

Homework 3

Sebastian Aguilera Novoa

Table of Contents

1. Limite on an Optimization Problem.....	1
2. Linear fitting in	5
1. Data generation.....	5
2. Problem solution.....	6
3. Atypical data.....	9
3. Linear Fitting in	11
1. Problem reformulation, matrix A.....	11
2. Problem solution.....	14
3.....	18
4. Integral Equation.....	18
a) Exact solution.....	19
b) Problem Discretization.....	19
c) Code Implementation.....	21
Solution by Gauss eq.....	21
Solution by CasADI.....	21
d) Add Regularization.....	28
Solution by Gauss eq.....	28
Solution by CasADI.....	29
Auxiliar Functions.....	34

1. Limite on an Optimization Problem

Given a matrix $J \in \mathbb{R}^{m \times n}$ and a matrix $Q \in \mathbb{R}^{n \times n}$ symmetric defined positive $Q > 0$, a vector of measures $\eta \in \mathbb{R}^m$ and a point $\bar{x} \in \mathbb{R}$, calculate the limite

$$\lim_{\alpha \rightarrow 0^+} \arg \min_x \frac{1}{2} \|\eta - Jx\|_2^2 + \frac{\alpha}{2} (x - \bar{x})^T Q (x - \bar{x})$$

In order to calculate this limit, the function $f(x)$ is defined to find its minimum value using the derivative

$$\begin{aligned}
 f(x) &= \frac{1}{2} \|\eta - Jx\|_2^2 + \frac{\alpha}{2} (x - \bar{x})^T Q (x - \bar{x}) = \frac{1}{2} (\eta - Jx)^T (\eta - Jx) + \frac{\alpha}{2} (x - \bar{x})^T Q (x - \bar{x}) \\
 &= \frac{1}{2} (\eta^T \eta - x^T J^T \eta - \eta^T Jx + x^T J^T Jx) + \frac{\alpha}{2} (x - \bar{x})^T Q (x - \bar{x}) \\
 &= \frac{1}{2} (\eta^T \eta - 2J^T \eta x + x^T J^T Jx) + \frac{\alpha}{2} (x - \bar{x})^T Q (x - \bar{x}) \\
 \nabla f(x) &= -J^T \eta + J^T Jx + \alpha Q (x - \bar{x}) \\
 \nabla^2 f(x) &= J^T J + \alpha Q > 0
 \end{aligned}$$

then, to find the minimum we have to solve $\nabla f(x) = 0$

$$\begin{aligned}
\nabla f(x) &= -J^T \eta + J^T Jx + \alpha Q(x - \bar{x}) = -J^T \eta + J^T Jx + \alpha Qx - \alpha \bar{x} Q \bar{1} = 0 \\
(J^T J + \alpha Q)x &= J^T \eta + \alpha \bar{x} Q \bar{1} \\
\Rightarrow x(\alpha) &= (J^T J + \alpha Q)^{-1} (J^T \eta + \alpha \bar{x} Q \bar{1})
\end{aligned}$$

here $\bar{1} \in \mathbb{R}^n$ is a vector of ones. Using SVD decomposition $J = U \Sigma V^T$, with $U \in \mathbb{R}^{m \times m}$, $\Sigma \in \mathbb{R}^{m \times n}$, $V^T \in \mathbb{R}^{n \times n}$

$$J^T J + \alpha Q = V \Sigma^T U^T U \Sigma V^T + \alpha Q = V \Sigma^T \Sigma V^T + \alpha Q$$

descomposing Q in terms of V , $Q = U' \Sigma' V^T$ since Q is square then $U', \Sigma', V^T \in \mathbb{R}^{n \times n}$

$$\begin{aligned}
J^T J + \alpha Q &= V \Sigma^T \Sigma V^T + \alpha U' \Sigma' V^T = (V \Sigma^T \Sigma + \alpha U' \Sigma') V^T = (V \Sigma^T \Sigma + \alpha V V^T U' \Sigma') V^T \\
J^T J + \alpha Q &= V (\Sigma^T \Sigma + \alpha V^T U' \Sigma') V^T
\end{aligned}$$

calculating the inverse

$$\begin{aligned}
(J^T J + \alpha Q)^{-1} &= (V (\Sigma^T \Sigma + \alpha V^T U' \Sigma') V^T)^{-1} = (V^T)^{-1} (\Sigma^T \Sigma + \alpha V^T U' \Sigma')^{-1} V^{-1} \\
&= V (\Sigma^T \Sigma + \alpha V^T U' \Sigma')^{-1} V^T = V (\Sigma^T \Sigma + \alpha W \Sigma')^{-1} V^T
\end{aligned}$$

writing in matricial form and defining $W = V^T U'$, $W \in \mathbb{R}^{n \times n}$

$$\begin{aligned}
(J^T J + \alpha Q)^{-1} &= V \left(\begin{array}{ccccccc} \sigma_1^2 + \alpha W_{1,1} \sigma'_1 & 0 & \dots & & & & 0 \\ 0 & \sigma_2^2 + \alpha W_{2,2} \sigma'_2 & 0 & \dots & \dots & & 0 \\ \vdots & \dots & \ddots & \dots & \dots & & \vdots \\ 0 & \dots & \dots & \sigma_r^2 + \alpha W_{r,r} \sigma'_r & \dots & & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \dots & \alpha W_{n-1,n-1} \sigma'_{n-1} & & 0 \\ 0 & 0 & \dots & \dots & \dots & & \alpha W_{n,n} \sigma'_n \end{array} \right)^{-1} V^T \\
(J^T J + \alpha Q)^{-1} &= V \left(\begin{array}{ccccccc} \frac{1}{\sigma_1^2 + \alpha W_{1,1} \sigma'_1} & 0 & \dots & & & & 0 \\ 0 & \frac{1}{\sigma_2^2 + \alpha W_{2,2} \sigma'_2} & 0 & \dots & \dots & & 0 \\ \vdots & \dots & \ddots & \dots & \dots & & \vdots \\ 0 & \dots & \dots & \frac{1}{\sigma_r^2 + \alpha W_{r,r} \sigma'_r} & \dots & & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \dots & \frac{1}{\alpha W_{n-1,n-1} \sigma'_{n-1}} & & 0 \\ 0 & 0 & \dots & \dots & \dots & & \frac{1}{\alpha W_{n,n} \sigma'_n} \end{array} \right) V^T
\end{aligned}$$

now let's calculate the product $(J^T J + \alpha Q)^{-1}(J^T \eta + \alpha \bar{x} Q)$

$$\begin{aligned} J^T \eta + \alpha \bar{x} Q \bar{1} &= V \Sigma^T U^T \eta + \alpha \bar{x} U' \Sigma' V^T \bar{1} = V \Sigma^T U^T \eta + \alpha \bar{x} V V^T U' \Sigma' V^T \bar{1} \\ &= V(\Sigma^T U^T \eta + \alpha \bar{x} W \Sigma' V^T \bar{1}) \end{aligned}$$

$$\begin{aligned} (J^T J + \alpha Q)^{-1}(J^T \eta + \alpha \bar{x} Q) &= V(\Sigma^T \Sigma + \alpha W \Sigma')^{-1} V^T V(\Sigma^T U^T \eta + \alpha \bar{x} W \Sigma' V^T) \\ &= V(\Sigma^T \Sigma + \alpha W \Sigma')^{-1}(\Sigma^T U^T \eta + \alpha \bar{x} W \Sigma' V^T) \\ &= V(\Sigma^T \Sigma + \alpha W \Sigma')^{-1}(\Sigma^T) U^T \eta + V(\Sigma^T \Sigma + \alpha W \Sigma')^{-1}(\alpha \bar{x} W \Sigma') V^T \end{aligned}$$

calculating each matrix by separated

- $V(\Sigma^T \Sigma + \alpha W \Sigma')^{-1} \Sigma^T U^T \eta$

$$V \begin{pmatrix} \frac{1}{\sigma_1^2 + \alpha W_{1,1} \sigma'_1} & 0 & \dots & & 0 \\ 0 & \frac{1}{\sigma_2^2 + \alpha W_{2,2} \sigma'_2} & 0 & \dots & 0 \\ \vdots & \dots & \ddots & \dots & \vdots \\ 0 & \dots & \dots & \frac{1}{\sigma_r^2 + \alpha W_{r,r} \sigma'_r} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & \frac{1}{\alpha W_{n-1,n-1} \sigma'_{n-1}} & 0 \\ 0 & 0 & \dots & \dots & \dots & \frac{1}{\alpha W_{n,n} \sigma'_n} \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 & \dots & & \\ 0 & \sigma_2 & 0 & \dots & \\ & & \ddots & \vdots & \dots \\ & & \dots & \sigma_r & \dots \\ & & & \vdots & 0 & \dots \\ & & & & \ddots & \vdots \\ & & & & \dots & 0 \end{pmatrix} U^T \eta$$

$$V(\Sigma^T \Sigma + \alpha W \Sigma')^{-1} \Sigma^T U^T \eta = V \begin{pmatrix} \frac{\sigma_1}{\sigma_1^2 + \alpha W_{1,1} \sigma'_1} & 0 & \dots & & 0 \\ 0 & \frac{\sigma_2}{\sigma_2^2 + \alpha W_{2,2} \sigma'_2} & 0 & \dots & \dots & 0 \\ \vdots & \dots & \ddots & \dots & \dots & \vdots \\ 0 & \dots & \dots & \frac{\sigma_r}{\sigma_r^2 + \alpha W_{r,r} \sigma'_r} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & 0 & 0 \\ 0 & 0 & \dots & \dots & \dots & 0 \end{pmatrix} V^T \quad (6)$$

- $V(\Sigma^T \Sigma + \alpha W \Sigma')^{-1}(\alpha \bar{x} W \Sigma') V^T$

$$V \begin{pmatrix} \frac{1}{\sigma_1^2 + \alpha W_{1,1}\sigma'_1} & 0 & \dots & & 0 \\ 0 & \frac{1}{\sigma_2^2 + \alpha W_{2,2}\sigma'_2} & 0 & \dots & \dots & 0 \\ \vdots & \dots & \ddots & \dots & \dots & \vdots \\ 0 & \dots & \dots & \frac{1}{\sigma_r^2 + \alpha W_{r,r}\sigma'_r} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & \frac{1}{\alpha W_{n-1,n-1}\sigma'_{n-1}} & 0 \\ 0 & 0 & \dots & \dots & \dots & \frac{1}{\alpha W_{n,n}\sigma'_n} \end{pmatrix} \begin{pmatrix} \alpha \bar{x} W_{1,1}\sigma'_1 & 0 & \dots & & & \\ 0 & \alpha \bar{x} W_{2,2}\sigma'_2 & 0 & \dots & & \\ & & \ddots & \vdots & \dots & \\ & & \dots & \alpha \bar{x} W_{r,r}\sigma'_r & \dots & \\ & & & \vdots & 0 & \dots \\ & & & & \ddots & \vdots \\ & & & & \dots & 0 \end{pmatrix} V^T$$

$$V(\Sigma^T \Sigma + \alpha W \Sigma')^{-1} (\alpha \bar{x} W \Sigma') V^T = V \begin{pmatrix} \alpha \frac{\bar{x} W_{1,1}\sigma'_1}{\sigma_1^2 + \alpha W_{1,1}\sigma'_1} & 0 & \dots & & 0 \\ 0 & \alpha \frac{\bar{x} W_{2,2}\sigma'_2}{\sigma_2^2 + \alpha W_{2,2}\sigma'_2} & 0 & \dots & \dots & 0 \\ \vdots & \dots & \ddots & \dots & \dots & \vdots \\ 0 & \dots & \dots & \alpha \frac{\bar{x} W_{r,r}\sigma'_r}{\sigma_r^2 + \alpha W_{r,r}\sigma'_r} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & 0 & 0 \\ 0 & 0 & \dots & \dots & \dots & 0 \end{pmatrix} V^T$$

(7)

Now let's calculate the limite of both matrixes

$$\lim_{\alpha \rightarrow 0^+} (J^T J + \alpha Q)^{-1} J^T = \lim_{\alpha \rightarrow 0^+} V(\Sigma^T \Sigma + \alpha W \Sigma')^{-1} \Sigma^T$$

$$= \lim_{\alpha \rightarrow 0^+} V \begin{pmatrix} \frac{\sigma_1}{\sigma_1^2 + \alpha W_{1,1}\sigma'_1} & 0 & \dots & & 0 \\ 0 & \frac{\sigma_2}{\sigma_2^2 + \alpha W_{2,2}\sigma'_2} & 0 & \dots & \dots & 0 \\ \vdots & \dots & \ddots & \dots & \dots & \vdots \\ 0 & \dots & \dots & \frac{\sigma_r}{\sigma_r^2 + \alpha W_{r,r}\sigma'_r} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & 0 & 0 \\ 0 & 0 & \dots & \dots & \dots & 0 \end{pmatrix} V^T = V \begin{pmatrix} \sigma_1^{-1} & 0 & \dots & & 0 \\ 0 & \sigma_2^{-1} & 0 & \dots & \dots & 0 \\ \vdots & \dots & \ddots & \dots & \dots & \vdots \\ 0 & \dots & \dots & \sigma_r^{-1} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & 0 & 0 \\ 0 & 0 & \dots & \dots & \dots & 0 \end{pmatrix} V^T = V \Sigma^+ V^T$$

$$\begin{aligned}
\lim_{\alpha \rightarrow 0^+} (J^T J + \alpha Q)^{-1} \alpha \bar{x} Q &= \lim_{\alpha \rightarrow 0^+} V(\Sigma^T \Sigma + \alpha W \Sigma')^{-1} (\alpha \bar{x} W \Sigma') V^T \\
&= \lim_{\alpha \rightarrow 0^+} V \begin{pmatrix} \alpha \frac{\bar{x} W_{1,1} \sigma'_1}{\sigma_1^2 + \alpha W_{1,1} \sigma'_1} & 0 & \dots & 0 \\ 0 & \alpha \frac{\bar{x} W_{2,2} \sigma'_2}{\sigma_2^2 + \alpha W_{2,2} \sigma'_2} & 0 & \dots & 0 \\ \vdots & \dots & \ddots & \dots & \vdots \\ 0 & \dots & \dots & \alpha \frac{\bar{x} W_{r,r} \sigma'_r}{\sigma_r^2 + \alpha W_{r,r} \sigma'_r} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & 0 & 0 \\ 0 & 0 & \dots & \dots & \dots & 0 \end{pmatrix} V^T = V \bar{\bar{0}} V^T
\end{aligned}$$

where $\bar{\bar{0}} \in \mathbb{R}^{n \times n}$ is a zero matrix.

