives two methods. This document focuses on one of the methods, *Euler*

## Objective

The objective is to calculate values of a real valued function $f(t)$ of a positive real variable $t$ for various $t$ from the Laplace transform

$$\hat{f}(s) = \int_0^\infty e^{-st} f(t) dt,$$

where $s$ is a complex variable with nonnegative real part. The following are the assumptions made about the function $f(t)$:

- $|f| \leq 1$ for all $t$ in the error analysis
- $f$ is sufficiently smooth.
- The algorithm requires that one is able to evaluate the real part of the Laplace transform $\hat{f}(s)$ at any desired complex $s$
- The algorithm is intended for computing $f(t)$ at single values of $t$.

## Method : Euler

This method is based on Bromwich contour inversion integral,

$$f(t) = \frac{1}{2\pi i}\int_{a-i\infty}^{a+i\infty} e^{st}\hat{f}(s)ds$$

Change of variable $a + iu = s$ turns the integral into

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{(a+iu)t}\hat{f}(a+iu)du$$

This implies that

$$f(t) = \frac{e^{at}}{2\pi}\int_{-\infty}^{\infty} e^{iut}\hat{f}(a+iu)du$$

$\Rightarrow$

$$f(t) = \frac{e^{at}}{2\pi}\int_{-\infty}^{\infty} (\cos ut + i\sin ut)\hat{f}(a+iu)du$$

$\Rightarrow$

$$f(t) = \frac{e^{at}}{2\pi}\int_{-\infty}^{0} (\cos ut + i\sin ut)\hat{f}(a+iu)du + \frac{e^{at}}{2\pi}\int_{0}^{\infty} (\cos ut + i\sin ut)\hat{f}(a+iu)du$$

$$f(t) = \frac{e^{at}}{2\pi}\int_{-\infty}^{0} (\cos ut + i\sin ut)\{\Re\hat{f}(a+iu) + i\,\Im\hat{f}(a+iu)\}du + \frac{e^{at}}{2\pi}\int_{0}^{\infty} (\cos ut + i\sin ut)\Re\hat{f}(a+iu) + i\,\Im\hat{f}(a+iu)du$$

Change of variable $u = -w$

$$f(t) = \frac{e^{at}}{2\pi}\int_{0}^{\infty} (\cos(-wt) + i\sin(-wt))\{\Re\hat{f}(a-iw) + i\,\Im\hat{f}(a-iw)\}dw + \frac{e^{at}}{2\pi}\int_{0}^{\infty} (\cos ut + i\sin ut)\Re\hat{f}(a+iu) + i\,\Im\hat{f}(a+iu)du$$

$\Rightarrow$

$$f(t) = \frac{e^{at}}{2\pi}\int_{0}^{\infty} (\cos(wt) - i\sin(wt))\{\Re\hat{f}(a+iw) - i\,\Im\hat{f}(a+iw)\}dw + \frac{e^{at}}{2\pi}\int_{0}^{\infty} (\cos ut + i\sin ut)\Re\hat{f}(a+iu) + i\,\Im\hat{f}(a+iu)du$$

$\Rightarrow$

$$\begin{aligned} f(t) &= \frac{e^{at}}{2\pi}\int_{0}^{\infty} 2\cos(ut)\Re\hat{f}(a+iu) - 2i\Im\hat{f}(a+iu)\cos utdu \\ &= \frac{e^{at}}{2\pi}\int_{0}^{\infty} 2\cos(ut)\Re\hat{f}(a+iu) + 2\Re\hat{f}(a+iu)\cos utdu \\ &= \frac{2e^{at}}{\pi}\int_{0}^{\infty} \cos(ut)\Re\hat{f}(a+iu)du \end{aligned}$$

The integral that would be used for inverting LT is

$$f(t) = \frac{2e^{at}}{\pi} \int_0^\infty \cos(ut) \Re \hat{f}(a + iu)$$

There are three main steps in this method :

1. Trapezoidal rule to discretize the Bromwich integral.
2. Use Fourier series method to replace the integral by a series with specified discretization error.
3. Apply Euler summation to accelerate convergence.

