Optimization of Complex Systems

Individual Task 2

Report

Fedak Kate

2019

1. Sensitivity analysis

In this task sensitivity analysis is used to calculate the gradient of the objective function and the constraints to improve quality of optimization.

1.1 Direct differentiation method (DDM)

As for the continuous case we will consider direct derivative of the characteristic functions.

$$\begin{cases} \dfrac{\partial\psi_0}{\partial b_i} = \displaystyle\int_{t_0}^{T} 2(\mathrm{y}_2(\mathrm{t}) - \mathrm{y}_d)\frac{\partial y}{\partial b_i}dt \\ \dfrac{\partial\psi_1}{\partial b_i} = \displaystyle\int_{t_0}^{T} 2(\left|\mathrm{y}_1(\mathrm{t}) - \mathrm{y}^+\right| + \mathrm{y}_1(\mathrm{t}) - \mathrm{y}^+)(\mathrm{sgn}(\mathrm{y}_1(\mathrm{t}) - \mathrm{y}^+) + 1)\frac{\partial y}{\partial b_i}dt \end{cases}$$

Function $y_1(\mathrm{t}), y_2(\mathrm{t})$ we can find from direct problem. The only unknown term in these equations is $\frac{\partial y}{\partial b_i}$. We can find it solve next problem.

<u>SA expressions for DDM</u>

$$\frac{d\psi}{db} = \left.\frac{\partial g_3}{\partial b}\right|_{t=T} + \int_{t_0}^{T}\frac{\partial g_1}{\partial b}dt + \left.\frac{\partial g_3}{\partial y}\frac{dy}{db}\right|_{t=T} + \int_{t_0}^{T}\frac{\partial g_1}{\partial y}\frac{dy}{db}dt$$

$$\frac{d}{dt}\left(\frac{dy}{db}\right) = \frac{\partial f}{\partial b} + \frac{\partial f}{\partial y}\frac{dy}{db}$$

$$\left.\frac{dy}{db}\right|_{t=t_0} = 0$$

$$\psi_0 = \int_{t_0}^{T}(\mathrm{y}_2(\mathrm{t}) - \mathrm{y}_d)^2 dt \Rightarrow g_1 = (\mathrm{y}_2(\mathrm{t}) - \mathrm{y}_d)^2$$

$$g_3 = 0$$

$$\frac{\partial f}{\partial b}=\begin{bmatrix}\frac{\partial f_1}{\partial b}\\ \frac{\partial f_2}{\partial b}\end{bmatrix}=\begin{bmatrix}0\\ p_4 y_1(1-\mathrm{e}^{-p_5 y_1})\frac{y_2}{p_6+p_7 y_2}\end{bmatrix}$$

$$\frac{\partial f}{\partial y}=\begin{bmatrix}\frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2}\\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2}\end{bmatrix}=$$

$$=\begin{bmatrix} p_1+\frac{p_4 y_2}{p_6+p_7 y_2}\left(1-e^{-p_5 y_1}+p_5 y_1 e^{-p_5 y_1}\right) & p_4 y_1\left(1-e^{-p_5 y_1}\right)\frac{p_4}{(p_6+p_7 y_2)^2}\\ \frac{p_8 p_4 y_2}{p_6+p_7 y_2}\left(1-e^{-p_5 y_1}+p_5 y_1 e^{-p_5 y_1}\right) & p_2+2p_3 y_2+p_8 p_4 y_1\left(1-e^{-p_5 y_1}\right)\frac{p_6}{(p_6+p_7 y_2)^2}\end{bmatrix}$$

$$\frac{\partial g_1}{\partial b}=0;\frac{\partial g_1}{\partial y}=\begin{bmatrix}0 & 2y_2(\mathrm{t})-1\end{bmatrix};$$

So,

$$\frac{dy}{db_i}=z,$$

$$\frac{d}{dt}(\mathrm{z})=\begin{bmatrix}z_1^{'}\\ z_2^{'}\end{bmatrix}=\frac{\partial f}{\partial b_i}\frac{\partial u}{\partial b_i}+\frac{\partial f}{\partial y}^{T}\begin{bmatrix}z_1\\ z_2\end{bmatrix}$$

$$\begin{cases} z_1^{'}=\frac{\partial f_1}{\partial b_i}\frac{\partial u}{\partial b_i}+\frac{\partial f_1}{\partial y_1}z_1+\frac{\partial f_1}{\partial y_2}z_2\\ z_2^{'}=\frac{\partial f_2}{\partial b_i}\frac{\partial u}{\partial b_i}+\frac{\partial f_2}{\partial y_1}z_1+\frac{\partial f_2}{\partial y_2}z_2\\ \quad z_1\big|_{t=t_0}=z_2\big|_{t=t_0}=0\end{cases}$$

We have new to the discrete case unknown $\frac{\partial u}{\partial b_i}$. It can be obtained from the form of u(x).

