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: Hshmat Sahak

Student Number: 1005903710
```

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,y)

% 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/(t - y)
```

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. part a

```
syms f F G s
f = exp(2*t)*t^3;
F = laplace(f)

% part b
G = ((s - 1)*(s - 2))/(s*(s + 2)*(s - 3));
f = ilaplace(G)

% part c
% We first declare symbolic function variable f, as well as symbolic
% variable a. We compute the result of |laplace(f(t))|. This gives
```

```
% laplace(f(t), t, s-a). Note that this verifies the relation as
 F(s)=laplace(f(t), t, s)
% => F(s-a) = laplace(f(t), t, s-a)
syms f(t) a
F_shift = laplace(exp(a*t)*f(t))
```

```
F =

6/(s - 2)^4


f =

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


F_shift =

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

Heaviside and Dirac functions

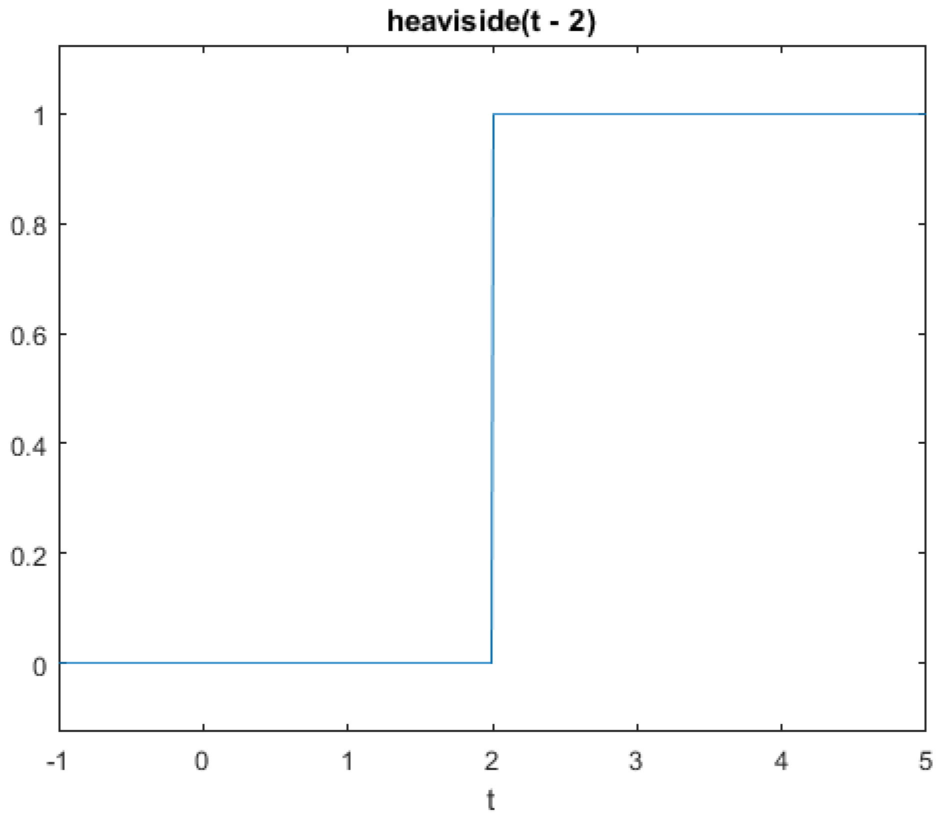
These two functions are builtin to MATLAB: `heaviside` is the Heaviside function `u_0(t)` at `0`

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.

