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.

MAT292, Fall 2019, Stinchcombe & Parsch, modified from MAT292, Fall 2018, Stinchcombe & Khovanskii, modified from MAT292, Fall 2017, Stinchcombe & Sinnamon, modified from MAT292, Fall 2015, Sousa, based on MAT292, Fall 2013, Sinnamon & Sousa

Student Information

```
Student Name: Prerak Chaudhari

Student Number: 1005114760
```

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)
```

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.

```
% a )
```

```
f = @(t) exp(2*t)* t^3
laplace(f(t))

% b)
F = @(s) ((s-1) * (s-2)) / (s*(s+2) *(s-3))
ilaplace(F(s))

% c)
syms f(t) F(s) a t s
F(s)= laplace(f(t))
laplace(exp(a*t)*f(t))
% Matlab understood that multiplying f(t) by e^(a*t) translates the
 laplace
% of f(t) by a
```

```
f =

  function_handle with value:

    @(t)exp(2*t)*t^3


ans =

6/(s - 2)^4


F =

  function_handle with value:

    @(s)((s-1)*(s-2))/(s*(s+2)*(s-3))


ans =

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


F(s) =

laplace(f(t), t, s)


ans =

laplace(f(t), t, s - a)
```

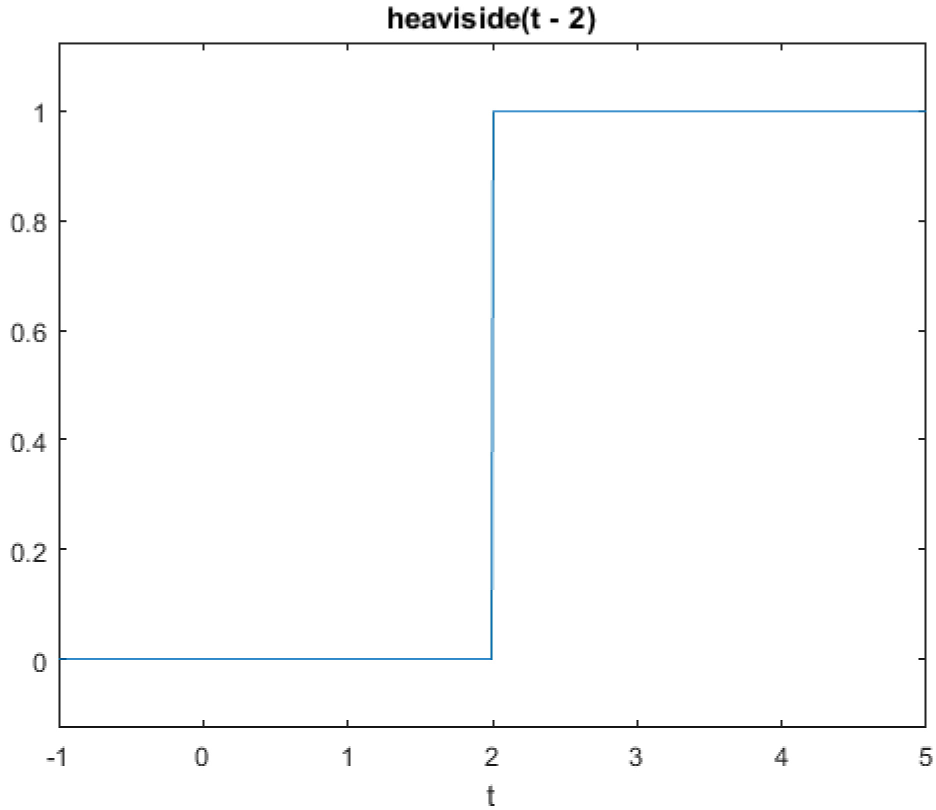
Heaviside and Dirac functions

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)
```

```
f =

heaviside(t - 2)


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.

```
%proof
% Let a be a value
% Let F(S) be the Laplace transform of f(t)
% Let G(S) be the Laplace transform of g(t) = u_a(t)*f(t-a)
% From laplace transform table, G(S) is exp(-a*s)*F(S)

syms f(t)
g = heaviside(t-3)*f(t-3);
F = laplace(f);
G = laplace(g);
disp(F)
disp(G)
```