1.2. Adjoint method (AM)

SA expressions for AM

$$\frac{d\psi}{db} = \left.\frac{\partial g_3}{\partial b}\right|_{t=T} + \int_{t_0}^{T}\left(\frac{\partial g_1}{\partial b} + \mu^T \frac{\partial f}{\partial b}\right)dt$$

$$\mu' = -\left(\frac{\partial f}{\partial y}\right)^T \mu - \left(\frac{\partial g_1}{\partial y}\right)^T$$

$$\mu(\mathrm{T}) = \left.\left(\frac{\partial g_3}{\partial y}\right)^T\right|_{t=T} = 0$$

$$\begin{bmatrix}\mu'_1\\ \mu'_2\end{bmatrix} = -\begin{bmatrix}\frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2}\\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2}\end{bmatrix}\begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix} - \begin{bmatrix}0 & 2y_2 - 1\end{bmatrix}^T$$

$$\begin{cases}\mu_1' = -\frac{\partial f_1}{\partial y_1}\mu_1 - \frac{\partial f_1}{\partial y_2}\mu_2\\ \mu_2' = -\frac{\partial f_2}{\partial y_1}\mu_1 - \frac{\partial f_2}{\partial y_2}\mu_2 - (2\,\mathrm{y}_2 - 1)\\ \mu_1|_{t=T} = \mu_2|_{t=T} = 0\end{cases}$$

And finally, sensitivity coefficients will be:

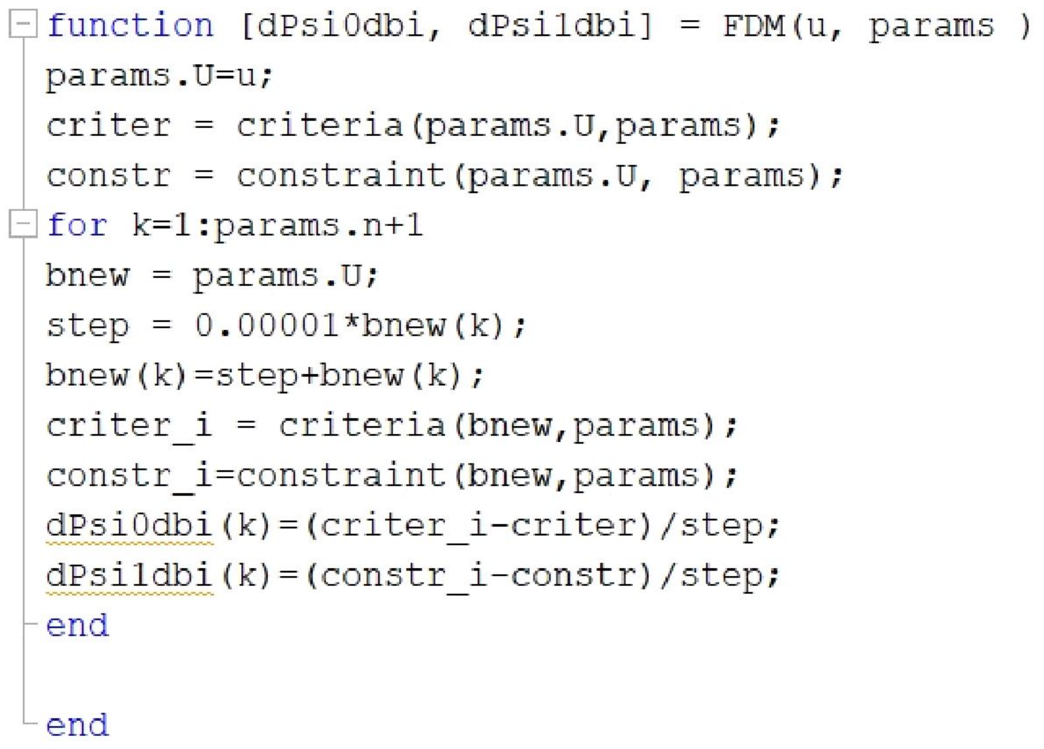
$$\begin{cases}\frac{\partial \psi_0}{\partial b_i} = \int_{t_0}^{T}\frac{\partial u}{\partial b_i}\mu_0 dt\\ \frac{\partial \psi_1}{\partial b_i} = \int_{t_0}^{T}\frac{\partial u}{\partial b_i}\mu_1 dt\end{cases}$$

2. Gradient calculation

In this section we will use Matlab to calculate the gradient with different methods and compare results.