```
a = 5;
syms f(t) g(t) F(s);
g(t) = heaviside(t-5)*f(t-5)
G = laplace(g(t))
```

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

% By observation(testing with different values of a), the formula is:
% G(s)=exp(-as)*F(s)
% This makes sense. We have seen already in lecture that a shift by a
 in
% the cartesian coordinates is equivalent to multiplication by exp(-
as) in
% the Laplace coordinates.
```

```
g(t) =

f(t - 5)*heaviside(t - 5)


G =

exp(-5*s)*laplace(f(t), t, s)


F =

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

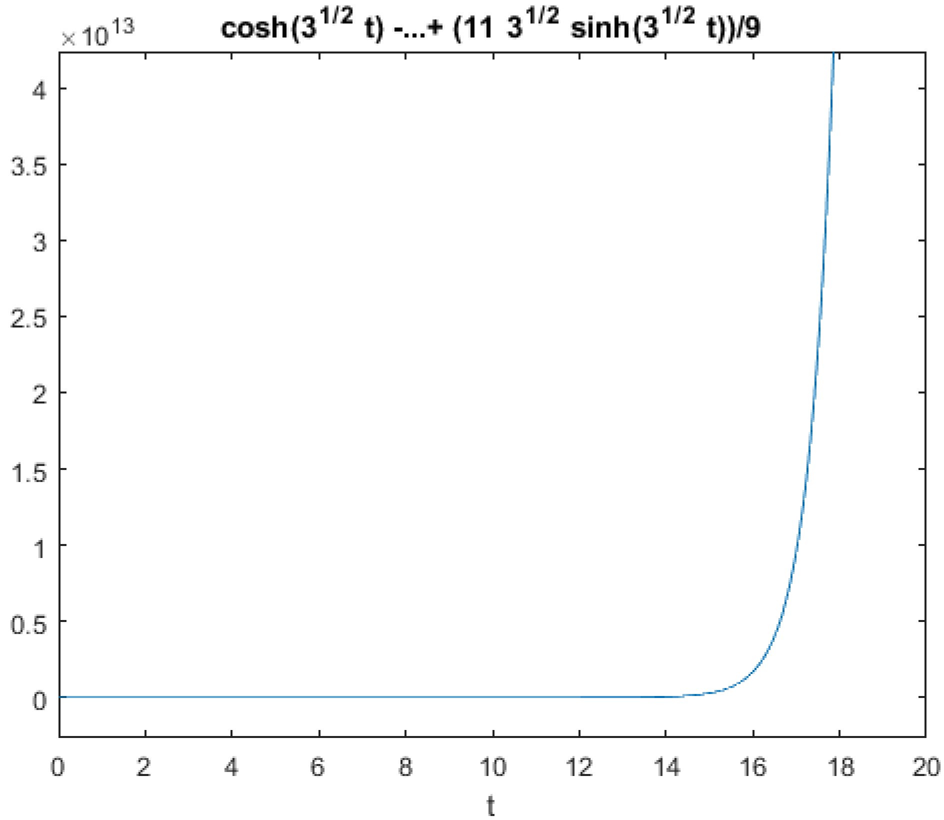
Solving IVPs using Laplace transforms

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

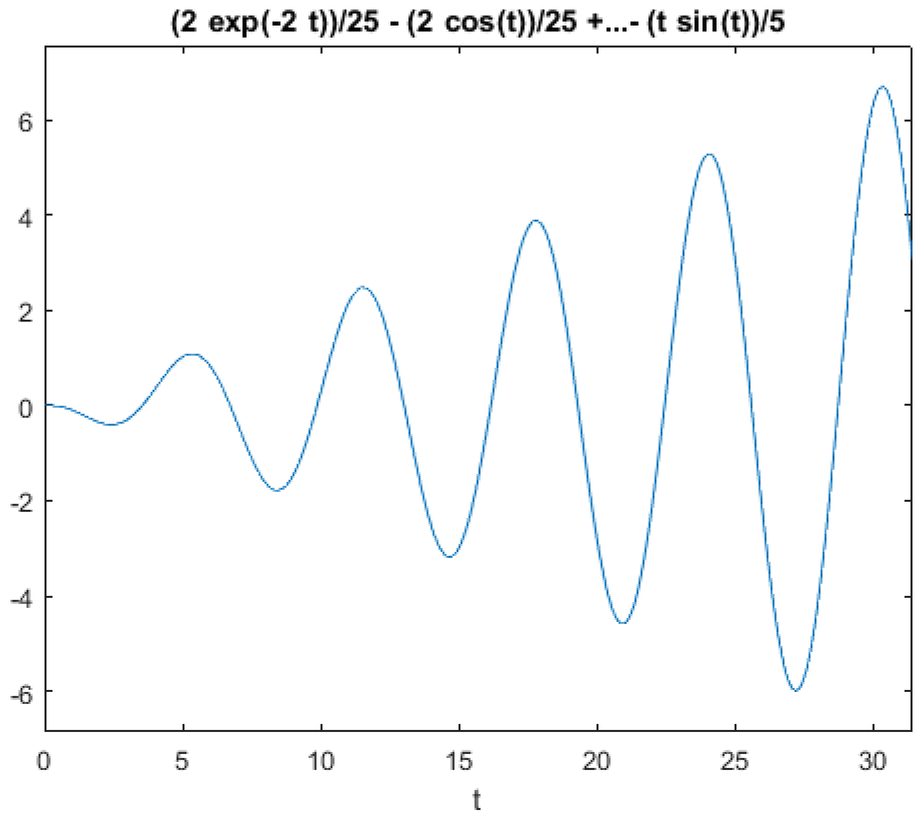
Details: Explain your steps using comments

- 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` emains 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, t), 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

% There is no intial condition for which y remains bounded as t goes
 to
% infinity. The general solution using the same method above, but
% commenting out L_ODE=subs(L_ODE,y(0),0) gives the solution as:
% y = (3*sin(t))/50 - (2*cos(t))/25 + (t*cos(t))/10 +
 (4*y(0)*cos(t))/5 - (t*sin(t))/5 + (2*y(0)*sin(t))/5 +
 exp(-2*t)*(y(0)/5 + 2/25)
% All terms here are either constant or decaying, except (t*cos(t))/10
 and
% -t*sin(t)/5. Since these do not depend on y(0), if one solution goes
 to
% inifinity as t->inf, all others will as those two terms would have
 that
% effect in all cases


