Homework 3

Sebastian Aguilera Novoa

Table of Contents

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 $Q \succ 0$, a vector of measures $\eta \in \mathbb{R}^m$ and a point $\bar{x} \in \mathbb{R}$, calculate the limite

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

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

$$f(x) = \frac{1}{2}||\eta - Jx||_2^2 + \frac{\alpha}{2}(x-\bar{x})^T Q(x-\bar{x}) = \frac{1}{2}(\eta - Jx)^T(\eta - Jx) + \frac{\alpha}{2}(x-\bar{x})^T Q(x-\bar{x})$$

$$= \frac{1}{2}(\eta^T\eta - x^T J^T\eta - \eta^T Jx + x^T J^T Jx) + \frac{\alpha}{2}(x-\bar{x})^T Q(x-\bar{x})$$

$$= \frac{1}{2}(\eta^T\eta - 2J^T\eta x + x^T J^T Jx) + \frac{\alpha}{2}(x-\bar{x})^T Q(x-\bar{x})$$

$$\nabla f(x) = -J^T\eta + J^T Jx + \alpha Q(x-\bar{x})$$

$$\nabla^2 f(x) = J^T J + \alpha Q \succ 0$$

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

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

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

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

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

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

calculating the inverse

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

writing in matricial form and defining $W = V^TU', W \in \mathbb{R}^{n\times n}$

$$
(J^TJ+\alpha Q)^{-1} = V\begin{pmatrix}
\sigma_1^2 + \alpha W_{1,1}\sigma'_1 & 0 & \cdots & & & 0\\
0 & \sigma_2^2 + \alpha W_{2,2}\sigma'_2 & 0 & \cdots & \cdots & 0\\
\vdots & \cdots & \ddots & \cdots & \cdots & \vdots\\
0 & \cdots & \cdots & \sigma_r^2 + \alpha W_{r,r}\sigma'_r & \cdots & 0\\
\vdots & \vdots & \vdots & \vdots & \vdots & \\
0 & 0 & \cdots & \cdots & \alpha W_{n-1,n-1}\sigma'_{n-1} & 0\\
0 & 0 & \cdots & \cdots & \cdots & \alpha W_{n,n}\sigma'_n
\end{pmatrix}^{-1} V^T
$$

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

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

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

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

calculating each matrix by separeted

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

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

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

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

$$
V \begin{pmatrix}
\frac{1}{\sigma_1^2+\alpha W_{1,1}\sigma_1^{'}} & 0 & \cdots & & & 0 \\
0 & \frac{1}{\sigma_2^2+\alpha W_{2,2}\sigma_2^{'}} & 0 & \cdots & \cdots & 0 \\
\vdots & \cdots & \ddots & \cdots & \cdots & \vdots \\
0 & \cdots & \cdots & \frac{1}{\sigma_r^2+\alpha W_{r,r}\sigma_r^{'}} & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \\
0 & 0 & \cdots & \cdots & \frac{1}{\alpha W_{n-1,n-1}\sigma_{n-1}^{'}} & 0 \\
0 & 0 & \cdots & \cdots & \cdots & \frac{1}{\alpha W_{n,n}\sigma_n^{'}}
\end{pmatrix}
\begin{pmatrix}
\alpha\bar{x}W_{1,1}\sigma_1^{'} & 0 & \cdots & & & & \\
0 & \alpha\bar{x}W_{2,2}\sigma_2^{'} & 0 & \cdots & & & \\
 & & \ddots & \vdots & \cdots & & \\
 & & \cdots & \alpha\bar{x}W_{r,r}\sigma_r^{'} & \cdots & & \\
 & & & \vdots & 0 & \cdots & \\
 & & & & & \ddots & \vdots \\
 & & & & & \cdots & 0
\end{pmatrix} V^T
$$

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

(7)