## Step 1 - Trapezoidal discretization

$$f(t) = \frac{he^{at}}{\pi} \Re \hat{f}(a) + \frac{2he^{at}}{\pi} \sum_{i=1}^{\infty} \Re \hat{f}(a + ikh) \cos(kht)$$

Let $h = \pi/2t$ and $a = A/2t$, we obtain

$$f(t) = \frac{e^{A/2}}{2t} \Re \hat{f}(A/2t) + \frac{e^{A/2}}{t} \sum_{i=1}^{\infty} (-1)^k \Re \hat{f}(A/2t + 2k\pi i/2t)$$

## Step 2 - Periodize the function and apply Poisson summation

Consider the damped function $g(t) = e^{-bt} f(t)$ of period $T = 2\pi/h$. Represent the periodized version of the function as

$$g_p(t) = \sum_{k=-\infty}^{\infty} g(t + kT)$$

Writing the Fourier series expansion of $g_p(t)$

$$g_p(t) = \sum_{n=-\infty}^{\infty} c_n e^{inht}$$

where

$$\begin{aligned} c_n &= \frac{1}{T} \int_0^T g_p(t) e^{-2\pi int/T} dt \\ &= \frac{1}{T} \int_0^T \left( \sum_{k=-\infty}^{\infty} g(t + kT) \right) e^{-2\pi int/T} dt \\ &= \frac{1}{T} \sum_{k=-\infty}^{\infty} \int_0^T g(t + kT) e^{-2\pi int/T} dt \end{aligned}$$

Let $\tau = t + kT$

$$\begin{aligned} c_n &= \frac{1}{T}\sum_{k=-\infty}^{\infty}\int_{k\tau}^{(k+1)\tau} g(\tau)e^{-2\pi in(\tau-kT)/T}d\tau \\ &= \frac{1}{T}\sum_{k=-\infty}^{\infty} e^{2\pi ink}\int_{k\tau}^{(k+1)\tau} g(\tau)e^{-2\pi in\tau/T}d\tau \\ &= \frac{1}{T}\int_{-\infty}^{\infty} g(\tau)e^{-2\pi in\tau/T}d\tau \\ &= \frac{1}{T}\int_{-\infty}^{\infty} f(\tau)e^{-b\tau}e^{-2\pi in\tau/T}d\tau \\ &= \frac{1}{T}\hat{f}(b+inh) \end{aligned}$$

In the above derivation, $e^{2\pi ink} = 1, \forall k \in \mathbb{Z}$ Thus

$$g_p(t) = \sum_{k=-\infty}^{\infty} g(t+2\pi k/h) = \sum_{k=-\infty}^{\infty} f(t+2\pi k/h)e^{-b(t+2\pi k/h)} = \frac{h}{2\pi}\sum_{k=-\infty}^{\infty}\hat{f}(b+ikh)$$

Letting $h = \pi/t$ and $b = A/2t$

$$f(t) = \frac{e^{A/2}}{2t}\sum_{-\infty}^{\infty}(-1)^k\Re\hat{f}(A/2t + 2k\pi i/2t) - \sum_{k=1}^{\infty}e^{-kA}f((2k+1)t)$$

One can see that the discretization error

$$|e_d| = |\sum_{k=1}^{\infty}e^{-kA}f((2k+1)t)| \leq \frac{e^{-A}}{1-e^{-A}}$$

The authors desire a discretization error of $10^{-8}$ and hence $A$ is chosen as 18.4.

## Step 3 - Apply Euler summation to accelerate convergence

$$E(m,n,t) = \sum_{k=0}^{m}\binom{m}{k}2^{-m}s_{n+k}(t)$$

where

$$s_n(t) = \frac{e^{A/2}}{2t}\Re\hat{f}(A/2t) + \frac{e^{A/2}}{t}\sum_{k=1}^{n}(-1)^k\Re\hat{f}(A/2t + 2k\pi i/2t)$$

The authors suggest $m = 11$ and $n = 15$.