```
laplace(f(t), t, s)

exp(-3*s)*laplace(f(t), t, s)
```

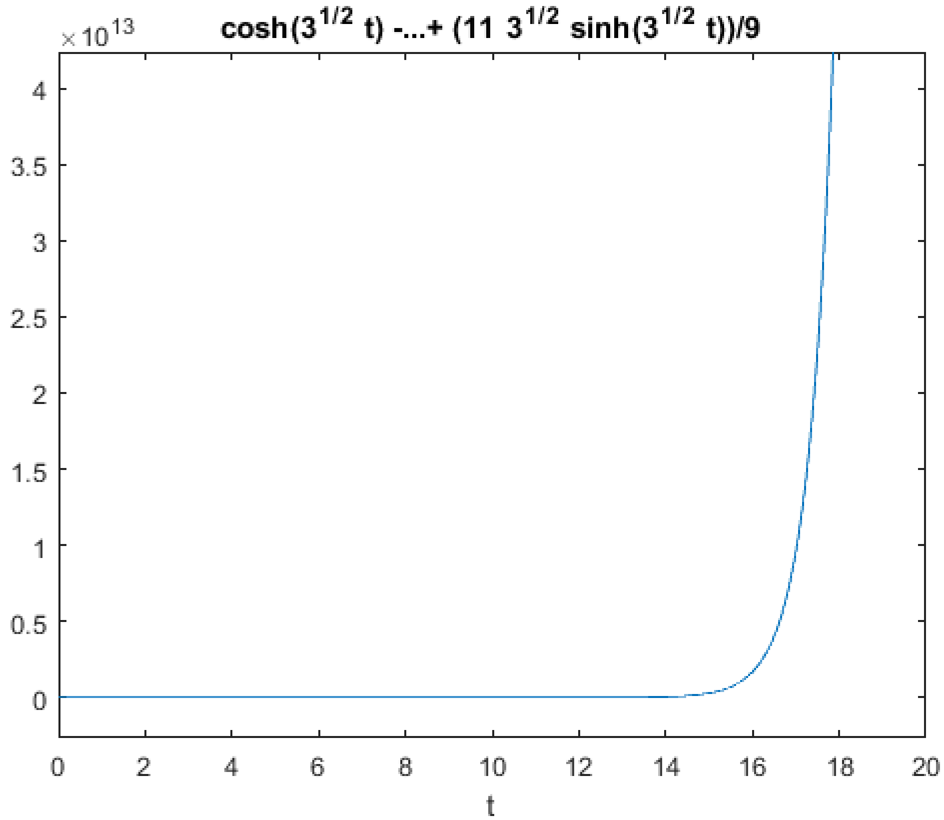
Solving IVPs using Laplace transforms

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
```

```
ODE =

diff(y(t), t, t) - 3*y(t) - 5*t == 0
```

```
L_ODE =

s^2*laplace(y(t), t, s) - s*y(0) - subs(diff(y(t), t), t, 0) - 5/s^2 -
 3*laplace(y(t), t, s) == 0


L_ODE =

s^2*laplace(y(t), t, s) - s - subs(diff(y(t), t), t, 0) - 5/s^2 -
 3*laplace(y(t), t, s) == 0


L_ODE =

s^2*laplace(y(t), t, s) - s - 5/s^2 - 3*laplace(y(t), t, s) - 2 == 0


L_ODE =

Y*s^2 - s - 3*Y - 5/s^2 - 2 == 0


Y =

(s + 5/s^2 + 2)/(s^2 - 3)


y =

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


ans =

5*t
```



