# Laplace Transform Lab: Solving ODEs using Laplace Transform in MATLAB

## Table of Contents



This lab will teach you to solve ODEs using a built in MATLAB Laplace transform function `laplace`. Also in this lab, you will write your own ODE solver using Laplace transforms and check whether the result yields the correct answer.

You will learn how to use the `laplace` routine.

There are five (5) exercises in this lab that are to be handed in. Write your solutions in the template, including appropriate descriptions in each step. Save the m-file and submit it on Quercus.

Include your name and student number in the submitted file.

# Student Information

```
Student Name: Emma Seabrook

Student Number: 1005834563
```

# Using symbolic variables to define functions

Recall the use of symbolic variables and function explained in the MATLAB assignment #2.

```
syms t s x y;

f = cos(t);
h = exp(2*x);
```

# Laplace transform and its inverse

```
% The routine |laplace| computes the Laplace transform of a function
```

```
F=laplace(f)
```

```
F =

s/(s^2 + 1)
```

By default it uses the variable `s` for the Laplace transform But we can specify which variable we want:

```
H=laplace(h)
laplace(h,y)

% Observe that the results are identical: one in the variable |s| and
 the
% other in the variable |y|
```

```
H =

1/(s - 2)


ans =

1/(y - 2)
```

We can also specify which variable to use to compute the Laplace transform:

```
j = exp(x*t)
laplace(j)
laplace(j,x,s)

% By default, MATLAB assumes that the Laplace transform is to be
 computed
% using the variable |t|, unless we specify that we should use the
 variable
% |x|
```

```
j =

exp(t*x)


ans =

1/(s - x)


ans =

1/(s - t)
```

We can also use inline functions with `laplace`. When using inline functions, we always have to specify the variable of the function.

```
l = @(t) t^2+t+1
laplace(l(t))


l =

  function_handle with value:

    @(t)t^2+t+1


ans =

(s + 1)/s^2 + 2/s^3
```

MATLAB also has the routine `ilaplace` to compute the inverse Laplace transform

```
ilaplace(F)
ilaplace(H)
ilaplace(laplace(f))


ans =

cos(t)


ans =

exp(2*t)


ans =

cos(t)
```

If `laplace` cannot compute the Laplace transform, it returns an unevaluated call.

```
g = 1/sqrt(t^2+1)
G = laplace(g)


g =

1/(t^2 + 1)^(1/2)


G =

laplace(1/(t^2 + 1)^(1/2), t, s)
```

But MATLAB "knows" that it is supposed to be a Laplace transform of a function. So if we compute the inverse Laplace transform, we obtain the original function

```
ilaplace(G)
```

```
ans =

1/(t^2 + 1)^(1/2)
```

The Laplace transform of a function is related to the Laplace transform of its derivative:

```
syms g(t)
laplace(diff(g,t),t,s)
```

```
ans =

s*laplace(g(t), t, s) - g(0)
```

# Exercise 1

Objective: Compute the Laplace transform and use it to show that MATLAB 'knows' some of its properties.

Details:

(a) Define the function `f(t)=exp(2t)*t^3`, and compute its Laplace transform `F(s)`. (b) Find a function `f(t)` such that its Laplace transform is `(s - 1)*(s - 2))/(s*(s + 2)*(s - 3)` (c) Show that MATLAB 'knows' that if `F(s)` is the Laplace transform of `f(t)`, then the Laplace transform of `exp(at)f(t)` is `F(s-a)`

(in your answer, explain part (c) using comments).

Observe that MATLAB splits the rational function automatically when solving the inverse Laplace transform.

```
clear;
syms t s a;
f = @(t) exp(2*t)*t^3;
F = laplace(f(t));
disp('F:');
disp(F);


L = ((s - 1)*(s - 2))/(s*(s + 2)*(s - 3));
l = ilaplace(L);
disp('l:');
disp(l);

e = exp(a*t)*f(t);
```

```
E = laplace(e);
disp('E:');
disp(E);

%(c) F(s) = 6/(s - 2)^4. When I plug in 'a' as a symbolic variable,
 matlab
%outputs E: 6/(a - s + 2)^4 thus showing that the Laplace transform of
%|exp(at)f(t)| is |F(s-a)|
```

```
F:
6/(s - 2)^4

l:
(6*exp(-2*t))/5 + (2*exp(3*t))/15 - 1/3

E:
6/(a - s + 2)^4
```