Even though the steps are straightforward, I think it takes a lot of ingenuity to come up with a neat and smart way of taming discretization error.

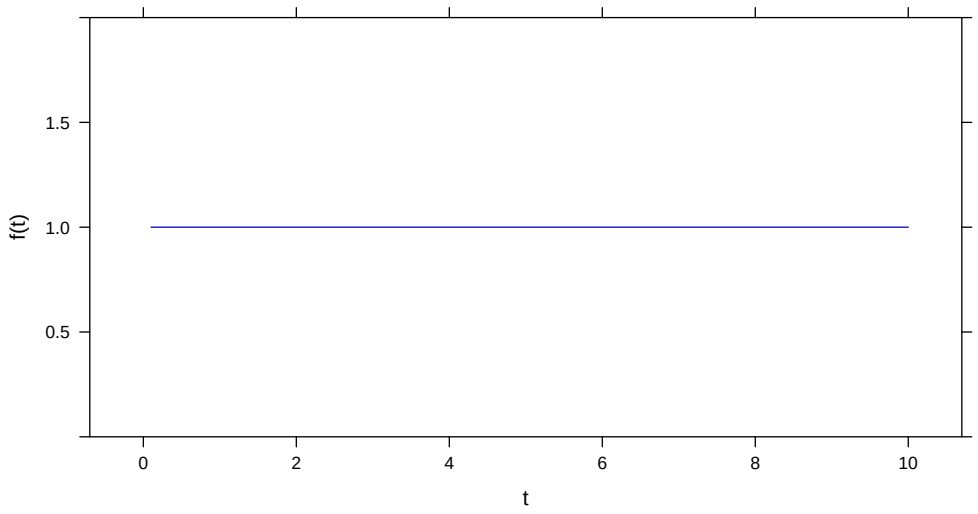
## R code for inverting Laplace Transform

```
inv_laplace_transform <- function(t,g){
  A   <- 18.4
  h   <- pi/(2*t)
  a   <- A/(2*t)
  n   <- 15
  m   <- 11
  s_n     <- function(n,t){
    temp  <- g( (A+2*(1:n)*pi*1)/(2*t) )
    exp(A/2)/t * (1/2 *Re(g(a)) +  sum( (-1)^(1:n) * Re(temp)))
  }
  sum(1/(2^(m)) * sapply( n:(n+m), s_n,t) * choose(m,0:m))
}
```

One can check the code for all the standard Laplace transforms. In this document, I have used the above function to invert all the standard examples.

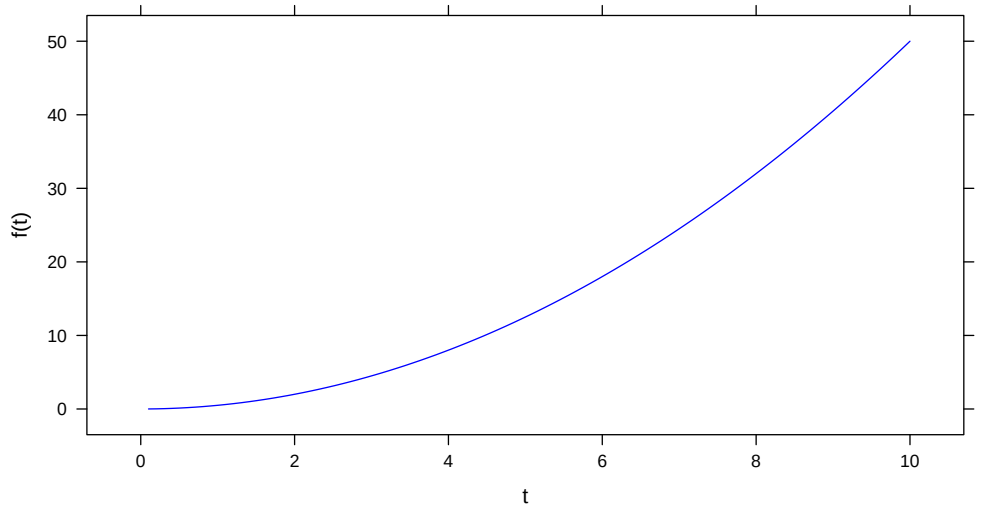
$$\mathcal{L}^{-1}\left\{\frac{1}{s}\right\} = 1$$

```
time     <- seq(0,10,length.out = 100)
g        <- function(s) {1/(s) +1*0}
f_t      <- sapply(time, inv_laplace_transform,g=g)
xyplot(f_t~time,ylab="f(t)",xlab="t",lwd=1,col = "blue",type="l", ylim=c(0,2),
       main = "" )
```