Then, the limit is

$$\lim_{\alpha \rightarrow 0} (J^T J + \alpha Q)^{-1} (J^T \eta + \alpha \bar{x} Q) = V \Sigma^+ V^T \eta + V \bar{\bar{0}} V^T = J^+ \eta$$

which gives the same result found for the laest square problem without regularization. \square

2. Linear fitting in L_2

Suppose there is a set of N noisy measures $(x_i, y_i) \in \mathbb{R}^2$ that we want to fit a line $y = ax + b$. The previous can be expressed as the following optimization problem

$$\min_{a, b} \sum_{i=1}^N (ax_i + b - y_i)^2 = \min_{a, b} \left\| J \begin{pmatrix} a \\ b \end{pmatrix} - y \right\|_2^2$$

Like was discussed in class, the optimal solution to the previous problem can be calculated explicitly by solving the normal Gauss equations

$$J^T J \begin{pmatrix} a \\ b \end{pmatrix} = J^T y$$

1. Data generation

Generate the problem data. Take $N = 30$ points in the interval $[0, 5]$ and generate the true outputs $y_i = 3x_i + 4$. Add gaussian noise with zero mean and standard deviation 1 to get the noisy data and plot them. *Advice:* check the commands `linspace` and `randn`. If you want a "random" reproducible serie, use `rng`.

```

N = 30; rng(7);

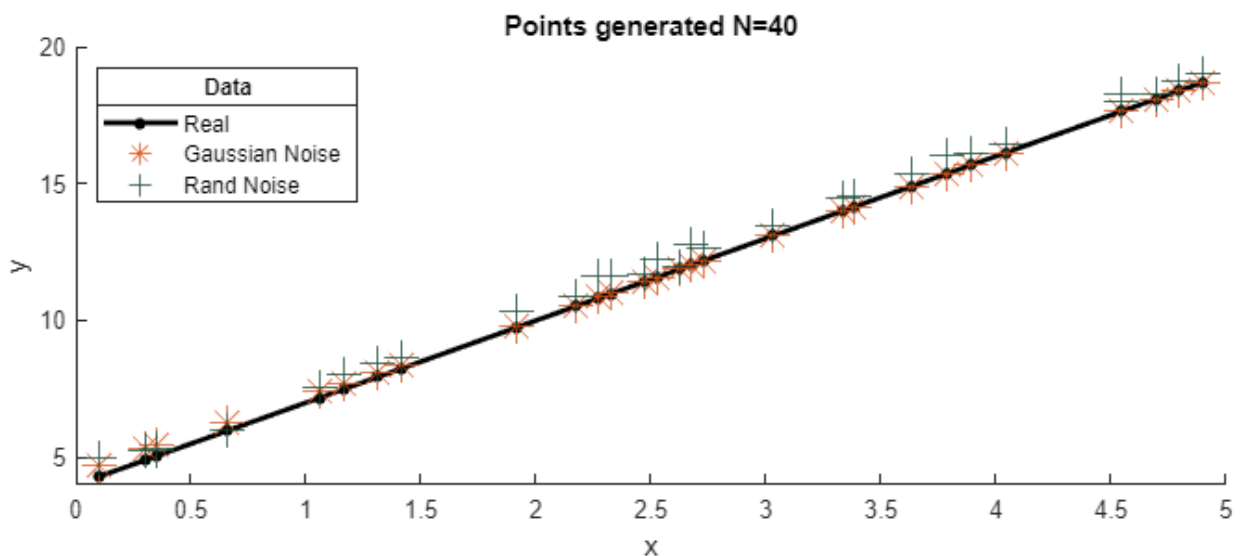
xrand = zeros(N,1);
x      = linspace(0,5,100);
xi     = randi(100,1,N);

for i=1:1:N; xrand(i) = x(xi(i)); end

y      = 3*xrand + 4;
ynoise = y + (1/sqrt(2*pi)).*exp(-xrand.^2/2);
ynoise1 = y + 0.9*rand(N,1);

clf
figure(Position=[100 100 800 300]);
hold on
plot(xrand,y, '.-', MarkerSize=14, Color='black', LineWidth=2)
plot(xrand,ynoise, '*', MarkerSize=14)
plot(xrand,ynoise1, '+', MarkerSize=14, Color=[21/250 71/250 52/250])
leg = legend('Real', 'Gaussian Noise','Rand Noise','Location','northwest');
title(leg,'Data');
xlabel('x'); ylabel('y'); title('Points generated N=40')
hold off

```



2. Problem solution

Write the matrix J . Calculate the coefficients a , b using the last equation and plot the measures and line getted in the same plane.

The equation (1) can be rewrite using residual vector r

$$r(x) = (ax_1 + b - y_1, ax_2 + b - y_2, \dots, ax_N + b - y_N)$$

then

$$\min_{a,b \in \mathbb{R}} \sum_{i=0}^N r_i^2(a,b) = \min_{a,b \in \mathbb{R}} \|r\|_2^2$$

Jacobian

$$J = \left[\frac{\partial r_i}{\partial x_j} \right]_{ij}$$

with $i = 1, \dots, N$, $j = 1, 2$, and $x_1 = a$, $x_2 = b$. Then the Jacobian is

$$J = \begin{pmatrix} \frac{\partial r_1}{\partial a} & \frac{\partial r_1}{\partial b} \\ \frac{\partial r_2}{\partial a} & \frac{\partial r_2}{\partial b} \\ \vdots & \vdots \\ \frac{\partial r_N}{\partial a} & \frac{\partial r_N}{\partial b} \end{pmatrix} = \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_{N-1} & 1 \\ x_N & 1 \end{pmatrix}$$

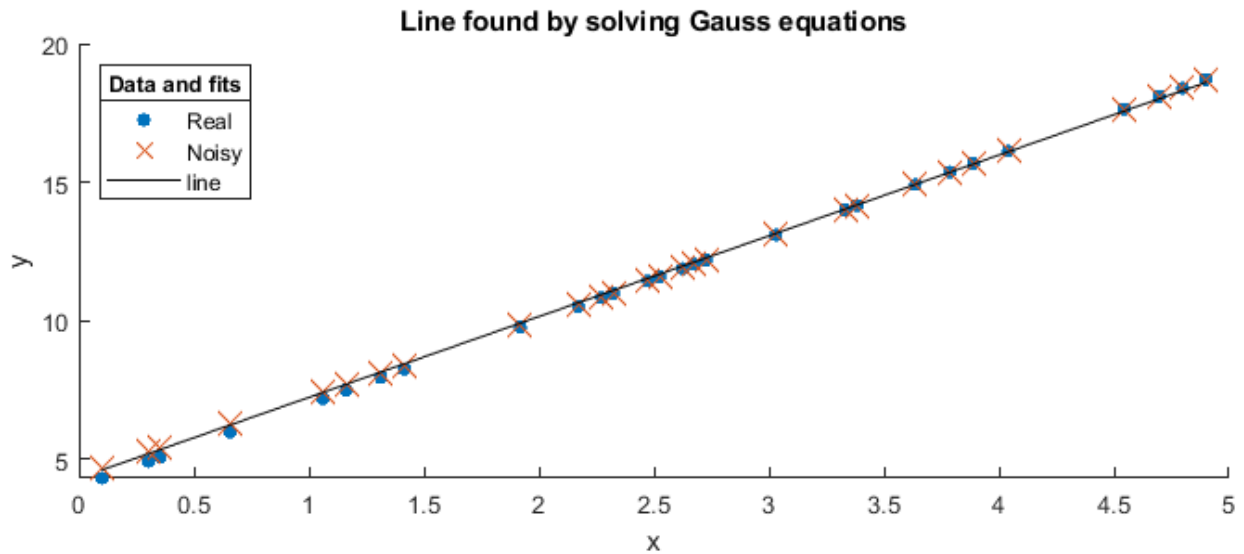
```
J = [xrand ones(N,1)];
ab = J \ ynoise;           % A \ b %Ax = b  '%dividir a izquierda'
disp(['Optimal solution found by solving Gauss equations: a = ' num2str(ab(1)) ' b = ' num2str(ab(2)) '])
```

Optimal solution found by solving Gauss equations: a = 2.9181 b = 4.3045

```
disp(['The residual error by solgin the Gauss equation is r = ' num2str(sum(ab(1)*xrand+ab(2)-y)) '])
```

The residual error by solgin the Gauss equation is r = 1.3145e-13

```
clf
figure(Position=[100 100 800 300]);
hold on
plot(xrand, y, '.', MarkerSize=14)
plot(xrand, ynoise, 'x', MarkerSize=14)
plot(xrand, ab(1)*xrand+ab(2), '-', Color='black')
leg = legend('Real', 'Noisy', 'line', 'Location', 'northwest');
title(leg, 'Data and fits');
xlabel('x'); ylabel('y'); title('Line found by solving Gauss equations')
hold off
```



Let's verify the coefficients using CasADi.

```
import casadi.*

opti = casadi.Opti();
a_opt = opti.variable();
b_opt = opti.variable();
opti.minimize(norm(a_opt*xrand+b_opt-ynoise,2));

opti.solver('ipopt');

sol = opti.solve();
```

This is Ipopt version 3.12.3, running with linear solver mumps.

NOTE: Other linear solvers might be more efficient (see Ipopt documentation).

```
Number of nonzeros in equality constraint Jacobian...:      0
Number of nonzeros in inequality constraint Jacobian.:      0
Number of nonzeros in Lagrangian Hessian.....:          3
```

```
Total number of variables.....:          2
   variables with only lower bounds:          0
   variables with lower and upper bounds:       0
   variables with only upper bounds:             0
Total number of equality constraints.....:          0
Total number of inequality constraints.....:          0
   inequality constraints with only lower bounds:  0
   inequality constraints with lower and upper bounds: 0
   inequality constraints with only upper bounds:  0
```

iter	objective	inf_pr	inf_du	lg(mu)	d	lg(rg)	alpha_du	alpha_pr	ls
0	6.8661733e+001	0.00e+000	1.59e+001	-1.0	0.00e+000	-	0.00e+000	0.00e+000	0
1	6.6373392e+001	0.00e+000	1.59e+001	-1.0	1.39e+005	-	1.00e+000	6.10e-005f	15
2	5.5605779e+001	0.00e+000	1.59e+001	-1.0	1.25e+005	-	1.00e+000	6.10e-005f	15
3	1.6121968e+001	0.00e+000	1.59e+001	-1.0	7.37e+004	-	1.00e+000	6.10e-005f	15
4	1.1849837e+001	0.00e+000	1.59e+001	-1.0	1.80e+003	-	1.00e+000	9.77e-004f	11
5	1.0363910e+001	0.00e+000	1.59e+001	-1.0	7.13e+002	-	1.00e+000	1.95e-003f	10
6	4.5094273e+000	0.00e+000	1.59e+001	-1.0	4.77e+002	-	1.00e+000	1.95e-003f	10
7	5.4311814e-001	0.00e+000	1.13e+001	-1.0	3.92e+001	-	1.00e+000	7.81e-003f	8
8	3.8251916e-001	0.00e+000	1.29e-001	-1.0	4.87e-002	-	1.00e+000	5.00e-001f	2


```

9 3.8250658e-001 0.00e+000 8.50e-006 -2.5 1.95e-004 - 1.00e+000 1.00e+000f 1
iter objective inf_pr inf_du lg(mu) ||d|| lg(rg) alpha_du alpha_pr ls
10 3.8250658e-001 0.00e+000 5.34e-014 -8.6 1.28e-008 - 1.00e+000 1.00e+000f 1

```

Number of Iterations.....: 10

	(scaled)	(unscaled)
Objective.....:	3.8250658000710780e-001	3.8250658000710780e-001
Dual infeasibility.....:	5.3401727484470030e-014	5.3401727484470030e-014
Constraint violation.....:	0.0000000000000000e+000	0.0000000000000000e+000
Complementarity.....:	0.0000000000000000e+000	0.0000000000000000e+000
Overall NLP error.....:	5.3401727484470030e-014	5.3401727484470030e-014

```

Number of objective function evaluations      = 121
Number of objective gradient evaluations      = 11
Number of equality constraint evaluations      = 0
Number of inequality constraint evaluations    = 0
Number of equality constraint Jacobian evaluations = 0
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations     = 10
Total CPU secs in IPOPT (w/o function evaluations) = 0.178
Total CPU secs in NLP function evaluations    = 0.001

```

EXIT: Optimal Solution Found.

solver	:	t_proc	(avg)	t_wall	(avg)	n_eval
nlp_f		0 (0)	0 (0)	121
nlp_grad_f		0 (0)	0 (0)	12
nlp_hess_l		0 (0)	0 (0)	10
total		180.00ms (180.00ms)		180.26ms (180.26ms)		1

```

aopt = sol.value(a_opt);
bopt = sol.value(b_opt);
disp(['Optimal solution found be CasADI: a = ' num2str(aopt) ' b = ' num2str(bopt)]);

```

Optimal solution found be CasADI: a = 2.9181 b = 4.3045

```

disp(['The residual error is by CasADI r = ' num2str(sum(aopt*xrand+bopt-ynoise))])

```

The residual error is by CasADI r = 7.9936e-15

```

disp(['CasADI - Gauss: a = ' num2str(aopt-ab(1)) ' b = ' num2str(bopt-ab(2))])

```

CasADI - Gauss: a = -4.4409e-16 b = -2.6645e-15

Although both solutions have the same value of slope and y -intercept (a, b) with difference in the 15-th decimal, the solution found by solving the Gauss equations has a smaller residual error then it is better.

3. Atypical data

Introduce 3 atypical data in the measures y and plot the new fitted line in the same plane.

```

rng(123);
ynoise_atypical = ynoise;
ynoise_atypical(randi(N,3,1)) = ynoise_atypical(randi(N,3,1))+0.7*rand(1);

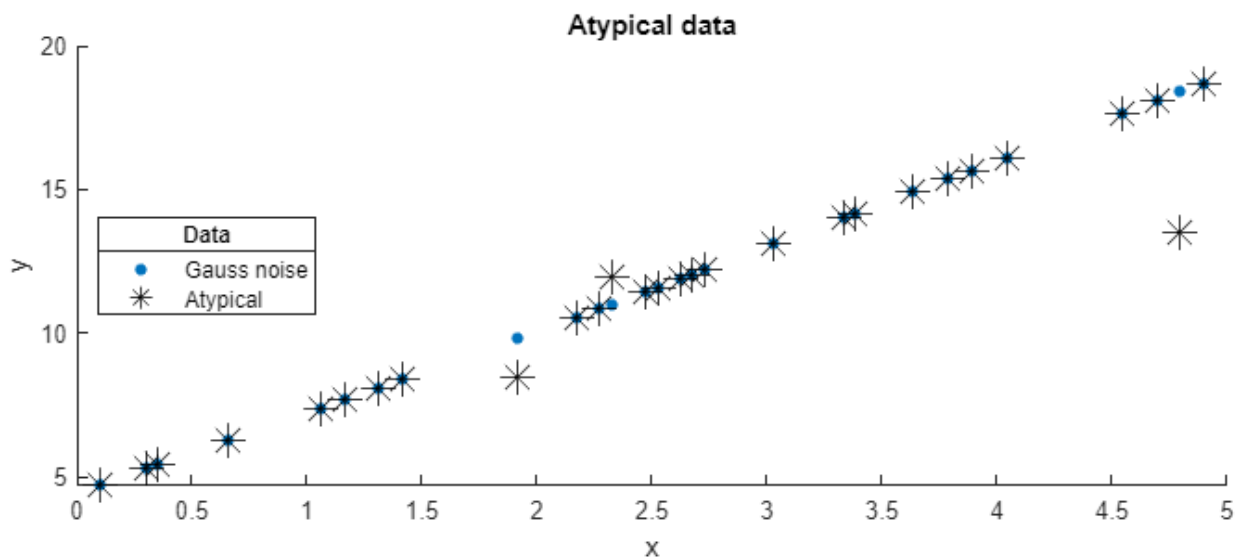
clf;
figure(Position=[100 100 800 300]);
hold on

```

```

plot(xrand, ynoise, '.', MarkerSize=14)
plot(xrand, ynoise_atypical, '*', MarkerSize=14, Color='black')
leg = legend('Gauss noise', 'Atypical', 'Location', 'west');
title(leg, 'Data');
xlabel('x'); ylabel('y'); title('Atypical data')
hold off

```



```

ab_atypical = J \ ynoise_atypical; % A \ b %Ax = b %'dividir a izquierda'
disp(['Optimal solution found by solving Gauss equations: a = ' num2str(ab_atypical(1)) ' b = '

```

Optimal solution found by solving Gauss equations: a = 2.755 b = 4.548

```

disp(['The residual error by solgin the Gauss equation is r = ' num2str(sum(ab_atypical(1)*xrand

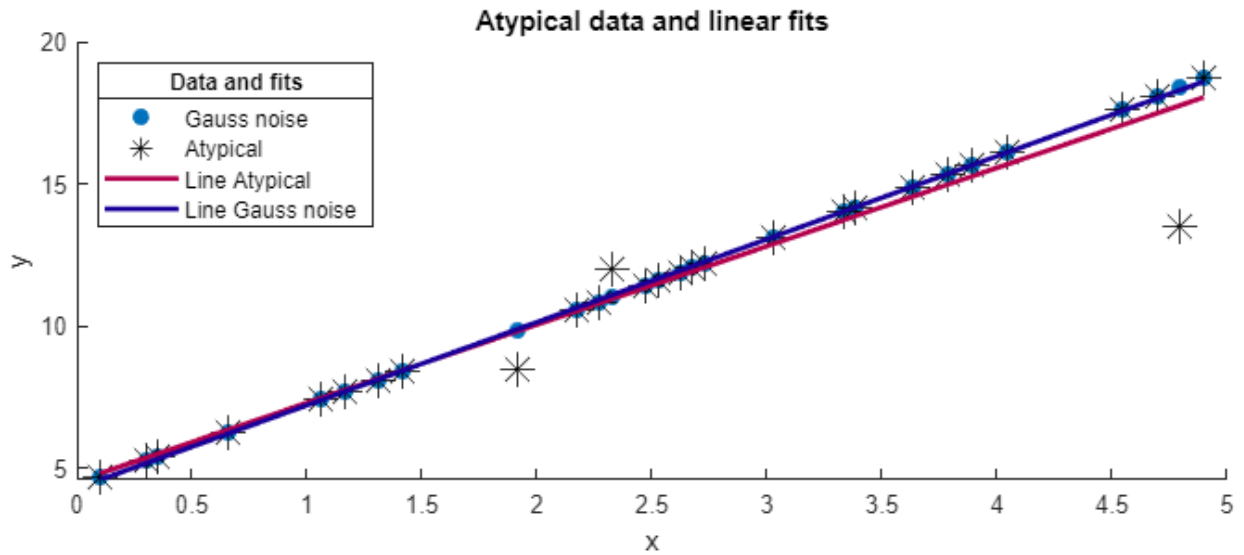
```

The residual error by solgin the Gauss equation is r = 9.9476e-14

```

clf;
figure(Position=[100 100 800 300]);
hold on
plot(xrand, ynoise, '.', MarkerSize=20, DisplayName='Gauss noise')
plot(xrand, ynoise_atypical, '*', MarkerSize=14, Color='black', DisplayName='Atypical')
plot(xrand, ab_atypical(1)*xrand+ab_atypical(2), '-', LineWidth=2, Color=[.7 .0 .3], DisplayName='Line Gauss noise')
plot(xrand, ab(1)*xrand+ab(2), '-', LineWidth=2, Color=[.1 0 .6], DisplayName='Line Gauss noise')
leg = legend('Location', 'northwest'); title(leg, 'Data and fits');
xlabel('x'); ylabel('y'); title('Atypical data and linear fits')
hold off

```



The measures y (with and without the atypical data) and the matrix J are necessary to the following point.