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:

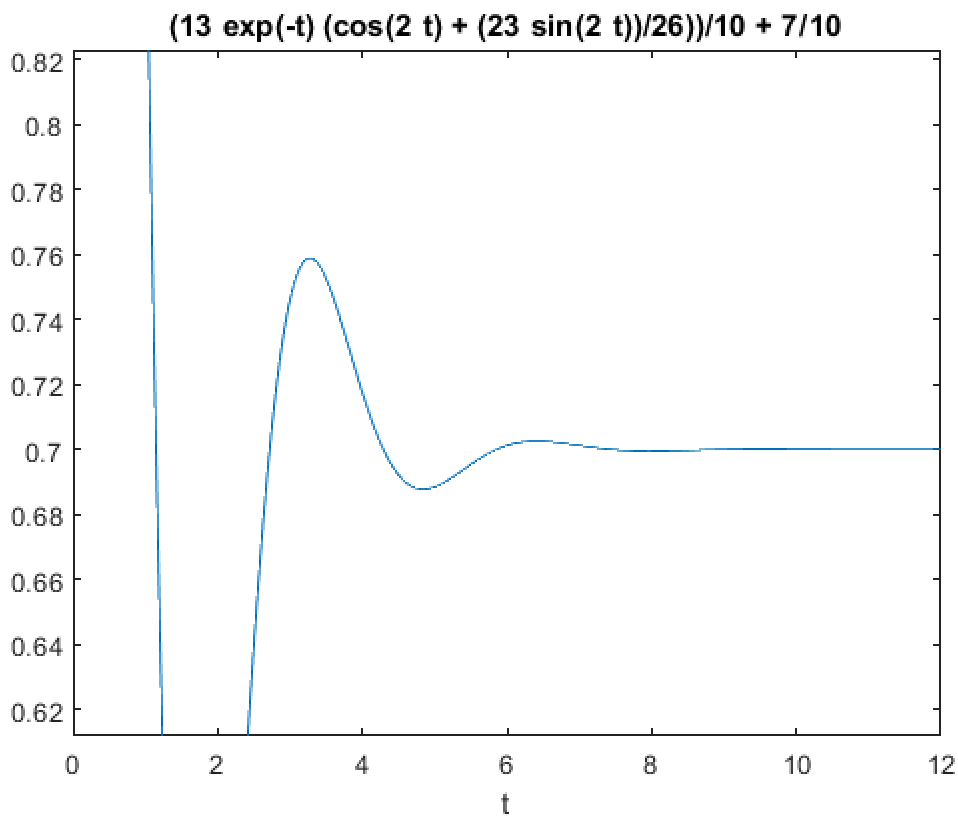
- Define
- `g(t) = 3 if 0 < t < 2`
- `g(t) = t+1 if 2 < t < 5`
- `g(t) = 5 if t > 5`
- Solve the IVP
- `y''+2y'+5y=g(t)`
- `y(0)=2 and y'(0)=1`
- Plot the solution for `t` in `[0,12]` and `y` in `[0,2.25]`.

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 y(t) t Y s g(t)
```

```
%Define g(t)
g(t) = 3*(heaviside(0)-heaviside(2)) + (t+1)*(heaviside(2)-
heaviside(5)) + 5*(heaviside(5))

% Then we define the ODE
ODE=diff(y(t),t,2)+2*diff(y(t),t,1)+5*y(t)-g(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),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])

% We can check that this is indeed the solution
diff(y,t,3)+2*diff(y,t,2)+diff(y,t,1)+2*y


g(t) =

7/2


ODE =

5*y(t) + 2*diff(y(t), t) + diff(y(t), t, t) - 7/2 == 0


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) - 7/(2*s) + 5*laplace(y(t), t, s) == 0


L_ODE =

2*s*laplace(y(t), t, s) - 2*s + s^2*laplace(y(t), t, s) -
 subs(diff(y(t), t), t, 0) - 7/(2*s) + 5*laplace(y(t), t, s) - 4 == 0


L_ODE =
```

```
2*s*laplace(y(t), t, s) - 2*s + s^2*laplace(y(t), t, s) - 7/(2*s) +
 5*laplace(y(t), t, s) - 5 == 0


L_ODE =

5*Y - 2*s + 2*Y*s + Y*s^2 - 7/(2*s) - 5 == 0


Y =

(2*s + 7/(2*s) + 5)/(s^2 + 2*s + 5)


y =

(13*exp(-t)*(cos(2*t) + (23*sin(2*t))/26))/10 + 7/10


ans =

(13*exp(-t)*(4*cos(2*t) + (46*sin(2*t))/13))/10 - (13*exp(-
t)*((92*cos(2*t))/13 - 8*sin(2*t)))/10 + (13*exp(-t)*(cos(2*t) +
 (23*sin(2*t))/26))/5 + 7/5
```