Now let's calculate the limite of both matrixes

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

$$
= \lim_{\alpha\to 0^+} V \begin{pmatrix}
\frac{\sigma_1}{\sigma_1^2+\alpha W_{1,1}\sigma_1^{'}} & 0 & \cdots & & & 0 \\
0 & \frac{\sigma_2}{\sigma_2^2+\alpha W_{2,2}\sigma_2^{'}} & 0 & \cdots & \cdots & 0 \\
\vdots & \cdots & \ddots & \cdots & \cdots & \vdots \\
0 & \cdots & \cdots & \frac{\sigma_r}{\sigma_r^2+\alpha W_{r,r}\sigma_r^{'}} & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \\
0 & 0 & \cdots & \cdots & 0 & 0 \\
0 & 0 & \cdots & \cdots & \cdots & 0
\end{pmatrix} V^T = V \begin{pmatrix}
\sigma_1^{-1} & 0 & \cdots & & & 0 \\
0 & \sigma_2^{-1} & 0 & \cdots & \cdots & 0 \\
\vdots & \cdots & \ddots & \cdots & \cdots & \vdots \\
0 & \cdots & \cdots & \sigma_r^{-1} & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \\
0 & 0 & \cdots & \cdots & 0 & 0 \\
0 & 0 & \cdots & \cdots & \cdots & 0
\end{pmatrix} V^T = V\Sigma^+V^T
$$

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

$$= \lim_{\alpha\to 0^+} V \begin{pmatrix} \alpha\dfrac{\bar{x}W_{1,1}\sigma_1'}{\sigma_1^2+\alpha W_{1,1}\sigma_1'} & 0 & \cdots & & & 0 \\ 0 & \alpha\dfrac{\bar{x}W_{2,2}\sigma_2'}{\sigma_2^2+\alpha W_{2,2}\sigma_2'} & 0 & \cdots & \cdots & 0 \\ \vdots & \cdots & \ddots & \cdots & \cdots & \vdots \\ 0 & \cdots & \cdots & \alpha\dfrac{\bar{x}W_{r,r}\sigma_r'}{\sigma_r^2+\alpha W_{r,r}\sigma_r'} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix} V^T = V\,\bar{\bar{0}}\,V^T$$

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

Then, the limit is

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

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

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

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

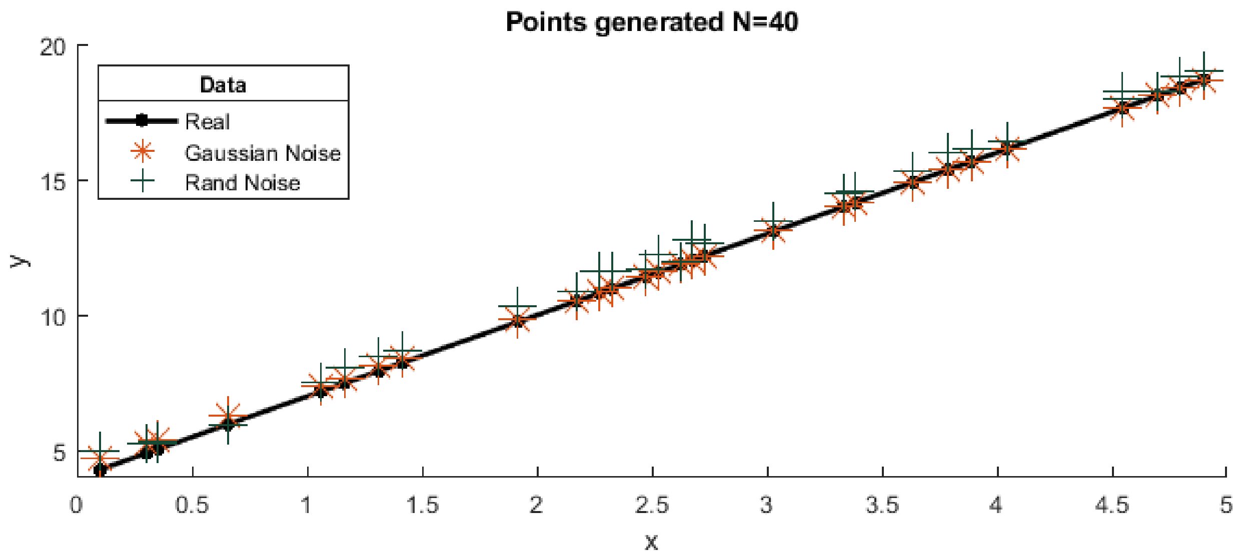
Like was discussed in class, the optimal solution to the previous problem can be calculated explicity by solving the normal Gauss equations