3. Linear Fitting in L_1

Now we are interesting to fit a line to the same measure set, but now using the following objective function

$$\min_{a, b} \sum_{i=1}^N |ax_i + b - y_i|$$

The objective function is not differentiable, then we are going to use width variables to get the following equivalent linear programming problem

$$\begin{aligned} \min_{a, b, s} \quad & \sum_{i=1}^N s_i \\ \text{s.t.} \quad & a - s_i \leq ax_i + b - y_i \leq s_i \quad i = 1, \dots, N \\ & -s_i \leq 0 \quad i = 1, \dots, N \end{aligned} \tag{3}$$

1. Problem reformulation, matrix A

In order to solve the previous linear programming problem use the MATLAB command `linprog`, to do this is necessary to write the problem as follows

$$\begin{aligned}
& \min_z f^T z \\
& \text{s.t. } Az \leq b \\
& \quad Cz = d \\
& \quad l_z \leq z \leq u_z
\end{aligned}$$

Write the matrix A and the vectors f, b . Organize the variables as $z^T = [a, b, s_1, \dots, s_N]$. Use the matrix J from the previous exercise to define A .

From the equation (3), note that the variable $s_i = ax_i + b - y_i$ and the product $f^T z$ must be a scalar. Then,

$$f = [0, 0, 1, \dots, 1], \quad \rightarrow \quad f^T z = \begin{pmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \cdot [a, b, s_1, \dots, s_N] = s_1 + s_2 + \dots + s_N$$

note that the i -th component of the residual vector can be written in terms of J as

$$ax_i + b - y_i = (x_i \quad 1) \begin{pmatrix} a \\ b \end{pmatrix} - y_i = J_i(a \ b)^T - y_i = (J(a \ b)^T - y)_i$$

respect to the constraints, it can be written as follows using the matrix J found in the previous point and the previous substitution

$$-z' \leq J \begin{pmatrix} a \\ b \end{pmatrix} - y \leq z', \quad \leftrightarrow \quad -z' - J \begin{pmatrix} a \\ b \end{pmatrix} \leq -y \leq z' - J \begin{pmatrix} a \\ b \end{pmatrix}$$

where $z = [a, b, s]$, $s = [s_1, s_2, \dots, s_N]$. For the components, since we are working in linear programming this constraint must be split into two

$$\begin{aligned}
-s - J \begin{pmatrix} a \\ b \end{pmatrix} &\leq -y & \leftrightarrow & \quad s + J \begin{pmatrix} a \\ b \end{pmatrix} \geq y \\
s - J \begin{pmatrix} a \\ b \end{pmatrix} &\geq -y & \leftrightarrow & \quad -s + J \begin{pmatrix} a \\ b \end{pmatrix} \leq y \\
-s &\leq 0 & \leftrightarrow & \quad s \geq 0
\end{aligned} \tag{4}$$

where the last 0 is a vector of $N + 2$ components. Now let's construct the matrix A taking into account the previous inequalities

$$\begin{aligned}
s_1 + x_1 a + b &= z_3 + x_1 z_1 + z_2 \geq y_1 \\
s_2 + x_2 a + b &= z_4 + x_2 z_1 + z_2 \geq y_2 \\
s_3 + x_3 a + b &= z_5 + x_3 z_1 + z_2 \geq y_3 \\
&\vdots \\
s_N + x_N a + b &= z_{N+2} + x_N z_1 + z_2 \geq y_N
\end{aligned}
\leftrightarrow
\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ x_1 & 1 & 1 & 0 & 0 & \dots & 0 \\ x_2 & 1 & 0 & 1 & 0 & \dots & 0 \\ x_2 & 1 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_N & 1 & 0 & 0 & 0 & \dots & 1 \end{pmatrix}
\begin{pmatrix} a \\ b \\ s_1 \\ s_2 \\ \vdots \\ s_{N-1} \\ s_N \end{pmatrix}
\geq
\begin{pmatrix} 0 \\ 0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{pmatrix}$$

that can be write as $A_1 z \geq b$ with

$$z = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_{N+2} \end{pmatrix} = \begin{pmatrix} a \\ b \\ s_1 \\ \vdots \\ s_N \end{pmatrix}, \quad A_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ x_1 & 1 & 1 & 0 & 0 & \dots & 0 \\ x_2 & 1 & 0 & 1 & 0 & \dots & 0 \\ x_2 & 1 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_N & 1 & 0 & 0 & 0 & \dots & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ 0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{pmatrix}$$

similarly the second constraint of (3) can be write as $A_2 z \leq b$

$$\begin{aligned}
-s_1 + x_1 a + b &= -z_3 + x_1 z_0 + z_1 \leq y_1 \\
-s_2 + x_2 a + b &= -z_4 + x_2 z_0 + z_1 \leq y_2 \\
-s_3 + x_3 a + b &= -z_5 + x_3 z_0 + z_1 \leq y_3 \\
&\vdots \\
-s_N + x_N a + b &= -z_{N+2} + x_N z_0 + z_1 \leq y_N
\end{aligned}
\leftrightarrow
\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ x_1 & 1 & -1 & 0 & 0 & \dots & 0 \\ x_2 & 1 & 0 & -1 & 0 & \dots & 0 \\ x_2 & 1 & 0 & 0 & -1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_N & 1 & 0 & 0 & 0 & \dots & -1 \end{pmatrix}
\begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ \vdots \\ z_{N+1} \\ z_{N+2} \end{pmatrix}
\leq
\begin{pmatrix} 0 \\ 0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{pmatrix}$$

with

$$A_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ x_1 & 1 & -1 & 0 & 0 & \dots & 0 \\ x_2 & 1 & 0 & -1 & 0 & \dots & 0 \\ x_2 & 1 & 0 & 0 & -1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_N & 1 & 0 & 0 & 0 & \dots & -1 \end{pmatrix}$$

and the final condition, all variables s bigger than one can be write as $-s = -I_z[2 : \text{end}] \leq 0$ where the zero is the vector of zeros.

Now implementing the matrixes A_1, A_2 and the vectors b, f in the code

```

clf;
f = [0; 0; ones(N,1)];
b = [0; 0; ynoise]; b_atypical = [0; 0; ynoise_atypical];

A1 = zeros(N+2); A1(3:end,1:2) = J; A1(3:end,3:end) = eye(N);
A2 = zeros(N+2); A2(3:end,1:2) = J; A2(3:end,3:end) = -eye(N);

```

2. Problem solution

Solve the problem using the y measures from the previous exercise (with and without the atypical data) and plot the results over the L_2 results. Which norm has a better result?

Without atypical data

```

%A1 \ b

opti3 = casadi.Opti();
z_opt3 = opti3.variable(N+2);

opti3.minimize(f'*z_opt3);

opti3.subject_to(A1*z_opt3 >= b);
opti3.subject_to(A2*z_opt3 <= b);
opti3.subject_to(-z_opt3(3:end) <= zeros(N,1));

opti3.solver('ipopt');
sol3 = opti3.solve();

```

This is Ipopt version 3.12.3, running with linear solver mumps.
NOTE: Other linear solvers might be more efficient (see Ipopt documentation).

```

Number of nonzeros in equality constraint Jacobian...:      0
Number of nonzeros in inequality constraint Jacobian.:    2078
Number of nonzeros in Lagrangian Hessian.....:          0

```

```

Total number of variables.....:    32
    variables with only lower bounds:      0
    variables with lower and upper bounds:  0
    variables with only upper bounds:      0
Total number of equality constraints.....:      0
Total number of inequality constraints.....:    94
    inequality constraints with only lower bounds:    32
    inequality constraints with lower and upper bounds:  0
    inequality constraints with only upper bounds:    62

```

iter	objective	inf_pr	inf_du	lg(mu)	d	lg(rg)	alpha_du	alpha_pr	ls
0	0.0000000e+000	1.87e+001	5.00e-001	-1.0	0.00e+000	-	0.00e+000	0.00e+000	0
1	3.1030639e+000	2.59e+000	5.37e+000	-1.0	1.84e+001	-	3.25e-001	8.63e-001h	1
2	1.3735394e+001	0.00e+000	3.90e+001	-1.0	1.56e+000	-	4.55e-001	9.90e-001h	1
3	1.1344994e+001	0.00e+000	6.71e+001	-1.0	3.73e-001	-	1.00e+000	9.91e-001f	1
4	9.3040192e+000	0.00e+000	1.00e-006	-1.0	4.44e-001	-	1.00e+000	1.00e+000f	1
5	2.4837846e+000	0.00e+000	1.58e+005	-5.7	3.19e-001	-	1.00e+000	7.84e-001f	1
6	2.0124862e+000	0.00e+000	7.33e+004	-5.7	7.06e-002	-	9.17e-001	5.37e-001f	1
7	1.8889895e+000	0.00e+000	3.74e+004	-5.7	5.89e-002	-	7.57e-001	4.89e-001f	1
8	1.8482983e+000	0.00e+000	1.77e+004	-5.7	2.25e-002	-	7.69e-001	5.27e-001f	1
9	1.8326806e+000	0.00e+000	6.43e+003	-5.7	2.60e-002	-	5.91e-001	6.37e-001f	1

```

iter    objective    inf_pr    inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls

```

```

10 1.8256614e+000 0.00e+000 1.80e+003 -5.7 2.79e-002 - 8.04e-001 7.20e-001f 1
11 1.8238984e+000 0.00e+000 2.80e+002 -5.7 6.50e-003 - 9.48e-001 8.44e-001f 1
12 1.8237706e+000 0.00e+000 1.85e-011 -5.7 7.17e-005 - 1.00e+000 1.00e+000f 1
13 1.8237082e+000 0.00e+000 1.96e+000 -8.6 1.68e-005 - 1.00e+000 9.89e-001f 1
14 1.8237077e+000 0.00e+000 5.65e-001 -8.6 1.06e-007 - 8.51e-001 7.11e-001f 1
15 1.8237076e+000 0.00e+000 2.55e-004 -8.6 8.52e-008 - 9.99e-001 1.00e+000f 1
16 1.8237076e+000 0.00e+000 2.51e-014 -8.6 2.91e-009 - 1.00e+000 1.00e+000h 1

```

Number of Iterations....: 16

	(scaled)	(unscaled)
Objective.....	1.8237075577790940e+000	1.8237075577790940e+000
Dual infeasibility.....	2.5091040356528538e-014	2.5091040356528538e-014
Constraint violation....	0.0000000000000000e+000	0.0000000000000000e+000
Complementarity.....	2.5745917457967955e-009	2.5745917457967955e-009
Overall NLP error.....	2.5745917457967955e-009	2.5745917457967955e-009

Number of objective function evaluations	= 17
Number of objective gradient evaluations	= 17
Number of equality constraint evaluations	= 0
Number of inequality constraint evaluations	= 17
Number of equality constraint Jacobian evaluations	= 0
Number of inequality constraint Jacobian evaluations	= 17
Number of Lagrangian Hessian evaluations	= 16
Total CPU secs in IPOPT (w/o function evaluations)	= 0.204
Total CPU secs in NLP function evaluations	= 0.003

EXIT: Optimal Solution Found.

	solver	t_proc	(avg)	t_wall	(avg)	n_eval
nlp_f		0	(0)	0	(0)	17
nlp_g		2.00ms	(117.65us)	2.00ms	(117.76us)	17
nlp_grad_f		0	(0)	0	(0)	18
nlp_hess_l		0	(0)	0	(0)	16
nlp_jac_g		1.00ms	(55.56us)	1.00ms	(55.61us)	18
total		211.00ms	(211.00ms)	211.05ms	(211.05ms)	1

```

zopt3 = sol3.value(z_opt3);
zopt3'

```

```

ans = 1x32
    2.9321    4.2577    0.1411    0.0068    0.0724    0.0101    0.0752    0.0647 ...

```

```

%plot(zopt3(3:end),'-o')
xlabel('i'); ylabel('s')
disp(['Optimal solution found: a = ' num2str(zopt3(1)) ' b = ' num2str(zopt3(2))]);

```

Optimal solution found: a = 2.9321 b = 4.2577

```

clf;
figure(Position=[100 100 900 400]);

subplot(1,2,1)
hold on
plot(xrand, ynoise, '.', MarkerSize=25, DisplayName='Gauss noise', Color='black')
plot(xrand, ab(1)*xrand+ab(2), '-', LineWidth=2, Color=[.7 .0 .3], DisplayName=['||_2: a=' num2str(ab(1)) ' b=' num2str(ab(2))])
plot(xrand,zopt3(1)*xrand+zopt3(2), '-', LineWidth=1, Color=[.1 0 .6], DisplayName=['||_1: a=' num2str(zopt3(1)) ' b=' num2str(zopt3(2))]);
leg = legend('Location','northwest');
title(leg,'Data and Fits')

```

```
xlabel('x'); ylabel('y'); title('without atypical data', 'Interpreter','latex')
hold off
disp(['Residual sum:  $\sum si =$  ' num2str(sum(zopt3(3:end))))];
```

Residual sum: $\sum si = 1.8237$

Atypical data

```
opti3_aty = casadi.Opti();
z_opt3_aty = opti3_aty.variable(N+2);

opti3_aty.minimize(f'*z_opt3_aty);

opti3_aty.subject_to(A1*z_opt3_aty >= b_atypical);
opti3_aty.subject_to(A2*z_opt3_aty <= b_atypical);
opti3_aty.subject_to(-z_opt3_aty(3:end) <= zeros(N,1));

opti3_aty.solver('ipopt');
sol3_aty = opti3_aty.solve();
```

This is Ipopt version 3.12.3, running with linear solver mumps.