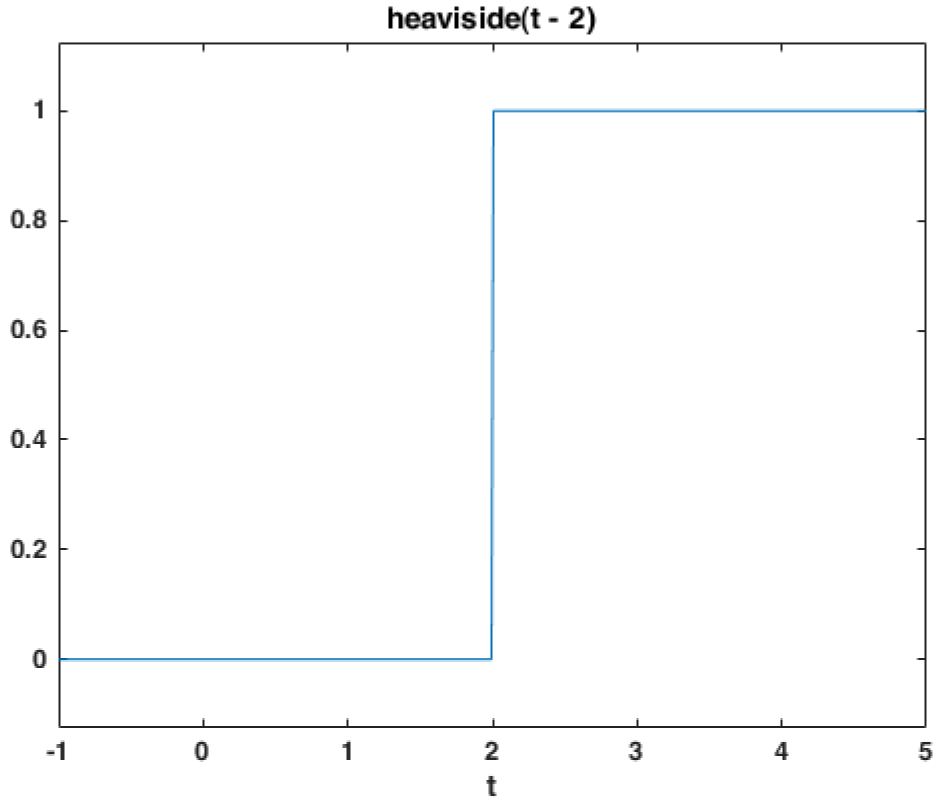
# Heaviside and Dirac functions

These two functions are builtin to MATLAB: `heaviside` is the Heaviside function `u_0(t)` at `0`

To define `u_2(t)`, we need to write

```
f=heaviside(t-2);
ezplot(f,[-1,5]);

% The Dirac delta function (at |0|) is also defined with the routine
% |dirac|

g = dirac(t-3)

% MATLAB "knows" how to compute the Laplace transform of these
 functions

laplace(f)
laplace(g)
```

```
g =

dirac(t - 3)


ans =

exp(-2*s)/s


ans =

exp(-3*s)
```




# Exercise 2

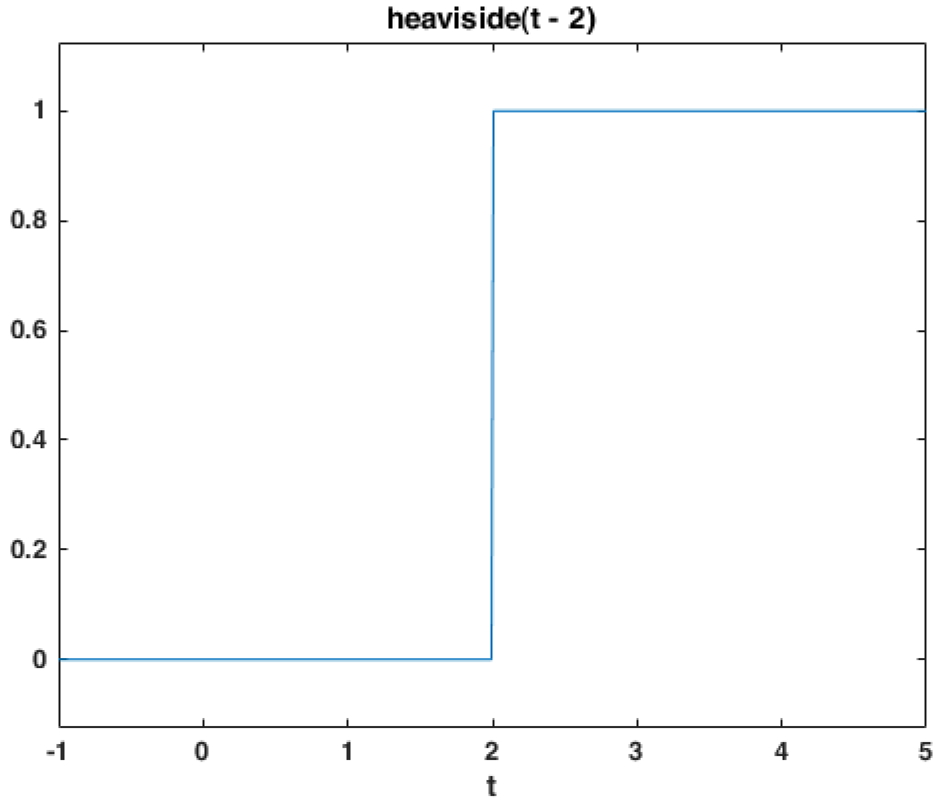
Objective: Find a formula comparing the Laplace transform of a translation of `f(t)` by `t-a` with the Laplace transform of `f(t)`