$$J^TJ\begin{pmatrix} a \\ b \end{pmatrix} = J^Ty$$

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

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

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

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

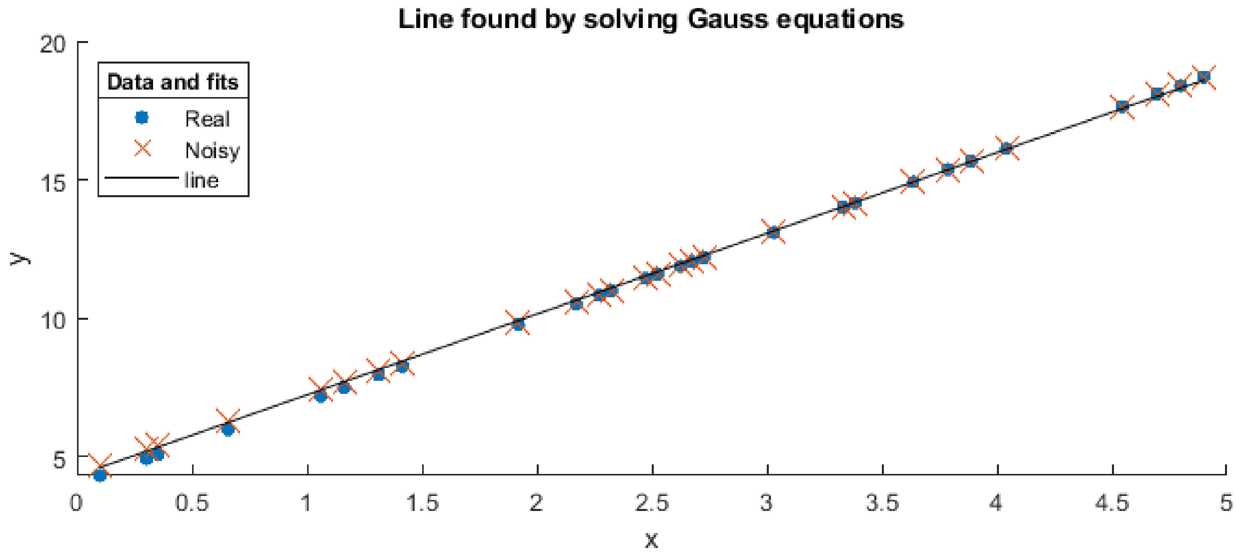
```
J = [xrand  ones(N,1)];
ab = J \ ynoise;             % A \ b %Ax = b  %'dividir a izquierda'
disp(['Optimal solution found by solving Gauss equations: a = ' num2str(ab(1)) ' b = ' num2str(
```

```
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 verifyt 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)   =      0.186
Total CPU secs in NLP function evaluations           =      0.001

EXIT: Optimal Solution Found.
      solver  :   t_proc      (avg)   t_wall      (avg)    n_eval
       nlp_f  |   1.00ms (  8.26us)   1.00ms (  8.26us)       121
  nlp_grad_f  |        0 (       0)        0 (       0)        12
  nlp_hess_l  |        0 (       0)        0 (       0)        10
       total  | 188.00ms (188.00ms) 188.02ms (188.02ms)         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
```