NOTE: Other linear solvers might be more efficient (see Ipopt documentation).

```
Number of nonzeros in equality constraint Jacobian...:      0
Number of nonzeros in inequality constraint Jacobian.:    2078
Number of nonzeros in Lagrangian Hessian.....:          0
```

```
Total number of variables.....:      32
      variables with only lower bounds:      0
      variables with lower and upper bounds:  0
      variables with only upper bounds:      0
Total number of equality constraints.....:      0
Total number of inequality constraints.....:     94
      inequality constraints with only lower bounds:  32
      inequality constraints with lower and upper bounds:  0
      inequality constraints with only upper bounds:  62
```

iter	objective	inf_pr	inf_du	lg(mu)	d	lg(rg)	alpha_du	alpha_pr	ls
0	0.0000000e+000	1.87e+001	5.00e-001	-1.0	0.00e+000	-	0.00e+000	0.00e+000	0
1	2.0015184e-001	1.77e+001	1.10e+000	-1.0	1.78e+001	-	3.40e-002	5.56e-002h	1
2	7.9248264e-001	1.53e+001	1.51e+000	-1.0	1.74e+001	-	5.61e-002	1.36e-001f	1
3	5.0000921e+000	4.84e+000	5.42e+000	-1.0	1.48e+001	-	2.67e-001	6.83e-001h	1
4	6.6812194e+000	4.54e+000	5.23e+000	-1.0	2.41e+000	-	5.71e-002	6.09e-002h	1
5	2.0516499e+001	1.54e+000	1.13e+001	-1.0	3.84e+000	-	4.15e-001	6.45e-001h	1
6	2.5562739e+001	0.00e+000	8.91e+001	-1.0	2.00e+000	-	2.22e-001	9.90e-001h	1
7	1.9362092e+001	0.00e+000	1.02e+002	-1.0	9.41e-001	-	1.00e+000	9.91e-001f	1
8	1.5794812e+001	0.00e+000	1.00e-006	-1.0	6.46e-001	-	1.00e+000	1.00e+000f	1
9	9.2959021e+000	0.00e+000	2.31e+005	-5.7	4.07e-001	-	9.92e-001	7.94e-001f	1
10	8.8876161e+000	0.00e+000	1.12e+005	-5.7	6.40e-002	-	9.71e-001	5.33e-001f	1
11	8.7500900e+000	0.00e+000	4.04e+004	-5.7	3.98e-002	-	8.15e-001	6.40e-001f	1
12	8.7315483e+000	0.00e+000	2.09e+004	-5.7	1.19e-002	-	8.92e-001	4.85e-001f	1
13	8.7153919e+000	0.00e+000	3.41e+003	-5.7	3.38e-002	-	6.30e-001	8.36e-001f	1
14	8.7129899e+000	0.00e+000	2.07e+002	-5.7	4.42e-003	-	9.19e-001	9.39e-001f	1
15	8.7128782e+000	0.00e+000	1.84e-011	-5.7	9.25e-005	-	1.00e+000	1.00e+000f	1
16	8.7128161e+000	0.00e+000	2.42e+000	-8.6	1.51e-005	-	1.00e+000	9.87e-001f	1
17	8.7128155e+000	0.00e+000	5.08e-001	-8.6	1.38e-007	-	7.97e-001	7.90e-001f	1
18	8.7128153e+000	0.00e+000	4.86e-003	-8.6	9.81e-008	-	9.79e-001	1.00e+000f	1
19	8.7128153e+000	0.00e+000	2.51e-014	-8.6	2.71e-009	-	1.00e+000	1.00e+000h	1

Number of Iterations.....: 19

	(scaled)	(unscaled)
Objective.....:	8.7128152997301669e+000	8.7128152997301669e+000
Dual infeasibility.....:	2.5091040356528538e-014	2.5091040356528538e-014
Constraint violation.....:	0.0000000000000000e+000	0.0000000000000000e+000
Complementarity.....:	2.6011777452244518e-009	2.6011777452244518e-009
Overall NLP error.....:	2.6011777452244518e-009	2.6011777452244518e-009

Number of objective function evaluations	=	20
Number of objective gradient evaluations	=	20
Number of equality constraint evaluations	=	0
Number of inequality constraint evaluations	=	20
Number of equality constraint Jacobian evaluations	=	0
Number of inequality constraint Jacobian evaluations	=	20
Number of Lagrangian Hessian evaluations	=	19
Total CPU secs in IPOPT (w/o function evaluations)	=	0.125
Total CPU secs in NLP function evaluations	=	0.003

EXIT: Optimal Solution Found.

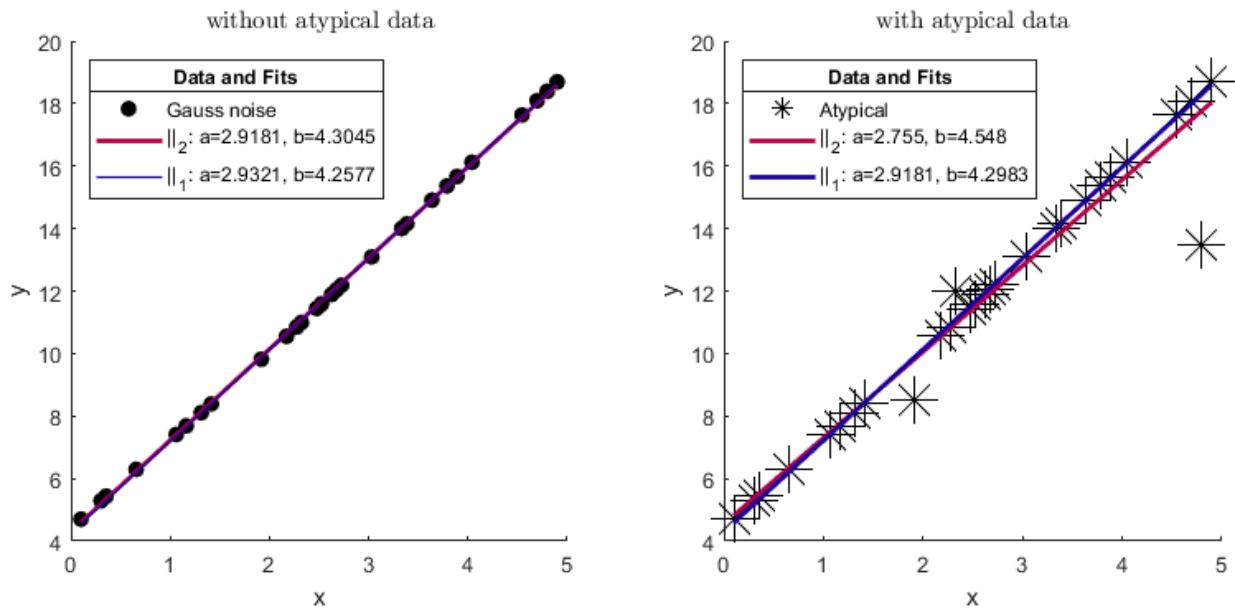
solver	:	t_proc	(avg)	t_wall	(avg)	n_eval
nlp_f		0	(0)	0	(0)	20
nlp_g		0	(0)	0	(0)	20
nlp_grad_f		0	(0)	0	(0)	21
nlp_hess_l		0	(0)	0	(0)	19
nlp_jac_g		3.00ms	(142.86us)	3.00ms	(142.81us)	21
total		130.00ms	(130.00ms)	130.03ms	(130.03ms)	1

```
zopt3_aty = sol3_aty.value(z_opt3_aty);
zopt3_aty'
```

```
ans = 1x32
    2.9181    4.2983    0.1054    0.0203    0.0827   -0.0000    0.1029    0.0680 ...
```

```
subplot(1,2,2)
hold on
%plot(xrand, ynoise, '.', MarkerSize=20, DisplayName='Gauss noise', Color='blue')
plot(xrand, ynoise_atypical, '*', MarkerSize=20, DisplayName='Atypical', Color='black')
plot(xrand, ab_atypical(1)*xrand+ab_atypical(2), '-', LineWidth=2, Color=[.7 .0 .3], DisplayName='')
plot(xrand,zopt3_aty (1)*xrand+zopt3_aty(2), '-', LineWidth=2, Color=[.1 0 .6], DisplayName='')
leg = legend('Location','northwest');
title(leg,'Data and Fits')
xlabel('x'); ylabel('y'); title('with atypical data', 'Interpreter','latex')
hold off
sgtitle('Fits comparation norms  $||\cdot||_1$  vs  $||\cdot||_2$ ',Interpreter='Latex');
```

Fits comparison norms $\|\cdot\|_1$ vs $\|\cdot\|_2$



```
disp(['RESIDUAL ERRORS' newline ...
    '- without atypical data:' newline ...
    '    Residual sum in ||_1: Σ si = ' num2str(sum(zopt3(3:end))) newline ...
    '    residual error in ||_2 r = ' num2str(sum(ab(1)*xrand+ab(2)-ynoise)) newline ...
    '- with atypical data' newline ...
    '    Residual sum in ||_1: Σ si = ' num2str(sum(zopt3_aty(3:end))) newline ...
    '    residual error in ||_2 r = ' num2str(sum(ab_atypical(1)*xrand+ab_atypical(2)-ynoise))
```

RESIDUAL ERRORS

- without atypical data:

Residual sum in $\|\cdot\|_1$: $\sum si = 1.8237$
 residual error in $\|\cdot\|_2$ $r = 1.3145e-13$

- with atypical data

Residual sum in $\|\cdot\|_1$: $\sum si = 8.7128$
 residual error in $\|\cdot\|_2$ $r = 9.9476e-14$

3.

Solve the problem with CasADi and compare the results.

```
%A1 \ b_atypical %ab = J \ ynoise;
```

4. Integral Equation

Consider the following integral equation

$$Ax(t) := \int_0^1 e^{st} x(t) dt = \frac{e^{s+1} - 1}{s + 1} =: y(s), \quad 0 \leq s \leq 1$$

a) Exact solution

Find the exact solution (*Advice:* Interpret the operator $A : L^2(0, 1) \rightarrow L^2(0, 1)$ as the Laplace transform).

Let's take the solution to be an exponential function $x(t) = e^{at}$

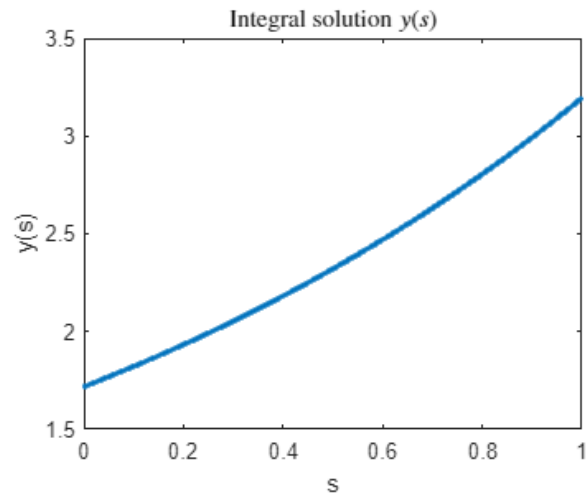
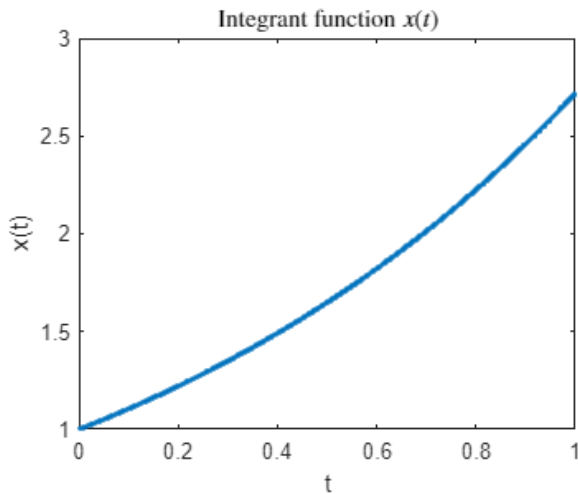
$$\int_0^1 e^{st} e^{at} dt = \int_0^1 e^{(s+a)t} dt = \frac{1}{s+a} e^{(s+a)t} \Big|_0^1 = \frac{e^{s+a} - 1}{s+a} = \frac{e^{s+1} - 1}{s+1}$$

then $a = 1$ and the solution is $x(x) = e^t$.

```
t = 0:0.001:1;

clf
figure(Position=[10 10 900 300])
subplot(1,2,1)
plot(t,exp(t),'.');
xlabel('t'); ylabel('x(t)');
xlim([0 1]); %ylim([-14 5]);
title('Integrand function $x(t)$', Interpreter='latex');

subplot(1,2,2)
plot(t, (exp(t+1)-1)./(t+1), '.')
xlabel('s'); ylabel('y(s)');
title('Integral solution $y(s)$', Interpreter='latex');
```



b) Problem Discretization

Use the Trapezoidal rule with an integer step $h = 1/n$ and $s = ih$ and $i = 1, \dots, n$ to get a linear system of equations.

Remembering the trapezoidal rule. Let $\{x_k\}$ be a partition of $[0, 1]$ such that $0 = t_1 < t_2 < \dots < t_{n-1} < t_n = 1$ and $\Delta t_k = \Delta t = 1/n$ be the length of the k -th subinterval

$$\int_a^b f(t)dx \approx \sum_{k=1}^{n-1} \frac{f(t_{k+1}) + f(t_k)}{2} \Delta t_k$$

and applying it to the integral equation

$$\begin{aligned} \int_0^1 e^{st} x(t) dt &\approx \sum_{k=1}^{n-1} \frac{e^{st_{k+1}} x_{k+1} + e^{st_k} x_k}{2} \cdot \frac{1}{n} = \frac{1}{2n} \sum_{k=1}^{n-1} (e^{st_{k+1}} x_{k+1} + e^{st_k} x_k) \\ &= \frac{1}{2n} [e^{st_2} x_2 + e^{st_1} x_1 + e^{st_3} x_3 + e^{st_2} x_2 + e^{st_4} x_4 + e^{st_3} x_3 + \dots + e^{st_n} x_n + e^{st_{n-1}} x_{n-1}] \\ &= \frac{1}{2n} [e^{st_1} x_1 + 2e^{st_2} x_2 + 2e^{st_3} x_3 + 2e^{st_4} x_4 + \dots + 2e^{st_{n-1}} x_{n-1} + e^{st_n} x_n] = \frac{e^{s+1} - 1}{s+1} \end{aligned}$$

with $h = 1/n$ and $s = ih$. Rewriting to the following system for any s . Then

$$y(s) = \int_0^1 e^{st} x(t) dt \approx \frac{1}{2n} (e^{st_1} \quad 2e^{st_2} \quad 2e^{st_3} \quad \dots \quad 2e^{st_{n-1}} \quad e^{st_n}) \cdot (x_1 \quad x_2 \quad x_3 \quad \dots \quad x_{n-1} \quad x_n) = \frac{e^{s+1} - 1}{s+1}$$

To build the system now let's evaluate the previous expression to each i

$$s = 0 : \quad \frac{1}{2n} (1 \quad 2 \quad 2 \quad \dots \quad 2 \quad 1) \cdot (x_1 \quad x_2 \quad x_3 \quad \dots \quad x_{n-1} \quad x_n) = e^1 - 1$$

$$s = h : \quad \frac{1}{2n} (e^{ht_1} \quad 2e^{ht_2} \quad 2e^{ht_3} \quad \dots \quad 2e^{ht_{n-1}} \quad e^{ht_n}) \cdot (x_1 \quad x_2 \quad x_3 \quad \dots \quad x_{n-1} \quad x_n) = \frac{e^{h+1} - 1}{h+1}$$

$$s = 2h : \quad \frac{1}{2n} (e^{2ht_1} \quad 2e^{2ht_2} \quad 2e^{2ht_3} \quad \dots \quad 2e^{2ht_{n-1}} \quad e^{2ht_n}) \cdot (x_1 \quad x_2 \quad x_3 \quad \dots \quad x_{n-1} \quad x_n) = \frac{e^{2h+1} - 1}{2h+1}$$

⋮

$$s = (n-1)h : \quad \frac{1}{2n} (e^{(n-1)ht_1} \quad 2e^{(n-1)ht_2} \quad 2e^{(n-1)ht_3} \quad \dots \quad 2e^{(n-1)ht_{n-1}} \quad e^{(n-1)ht_n}) \cdot (x_1 \quad x_2 \quad x_3 \quad \dots \quad x_{n-1} \quad x_n) = \frac{e^{(n-1)h+1} - 1}{(n-1)h+1}$$

$$s = nh : \quad \frac{1}{2n} (e^{nht_1} \quad 2e^{nht_2} \quad 2e^{nht_3} \quad \dots \quad 2e^{nht_{n-1}} \quad e^{nht_n}) \cdot (x_1 \quad x_2 \quad x_3 \quad \dots \quad x_{n-1} \quad x_n) = \frac{e^{nh+1} - 1}{nh+1}$$

that can be write as $Ax = b$ where

$$A_{(n \times n)} = \frac{1}{2n} \begin{pmatrix} e^{ht_1} & 2e^{ht_2} & 2e^{ht_3} & \dots & 2e^{ht_{n-1}} & e^{ht_n} \\ e^{2ht_1} & 2e^{2ht_2} & 2e^{2ht_3} & \dots & 2e^{2ht_{n-1}} & e^{2ht_n} \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ e^{(n-1)ht_1} & 2e^{(n-1)ht_2} & 2e^{(n-1)ht_3} & \dots & 2e^{(n-1)ht_{n-1}} & e^{(n-1)ht_n} \\ e^{nht_1} & 2e^{nht_2} & 2e^{nht_3} & \dots & 2e^{nht_{n-1}} & e^{nht_n} \end{pmatrix}$$

$$x_{(n \times 1)} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix}, \quad b_{(n \times 1)} = \begin{pmatrix} \frac{e^{h+1} - 1}{h + 1} \\ \frac{e^{2h+1} - 1}{2h + 1} \\ \vdots \\ \frac{e^{(n-1)h+1} - 1}{(n-1)h + 1} \\ \frac{e^{nh+1} - 1}{nh + 1} \end{pmatrix}$$

c) Code Implementation

Find the solution to the linear system for $n = 4, 8, 16, 32$ and plot them. Point out the number condition of each linear system using the command **cond** on MATLAB.