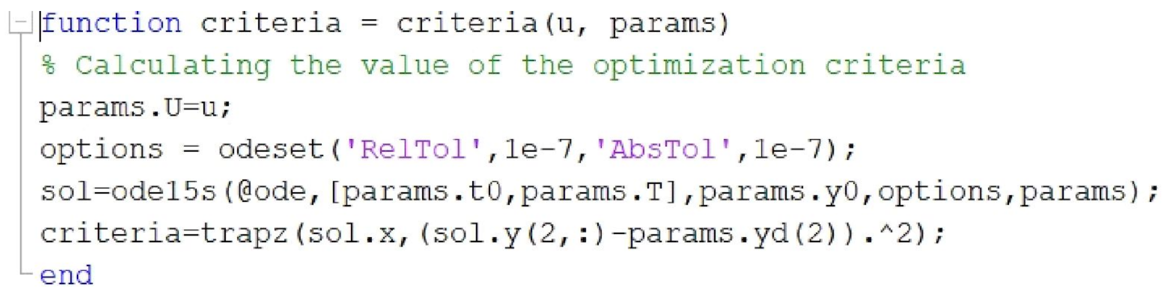
2.1 Finite difference method (FDM)

Although, when nothing is given to fmincon it uses optimized version of finite difference method, we need something to compare our DDM and AM results with. This small helper method will numerically calculate the gradient by taking the difference in function's value divided by difference in function's argument. By default we set this difference to be 0.01.

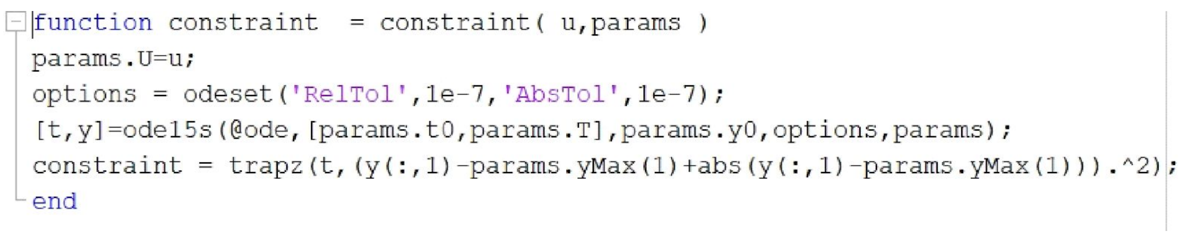


```
function [dPsi0dbi, dPsi1dbi] = FDM(u, params )
params.U=u;
criter = criteria(params.U,params);
constr = constraint(params.U, params);
for k=1:params.n+1
bnew = params.U;
step = 0.00001*bnew(k);
bnew(k)=step+bnew(k);
criter_i = criteria(bnew,params);
constr_i=constraint(bnew,params);
dPsi0dbi(k)=(criter_i-criter)/step;
dPsi1dbi(k)=(constr_i-constr)/step;
end

end
```



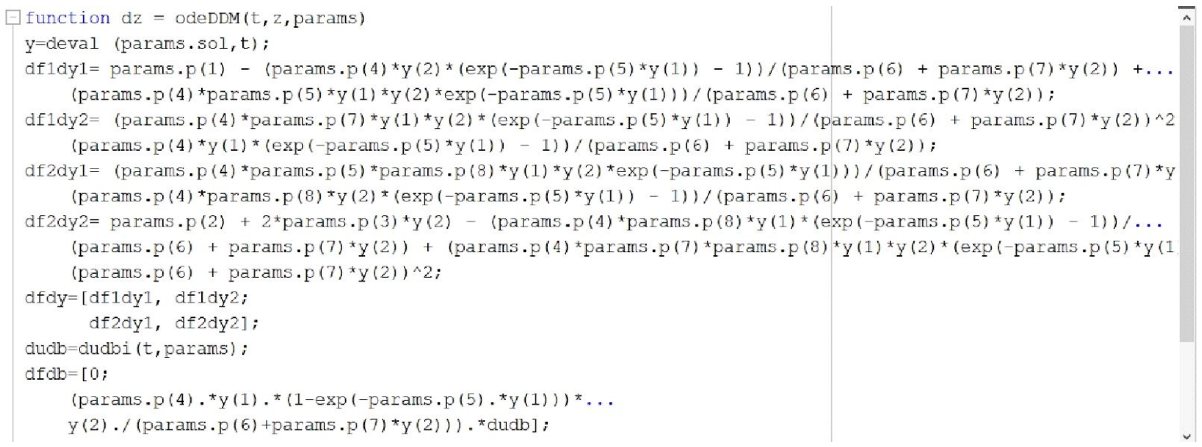
```
function criteria = criteria(u, params)
% Calculating the value of the optimization criteria
params.U=u;
options = odeset('RelTol',1e-7,'AbsTol',1e-7);
sol=ode15s(@ode,[params.t0,params.T],params.y0,options,params);
criteria=trapz(sol.x,(sol.y(2,:)-params.yd(2)).^2);
end
```



```
function constraint  = constraint( u,params )
params.U=u;
options = odeset('RelTol',1e-7,'AbsTol',1e-7);
[t,y]=ode15s(@ode,[params.t0,params.T],params.y0,options,params);
constraint = trapz(t,(y(:,1)-params.yMax(1)+abs(y(:,1)-params.yMax(1))).^2);
end
```