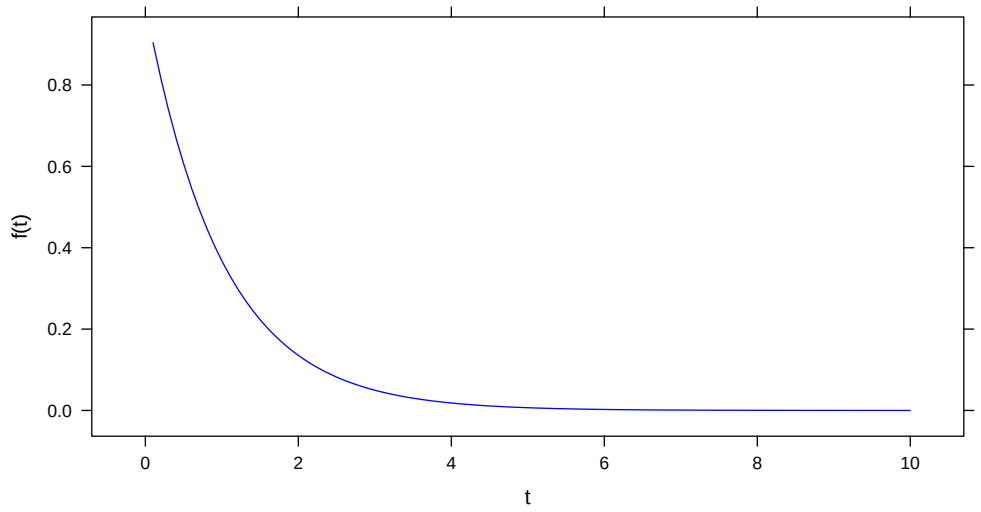
$$\mathcal{L}^{-1}\left\{\frac{1}{s^3}\right\} = t^2/2$$

```
g        <- function(s) {1/(s^3) +1*0}
f_t      <- sapply(time, inv_laplace_transform,g=g)
xyplot(f_t~time,ylab="f(t)",xlab="t",lwd=1,col = "blue",type="l",main = "" )
```