Details:

- Give a value to `a`
- Let `G(s)` be the Laplace transform of `g(t)=u_a(t)f(t-a)` and `F(s)` is the Laplace transform of `f(t)`, then find a formula relating `G(s)` and `F(s)`

In your answer, explain the 'proof' using comments.

```
syms a t;
u = heaviside(t-a);
f = @(t) exp(2*t)*t^3;
g = @(t) u*f(t-a);
F = laplace(f(t));
G = laplace(g(t));

% F = 6/(s - 2)^4 G = -laplace(exp(2*t - 2*a)*heaviside(t - a)*(a -
 t)^3,
% t, s) if we give a value to a such as 5: G = (6*exp(-5*s))/(s - 2)^4
 From
% this we can derive that |G = F * exp(-a*s)|
```

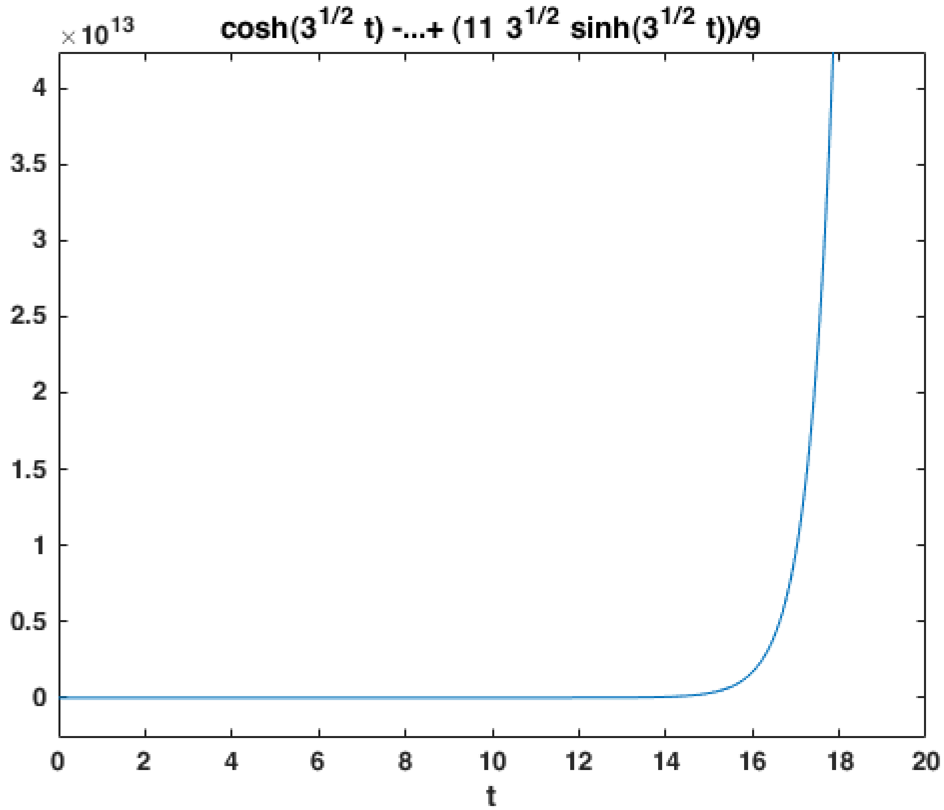



# Solving IVPs using Laplace transforms

Consider the following IVP, `y''-3y = 5t` with the initial conditions `y(0)=1` and `y'(0)=2`. We can use MATLAB to solve this problem using Laplace transforms:

```
% First we define the unknown function and its variable and the
 Laplace
% tranform of the unknown

syms y(t) t Y s;

% Then we define the ODE

ODE = diff(y(t),t,2)-3*y(t)-5*t == 0;

% Now we compute the Laplace transform of the ODE.

L_ODE = laplace(ODE);

% Use the initial conditions

L_ODE=subs(L_ODE,y(0),1);
L_ODE=subs(L_ODE, subs(diff(y(t), t), t, 0), 2);

% We then need to factor out the Laplace transform of |y(t)|