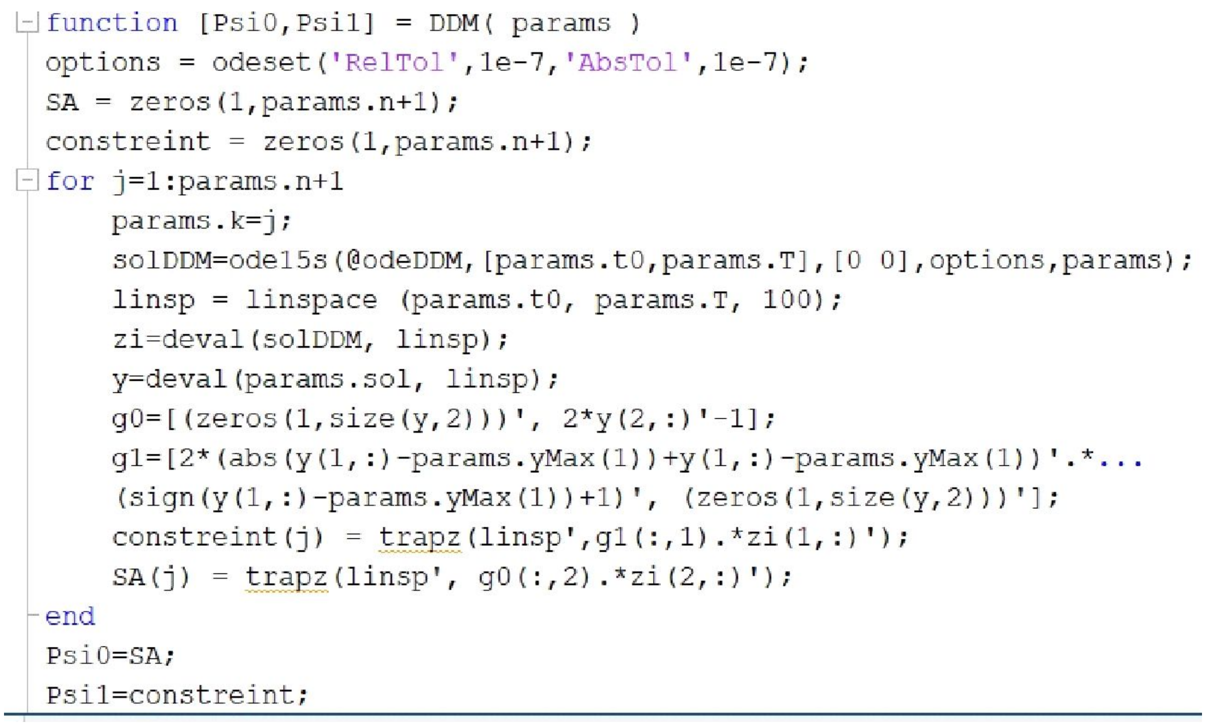
2.2 Direct differentiation method (DDM)

In this method we need to solve the following ODE(converted to system of first-order ODEs) with 0 initial value:



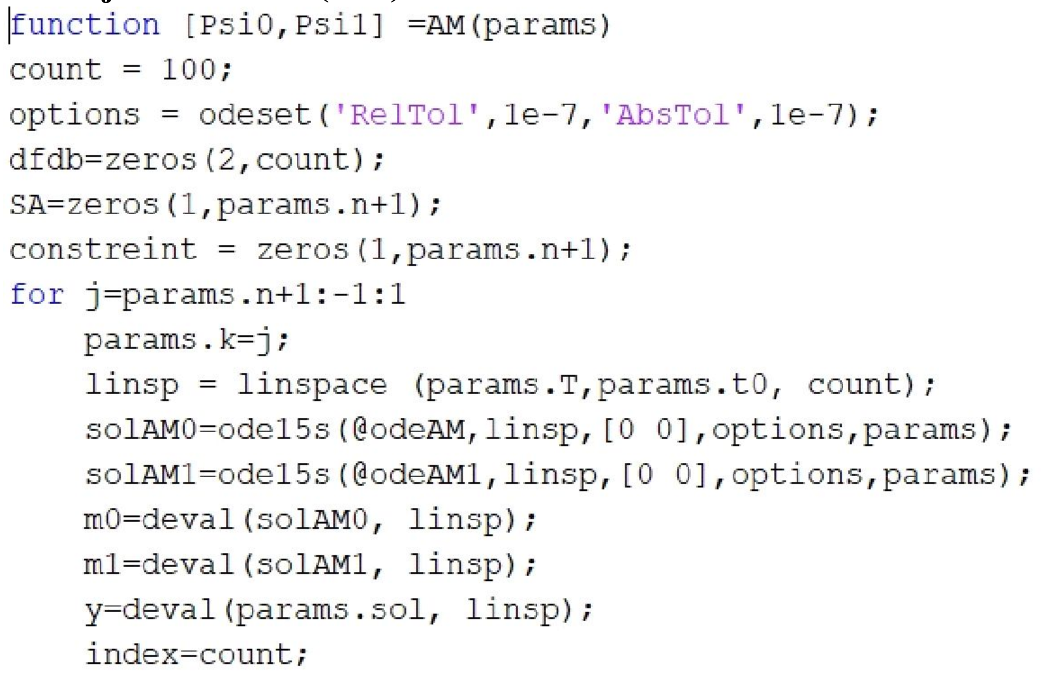
```
function dz = odeDDM(t,z,params)
y=deval (params.sol,t);
df1dy1= params.p(1) - (params.p(4)*y(2)*(exp(-params.p(5)*y(1)) - 1))/(params.p(6) + params.p(7)*y(2)) +...
    (params.p(4)*params.p(5)*y(1)*y(2)*exp(-params.p(5)*y(1)))/(params.p(6) + params.p(7)*y(2));
df1dy2= (params.p(4)*params.p(7)*y(1)*y(2)*(exp(-params.p(5)*y(1)) - 1))/(params.p(6) + params.p(7)*y(2))^2
    (params.p(4)*y(1)*(exp(-params.p(5)*y(1)) - 1))/(params.p(6) + params.p(7)*y(2));
df2dy1= (params.p(4)*params.p(5)*params.p(8)*y(1)*y(2)*exp(-params.p(5)*y(1)))/(params.p(6) + params.p(7)*y
    (params.p(4)*params.p(8)*y(2)*(exp(-params.p(5)*y(1)) - 1))/(params.p(6) + params.p(7)*y(2));
df2dy2= params.p(2) + 2*params.p(3)*y(2) - (params.p(4)*params.p(8)*y(1)*(exp(-params.p(5)*y(1)) - 1))/...
    (params.p(6) + params.p(7)*y(2)) + (params.p(4)*params.p(7)*params.p(8)*y(1)*y(2)*(exp(-params.p(5)*y(1
    (params.p(6) + params.p(7)*y(2))^2;
dfdy=[df1dy1, df1dy2;
      df2dy1, df2dy2];
dudb=dudbi(t,params);
dfdb=[0;
    (params.p(4).*y(1).*(1-exp(-params.p(5).*y(1)))*...
    y(2)./(params.p(6)+params.p(7)*y(2))).*dudb];
```

Next methods use trapz function to calculate sensitivity coefficients for this method by formulas from the previous section.

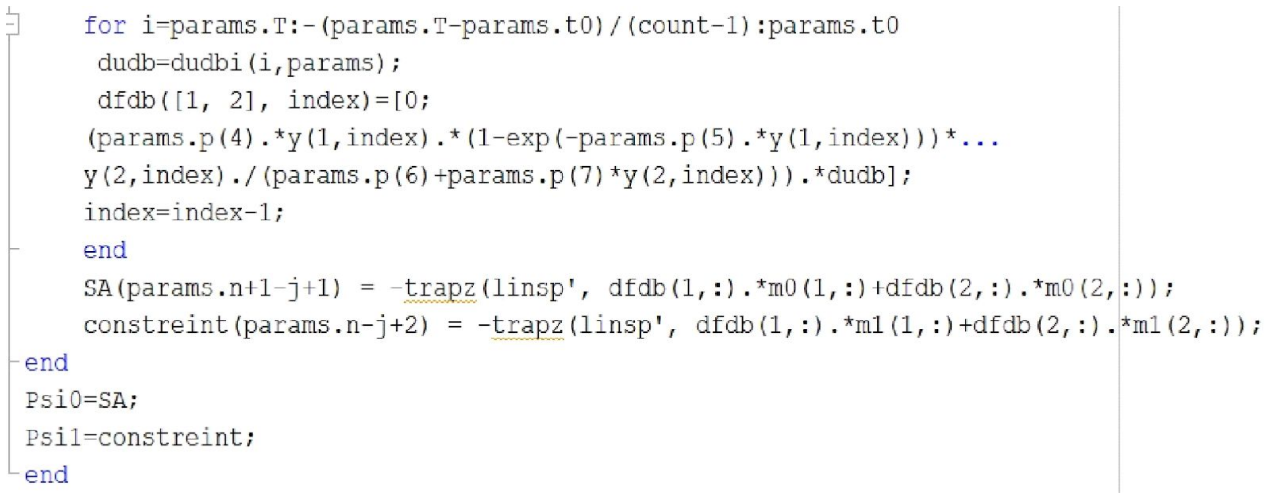


```
function [Psi0,Psi1] = DDM( params )
options = odeset('RelTol',1e-7,'AbsTol',1e-7);
SA = zeros(1,params.n+1);
constreint = zeros(1,params.n+1);
for j=1:params.n+1
    params.k=j;
    solDDM=ode15s(@odeDDM,[params.t0,params.T],[0 0],options,params);
    linsp = linspace (params.t0, params.T, 100);
    zi=deval(solDDM, linsp);
    y=deval(params.sol, linsp);
    g0=[(zeros(1,size(y,2)))', 2*y(2,:)'-1];
    g1=[2*(abs(y(1,:)-params.yMax(1))+y(1,:)-params.yMax(1))'.*...
    (sign(y(1,:)-params.yMax(1))+1)', (zeros(1,size(y,2)))'];
    constreint(j) = trapz(linsp',g1(:,1).*zi(1,:)');
    SA(j) = trapz(linsp', g0(:,2).*zi(2,:)');
end
Psi0=SA;
Psi1=constreint;
```