Exercise 3

Objective: Solve an IVP using the Laplace transform

Details: Explain your steps using comments

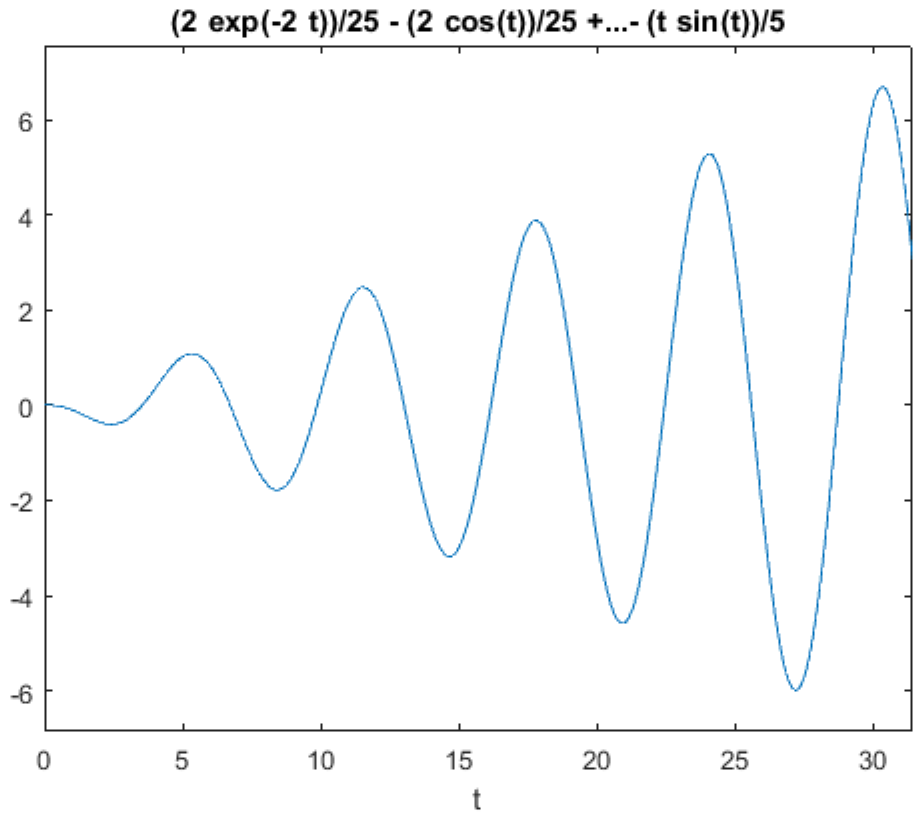
- Solve the IVP
- `y'''+2y''+y'+2*y=-cos(t)`
- `y(0)=0`, `y'(0)=0`, and `y''(0)=0`
- for `t` in `[0,10*pi]`
- Is there an initial condition for which `y` remains bounded as `t` goes to infinity? If so, find it.

```
% 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,3) + 2*diff(y(t),t,2) + diff(y(t),t,1) + 2*y(t) +
 cos(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),0)
L_ODE=subs(L_ODE,subs(diff(y(t), t), t, 0),0)
L_ODE=subs(L_ODE,subs(diff(y(t), t,2),t, 0),0)

% 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,[0,10*pi])

% We can check that this is indeed the solution

diff(y,t,3) + 2*diff(y,t,2) + diff(y,t,1) + 2*y

% The general solution to this ODE is
% y(t) = c_3*exp(-2t) + c_2*sin(t) + c_1*cos(t) - tsin(t)/5 +
 tcos(t)/10.
% (derivation steps skipped; left as an exercise for the reader)

% The terms -tsin(t)/5 and tcos(t)/10 will cause this function to grow
 while
% oscillating as t goes to infinity. Since these two terms are not
 reliant
% on the coefficients, no set of initial conditions will cause the
 function
% to be bounded as t goes to infinity.
```

```
ODE =

cos(t) + 2*y(t) + diff(y(t), t) + 2*diff(y(t), t, t) + diff(y(t), t,
 t, t) == 0


L_ODE =

s*laplace(y(t), t, s) - y(0) - 2*s*y(0) - s*subs(diff(y(t), t), t, 0)
 + s/(s^2 + 1) + 2*s^2*laplace(y(t), t, s) + s^3*laplace(y(t), t, s) -
 2*subs(diff(y(t), t), t, 0) - s^2*y(0) - subs(diff(y(t), t, t), t, 0)
 + 2*laplace(y(t), t, s) == 0
```

```
L_ODE =

s*laplace(y(t), t, s) - s*subs(diff(y(t), t), t, 0) + s/(s^2
 + 1) + 2*s^2*laplace(y(t), t, s) + s^3*laplace(y(t), t, s) -
 2*subs(diff(y(t), t), t, 0) - subs(diff(y(t), t, t), t, 0) +
 2*laplace(y(t), t, s) == 0


L_ODE =

s*laplace(y(t), t, s) + s/(s^2 + 1) + 2*s^2*laplace(y(t), t,
 s) + s^3*laplace(y(t), t, s) - subs(diff(y(t), t, t), t, 0) +
 2*laplace(y(t), t, s) == 0


L_ODE =

s*laplace(y(t), t, s) + s/(s^2 + 1) + 2*s^2*laplace(y(t), t, s) +
 s^3*laplace(y(t), t, s) + 2*laplace(y(t), t, s) == 0


L_ODE =

2*Y + Y*s + s/(s^2 + 1) + 2*Y*s^2 + Y*s^3 == 0


Y =

-s/((s^2 + 1)*(s^3 + 2*s^2 + s + 2))


y =

(2*exp(-2*t))/25 - (2*cos(t))/25 + (3*sin(t))/50 + (t*cos(t))/10 -
 (t*sin(t))/5


ans =

-cos(t)
```