Solution by Gauss eq

First, let's find the solution by solving the Gauss equations with the function **System(n,α)**

```
[time1, sol1, co1, M1, b1] = System(4,0);
[time2, sol2, co2, M2, b2] = System(8,0);
[time3, sol3, co3, M3, b3] = System(16,0);
```

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND = 8.993789e-19.

```
[time4, sol4, co4, M4, b4] = System(32,0);
```

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND = 7.263197e-20.

```
[time100, sol100, co100, M100, b100] = System(128,0);
```

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND = 4.634772e-21.

```
disp(['cond n=4: ' num2str(round(co1,2)) newline 'cond n=8: ' num2str(co2) newline ...
      'cond n=16: ' num2str(co3) newline 'cond n=32: ' num2str(co4) newline 'cond n=128: ' num2str(co5)]
```

```
cond n=4: 23063.77
cond n=8: 9058044760132.311
cond n=16: 3.542464016391407e+17
cond n=32: 3.190396294578456e+18
cond n=128: 2.742725726910894e+19
```

where this variable **cond** is the condition number for inversion, it measures the sensitivity of the solution of a system of linear equations to errors in the data. By default, Matlab calculates the 2-norm

$$\text{cond} = \kappa(A) = \|A\|_2 \|A^{-1}\|_2$$

Solution by CasADI

Now let solve the problem with CasADI as the following optimization problem

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2$$

the function **SolCasADI(n,A,b)** is used here.

```
x1 = SolCasADI(4, M1, b1);
```

This is Ipopt version 3.12.3, running with linear solver mumps.

NOTE: Other linear solvers might be more efficient (see Ipopt documentation).

```
Number of nonzeros in equality constraint Jacobian...: 0
Number of nonzeros in inequality constraint Jacobian.: 0
Number of nonzeros in Lagrangian Hessian.....: 10
```

```
Total number of variables.....: 4
      variables with only lower bounds: 0
      variables with lower and upper bounds: 0
      variables with only upper bounds: 0
Total number of equality constraints.....: 0
Total number of inequality constraints.....: 0
      inequality constraints with only lower bounds: 0
      inequality constraints with lower and upper bounds: 0
      inequality constraints with only upper bounds: 0
```

```
iter   objective   inf_pr   inf_du lg(mu)  ||d|| lg(rg) alpha_du alpha_pr ls
  0  2.6943424e+001  0.00e+000  8.15e+000  -1.0  0.00e+000  -  0.00e+000  0.00e+000  0
  1  4.6276502e-024  0.00e+000  8.34e-016  -1.0  3.52e+000  -  1.00e+000  1.00e+000f  1
```

Number of Iterations.....: 1

	(scaled)	(unscaled)
Objective.....	4.6276501576568773e-024	4.6276501576568773e-024
Dual infeasibility.....	8.3385863701727979e-016	8.3385863701727979e-016
Constraint violation.....	0.0000000000000000e+000	0.0000000000000000e+000
Complementarity.....	0.0000000000000000e+000	0.0000000000000000e+000
Overall NLP error.....	8.3385863701727979e-016	8.3385863701727979e-016

```
Number of objective function evaluations = 2
Number of objective gradient evaluations = 2
Number of equality constraint evaluations = 0
Number of inequality constraint evaluations = 0
Number of equality constraint Jacobian evaluations = 0
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations = 1
Total CPU secs in IPOPT (w/o function evaluations) = 0.148
Total CPU secs in NLP function evaluations = 0.000
```

EXIT: Optimal Solution Found.

solver	:	t_proc	(avg)	t_wall	(avg)	n_eval
nlp_f		0 (0)	0 (0)	2
nlp_grad_f		0 (0)	0 (0)	3
nlp_hess_l		0 (0)	0 (0)	1
total		154.00ms	(154.00ms)	154.03ms	(154.03ms)	1

```
x2 = SolCasADI(8, M2, b2);
```

This is Ipopt version 3.12.3, running with linear solver mumps.

NOTE: Other linear solvers might be more efficient (see Ipopt documentation).

```
Number of nonzeros in equality constraint Jacobian...: 0
Number of nonzeros in inequality constraint Jacobian.: 0
```

```

Number of nonzeros in Lagrangian Hessian.....: 36

Total number of variables.....: 8
    variables with only lower bounds: 0
    variables with lower and upper bounds: 0
    variables with only upper bounds: 0
Total number of equality constraints.....: 0
Total number of inequality constraints.....: 0
    inequality constraints with only lower bounds: 0
    inequality constraints with lower and upper bounds: 0
    inequality constraints with only upper bounds: 0

iter   objective   inf_pr   inf_du lg(mu)  ||d|| lg(rg) alpha_du alpha_pr ls
  0 4.9950145e+001 0.00e+000 8.57e+000 -1.0 0.00e+000 - 0.00e+000 0.00e+000 0
  1 3.6378098e-006 0.00e+000 3.08e-004 -1.0 3.08e+000 -4.0 1.00e+000 1.00e+000f 1
  2 1.4262148e-006 0.00e+000 3.79e-006 -5.7 1.14e-001 -4.5 1.00e+000 1.00e+000f 1
  3 2.0457279e-007 0.00e+000 1.41e-006 -8.6 1.27e-001 -5.0 1.00e+000 1.00e+000f 1
  4 5.8550796e-009 0.00e+000 2.39e-007 -8.6 6.45e-002 -5.4 1.00e+000 1.00e+000f 1
  5 4.4693305e-011 0.00e+000 1.55e-008 -8.6 1.25e-002 -5.9 1.00e+000 1.00e+000f 1
  6 2.0353632e-011 0.00e+000 6.50e-010 -9.0 1.58e-003 -6.4 1.00e+000 1.00e+000f 1

```

Number of Iterations.....: 6

	(scaled)	(unscaled)
Objective.....	2.0353632495740268e-011	2.0353632495740268e-011
Dual infeasibility.....	6.4957062359547385e-010	6.4957062359547385e-010
Constraint violation.....	0.0000000000000000e+000	0.0000000000000000e+000
Complementarity.....	0.0000000000000000e+000	0.0000000000000000e+000
Overall NLP error.....	6.4957062359547385e-010	6.4957062359547385e-010

```

Number of objective function evaluations      = 7
Number of objective gradient evaluations      = 7
Number of equality constraint evaluations      = 0
Number of inequality constraint evaluations    = 0
Number of equality constraint Jacobian evaluations = 0
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations      = 6
Total CPU secs in IPOPT (w/o function evaluations) = 0.008
Total CPU secs in NLP function evaluations      = 0.000

```

EXIT: Optimal Solution Found.

solver	:	t_proc	(avg)	t_wall	(avg)	n_eval
nlp_f		0 (0)	0 (0)	7
nlp_grad_f		0 (0)	0 (0)	8
nlp_hess_l		0 (0)	0 (0)	6
total		9.00ms (9.00ms)	9.00ms (9.00ms)	1

```
x3 = SolCasADI(16, M3, b3);
```

This is Ipopt version 3.12.3, running with linear solver mumps.
NOTE: Other linear solvers might be more efficient (see Ipopt documentation).

```

Number of nonzeros in equality constraint Jacobian...: 0
Number of nonzeros in inequality constraint Jacobian.: 0
Number of nonzeros in Lagrangian Hessian.....: 136

```

```

Total number of variables.....: 16
    variables with only lower bounds: 0
    variables with lower and upper bounds: 0
    variables with only upper bounds: 0
Total number of equality constraints.....: 0
Total number of inequality constraints.....: 0

```

```

inequality constraints with only lower bounds:      0
inequality constraints with lower and upper bounds: 0
inequality constraints with only upper bounds:      0

iter   objective   inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr ls
  0 9.6118472e+001 0.00e+000 8.60e+000 -1.0 0.00e+000 - 0.00e+000 0.00e+000 0
  1 2.6605883e-006 0.00e+000 2.93e-004 -1.0 2.93e+000 -4.0 1.00e+000 1.00e+000f 1
  2 8.8600597e-007 0.00e+000 2.49e-006 -5.7 7.48e-002 -4.5 1.00e+000 1.00e+000f 1
  3 1.0617175e-007 0.00e+000 8.82e-007 -8.6 7.94e-002 -5.0 1.00e+000 1.00e+000f 1
  4 2.6596095e-009 0.00e+000 1.32e-007 -8.6 3.55e-002 -5.4 1.00e+000 1.00e+000f 1
  5 2.8730167e-010 0.00e+000 7.25e-009 -8.6 5.87e-003 -5.9 1.00e+000 1.00e+000f 1

```

Number of Iterations.....: 5

```

                                (scaled)                (unscaled)
Objective.....: 2.8730166532400370e-010 2.8730166532400370e-010
Dual infeasibility.....: 7.2518550442247229e-009 7.2518550442247229e-009
Constraint violation.....: 0.0000000000000000e+000 0.0000000000000000e+000
Complementarity.....: 0.0000000000000000e+000 0.0000000000000000e+000
Overall NLP error.....: 7.2518550442247229e-009 7.2518550442247229e-009

```

```

Number of objective function evaluations      = 6
Number of objective gradient evaluations      = 6
Number of equality constraint evaluations      = 0
Number of inequality constraint evaluations    = 0
Number of equality constraint Jacobian evaluations = 0
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations      = 5
Total CPU secs in IPOPT (w/o function evaluations) = 0.121
Total CPU secs in NLP function evaluations    = 0.001

```

EXIT: Optimal Solution Found.

```

solver : t_proc      (avg)  t_wall      (avg)  n_eval
nlp_f   |      0 (      0)      0 (      0)      6
nlp_grad_f |      0 (      0)      0 (      0)      7
nlp_hess_l | 1.00ms (200.00us) 1.00ms (200.20us) 5
total   | 155.00ms (155.00ms) 154.90ms (154.90ms) 1

```

```
x4 = SolCasADI(32, M4, b4);
```

This is Ipopt version 3.12.3, running with linear solver mumps.

NOTE: Other linear solvers might be more efficient (see Ipopt documentation).

```

Number of nonzeros in equality constraint Jacobian...: 0
Number of nonzeros in inequality constraint Jacobian.: 0
Number of nonzeros in Lagrangian Hessian.....: 528

```

```

Total number of variables.....: 32
    variables with only lower bounds: 0
    variables with lower and upper bounds: 0
    variables with only upper bounds: 0
Total number of equality constraints.....: 0
Total number of inequality constraints.....: 0
    inequality constraints with only lower bounds: 0
    inequality constraints with lower and upper bounds: 0
    inequality constraints with only upper bounds: 0

```

```

iter   objective   inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr ls
  0 1.8853286e+002 0.00e+000 8.59e+000 -1.0 0.00e+000 - 0.00e+000 0.00e+000 0
  1 1.8820221e-006 0.00e+000 2.82e-004 -1.0 2.82e+000 -4.0 1.00e+000 1.00e+000f 1
  2 4.8321288e-007 0.00e+000 1.70e-006 -5.7 5.10e-002 -4.5 1.00e+000 1.00e+000f 1
  3 5.1288904e-008 0.00e+000 5.33e-007 -8.6 4.80e-002 -5.0 1.00e+000 1.00e+000f 1

```



```

4 1.1985394e-009 0.00e+000 7.44e-008 -8.6 2.01e-002 -5.4 1.00e+000 1.00e+000f 1
5 2.1944447e-010 0.00e+000 4.45e-009 -8.6 3.60e-003 -5.9 1.00e+000 1.00e+000f 1

```

Number of Iterations....: 5

	(scaled)	(unscaled)
Objective.....	2.1944447377641908e-010	2.1944447377641908e-010
Dual infeasibility.....	4.4451118540402028e-009	4.4451118540402028e-009
Constraint violation....	0.0000000000000000e+000	0.0000000000000000e+000
Complementarity.....	0.0000000000000000e+000	0.0000000000000000e+000
Overall NLP error.....	4.4451118540402028e-009	4.4451118540402028e-009

```

Number of objective function evaluations      = 6
Number of objective gradient evaluations     = 6
Number of equality constraint evaluations     = 0
Number of inequality constraint evaluations   = 0
Number of equality constraint Jacobian evaluations = 0
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations    = 5
Total CPU secs in IPOPT (w/o function evaluations) = 0.006
Total CPU secs in NLP function evaluations    = 0.000

```

EXIT: Optimal Solution Found.

solver	:	t_proc	(avg)	t_wall	(avg)	n_eval
nlp_f		0 (0)	0 (0)	6
nlp_grad_f		0 (0)	0 (0)	7
nlp_hess_l		0 (0)	0 (0)	5
total		8.00ms (8.00ms)	8.00ms (8.00ms)	1

```
x100 = SolCasADI(128, M100, b100);
```

This is Ipopt version 3.12.3, running with linear solver mumps.