2.3 Adjoint method (AM)



```
function [Psi0,Psi1] =AM(params)
count = 100;
options = odeset('RelTol',1e-7,'AbsTol',1e-7);
dfdb=zeros(2,count);
SA=zeros(1,params.n+1);
constreint = zeros(1,params.n+1);
for j=params.n+1:-1:1
    params.k=j;
    linsp = linspace (params.T,params.t0, count);
    solAM0=ode15s(@odeAM,linsp,[0 0],options,params);
    solAM1=ode15s(@odeAM1,linsp,[0 0],options,params);
    m0=deval(solAM0, linsp);
    m1=deval(solAM1, linsp);
    y=deval(params.sol, linsp);
    index=count;
```



```
    for i=params.T:-(params.T-params.t0)/(count-1):params.t0
     dudb=dudbi(i,params);
     dfdb([1, 2], index)=[0;
    (params.p(4).*y(1,index).*(1-exp(-params.p(5).*y(1,index)))*...
    y(2,index)./(params.p(6)+params.p(7)*y(2,index))).*dudb];
    index=index-1;
    end
    SA(params.n+1-j+1) = -trapz(linsp', dfdb(1,:).*m0(1,:)+dfdb(2,:).*m0(2,:));
    constreint(params.n-j+2) = -trapz(linsp', dfdb(1,:).*m1(1,:)+dfdb(2,:).*m1(2,:));
end
Psi0=SA;
Psi1=constreint;
end
```

2.4 Results

The results of gradient evaluation for initial values of control function are given in the table 1. We can find gradient by calling the function *fmincon* with `'MaxFunEvals' equals 0.` We will test the algorithm for n = 2, 3,5 and constant values of control function bi = -0.1.

	n	2	3	5
Grad (*fmincon*)		[3.3402; -1.0479]	[3.8458; -1.0111; -0.5423]	[4.9299; -2.1681;-0.2781;0.7024 -0.8935]
		[1.0e+03 * 1.4261; 1.0e+03 * 0.1753]	[1.0e+03 * 1.2508; 1.0e+03 * 0.3507; 0]	[900.4901;700.6336;0.3380;0;0]
	time	0.425857	0.406944	0.588621

	n	2	3	5
FDM		[3.3402; -1.0479]	[3.8481; -1.0088; -0.5399]	[4.9298 ; -2.1639 ; 0.9350 ; 0.7045 ; -0.8935]
		[1.0e+03 *1.4261; 1.0e+03 * 0.1753]	[1.0e+03 * 1.2508 ; 1.0e+03 * 0.3506 ; 0]	[900.4660; 700.6091; 0.3380; 0; 0]
	time	0.293733	0.362454	0.548099

	n	2	3	5
DDM		[3.3394; -1.0476]	[3.8355; -0.9923; -0.5515]	[4.9289; -2.1868; -0.2379; 0.6780; -0.8905]
		[1.0e+03 * 1.4272 ; 1.0e+03 * 0.1755]	[1.0e+03 * 1.2516; 1.0e+03 * 0.3511; 0]	[900.9275; 701.4399 ; 0.3409 ; 0 ;0]
	time	0.477773	0.505248	0.866145

	n	2	3	5
AM		[3.3387; -1.0471]	[3.8351 ; -0.9927; -0.5507]	[4.9291; -2.1880; -0.2378; 0.6783 ; -0.8899]
		[1.0e+03 * 1.4255; 1.0e+03 * 0.1752]	[1.0e+03 * 1.2503 ; 1.0e+03 * 0.3505; 0]	[900.1366; 700.2616 ; 0.3372 ; 0 ; 0]
	time	0.681703	0.776904	1.277655

Table 1.(SA coef)

I've solved the optimization problem with different sets of constraints using gradient evaluation obtained by different methods. It is easy to see that all the methods used give the gradients values close to the values calculated by *fmincon* function.

3. Solving optimization problem

Optimization is done by the inbuilt method fmincon the same way as in
the Individual task 1. The only difference is additional options to instruct the solver to use the gradient calculated by us.