Exercise 4

Objective: Solve an IVP using the Laplace transform

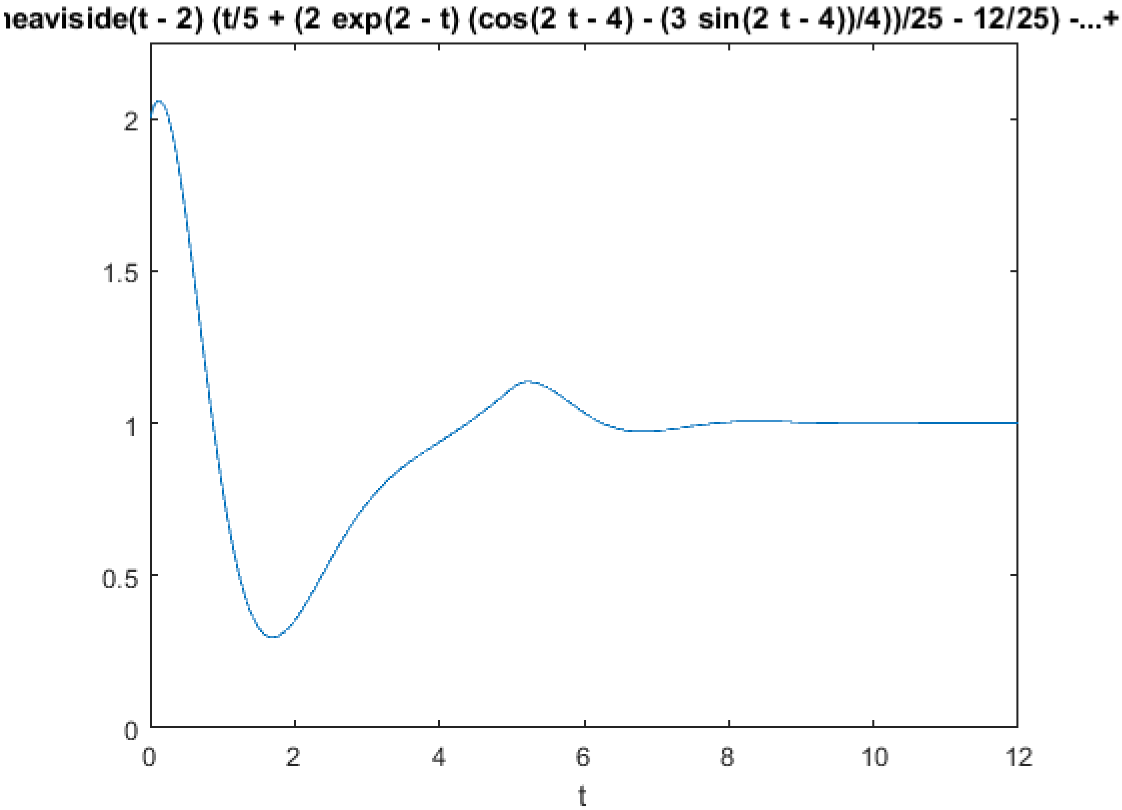
Details:

- 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.

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

```
% tranform of the unknown

syms syms y(t) t Y s g(t)

% Define g(t) using heaviside function for discontuinity
g(t) =3*heaviside(t) + (t-2)*heaviside(t-2) + (4-t)*heaviside(t-5);

% Then we define the ODE

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

% Now we compute the Laplace transform of the ODE.

L_ODE = laplace(ODE)

% Use the initial conditions

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

% 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,[0,12,0,2.25])


L_ODE =

2*s*laplace(y(t), t, s) - 2*y(0) - s*y(0) + s^2*laplace(y(t), t, s) -
 subs(diff(y(t), t), t, 0) + 5*laplace(y(t), t, s) == exp(-2*s)/s^2 +
 3/s - (exp(-5*s)*(s + 1))/s^2
```



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)
laplace(I,t,s)

% The laplace transform of the convolution of exp(-2t) and y(t) will
 be the product
% of the individual functions' laplace transformations. The laplace
% transform of exp(-2t) is 1/(s+2) and the transform for y(t) is
% laplace(y(t),t,s) in MATLAB notation. So the computed laplace should
 be
% laplace(y(t),t,s)/(s+2), which matches the MATLAB computation.
 Therefore, MATLAB knows about the convolution
% theorm and was able to correctly compute the laplace transformation.
```

```
I =

int(exp(2*tau - 2*t)*y(tau), tau, 0, t)
```

```
ans =

laplace(y(t), t, s)/(s + 2)
```

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.