NOTE: Other linear solvers might be more efficient (see Ipopt documentation).

```

Number of nonzeros in equality constraint Jacobian...: 0
Number of nonzeros in inequality constraint Jacobian.: 0
Number of nonzeros in Lagrangian Hessian.....: 8256

```

```

Total number of variables.....: 128
    variables with only lower bounds: 0
    variables with lower and upper bounds: 0
    variables with only upper bounds: 0
Total number of equality constraints.....: 0
Total number of inequality constraints.....: 0
    inequality constraints with only lower bounds: 0
    inequality constraints with lower and upper bounds: 0
    inequality constraints with only upper bounds: 0

```

iter	objective	inf_pr	inf_du	lg(mu)	d	lg(rg)	alpha_du	alpha_pr	ls
0	7.4315539e+002	0.00e+000	8.56e+000	-1.0	0.00e+000	-	0.00e+000	0.00e+000	0
1	2.3519607e-006	0.00e+000	2.73e-004	-1.0	2.73e+000	-4.0	1.00e+000	1.00e+000f	1
2	2.0594929e-007	0.00e+000	6.94e-007	-5.7	2.08e-002	-4.5	1.00e+000	1.00e+000f	1
3	1.9813707e-008	0.00e+000	1.97e-007	-8.6	1.78e-002	-5.0	1.00e+000	1.00e+000f	1
4	4.0196681e-010	0.00e+000	2.59e-008	-8.6	6.99e-003	-5.4	1.00e+000	1.00e+000f	1
5	6.7343212e-011	0.00e+000	1.49e-009	-8.6	1.21e-003	-5.9	1.00e+000	1.00e+000f	1

Number of Iterations....: 5

	(scaled)	(unscaled)
Objective.....	6.7343212389345426e-011	6.7343212389345426e-011
Dual infeasibility.....	1.4908838812067784e-009	1.4908838812067784e-009
Constraint violation....	0.0000000000000000e+000	0.0000000000000000e+000

```

Complementarity.....:  0.0000000000000000e+000  0.0000000000000000e+000
Overall NLP error.....:  1.4908838812067784e-009  1.4908838812067784e-009

```

```

Number of objective function evaluations      = 6
Number of objective gradient evaluations     = 6
Number of equality constraint evaluations    = 0
Number of inequality constraint evaluations  = 0
Number of equality constraint Jacobian evaluations = 0
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations    = 5
Total CPU secs in IPOPT (w/o function evaluations) = 0.017
Total CPU secs in NLP function evaluations   = 0.010

```

EXIT: Optimal Solution Found.

solver	:	t_proc	(avg)	t_wall	(avg)	n_eval
nlp_f		0	(0)	0	(0)	6
nlp_grad_f		0	(0)	0	(0)	7
nlp_hess_l		10.00ms	(2.00ms)	10.00ms	(2.00ms)	5
total		28.00ms	(28.00ms)	28.01ms	(28.01ms)	1

```

clf
figure(Position=[10 10 900 600])
subplot(3,1,1)
hold on
plot(time1, sol1', 'o-', DisplayName='Gauss eq, n=4', MarkerSize=8);
plot(time2, sol2', 'o-', DisplayName='Gauss eq, n=8');
plot(time3, sol3', 'o-', DisplayName='Gauss eq, n=15');
plot(time4, sol4', 'o-', DisplayName='Gauss eq, n=32');
plot(time100, sol100', 'o-', DisplayName='Gauss eq, n=128');
title('$x(t)$ solution solving the Gauss eq', Interpreter='latex')
plot(t, exp(t), Color='black', DisplayName='Exact Solution', LineWidth=2)
leg = legend('Location', 'northeastoutside');
title(leg, 'Number of points')
ylim([0 4]); ylabel('x')
hold off

subplot(3,1,2)
hold on
plot(time1, x1', '+-', DisplayName='CasADI, n=4', MarkerSize=8);
plot(time2, x2', '+-', DisplayName='CasADI, n=8');
plot(time3, x3', '+-', DisplayName='CasADI, n=15');
plot(time4, x4', '+-', DisplayName='CasADI, n=32');
plot(time100, x100', '-.', DisplayName='CasADI, n=128', MarkerSize=14);
plot(t, exp(t), Color='black', DisplayName='Exact Solution', LineWidth=2)
xlim([0 1]); ylabel('x')

title('$x(t)$ solution by CasADI', Interpreter='latex')
leg = legend('Location', 'northeastoutside');
title(leg, 'Number of points')
hold off
subplot(3,1,3)
hold on

```

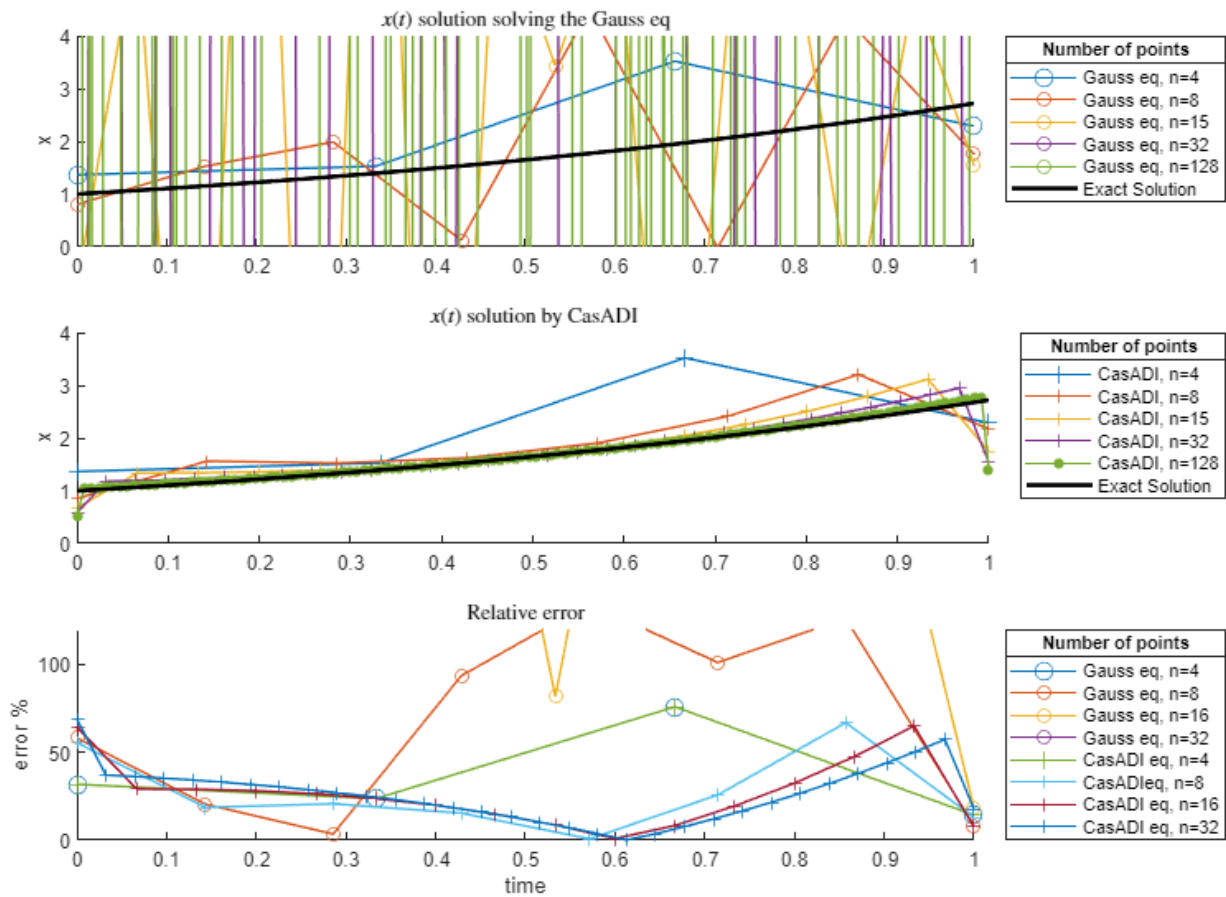
```

plot(time1', 100*abs((sol1-exp( 0:1/3:1))/exp( 0:1/3:1)),'o-', DisplayName='Gauss eq, n=4',MarkerSize=10);
plot(time2', 100*abs((sol2-exp( 0:1/7:1))/exp( 0:1/7:1)),'o-', DisplayName='Gauss eq, n=8');
plot(time3', 100*abs((sol3-exp( 0:1/15:1))/exp( 0:1/15:1)),'o-', DisplayName='Gauss eq, n=16');
plot(time4', 100*abs((sol4-exp( 0:1/31:1))/exp( 0:1/31:1)),'o-', DisplayName='Gauss eq, n=32');

plot(time1', 100*abs((x1-exp( 0:1/3:1))/exp( 0:1/3:1)),'+-', DisplayName='CasADI eq, n=4',MarkerSize=10);
plot(time2', 100*abs((x2-exp( 0:1/7:1))/exp( 0:1/7:1)),'+-', DisplayName='CasADI eq, n=8');
plot(time3', 100*abs((x3-exp( 0:1/15:1))/exp( 0:1/15:1)),'+-', DisplayName='CasADI eq, n=16');
plot(time4', 100*abs((x4-exp( 0:1/31:1))/exp( 0:1/31:1)),'+-', DisplayName='CasADI eq, n=32');

title('Relative error ',Interpreter='latex')
leg = legend('Location','northeastoutside');
title(leg,'Number of points')
ylim([0 120]); xlabel('time'); ylabel('error %')
hold off