3.1 Without integral constraint

Method	b*			time
FDM	[-0.1100 ;-0.0900]	0.1450	[2.1675 ; -0.8779]	3.424794
DDM	[-0.1100 ;-0.0900]	0.1450	[2.0922; -0.9809]	4.257516
AM	[-0.1098 ;-0.0911]	0.1465	[4.1984;-0.7278]	5.703713

Table 2. Results in case without constraint, n=2;

Method	b*			time
FDM	[-0.1099 ;-0.0901; -0.0909 ;-0.0993; -0.0904]	0.1133	[2.5565;-2.2776 -0.6597;-0.3276; -0.5208]	45.684944
DDM	[-0.1097 ;-0.0911 -0.0935;-0.1011;- 0.0917]	0.1168	[3.6651;-0.8484 -0.2235;0.0197; -0.5205]	29.935747
AM	-0.1100 ;-0.0900; -0.0914 ;-0.1100; -0.0900]	0.1162	[5.0346;-2.0235; -0.0206; 1.0340; -0.7348]	18.787785

Table 3. Results in case without constraint, n=5;

We can see that FDM gives us the smallest values of while AM optimizes worse but is less time-consuming. FDM is the most time-consuming.

3.2 With integral constraint

Method	b*			time
FDM	[-0.1100 ;-0.1100]	0.1647	[2.2792;-1.0760]	6.890902
DDM	[-0.1100 ;-0.1100]	0.1647	[1.7894;-1.3849]	8.189049
AM	[-0.1100 ;-0.1100]	0.1647	[4.3748;-0.5346]	4.752103

Table 4. Results in case with constraint, n=2;

Method	b*			time
FDM	[-0.1100 ; -0.1100; -0.1005 ;-0.1002 -0.0996]	0.1652	[3.9580;-3.5703; 0.9403;0.3971; -0.8953]	74.541801
DDM	[-0.1100 ;-0.1100; -0.1003 ; -0.1002 ; --0.0995]	0.1654	[4.0683;-3.8763; 0.6842;0.3096; -0.9187]	33.218342
AM	[-0.1100;-0.1100; -0.1000; -0.1004; -0.0996]	0.1656	[5.1565;-1.2992; 0.0963;0.7508; -0.8744]	11.327486

Table 5. Results in case with constraint, n=5;

It's easy to see that in case with constraint there are larger values of optimization criteria. It confirms the fact that the optimization criteria cannot be reduced with increasing constraints. In this case FDM is the most time-consuming but it still gives the best results.

4. Optimal solutions

Here, we will consider solution of the problem with n = 5.

The optimal values of optimization parameters are found by the usage of function *fminsearch*. The optimal values is b=[-0.1577;-0.0822;-0.0823; -0.0692;-0.0692] and 0.0266. On the figures the illustrations of $y_1(t)$, $y_2(t)$ and control function u.

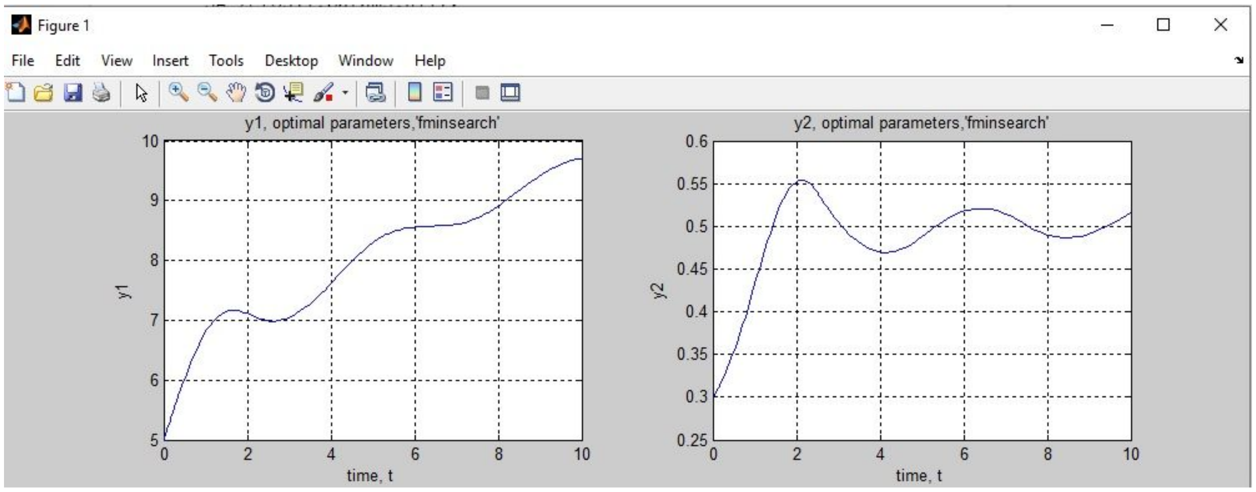


Figure 1.($y_1(t)$, $y_2(t)$)