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)
% Consider the expression I=int(exp(-2*(t-tau))*y(tau),tau,0,t). Note
 that
% the RHS is just the convolution of f(t)=e^-2t and y(t). So we are
 taking
% the laplace transform of the convolution of f and y. By convolution
% theorem, this is equal to the product of the laplace transform of y
 and
% e^-2t. The laplace transform of y is Y=laplace(y(t), t, s) and the
% laplace transform of e^-2t is 1/(s+2). The product of these matches
 the
% answer produced by MATLAB
```

```
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 integral 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 mean number of failures satisfies the integral equation for
 reasons
```

```
% that were never taught in lecture. Specifically, the mean number of
% failures is equal to the integral from time 0 to time t of the
% probability that we see a failure, multiplied by the mean number of
% failures starting from that point. In a more mathematical sense, the
% probability we get a failure at time t' in [0, t] is f(tau=t)dtau.
 The
% expected value of mean number of failures from time=0 to time=t
% associated with this event is then 1+m(t-tau), which represents 1
 for
% failure we have already seen, and them m(t-tau) is number of
 failures
% since that first failure, which is valid since the probability
 density
% function is identically distributed.

% solve renewal equation
syms t tau f(t) ma(t) mb(t) Ma Mb k
f(t) = exp(-1*t)
IE = ma(t) - int((1+ma(t-tau))*f(tau),tau,0,t) == 0
L_IE = laplace(IE)
L_IE = subs(L_IE,laplace(ma(t), t, s), Ma)
Ma=solve(L_IE,Ma)
ma(t) = ilaplace(Ma)

% verify renewal theorem
limit(ma(t)/t, t, inf)
limit(1/int(t*f(t), 0, t), t, inf)

% Solve renewal equation
k = 4; % Give k a value
f(t) = (t^(k-1)/factorial(k-1))*exp(-t)
IE2 = mb(t) - int((1+mb(t-tau))*f(tau),tau,0,t) == 0
L_IE2 = laplace(IE2)
L_IE2 = subs(L_IE2,laplace(mb(t), t, s), Mb)
Mb=solve(L_IE2,Mb)
mb(t) = ilaplace(Mb)

% Verify renewal theorem
limit(mb(t)/t, t, inf)
limit(1/int(t*f(t), 0, t), t, inf)

%Now try k=5
syms t tau f(t) mb(t) Mb k
k = 5; % Give k a value
f(t) = (t^(k-1)/factorial(k-1))*exp(-t)
IE2 = mb(t) - int((1+mb(t-tau))*f(tau),tau,0,t) == 0
L_IE2 = laplace(IE2)
L_IE2 = subs(L_IE2,laplace(mb(t), t, s), Mb)
Mb=solve(L_IE2,Mb)
mb(t) = ilaplace(Mb)
% m = (1+m)*f (* denotes convolution)
% M = L(m) = L(1+m)L(f)
% M = (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, M = 1/(s((s+1)^k-1)).
```