```



```

disp(['cond n=4: ' num2str(round(co1,2)) newline 'cond n=8: ' num2str(co2) newline ...
      'cond n=16: ' num2str(co3) newline 'cond n=32: ' num2str(co4) newline 'cond n=128: ' num2str(co5) newline ...
      'cond n=256: ' num2str(co6) newline 'cond n=512: ' num2str(co7) newline 'cond n=1024: ' num2str(co8) newline ...
      'cond n=2048: ' num2str(co9) newline 'cond n=4096: ' num2str(co10) newline 'cond n=8192: ' num2str(co11) newline ...
      'cond n=16384: ' num2str(co12) newline 'cond n=32768: ' num2str(co13) newline 'cond n=65536: ' num2str(co14) newline ...
      'cond n=131072: ' num2str(co15) newline 'cond n=262144: ' num2str(co16) newline 'cond n=524288: ' num2str(co17) newline ...
      'cond n=1048576: ' num2str(co18) newline 'cond n=2097152: ' num2str(co19) newline 'cond n=4194304: ' num2str(co20) newline ...
      'cond n=8388608: ' num2str(co21) newline 'cond n=16777216: ' num2str(co22) newline 'cond n=33554432: ' num2str(co23) newline ...
      'cond n=67108864: ' num2str(co24) newline 'cond n=134217728: ' num2str(co25) newline 'cond n=268435456: ' num2str(co26) newline ...
      'cond n=536870912: ' num2str(co27) newline 'cond n=1073741824: ' num2str(co28) newline 'cond n=2147483648: ' num2str(co29) newline ...
      'cond n=4294967296: ' num2str(co30) newline 'cond n=8589934592: ' num2str(co31) newline 'cond n=17179869184: ' num2str(co32) newline ...
      'cond n=34359738368: ' num2str(co33) newline 'cond n=68719476736: ' num2str(co34) newline 'cond n=137438953472: ' num2str(co35) newline ...
      'cond n=274877906944: ' num2str(co36) newline 'cond n=549755813888: ' num2str(co37) newline 'cond n=1099511627776: ' num2str(co38) newline ...
      'cond n=2199023255552: ' num2str(co39) newline 'cond n=4398046511104: ' num2str(co40) newline 'cond n=8796093022208: ' num2str(co41) newline ...
      'cond n=17592186044416: ' num2str(co42) newline 'cond n=35184372088832: ' num2str(co43) newline 'cond n=70368744177664: ' num2str(co44) newline ...
      'cond n=140737488355328: ' num2str(co45) newline 'cond n=281474976710656: ' num2str(co46) newline 'cond n=562949953421312: ' num2str(co47) newline ...
      'cond n=1125899906842624: ' num2str(co48) newline 'cond n=2251799813685248: ' num2str(co49) newline 'cond n=4503599627370496: ' num2str(co50) newline ...
      'cond n=9007199254740992: ' num2str(co51) newline 'cond n=18014398509481984: ' num2str(co52) newline 'cond n=36028797018963968: ' num2str(co53) newline ...
      'cond n=72057594037927936: ' num2str(co54) newline 'cond n=144115188075855872: ' num2str(co55) newline 'cond n=288230376151711744: ' num2str(co56) newline ...
      'cond n=576460752303423488: ' num2str(co57) newline 'cond n=1152921504606846976: ' num2str(co58) newline 'cond n=2305843009213693952: ' num2str(co59) newline ...
      'cond n=4611686018427387904: ' num2str(co60) newline 'cond n=9223372036854775808: ' num2str(co61) newline 'cond n=18446744073709551616: ' num2str(co62) newline ...
      'cond n=36893488147419103232: ' num2str(co63) newline 'cond n=73786976294838206464: ' num2str(co64) newline 'cond n=147573952589676412928: ' num2str(co65) newline ...
      'cond n=295147905179352825856: ' num2str(co66) newline 'cond n=590295810358705651712: ' num2str(co67) newline 'cond n=1180591620717411303424: ' num2str(co68) newline ...
      'cond n=2361183241434822606848: ' num2str(co69) newline 'cond n=4722366482869645213696: ' num2str(co70) newline 'cond n=9444732965739290427392: ' num2str(co71) newline ...
      'cond n=18889465931478580854784: ' num2str(co72) newline 'cond n=37778931862957161709568: ' num2str(co73) newline 'cond n=75557863725914323419136: ' num2str(co74) newline ...
      'cond n=151115727451828646838272: ' num2str(co75) newline 'cond n=302231454903657293676544: ' num2str(co76) newline 'cond n=604462909807314587353088: ' num2str(co77) newline ...
      'cond n=1208925819614629174706176: ' num2str(co78) newline 'cond n=2417851639229258349412352: ' num2str(co79) newline 'cond n=4835703278458516698824704: ' num2str(co80) newline ...
      'cond n=9671406556917033397649408: ' num2str(co81) newline 'cond n=19342813113834066795298816: ' num2str(co82) newline 'cond n=38685626227668133590597632: ' num2str(co83) newline ...
      'cond n=77371252455336267181195264: ' num2str(co84) newline 'cond n=154742504910672534362390528: ' num2str(co85) newline 'cond n=309485009821345068724781056: ' num2str(co86) newline ...
      'cond n=618970019642690137449562112: ' num2str(co87) newline 'cond n=1237940039285380274899124224: ' num2str(co88) newline 'cond n=2475880078570760549798248448: ' num2str(co89) newline ...
      'cond n=4951760157141521099596496896: ' num2str(co90) newline 'cond n=9903520314283042199192993792: ' num2str(co91) newline 'cond n=19807040628566084398385987584: ' num2str(co92) newline ...
      'cond n=39614081257132168796771975168: ' num2str(co93) newline 'cond n=79228162514264337593543950336: ' num2str(co94) newline 'cond n=158456325028528675187087900672: ' num2str(co95) newline ...
      'cond n=316912650057057350374175801344: ' num2str(co96) newline 'cond n=633825300114114700748351602688: ' num2str(co97) newline 'cond n=1267650600228229401496703205376: ' num2str(co98) newline ...
      'cond n=2535301200456458802993406410752: ' num2str(co99) newline 'cond n=5070602400912917605986812821504: ' num2str(co100) newline 'cond n=10141204801825835211973625643008: ' num2str(co101) newline ...
      'cond n=20282409603651670423947251286016: ' num2str(co102) newline 'cond n=40564819207303340847894502572032: ' num2str(co103) newline 'cond n=81129638414606681695789005144064: ' num2str(co104) newline ...
      'cond n=162259276829213363391578010288128: ' num2str(co105) newline 'cond n=324518553658426726783156020576256: ' num2str(co106) newline 'cond n=649037107316853453566312041152512: ' num2str(co107) newline ...
      'cond n=1298074214633706907132624082305024: ' num2str(co108) newline 'cond n=2596148429267413814265248164610048: ' num2str(co109) newline 'cond n=5192296858534827628530496329220096: ' num2str(co110) newline ...
      'cond n=10384593717069655257060992658440192: ' num2str(co111) newline 'cond n=20769187434139310514121985316880384: ' num2str(co112) newline 'cond n=41538374868278621028243970633760768: ' num2str(co113) newline ...
      'cond n=83076749736557242056487941267521536: ' num2str(co114) newline 'cond n=166153499473114484112975882535043072: ' num2str(co115) newline 'cond n=332306998946228968225951765070086144: ' num2str(co116) newline ...
      'cond n=664613997892457936451903530140172288: ' num2str(co117) newline 'cond n=1329227995784915872903807060280344576: ' num2str(co118) newline 'cond n=2658455991569831745807614120560689152: ' num2str(co119) newline ...
      'cond n=5316911983139663491615228241121378304: ' num2str(co120) newline 'cond n=10633823966279326983230456482242756608: ' num2str(co121) newline 'cond n=21267647932558653966460912964485513216: ' num2str(co122) newline ...
      'cond n=42535295865117307932921825928971026432: ' num2str(co123) newline 'cond n=85070591730234615865843651857942052864: ' num2str(co124) newline 'cond n=170141183460469231731687303715884105728: ' num2str(co125) newline ...
      'cond n=340282366920938463463374607431768211456: ' num2str(co126) newline 'cond n=680564733841876926926749214863536422912: ' num2str(co127) newline 'cond n=1361129467683753853853498429727072845824: ' num2str(co128) newline ...
      'cond n=2722258935367507707706996859454145691648: ' num2str(co129) newline 'cond n=5444517870735015415413993718908291383296: ' num2str(co130) newline 'cond n=10889035741470030830827987437816582766592: ' num2str(co131) newline ...
      'cond n=21778071482940061661655974875633165533184: ' num2str(co132) newline 'cond n=43556142965880123323311949751266331066368: ' num2str(co133) newline 'cond n=87112285931760246646623899502532662132736: ' num2str(co134) newline ...
      'cond n=174224571863520493293247799005065324265472: ' num2str(co135) newline 'cond n=348449143727040986586495598010130648530944: ' num2str(co136) newline 'cond n=696898287454081973172991196020261297061888: ' num2str(co137) newline ...
      'cond n=1393796574908163946345982392040522594123776: ' num2str(co138) newline 'cond n=2787593149816327892691964784081045188247552: ' num2str(co139) newline 'cond n=5575186299632655785383929568162090376495104: ' num2str(co140) newline ...
      'cond n=11150372599265311570767859136324180752990208: ' num2str(co141) newline 'cond n=22300745198530623141535718272648361505980416: ' num2str(co142) newline 'cond n=44601490397061246283071436545296723011960832: ' num2str(co143) newline ...
      'cond n=89202980794122492566142873090593446023921664: ' num2str(co144) newline 'cond n=178405961588244985132285746181186892047843328: ' num2str(co145) newline 'cond n=356811923176489970264571492362373784095686656: ' num2str(co146) newline ...
      'cond n=713623846352979940529142984724747568191373312: ' num2str(co147) newline 'cond n=1427247692705959881058285969449495136382746624: ' num2str(co148) newline 'cond n=2854495385411919762116571938898990272765493248: ' num2str(co149) newline ...
      'cond n=5708990770823839524233143877797980545530986496: ' num2str(co150) newline 'cond n=11417981541647679048466287755595961091061972992: ' num2str(co151) newline 'cond n=22835963083295358096932575511191922182123945984: ' num2str(co152) newline ...
      'cond n=45671926166590716193865151022383844364247891968: ' num2str(co153) newline 'cond n=91343852333181432387730302044767688728495783936: ' num2str(co154) newline 'cond n=182687704666362864775460604089535377456991567872: ' num2str(co155) newline ...
      'cond n=365375409332725729550921208179070754913983135744: ' num2str(co156) newline 'cond n=730750818665451459101842416358141509827966271488: ' num2str(co157) newline 'cond n=1461501637330902918203684832716283019655932542976: ' num2str(co158) newline ...
      'cond n=2923003274661805836407369665432566039311865085952: ' num2str(co159) newline 'cond n=5846006549323611672814739330865132078623730171904: ' num2str(co160) newline 'cond n=11692013098647223345629478661730264157247460343808: ' num2str(co161) newline ...
      'cond n=23384026197294446691258957323460528314494920687616: ' num2str(co162) newline 'cond n=46768052394588893382517914646921056628989841375232: ' num2str(co163) newline 'cond n=93536104789177786765035829293842113257979682750464: ' num2str(co164) newline ...
      'cond n=187072209578355573530071658587684226515959365500928: ' num2str(co165) newline 'cond n=374144419156711147060143317175368453031918731001856: ' num2str(co166) newline 'cond n=748288838313422294120286634350736906063837462003712: ' num2str(co167) newline ...
      'cond n=1496577676626844588240573268701473812127674924007424: ' num2str(co168) newline 'cond n=2993155353253689176481146537402947624255349848014848: ' num2str(co169) newline 'cond n=5986310706507378352962293074805895248510699696029696: ' num2str(co170) newline ...
      'cond n=11972621413014756705924586149611790497021399392059392: ' num2str(co171) newline 'cond n=23945242826029513411849172299223580994042798784118784: ' num2str(co172) newline 'cond n=47890485652059026823698344598447161988085597568237568: ' num2str(co173) newline ...
      'cond n=95780971304118053647396689196894323976171195136475136: ' num2str(co174) newline 'cond n=191561942608236107294793378393788647952342390272950272: ' num2str(co175) newline 'cond n=383123885216472214589586756787577295904684780545900544: ' num2str(co176) newline ...
      'cond n=766247770432944429179173513575154591809369561091801088: ' num2str(co177) newline 'cond n=1532495540865888858358347027150309183618739122183602176: ' num2str(co178) newline 'cond n=3064991081731777716716694054300618367237478244367204352: ' num2str(co179) newline ...
      'cond n=6129982163463555433433388108601236734474956488734408704: ' num2str(co180) newline 'cond n=12259964326927110866866776217202473468949912977468817408: ' num2str(co181) newline 'cond n=24519928653854221733733552434404946937899825954937634816: ' num2str(co182) newline ...
      'cond n=49039857307708443467467104868809893875799651909875269632: ' num2str(co183) newline 'cond n=98079714615416886934934209737619787751599303819750539264: ' num2str(co184) newline 'cond n=196159429230833773869868419475239575503198607639501078528: ' num2str(co185) newline ...
      'cond n=392318858461667547739736838950479151006397215279002157056: ' num2str(co186) newline 'cond n=784637716923335095479473677900958302012794430558004314112: ' num2str(co187) newline 'cond n=1569275433846670190958947355801916604025588861116008628224: ' num2str(co188) newline ...
      'cond n=3138550867693340381917894711603833208051177722232017256448: ' num2str(co189) newline 'cond n=6277101735386680763835789423207666416102355444464034512896: ' num2str(co190) newline 'cond n=12554203470773361527671578846415332832204710888928069025792: ' num2str(co191) newline ...
      'cond n=25108406941546723055343157692830665664409421777856138051584: ' num2str(co192) newline 'cond n=50216813883093446110686315385661331328818843555712276103168: ' num2str(co193) newline 'cond n=100433627766186892221372630771322662657637687111424552206336: ' num2str(co194) newline ...
      'cond n=200867255532373784442745261542645325315275374222849104412672: ' num2str(co195) newline 'cond n=401734511064747568885490523085290650630550748445698208825344: ' num2str(co196) newline 'cond n=803469022129495137770981046170581301261101496891396417650688: ' num2str(co197) newline ...
      'cond n=1606938044258990275541962092341162602522202993782792835301376: ' num2str(co198) newline 'cond n=3213876088517980551083924184682325205044405987565585670602752: ' num2str(co199) newline 'cond n=6427752177035961102167848369364650410088811975131171341205504: ' num2str(co200) newline ...
      'cond n=12855504354071922204335696738729300820177623950262342682411008: ' num2str(co201) newline 'cond n=25711008708143844408671393477458601640355247900524685364822016: ' num2str(co202) newline 'cond n=51422017416287688817342786954917203280710495801049370729644032: ' num2str(co203) newline ...
      'cond n=102844034832575377634685573909834406561420991602098741459288064: ' num2str(co204) newline 'cond n=205688069665150755269371147819668813122841983204197482918576128: ' num2str(co205) newline 'cond n=411376139330301510538742295639337626245683966408394965837152256: ' num2str(co206) newline ...
      'cond n=822752278660603021077484591278675252491367932816789931674304512: ' num2str(co207) newline 'cond n=1645504557321206042154969182557350504982735865633579863348609024: ' num2str(co208) newline 'cond n=3291009114642412084309938365114701009965471731267159726697218048: ' num2str(co209) newline ...
      'cond n=6582018229284824168619876730229402019930943462534319453394436096: ' num2str(co210) newline 'cond n=13164036458569648337239753460458804039861886925068638906788872192: ' num2str(co211) newline 'cond n=26328072917139296674479506920917608079723773850137277813577744384: ' num2str(co212) newline ...
      'cond n=52656145834278593348959013841835216159447547700274555627155488768: ' num2str(co213) newline 'cond n=105312291668557186697918027683670432318895095400549111254310977536: ' num2str(co214) newline 'cond n=210624583337114373395836055367340864637790190801098222508621955072: ' num2str(co215) newline ...
      'cond n=421249166674228746791672110734681729275580381602196445017243910144: ' num2str(co216) newline 'cond n=842498333348457493583344221469363458551160763204392890034487820288: ' num2str(co217) newline 'cond n=1684996666696914987166688442938726917102321526408785780068975640576: ' num2str(co218) newline ...
      'cond n=3369993333393829974333376885877453834204643052817571560137951281152: ' num2str(co219) newline 'cond n=6739986666787659948666753771754907668409286105635143120275902562304: ' num2str(co220) newline 'cond n=13479973333575319897333507543509815336818572211270286240551805124608: ' num2str(co221) newline ...
      'cond n=26959946667150639794667015087019630673637144422540572481103610249216: ' num2str(co222) newline 'cond n=53919893334301279589334030174039261347274288845081144962207220498432: ' num2str(co223) newline 'cond n=107839786668602559178668060348078522694548577690162289924414440996864: ' num2str(co224) newline ...
      'cond n=215679573337205118357336120696157045389097155380324579848828881993728: ' num2str(co225) newline 'cond n=431359146674410236714672241392314090778194310760649159697657763987456: ' num2str(co226) newline 'cond n=86271829334882
```

Note that the first and last point are atypical data, this can be solved using the following matrix

$$A_{(n \times n)} = \frac{1}{n} \begin{pmatrix} e^{ht_1} & e^{ht_2} & e^{ht_3} & \dots & e^{ht_{n-1}} & e^{ht_n} \\ e^{2ht_1} & e^{2ht_2} & e^{2ht_3} & \dots & e^{2ht_{n-1}} & e^{2ht_n} \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ e^{(n-1)ht_1} & e^{(n-1)ht_2} & e^{(n-1)ht_3} & \dots & e^{(n-1)ht_{n-1}} & e^{(n-1)ht_n} \\ e^{nht_1} & e^{nht_2} & e^{nht_3} & \dots & e^{nht_{n-1}} & e^{nht_n} \end{pmatrix}$$

It is also important to mention that the result given by solving the Gauss equations is counterintuitive, when the number of points n increase the function starts to behave chaotically, the solution starts to oscillate drastically, then the best solution in this case is when there are a few number of points.

In contrast, the solution with CasADI is different, the solution improves when n increases then the best solution for this point is use CasADI library with high number of points. Furthermore, the relative error is plotted for both solution methods, here the result shows that CasADI has lower relative errors than by solving Gauss eq.

d) Add Regularization

Solve the integral equation using the regularization method for each of the following parameters: i)

$n = 100$, $\alpha = 0.2$, ii) $n = 100$, $\alpha = 0.1$, and iii) $n = 100$, $\alpha = 10^{-3}$. Plot the solutions.

Solution by Gauss eq

In this point, a regularization problem is added. The problem solution is found by solving the gauss equations

$$x = (A + \alpha I)^{-1}b = (A + \alpha I) \setminus b$$

```
[time_0, sol_0, co_0, M_0, b_0] = System(100, 0);
```

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND = 6.209026e-21.

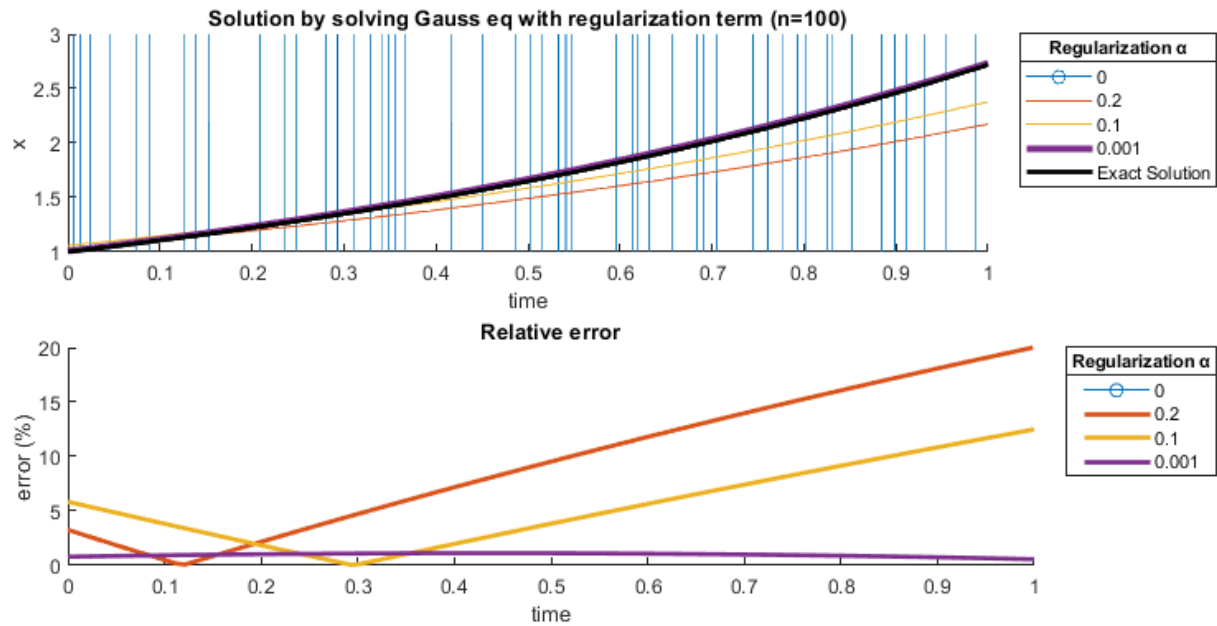
```
[time_i, sol_i, co_i, M_i, b_i] = System(100, 0.2);
[time_ii, sol_ii, co_ii, M_ii, b_ii] = System(100, 0.1);
[time_iii, sol_iii, co_iii, M_iii, b_iii] = System(100, 0.001);

clf;
figure(Position=[10 10 900 400])
subplot(2,1,1)
title('Solution by solving Gauss eq with regularization term (n=100)')
hold on
plot(time_0, sol_0, '-o', DisplayName='0')
plot(time_i, sol_i, DisplayName='0.2')
plot(time_ii, sol_ii, DisplayName='0.1')
plot(time_iii, sol_iii, DisplayName='0.001', LineWidth=3)
plot(t, exp(t), Color='black', DisplayName='Exact Solution', LineWidth=2)
leg = legend('Location', 'northeastoutside');
title(leg, 'Regularization  $\alpha$ ')
xlabel('time'); ylabel('x'); ylim([1 3])
hold off
```

```

subplot(2,1,2)
title('Relative error')
hold on
plot(time_0, 100*abs((sol_0-exp(time_0)')./exp(time_0)'), '-o', DisplayName='0')
plot(time_i, 100*abs((sol_i-exp(time_i)')./exp(time_i)'), '-', DisplayName='0.2', LineWidth=2)
plot(time_ii, 100*abs((sol_ii-exp(time_ii)')./exp(time_ii)'), '-', DisplayName='0.1', LineWidth=2)
plot(time_iii, 100*abs((sol_iii-exp(time_iii)')./exp(time_iii)'), '-', DisplayName='0.001', LineWidth=2)
leg = legend('Location', 'northeastoutside');
title(leg, 'Regularization  $\alpha$ ')
xlabel('time'); ylabel('error (%)'); ylim([0 20])
hold off

```



```

disp(['The higher relative error for  $\alpha=0.2$  is ' num2str(max(100*abs((sol_i-exp(time_i)')./exp(time_i))))
      'The higher relative error for  $\alpha=0.1$  is ' num2str(max(100*abs((sol_ii-exp(time_ii)')./exp(time_ii))))
      'The higher relative error for  $\alpha=0.001$  is ' num2str(max(100*abs((sol_iii-exp(time_iii)')./exp(time_iii))))

```

```

The higher relative error for  $\alpha=0.2$  is 20.0283
The higher relative error for  $\alpha=0.1$  is 12.4936
The higher relative error for  $\alpha=0.001$  is 1.1288

```

```

disp(['cond n=100,  $\alpha=0$ : ' num2str(round(co_0,2)) newline 'cond n=100,  $\alpha=0.2$ : ' num2str(co_i)
      'cond n=100,  $\alpha=0.01$ : ' num2str(co_ii) newline 'cond n=100,  $\alpha=0.001$ : ' num2str(co_iii)

```

```

cond n=100,  $\alpha=0$ : 1.855424207205099e+19
cond n=100,  $\alpha=0.2$ : 7.7493
cond n=100,  $\alpha=0.01$ : 14.5093
cond n=100,  $\alpha=0.001$ : 1361.5892

```

In order to fix the condition of the matrix A a regularization term is added. The result shows that as the regularization term decreases the solution improves, without the regularization term the solution is still unstable, and the best solution is when $\alpha = 0.001$ because the relative error is always smaller than 1.5 %.