```

```
L_ODE = subs(L_ODE,laplace(y(t), t, s), Y);
Y=solve(L_ODE,Y);

% We now need to use the inverse Laplace transform to obtain the
 solution
% to the original IVP

y = ilaplace(Y)

% We can plot the solution

ezplot(y,[0,20]);

% We can check that this is indeed the solution

diff(y,t,2)-3*y;


y =

cosh(3^(1/2)*t) - (5*t)/3 + (11*3^(1/2)*sinh(3^(1/2)*t))/9
```



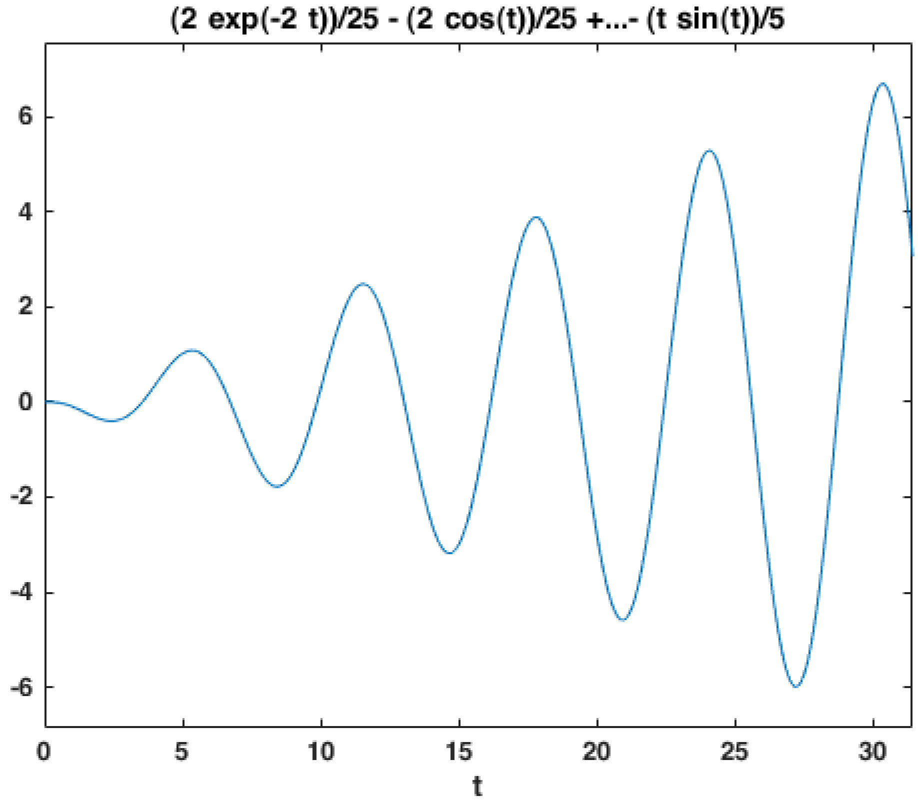

# Exercise 3

Objective: Solve an IVP using the Laplace transform

Details: Explain your steps using comments

- Solve the IVP
- `y'''+2y''+y'+2*y=-cos(t)`
- `y(0)=0`, `y'(0)=0`, and `y''(0)=0`
- for `t` in `[0,10*pi]`
- Is there an initial condition for which `y` remains bounded as `t` goes to infinity? If so, find it.

```
syms y(t) t Y s;


ODE = diff(y(t),t,3)+ 2*diff(y(t),t,2) + diff(y(t),t,1)+2*y(t) +
 cos(t) == 0;

L_ODE = laplace(ODE);

L_ODE=subs(L_ODE,y(0),0);
L_ODE=subs(L_ODE,subs(diff(y(t), 1), t, 0), 0);
L_ODE=subs(L_ODE,subs(diff(y(t), 2), t, 0), 0);


L_ODE = subs(L_ODE, laplace(y(t), t, s), Y);
Y=solve(L_ODE,Y);



y = ilaplace(Y);


ezplot(y,[0,10*pi]);

% No, there is no initial condition where y(t) is bounded because the
% initial conditions do not affect the eigenvalues of an ODE.
```