```
% The integrand represents the probability of the machine failing at
 time t.
% The integration computes the mean number of failures over the time
 inverval [0,t].

% From the expected value formula, the mean/expected number of failues
 at time t, m(t), is
% int_0^t[n(t-tau)*f(tau)dt] where f(tau) is the probability of a
 failure occuring at time
% tau and n(t-tau) is the total number of failures between time tau
 and time t. Now, the mean
% number of failures between time tau and time t is m(t-tau) since the
 number of failures
% at time t is m(t). When tau=t, m(t-tau)=m(t-t)=m(0)=0 as stated in
 the exercise. If it is
% assumed failure always occurs at the interval endpoint, n(t-
tau)=1+m(t-tau).
% Therefore m(t) = int_0^t[(1+m(t-tau))*f(tau)dt].

% m(t) is defined as the convolution between (1+m) and f.
% Then the laplace transform of m(t), M, is the product of the laplace
 of (1+m) and the laplace of f.
%   i.e. M = L(1+m)*L(f)
%          = (1/s + M)*L((t^(k-1)/(k-1)!)*e^(-t))
%          = (1/s + M)*((s+1)^-k)
%          = (1/s + M)/((s+1)^k)
% Solving for M gives 1/(s((s+1)^k-1)).
% To calculate the inverse laplace of M, partial  fractions must be
 used. Note that as k increases,
```

```
% this becomes computationally intensive because the higher order
 denominator decomposes into more terms.
% Therefore, increasing k will result in an increase in MATLAB's
 computation time for L^-1(M).
% This effect starts to become noticable when k equals 5.

% Define variables
syms t tau f(t) ma(t) mb(t) Ma Mb

% Probability function is exp(-t)
f = exp(-t); % Probability distribution function
eq = ma(t)-int((ma(t-tau)+1)*subs(f,t,tau),tau,0,t)==0; % ODE form of
 m(t)
leq = laplace(eq); % Laplace transform
leq = subs(leq,laplace(ma),Ma); % Replace laplace variable
Ma = solve(leq,Ma); % Solve laplace equation
ma = ilaplace(Ma) % Take the inverse laplace
mean = subs(int(t*f,t,0,inf)) % Mean of f(t)
check = subs(ma/t,t,inf) % ma/t as t->inf
% Elementary renewal theorem is verified

% Probability function is t^(k-1)/(k-1)! exp(-t)
k = 5; % Let k = 5
f = (t^(k-1)/factorial(k-1)) * exp(-t); % Define probability
 distribution function
eq = mb(t)-int((mb(t-tau)+1)*subs(f,t,tau),tau,0,t)==0; % ODE form of
 m(t)
leq = laplace(eq); % Laplace transform
leq = subs(leq,laplace(mb),Mb); % Replace laplace variable
Mb = solve(leq,Mb); % Solve laplace equation
mb = vpa(simplify(ilaplace(Mb))) % Take the inverse laplace, vpa is
 used to simplify the equation but introduces computation error
mean = subs(int(t*f,t,0,inf)) % Mean of f(t)
check = subs(mb/t,t,1e20) % A large number is used to represent
 infinity as the integral becomes too complex for MATLAB to compute
% Elementary renewal theorem is verified
```

```
ma =

t


mean =

1


check =

1


mb =
```

```
0.2*t +
 exp(-1.8090169943749474241022934171828*t)*cos(0.58778525229247312916870595463907*
 + 0.032491969623290632615587141221513i) +
 exp(-1.8090169943749474241022934171828*t)*cos(0.58778525229247312916870595463907*
 - 0.032491969623290632615587141221513i) +
 exp(-1.8090169943749474241022934171828*t)*sin(0.58778525229247312916870595463907*
 - 0.1i) +
 exp(-1.8090169943749474241022934171828*t)*sin(0.58778525229247312916870595463907*
 + 0.1i) +
 exp(-0.69098300562505257589770658281718*t)*cos(0.95105651629515357211643933337938
 + 0.13763819204711735382072095819109i) +
 exp(-0.69098300562505257589770658281718*t)*cos(0.95105651629515357211643933337938
 - 0.13763819204711735382072095819109i) +
 exp(-0.69098300562505257589770658281718*t)*sin(0.95105651629515357211643933337938
 - 0.1i) +
 exp(-0.69098300562505257589770658281718*t)*sin(0.95105651629515357211643933337938
 + 0.1i) - 0.4


mean =

5


check =

0.19999999999999999996
```

Published with MATLAB® R2019a