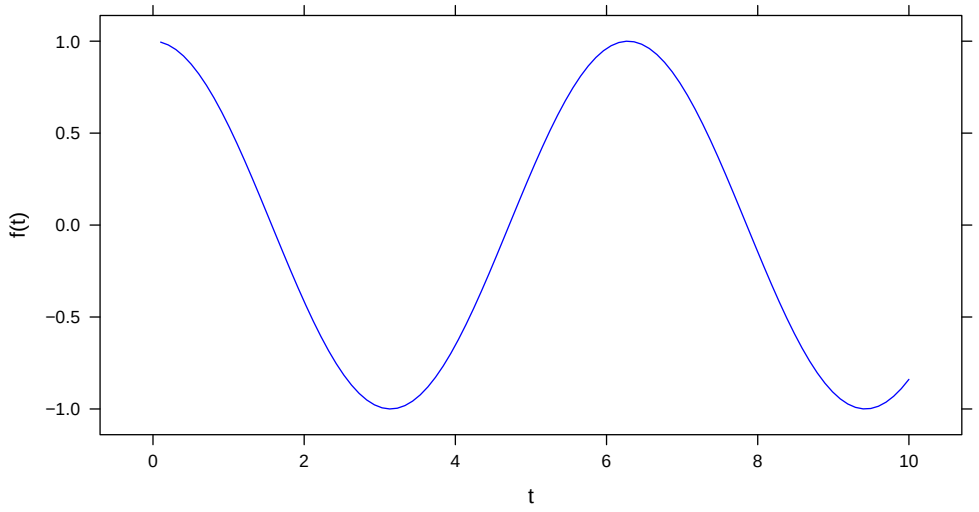

$$\mathcal{L}^{-1}\left\{\frac{1}{s+1}\right\} = e^{-t}$$

```
g        <- function(s) {1/(s+1) +1*0}
f_t      <- sapply(time, inv_laplace_transform,g=g)
xyplot(f_t~time,ylab="f(t)",xlab="t",lwd=1,col = "blue",type="l",main = "" )
```



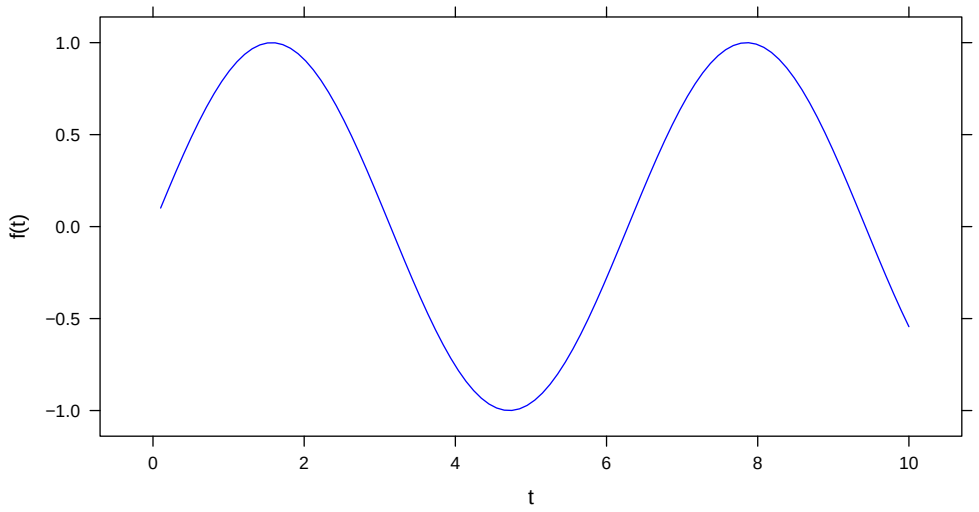
$$\mathcal{L}^{-1}\left\{\frac{s}{s^2+1}\right\} = \cos t$$

```
g        <- function(s) {s/(s^2+1) +1*0}
f_t      <- sapply(time, inv_laplace_transform,g=g)
xyplot(f_t~time,ylab="f(t)",xlab="t",lwd=1,col = "blue",type="l",main = "" )
```