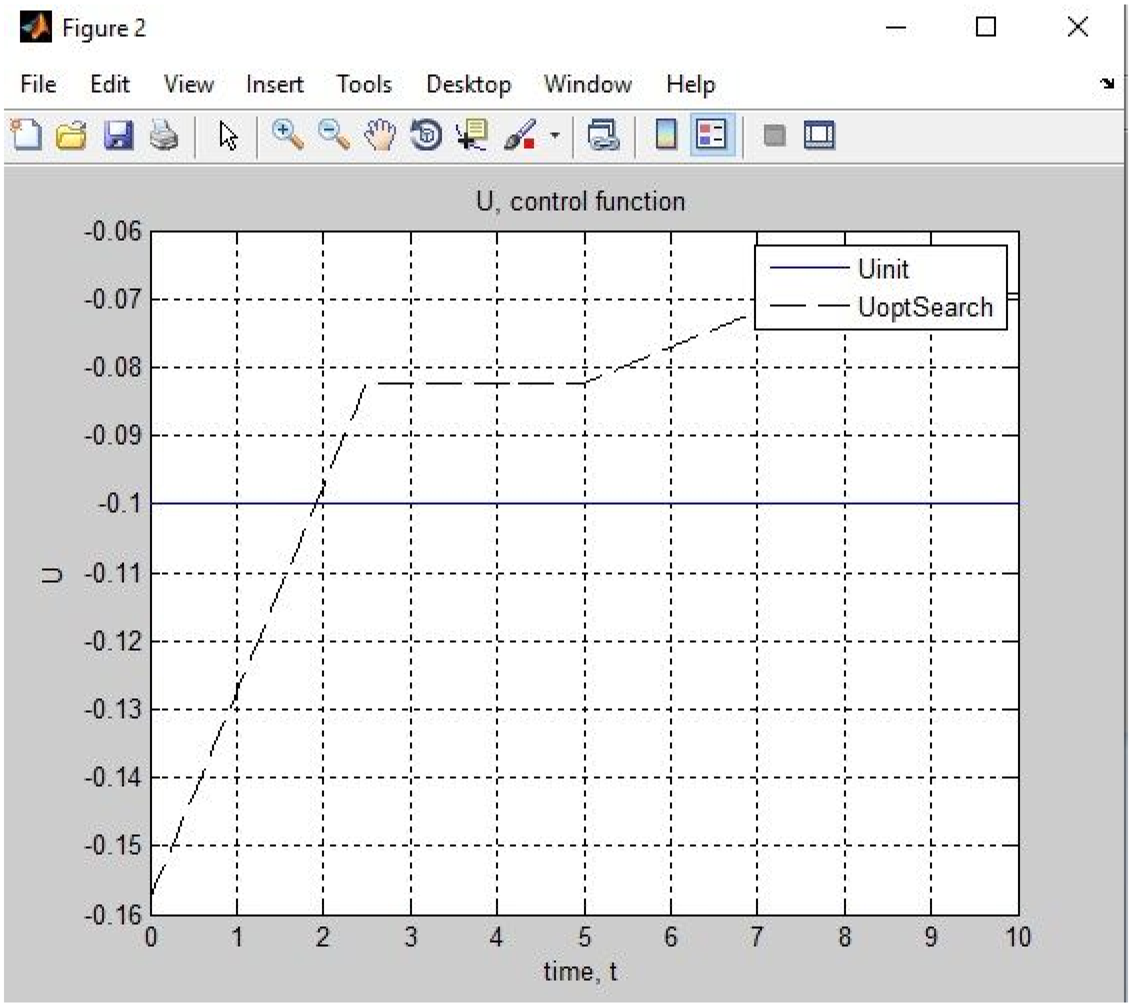


Figure 2.(Control function)

5. Necessary optimality conditions

Here, we will consider our problem with all the conditions and constraints it was initially set with. Like in the case solved before, there is a IVP constraint c(u, y). New additions are integral constraint Ψ1(u) and dual constraints on control function $u^{-} \leq u \leq u^{+}$. . We can write the Lagrangian function.

$$L(\mathrm{u},\mathrm{y}\,\mu,\lambda^{-},\lambda^{+},\lambda_1)=\tilde{\psi}_0+(\mu,\mathrm{c})+(\lambda^{-},\mathrm{u}^{-}-\mathrm{u})+(\lambda^{+},\mathrm{u}-\mathrm{u}^{+})+(\lambda_1,\psi_1)$$

Now, we can write KKT conditions for this problem.

$$1.\begin{cases} Lu\delta u = 0 \\ Ly\delta y = 0 \\ L\mu\delta\mu = 0 \end{cases}$$

$$2.\begin{cases} c(\mathrm{u},\mathrm{y}) = 0 \\ \tilde{\psi}_1(\mathrm{u},\mathrm{y}) = 0 \end{cases}$$

$$3.\begin{cases} u^- - u \le 0 \\ u - u^+ \le 0 \end{cases}$$

$$4.\begin{cases} (\lambda^-, u^- - u) = 0 \\ (\lambda^+, u - u^+) = 0 \\ \lambda_1\tilde{\psi}_1(\mathrm{u},\mathrm{y}) = 0 \end{cases}$$

$$5.\lambda^-, \lambda^+, \lambda_1 \ge 0$$

Solving system of all these equations with satisfying all the conditions we will get an optimal solution to our constrained problem. In this report we will not go into details of how it can be done, but just say that it requires a lot of analytic transformations and even then will be hard to solve numeric.