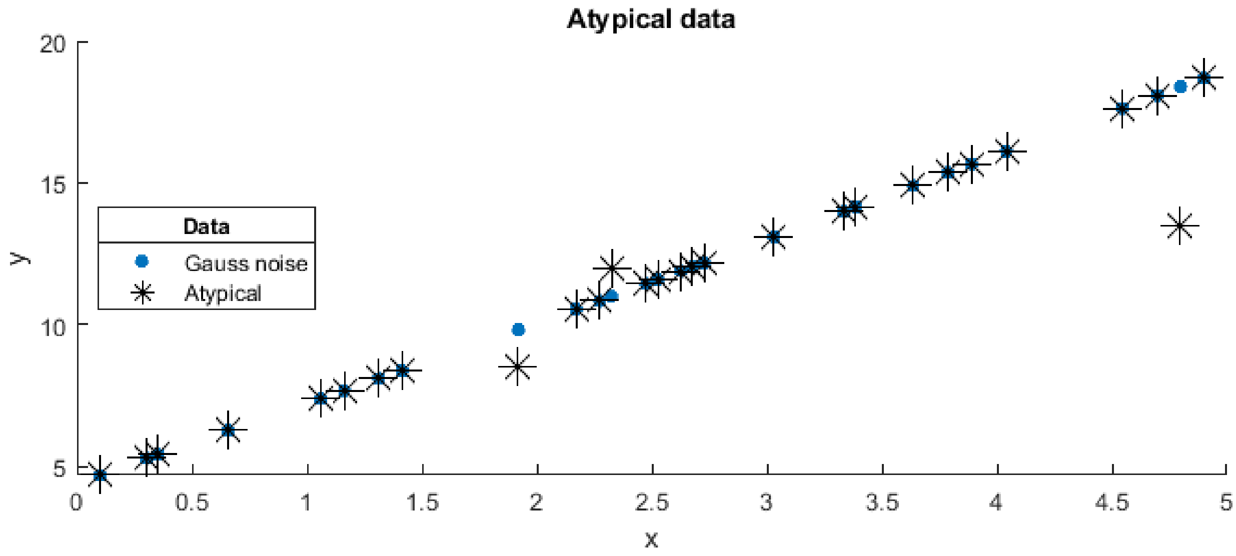
Althouh both solutions have the same value of slope and y-intercept (a, b) with difference in the 15-th decimal, the solution found by solving the Gauss equations has a smaller residual error then it is better.

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
```



Gauss Eq

```
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)*xran
```

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

CasADI

```
opti = casadi.Opti();
a_opt_atypical = opti.variable();
b_opt_atypical = opti.variable();
opti.minimize(norm(a_opt_atypical*xrand+b_opt_atypical-ynoise_atypical,2));

opti.solver('ipopt');

sol_atypical = opti.solve();
```

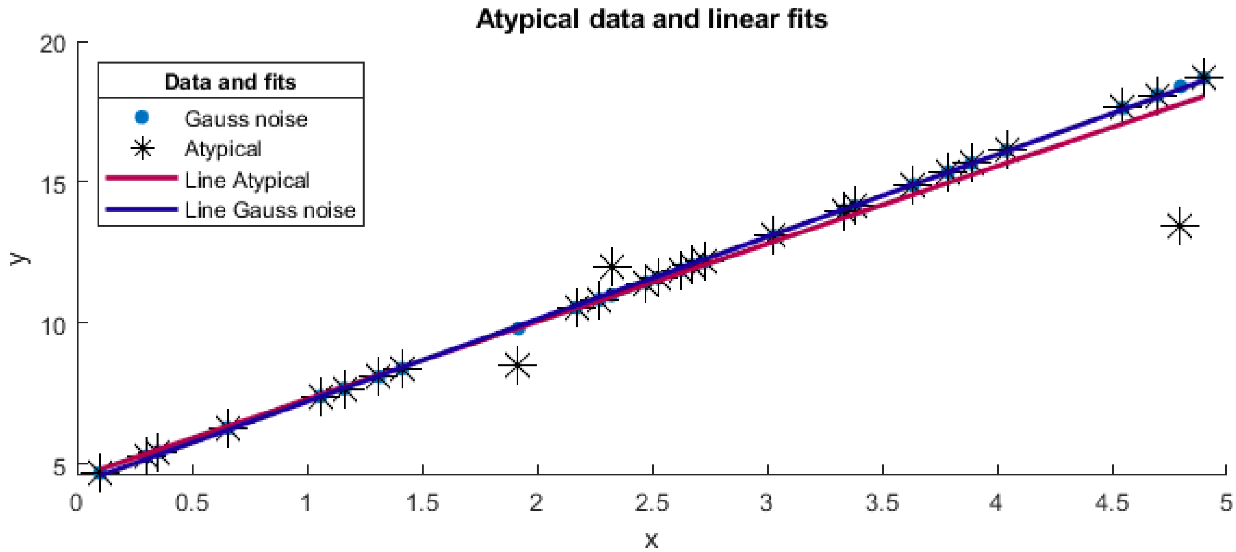
```
This is Ipopt version 3.12.3, running with linear solver mumps.
NOTE: Other linear solvers might be more efficient (see Ipopt documentation).