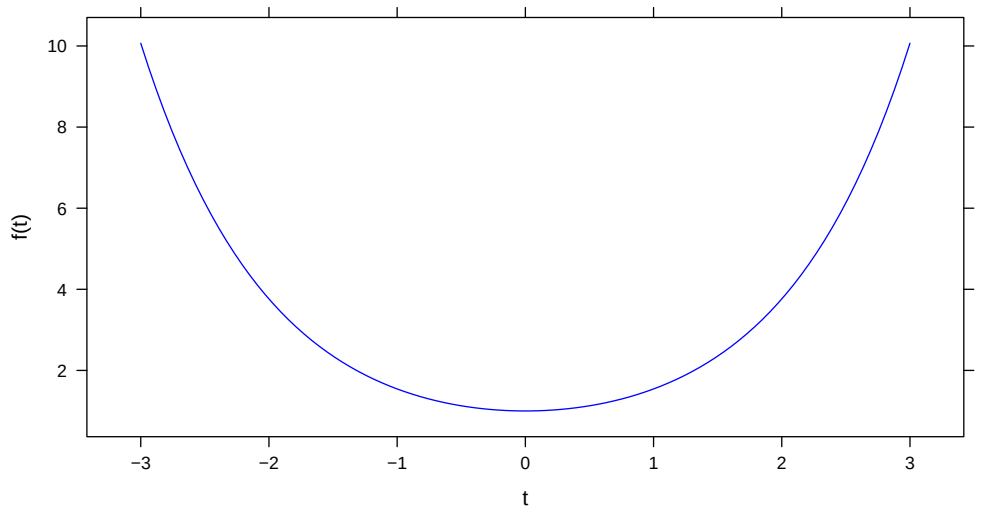

$$\mathcal{L}^{-1}\left\{\frac{1}{s^2+1}\right\} = \sin t$$

```
g        <- function(s) {1/(s^2+1) +1*0}
f_t      <- sapply(time, inv_laplace_transform,g=g)
xyplot(f_t~time,ylab="f(t)",xlab="t",lwd=1,col = "blue",type="l",main = "" )
```



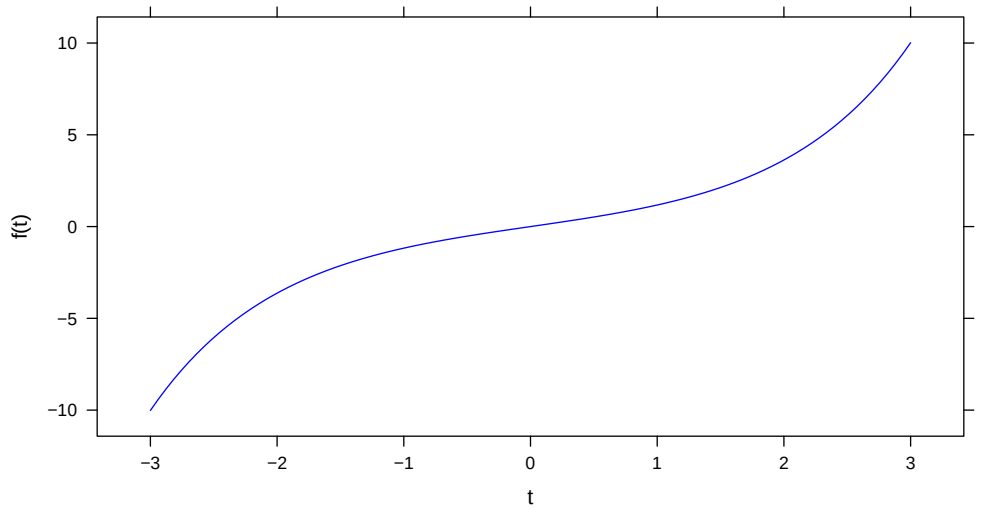
$$\mathcal{L}^{-1}\left\{\frac{s}{s^2-1}\right\} = \cosh t$$

```
g        <- function(s) {s/(s^2-1) +1*0}
f_t      <- sapply(time, inv_laplace_transform,g=g)
xyplot(f_t~time,ylab="f(t)",xlab="t",lwd=1,col = "blue",type="l",main = "" )
```