Solution by CasADI

Now let's solve the problem, least squares, but modifying the matrix with a regularization parameter, $A = A + \alpha I$

```
x_casadi_0 = SolCasADI(100, M_0, b_0);
```

This is Ipopt version 3.12.3, running with linear solver mumps.

NOTE: Other linear solvers might be more efficient (see Ipopt documentation).

```
Number of nonzeros in equality constraint Jacobian...:      0
Number of nonzeros in inequality constraint Jacobian.:      0
Number of nonzeros in Lagrangian Hessian.....:      5050
```

```
Total number of variables.....:      100
      variables with only lower bounds:      0
      variables with lower and upper bounds:      0
      variables with only upper bounds:      0
Total number of equality constraints.....:      0
Total number of inequality constraints.....:      0
      inequality constraints with only lower bounds:      0
      inequality constraints with lower and upper bounds:      0
      inequality constraints with only upper bounds:      0
```

iter	objective	inf_pr	inf_du	lg(mu)	d	lg(rg)	alpha_du	alpha_pr	ls
0	5.8138663e+002	0.00e+000	8.56e+000	-1.0	0.00e+000	-	0.00e+000	0.00e+000	0
1	2.0533839e-006	0.00e+000	2.74e-004	-1.0	2.74e+000	-4.0	1.00e+000	1.00e+000f	1
2	2.2727703e-007	0.00e+000	8.03e-007	-5.7	2.41e-002	-4.5	1.00e+000	1.00e+000f	1
3	2.2075292e-008	0.00e+000	2.32e-007	-8.6	2.09e-002	-5.0	1.00e+000	1.00e+000f	1
4	4.6139277e-010	0.00e+000	3.07e-008	-8.6	8.28e-003	-5.4	1.00e+000	1.00e+000f	1
5	8.4276125e-011	0.00e+000	1.79e-009	-8.6	1.45e-003	-5.9	1.00e+000	1.00e+000f	1

Number of Iterations.....: 5

	(scaled)	(unscaled)
Objective.....	8.4276125324613342e-011	8.4276125324613342e-011
Dual infeasibility.....	1.7862071497248228e-009	1.7862071497248228e-009
Constraint violation....	0.0000000000000000e+000	0.0000000000000000e+000
Complementarity.....	0.0000000000000000e+000	0.0000000000000000e+000
Overall NLP error.....	1.7862071497248228e-009	1.7862071497248228e-009

```
Number of objective function evaluations      = 6
Number of objective gradient evaluations      = 6
Number of equality constraint evaluations      = 0
Number of inequality constraint evaluations    = 0
Number of equality constraint Jacobian evaluations = 0
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations      = 5
Total CPU secs in IPOPT (w/o function evaluations) =      1.679
Total CPU secs in NLP function evaluations      =      0.007
```

EXIT: Optimal Solution Found.

solver	t_proc	(avg)	t_wall	(avg)	n_eval
nlp_f	0 (0)	0 (0)	6
nlp_grad_f	0 (0)	0 (0)	7
nlp_hess_l	7.00ms (1.40ms)	7.00ms (1.40ms)	5
total	1.71 s (1.71 s)	1.71 s (1.71 s)	1

```
x_casadi_i = SolCasADI(100, M_i, b_i);
```

This is Ipopt version 3.12.3, running with linear solver mumps.

NOTE: Other linear solvers might be more efficient (see Ipopt documentation).

```
Number of nonzeros in equality constraint Jacobian...:      0
Number of nonzeros in inequality constraint Jacobian.:      0
```

```

Number of nonzeros in Lagrangian Hessian.....: 5050

Total number of variables.....: 100
    variables with only lower bounds: 0
    variables with lower and upper bounds: 0
    variables with only upper bounds: 0
Total number of equality constraints.....: 0
Total number of inequality constraints.....: 0
    inequality constraints with only lower bounds: 0
    inequality constraints with lower and upper bounds: 0
    inequality constraints with only upper bounds: 0

iter   objective   inf_pr   inf_du lg(mu)  ||d|| lg(rg) alpha_du alpha_pr ls
  0 5.8138663e+002 0.00e+000 9.83e+000 -1.0 0.00e+000 - 0.00e+000 0.00e+000 0
  1 3.1715660e-027 0.00e+000 6.58e-015 -1.0 2.17e+000 - 1.00e+000 1.00e+000f 1

```

Number of Iterations.....: 1

	(scaled)	(unscaled)
Objective.....:	3.1715659656345017e-027	3.1715659656345017e-027
Dual infeasibility.....:	6.5825710206764181e-015	6.5825710206764181e-015
Constraint violation.....:	0.0000000000000000e+000	0.0000000000000000e+000
Complementarity.....:	0.0000000000000000e+000	0.0000000000000000e+000
Overall NLP error.....:	6.5825710206764181e-015	6.5825710206764181e-015

```

Number of objective function evaluations      = 2
Number of objective gradient evaluations      = 2
Number of equality constraint evaluations      = 0
Number of inequality constraint evaluations    = 0
Number of equality constraint Jacobian evaluations = 0
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations      = 1
Total CPU secs in IPOPT (w/o function evaluations) = 0.031
Total CPU secs in NLP function evaluations      = 0.001

```

EXIT: Optimal Solution Found.

solver	:	t_proc	(avg)	t_wall	(avg)	n_eval
nlp_f		0 (0)	0 (0)	2
nlp_grad_f		0 (0)	0 (0)	3
nlp_hess_l		1.00ms (1.00ms)	1.00ms (1.00ms)	1
total		54.00ms (54.00ms)	54.01ms (54.01ms)	1

```
x_casadi_ii = SolCasADI(100, M_ii, b_ii);
```

This is Ipopt version 3.12.3, running with linear solver mumps.
NOTE: Other linear solvers might be more efficient (see Ipopt documentation).

```

Number of nonzeros in equality constraint Jacobian...: 0
Number of nonzeros in inequality constraint Jacobian.: 0
Number of nonzeros in Lagrangian Hessian.....: 5050

Total number of variables.....: 100
    variables with only lower bounds: 0
    variables with lower and upper bounds: 0
    variables with only upper bounds: 0
Total number of equality constraints.....: 0
Total number of inequality constraints.....: 0
    inequality constraints with only lower bounds: 0
    inequality constraints with lower and upper bounds: 0
    inequality constraints with only upper bounds: 0

iter   objective   inf_pr   inf_du lg(mu)  ||d|| lg(rg) alpha_du alpha_pr ls

```

```

0 5.8138663e+002 0.00e+000 9.20e+000 -1.0 0.00e+000 - 0.00e+000 0.00e+000 0
1 1.0402265e-026 0.00e+000 5.39e-015 -1.0 2.38e+000 - 1.00e+000 1.00e+000f 1

```

Number of Iterations....: 1

	(scaled)	(unscaled)
Objective.....	1.0402265022890296e-026	1.0402265022890296e-026
Dual infeasibility.....	5.3862704223432385e-015	5.3862704223432385e-015
Constraint violation....	0.0000000000000000e+000	0.0000000000000000e+000
Complementarity.....	0.0000000000000000e+000	0.0000000000000000e+000
Overall NLP error.....	5.3862704223432385e-015	5.3862704223432385e-015

```

Number of objective function evaluations      = 2
Number of objective gradient evaluations     = 2
Number of equality constraint evaluations     = 0
Number of inequality constraint evaluations   = 0
Number of equality constraint Jacobian evaluations = 0
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations    = 1
Total CPU secs in IPOPT (w/o function evaluations) = 0.025
Total CPU secs in NLP function evaluations    = 0.001

```

EXIT: Optimal Solution Found.

solver	t_proc	(avg)	t_wall	(avg)	n_eval
nlp_f	0	(0)	0	(0)	2
nlp_grad_f	0	(0)	0	(0)	3
nlp_hess_l	1.00ms	(1.00ms)	1.00ms	(1.00ms)	1
total	47.00ms	(47.00ms)	47.01ms	(47.01ms)	1

```
x_casadi_iii = SolCasADI(100, M_iii, b_iii);
```

This is Ipopt version 3.12.3, running with linear solver mumps.

NOTE: Other linear solvers might be more efficient (see Ipopt documentation).

```

Number of nonzeros in equality constraint Jacobian...: 0
Number of nonzeros in inequality constraint Jacobian.: 0
Number of nonzeros in Lagrangian Hessian.....: 5050

```

```

Total number of variables.....: 100
    variables with only lower bounds: 0
    variables with lower and upper bounds: 0
    variables with only upper bounds: 0
Total number of equality constraints.....: 0
Total number of inequality constraints.....: 0
    inequality constraints with only lower bounds: 0
    inequality constraints with lower and upper bounds: 0
    inequality constraints with only upper bounds: 0

```

```

iter   objective   inf_pr   inf_du lg(mu)  ||d|| lg(rg) alpha_du alpha_pr ls
0 5.8138663e+002 0.00e+000 8.57e+000 -1.0 0.00e+000 - 0.00e+000 0.00e+000 0
1 9.4746989e-023 0.00e+000 6.75e-015 -1.0 2.73e+000 - 1.00e+000 1.00e+000f 1

```

Number of Iterations....: 1

	(scaled)	(unscaled)
Objective.....	9.4746989019194999e-023	9.4746989019194999e-023
Dual infeasibility.....	6.7519697818798492e-015	6.7519697818798492e-015
Constraint violation....	0.0000000000000000e+000	0.0000000000000000e+000
Complementarity.....	0.0000000000000000e+000	0.0000000000000000e+000
Overall NLP error.....	6.7519697818798492e-015	6.7519697818798492e-015


```

Number of objective function evaluations      = 2
Number of objective gradient evaluations     = 2
Number of equality constraint evaluations     = 0
Number of inequality constraint evaluations  = 0
Number of equality constraint Jacobian evaluations = 0
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations    = 1
Total CPU secs in IPOPT (w/o function evaluations) = 0.035
Total CPU secs in NLP function evaluations   = 0.001

```

EXIT: Optimal Solution Found.

solver	:	t_proc	(avg)	t_wall	(avg)	n_eval
nlp_f		0	(0)	0	(0)	2
nlp_grad_f		0	(0)	0	(0)	3
nlp_hess_l		1.00ms	(1.00ms)	1.00ms	(1.00ms)	1
total		38.00ms	(38.00ms)	37.40ms	(37.40ms)	1

```
clf;
```

```
figure(Position=[10 10 900 400])
```

```
subplot(211)
```

```
hold on
```

```
plot(time_0, x_casadi_0, '+', DisplayName='0', LineWidth=2)
```

```
plot(time_iii, x_casadi_iii, DisplayName='0.001', LineWidth=5)
```

```
plot(time_ii, x_casadi_ii, DisplayName='0.1', LineWidth=2)
```

```
plot(time_i, x_casadi_i, '-o', DisplayName='0.2')
```

```
plot(time_i, exp(time_i), DisplayName='Exact Solution')
```

```
leg = legend('Location', 'northeastoutside');
```

```
title(leg, 'Regularization  $\alpha$ ')
```

```
ylabel('x'); xlabel('time'); title('Solution with CasADI')
```

```
hold off
```

```
subplot(212)
```

```
hold on
```

```
plot(time_0(2:end-1), 100*abs((x_casadi_0(2:end-1)-exp(time_0(2:end-1))) ./ exp(time_0(2:end-1))))
```

```
plot(time_iii(2:end-1), 100*abs((x_casadi_iii(2:end-1)-exp(time_iii(2:end-1))) ./ exp(time_iii(2:end-1))))
```

```
plot(time_ii(2:end-1), 100*abs((x_casadi_ii(2:end-1)-exp(time_ii(2:end-1))) ./ exp(time_ii(2:end-1))))
```

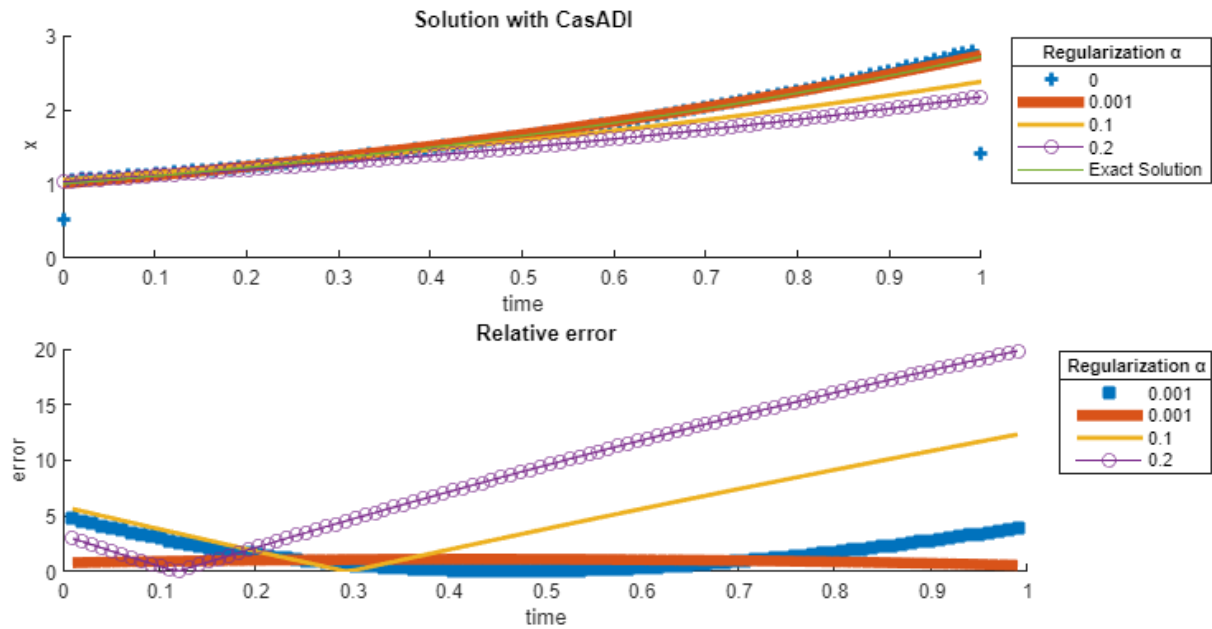
```
plot(time_i(2:end-1), 100*abs((x_casadi_i(2:end-1)-exp(time_i(2:end-1))) ./ exp(time_i(2:end-1))))
```

```
leg = legend('Location', 'northeastoutside');
```

```
title(leg, 'Regularization  $\alpha$ ')
```

```
ylabel('error'); xlabel('time'); title('Relative error')
```

```
hold off
```



```
disp(['Maximum relative error ' num2str(max(100*abs((x_casadi_i(2:end-1)-exp(time_i(2:end-1)))'))'])
```

Maximum relative error 19.8358

The relative error is plotted without the first and last point since CasADI calculate these points wrong, the value getted is the half of the real value and can be fixed by multiplying by 2 the first and last columns of the matrix A . Note that the solutions are the same since CasADI solve the problem correctly for 100 points, then the regularization term is not necessary when CasADI is used.

Auxiliar Functions

```
function y=yfunc(s)
    y = (exp(s+1)-1)./(s+1);
end

function [time, sol, co, M, b]=System(n, alfa)
    time = 0:1/(n-1):1;
    M = zeros(n,n);
    for i=1:n
        s = i*(1/n);

        v = exp(s*time);
        v(2:end-1) = 2*v(2:end-1);

        M(i,:)= (1/(2*n))*v;
    end
    b = yfunc((1:n)*(1/n))';
    M = M+alfa*eye(size(M));
    sol = M \ b;
    co = cond(M);
```

```
end

function x_int_opt=SolCasADI(n, M, b)
    opti_int = casadi.Opti();
    x_int = opti_int.variable(n);

    opti_int.minimize(norm(M*x_int-b,2)^2);

    opti_int.solver('ipopt');

    sol_int = opti_int.solve();
    x_int_opt = sol_int.value(x_int);
end
```