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 6.7498721e+001 0.00e+000 1.58e+001  -1.0 0.00e+000    -  0.00e+000 0.00e+000   0
   1 3.4624970e+001 0.00e+000 1.57e+001  -1.0 8.79e+002    -  1.00e+000 7.81e-003f  8
   2 2.0753266e+001 0.00e+000 1.54e+001  -1.0 1.18e+002    -  1.00e+000 3.12e-002f  6
   3 5.6320881e+000 0.00e+000 8.05e+000  -1.0 2.49e+001    -  1.00e+000 6.25e-002f  5
   4 4.9550731e+000 0.00e+000 3.16e+000  -1.0 2.59e-001    -  1.00e+000 1.00e+000f  1
   5 4.8565641e+000 0.00e+000 1.32e-001  -1.0 6.92e-002    -  1.00e+000 1.00e+000f  1
   6 4.8563961e+000 0.00e+000 9.14e-006  -2.5 2.73e-003    -  1.00e+000 1.00e+000f  1
   7 4.8563961e+000 0.00e+000 1.08e-015  -8.6 1.89e-007    -  1.00e+000 1.00e+000f  1

Number of Iterations....: 7

                                   (scaled)                 (unscaled)
Objective...............:  4.8563960975416594e+000   4.8563960975416594e+000
Dual infeasibility......:  1.0824674490095276e-015   1.0824674490095276e-015
Constraint violation....:  0.0000000000000000e+000   0.0000000000000000e+000
Complementarity.........:  0.0000000000000000e+000   0.0000000000000000e+000
Overall NLP error.......:  1.0824674490095276e-015   1.0824674490095276e-015


Number of objective function evaluations             = 36
Number of objective gradient evaluations             = 8
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             = 7
Total CPU secs in IPOPT (w/o function evaluations)   =      0.229
Total CPU secs in NLP function evaluations           =      0.002

EXIT: Optimal Solution Found.
      solver  :   t_proc      (avg)   t_wall      (avg)    n_eval
       nlp_f  |   1.00ms ( 27.78us)   1.00ms ( 27.78us)        36
  nlp_grad_f  |   2.00ms (222.22us)   2.00ms (222.22us)         9
  nlp_hess_l  |        0 (       0)        0 (       0)         7
       total  | 244.00ms (244.00ms) 244.05ms (244.05ms)         1
```

```
aopt_atypical = sol_atypical.value(a_opt_atypical); bopt_atypical = sol_atypical.value(b_opt_at
disp(['Optimal solution found be CasADI atypical data: a = ' num2str(aopt_atypical) ' b = ' nur
```

```
Optimal solution found be CasADI atypical data: a = 2.755 b = 4.548
```

```
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], DisplayNar
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

$$\begin{aligned} &\min_{a,b,s} \sum s_i \\ &\text{s.a} - s_i \leq ax_i + b - y_i \leq s_i \quad i = 1, \ldots, N \\ &\quad - si \leq 0 \qquad\qquad\qquad\quad i = 1, \ldots, N \end{aligned} \tag{3}$$

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

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

Write the matrix A and the vectors f, b. Organize he 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 vairable $s_i = ax_i + b - y_i$ and the product $f^T z$ must be a scalar. Then,

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

Let's divide the problem in two

$$
\begin{aligned}
ax_i + b - y_i \leq s_i & \qquad s_i \leq ax_i + b - y_i \\
ax_i + b - s_i \leq y_i & \qquad -ax_i - b + s_i \leq -y_i \\
z_0 x_i + z_1 - z_{i+2} \leq y_i & \quad -z_0 x_i - z_1 + z_{i+2} \leq -y_i
\end{aligned}
$$

for $i = 1, \dots, N$. Merging both inequalities

$$
\begin{aligned}
z_0 x_i + z_1 - z_{i+2} \leq y_i \\
-z_0 x_i - z_1 + z_{i+2} \leq -y_i
\end{aligned}
$$