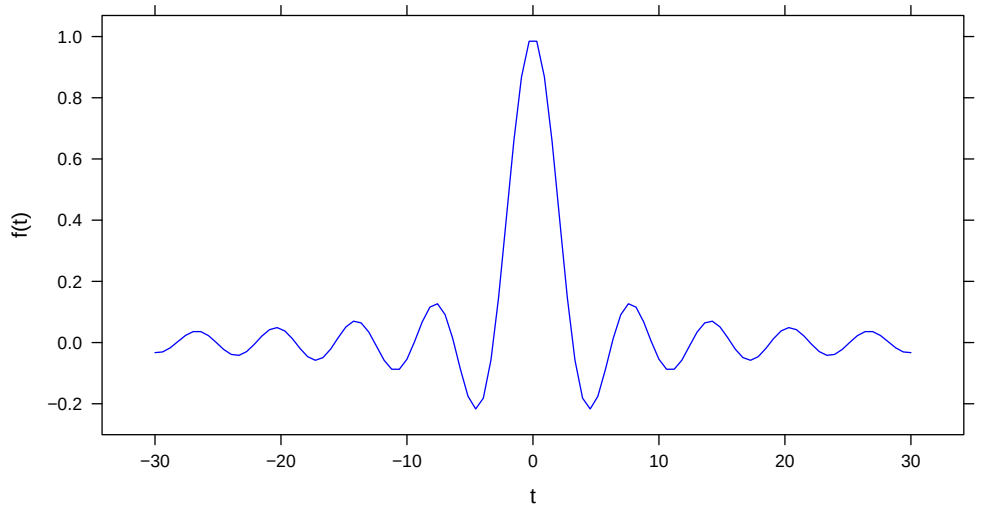

$$\mathcal{L}^{-1}\left\{\frac{1}{s^2-1}\right\} = \sinh t$$

```
g        <- function(s) {1/(s^2-1) +1*0}
f_t      <- sapply(time, inv_laplace_transform,g=g)
xyplot(f_t~time,ylab="f(t)",xlab="t",lwd=1,col = "blue",type="l",main = "" )
```



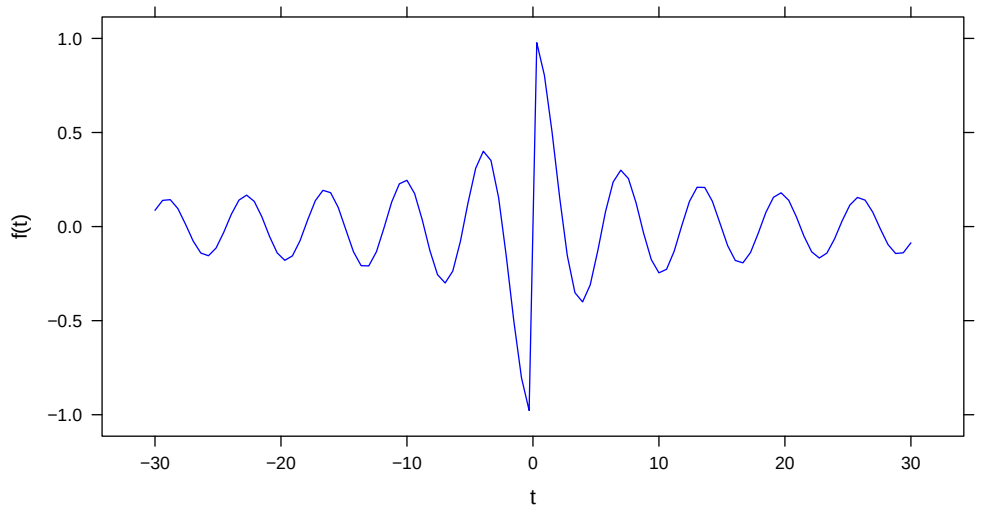
$$\mathcal{L}^{-1}\left\{\tan^{-1}\frac{1}{s}\right\} = \frac{\sin t}{t}$$

```
g        <- function(s) {atan(1/s) +1*0}
f_t      <- sapply(time, inv_laplace_transform,g=g)
xyplot(f_t~time,ylab="f(t)",xlab="t",lwd=1,col = "blue",type="l",main = "" )
```




$$\mathcal{L}^{-1}\left\{\frac{1}{\sqrt{s^2+1}}\right\} = J_0(t)$$

```
g        <- function(s) {1/sqrt(s^2+1) +1*0}
f_t      <- sapply(time, inv_laplace_transform,g=g)
xyplot(f_t~time,ylab="f(t)",xlab="t",lwd=1,col = "blue",type="l",main = "" )
```



## Takeaway

The method developed by the authors in the paper has been used in a ton of places. All the textbook examples match with that of code output. In reality, most of the Laplace transforms for which inversion is desired have no closed form solutions and hence this method is immensely useful to get numerical results. However the authors do caution the user about the method by saying,

- When $f$ does not have enough smoothness, one can think of performing convolution smoothing. This is achieved by multiplying the transform $\hat{f}$ by some other transform $\hat{g}$ before doing the inverse
- With an increase in $t$, the value of $n$ increases and this often is the case. The value of $n(t)$ to achieve the prescribed accuracy is often approximately a linear function of $t$