```
% To calculate the inverse Laplace transform:
% the method of partial fractions is required
% as k increases this becomes harder to compute as decomposition
 contains more terms (e.g. decomposition of (s+1)^k will
% contain k terms with denominators s+1, (s+1)^2, (s+1)^3, ... , (s
+1)^k).
% k increases => the computation time of L^-1(M) increases.


f(t) =

exp(-t)


IE =

ma(t) - int(exp(-tau)*(ma(t - tau) + 1), tau, 0, t) == 0


L_IE =

laplace(ma(t), t, s) - (1/s + laplace(ma(t), t, s))/(s + 1) == 0


L_IE =

Ma - (Ma + 1/s)/(s + 1) == 0


Ma =

1/s^2


ma(t) =

t


ans =

1


ans =

1


f(t) =

(t^3*exp(-t))/6
```

```
IE2 =

mb(t) - int((tau^3*exp(-tau)*(mb(t - tau) + 1))/6, tau, 0, t) == 0


L_IE2 =

laplace(mb(t), t, s) - (1/s + laplace(mb(t), t, s))/(s + 1)^4 == 0


L_IE2 =

Mb - (Mb + 1/s)/(s + 1)^4 == 0


Mb =

1/(s^5 + 4*s^4 + 6*s^3 + 4*s^2)


mb(t) =

t/4 + exp(-2*t)/8 + (exp(-t)*(cos(t) + sin(t)))/4 - 3/8


ans =

1/4


ans =

1/4


f(t) =

(t^4*exp(-t))/24


IE2 =

mb(t) - int((tau^4*exp(-tau)*(mb(t - tau) + 1))/24, tau, 0, t) == 0


L_IE2 =

laplace(mb(t), t, s) - (1/s + laplace(mb(t), t, s))/(s + 1)^5 == 0


L_IE2 =

Mb - (Mb + 1/s)/(s + 1)^5 == 0
```

```
Mb =

-1/(s*(1/(s + 1)^5 - 1)*(s + 1)^5)


mb(t) =

t/5 + (9*symsum((exp(t*root(s3^4 + 5*s3^3 + 10*s3^2 + 10*s3 + 5, s3,
 k))*root(s3^4 + 5*s3^3 + 10*s3^2 + 10*s3 + 5, s3, k)^2)/(15*root(s3^4
 + 5*s3^3 + 10*s3^2 + 10*s3 + 5, s3, k)^2 + 4*root(s3^4 + 5*s3^3 +
 10*s3^2 + 10*s3 + 5, s3, k)^3 + 20*root(s3^4 + 5*s3^3 + 10*s3^2 +
 10*s3 + 5, s3, k) + 10), k, 1, 4))/5 + (2*symsum((exp(t*root(s3^4 +
 5*s3^3 + 10*s3^2 + 10*s3 + 5, s3, k))*root(s3^4 + 5*s3^3 + 10*s3^2
 + 10*s3 + 5, s3, k)^3)/(15*root(s3^4 + 5*s3^3 + 10*s3^2 + 10*s3
 + 5, s3, k)^2 + 4*root(s3^4 + 5*s3^3 + 10*s3^2 + 10*s3 + 5, s3,
 k)^3 + 20*root(s3^4 + 5*s3^3 + 10*s3^2 + 10*s3 + 5, s3, k) + 10),
 k, 1, 4))/5 + 2*symsum(exp(t*root(s3^4 + 5*s3^3 + 10*s3^2 + 10*s3
 + 5, s3, k))/(15*root(s3^4 + 5*s3^3 + 10*s3^2 + 10*s3 + 5, s3,
 k)^2 + 4*root(s3^4 + 5*s3^3 + 10*s3^2 + 10*s3 + 5, s3, k)^3 +
 20*root(s3^4 + 5*s3^3 + 10*s3^2 + 10*s3 + 5, s3, k) + 10), k, 1,
 4) + 3*symsum((exp(root(s3^4 + 5*s3^3 + 10*s3^2 + 10*s3 + 5, s3,
 k)*t)*root(s3^4 + 5*s3^3 + 10*s3^2 + 10*s3 + 5, s3, k))/(20*root(s3^4
 + 5*s3^3 + 10*s3^2 + 10*s3 + 5, s3, k) + 15*root(s3^4 + 5*s3^3 +
 10*s3^2 + 10*s3 + 5, s3, k)^2 + 4*root(s3^4 + 5*s3^3 + 10*s3^2 +
 10*s3 + 5, s3, k)^3 + 10), k, 1, 4) - 2/5
```

Published with MATLAB® R2020a