# Exercise 4

Objective: Solve an IVP using the Laplace transform

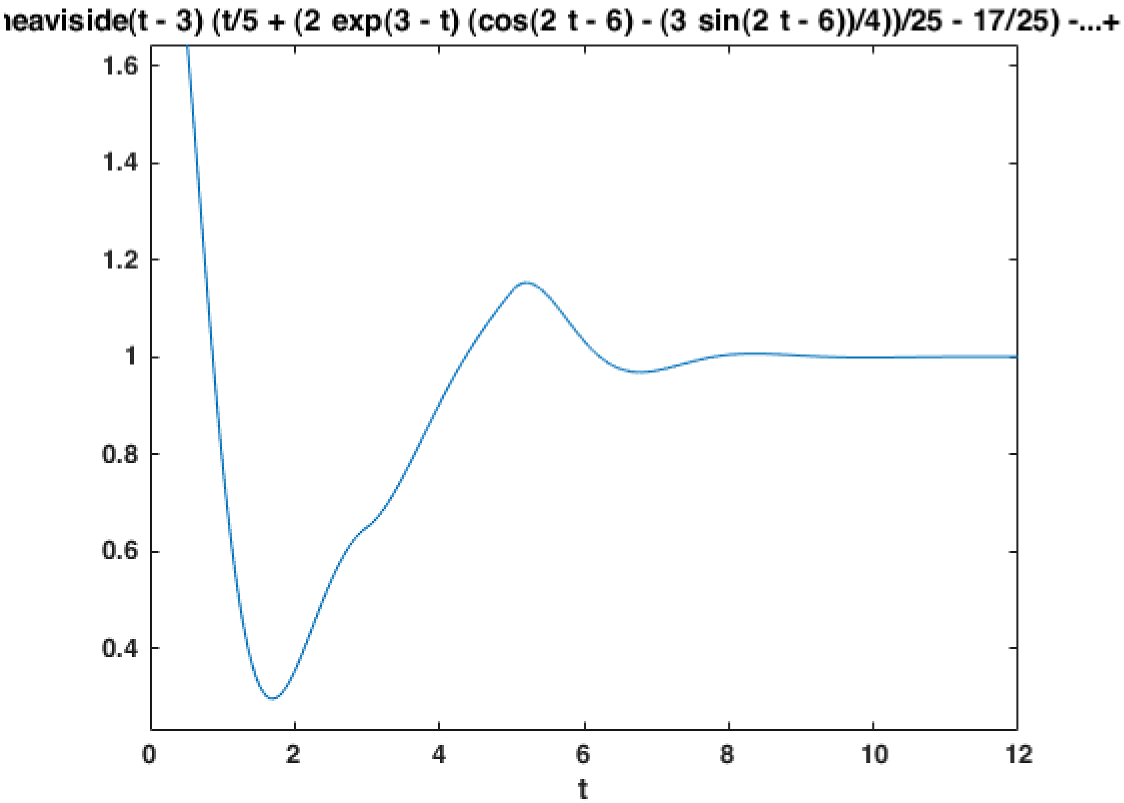
Details:

- Define
- `g(t) = 3 if 0 < t < 2`
- `g(t) = t+1 if 2 < t < 5`
- `g(t) = 5 if t > 5`
- Solve the IVP
- `y''+2y'+5y=g(t)`
- `y(0)=2 and y'(0)=1`
- Plot the solution for `t` in `[0,12]` and `y` in `[0,2.25]`.

In your answer, explain your steps using comments.

```
syms y(t) t Y s;

g = 3 + ((t+1)-3)*heaviside(t-3) + (5-(t+1))*heaviside(t-5);
```

```
ODE = diff(y(t),t,2) + 2*diff(y(t),t,1)+5*y(t) - g==0;

L_ODE = laplace(ODE);

L_ODE=subs(L_ODE,y(0),2);
L_ODE=subs(L_ODE,subs(diff(y(t), 1), t, 0), 1);


L_ODE = subs(L_ODE, laplace(y(t), t, s), Y);

Y=solve(L_ODE,Y);

y = ilaplace(Y);

ezplot(y,[0,12]);
```




# Exercise 5a

Objective: Use the Laplace transform to solve an integral equation

Verify that MATLAB knowns about the convolution theorem by explaining why the following transform is computed correctly.

```
syms t tau y(tau) s;
I=int(exp(-2*(t-tau))*y(tau),tau,0,t);
```

```
L_i = laplace(I,t,s);


% The laplace of the convolution integral is F(s)*G(s). Matlab
 demonstrates
% this because f(t-tau) is exp(-2*(t-tau)) and g(tau)is y(tau). In the
% solution from matlab, L_i is laplace(y(t), t, s)/(s + 2) with the
% laplace(y(t), t, s) being the laplace of f(t-tau) and the 1/(s+2)
 being
% the laplace of g(tau). This demonstrates how the convolution in the
% t-space is equal to multiplication in the s-space.
%
```

# Exercise 5b

A particular machine in a factory fails randomly and needs to be replaced. Suppose that the times `t>=0` between failures are independent and identically distributed with probability density function `f(t)`. The mean number of failures `m(t)` at time `t` satisfies the renewal equation `m(t) = \int_0^t [1+m(t-tau)] f(tau) dtau`

Details:

- Explain why the mean number of failures satisfies this intergal equation. Note that `m(0)=0`.
- Solve the renewal equation for `m(t)` using MATLAB symbolic computation in the cases of i) exponential failure times `f(t) = exp(-t)` and ii) gamma-distributed failure times `f(t) = t^(k-1)/(k-1)! exp(-t)` for natural number `k`. Why does MATLAB have difficulty with the calculation for `k>=5`?
- Verify the elementary renewal theorem: `m(t)/t` approaches the reciprocal of the mean of `f(t)` as `t` goes to infinity.

```
syms t1 t2 Y1 Y2 tau1 tau2 k s1 s2 m_1(t) m_2(t);

f1 = @(t1) exp(-t1);
f2 = @(t2) t2^(k-1)/(factorial(k-1))*exp(-t2);


m1_1 = int((1+m_1(t1-tau1))*f1(tau1),tau1,0,t1)== m_1(t1);
m2_2 = int((1+m_2(t2-tau2))*f2(tau2),tau2,0,t2)== m_2(t2);

F_1 = laplace(m1_1,t1,s1);
F_2 = laplace(m2_2,t2,s2);

F_1 = subs(F_1,laplace(m_1(t1), t1, s1),Y1);
F_2 = subs(F_2,laplace(m_2(t2), t2, s2),Y2);

Y1 = solve(F_1, Y1);
Y2 = solve(F_2, Y2);

con_1 = ilaplace(Y1);
con_2 = ilaplace(Y2);
```

*Warning: Solutions are valid under the following*

```
conditions: 0 < real(k) | 2 <= k & in(k, 'integer').
To include parameters and conditions in the solution,
specify the 'ReturnConditions' value as 'true'.
```

# My Answer:

(a) In the case of a probability distribution, all of the possibilities must also satisfy the integral of the distribution function. Since the integral is also equal to the convolution + the laplace transform of F which together = m(t), and m(0) = 0, the mean nuber of failiures must satisfy the integral equation.

```
% (b) Matlab has difficulty with the calculation for k>5 because as k
% increases, the value of it's factorial does too which needs to be
% calculated at each time step. This means that the calculation gets
 more
% and more complex every time k gets larger. Also, Matlab needs to
 take the
% laplace transform of the function every time which gets more and
 more
% complex every time k increases because k is factorialised.

% (c) To verify this, we calculate the limit as t->inf
syms t_lim;

g2 = int(t_lim * f2(t_lim), 0,inf);

lim_f2 = 1/limit(g2, t_lim, inf);

m_t = con_2/t;

lim2 = limit(m_t, t, inf);


disp(lim2==lim_f2);

%Since lim2 == lim_f2, the elementary renewal theorem holds
```

```
piecewise(gamma(k) == 0, 0, gamma(k) ~= 0, -
limit((gamma(k)*ilaplace(1/(s2*gamma(k) - s2*factorial(k -
 1)*(s2 + 1)^k), s2, t))/t, t, Inf)) == piecewise(-1 < real(k),
 factorial(k - 1)/gamma(k + 1), real(k) <= -1, factorial(k - 1)/
limit(int(t_lim*t_lim^(k - 1)*exp(-t_lim), t_lim, 0, Inf), t_lim,
 Inf))
```

*Published with MATLAB® R2019b*