In matricial form, $Az \leq b$

$$
\begin{pmatrix}
x_1 & 1 & -1 & 0 & 0 & \cdots & 0 \\
x_2 & 1 & 0 & -1 & 0 & \cdots & \vdots \\
\vdots & \vdots & 0 & 0 & \ddots & \cdots & \vdots \\
x_N & 1 & 0 & 0 & 0 & \cdots & -1 \\
-x_1 & -1 & 1 & 0 & 0 & \cdots & 0 \\
-x_2 & -1 & 0 & 1 & 0 & \cdots & 0 \\
\vdots & \vdots & 0 & 0 & \ddots & \cdots & 0 \\
-x_N & -1 & 0 & 0 & 0 & \cdots & 1
\end{pmatrix}
\begin{pmatrix} a \\ b \\ s_1 \\ s_2 \\ \vdots \\ s_{N-1} \\ s_N \end{pmatrix}
\leq
\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \\ -y_1 \\ -y_2 \\ \vdots \\ y_n \end{pmatrix}
$$

$$A = \begin{pmatrix} J_{N\times 2} & -I_{N\times N} \\ -J_{N\times 2} & I_{N\times N} \end{pmatrix}_{2N\times(N+2)}, \qquad z = \begin{pmatrix} a \\ b \\ s_1 \\ \vdots \\ s_N \end{pmatrix}_{(N+2)\times 1}, \qquad b = \begin{pmatrix} y \\ -y \end{pmatrix}_{2N\times 1}$$

```
A = zeros(2*N, N+2);
A(1:N,1:2) = J;          A(N+1:end,1:2) = -J;
A(1:N,3:end) = -eye(N); A(N+1:end,3:end) = eye(N);

f = [0; 0; ones(N,1)];

b = [ynoise; -ynoise]; b_atypical = [ynoise_atypical; -ynoise_atypical];
```

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?

The matrix A_{eq} does not exist and also the vectors $d, u_z = []$; but the vector $u_l = [1, \ldots, 1]^T_{N+2}$ is an unitary vector

Solving using Gauss Eq

Without atypical data

```
SolGeq = A \ b;
```

```
Warning: Rank deficient, rank = 30, tol =  3.044537e-13.
```

```
disp(['Without atypical data a=' num2str(SolGeq(1)) ' b=' num2str(SolGeq(2))])
```

```
Without atypical data a=2.9173 b=4.4053
```

With atypical data

```
SolGeq_atypical = A \ b_atypical;
```

```
Warning: Rank deficient, rank = 30, tol =  3.044537e-13.
```

```
disp(['With atypical data: a=' num2str(SolGeq_atypical(1)) ' b=' num2str(SolGeq_atypical(2))])
```

```
With atypical data: a=2.9173 b=4.4053
```

Solving with linprog

Defining matrices and vectors necessaries

```
Aeq = []; d = []; uz =[];
lz = zeros(N+2,1);
```

With atypical data

```
x_linprog =  linprog(f, A, b, Aeq, d, lz, uz);
```

```
Optimal solution found.
```

```
disp(['Without atypical data: a=' num2str(x_linprog(1)) ' b=' num2str(x_linprog(2))])
```

```
Without atypical data: a=2.9171 b=4.4062
```

With atypical data

```
x_linprog_atypical =  linprog(f, A, b_atypical, Aeq, d, lz, uz);
```

```
Optimal solution found.
```

```
disp(['With atypical data: a=' num2str(x_linprog_atypical(1)) ' b=' num2str(x_linprog_atypical(
```

```
With atypical data: a=2.6085 b=5.918
```

3. Solution by CasADI

Solve the problem with CasADi and compare the results.

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

opti3.minimize(f'*z_opt3);

opti3.subject_to(A*z_opt3 >= b);
opti3.subject_to(lz <= z_opt3 );

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.:     1952
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...............:       92
        inequality constraints with only lower bounds:       92
   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 0.0000000e+000 1.87e+001 7.48e-002  -1.0 0.00e+000    -  0.00e+000 0.00e+000   0
   1 1.0917844e-002 1.81e+001 9.43e+000  -1.0 1.88e+001    -  3.11e-003 3.27e-002h  1
   2 2.2715608e+000 4.19e+000 6.82e+000  -1.0 1.80e+001    -  2.50e-002 7.70e-001f  1
   3 3.0402076e+000 2.47e-001 5.48e-001  -1.0 4.29e+000    -  1.00e+000 9.20e-001h  1
   4 2.2238389e+000 9.06e-002 2.79e-001  -1.7 2.27e-001    -  1.00e+000 7.27e-001h  1
   5 2.8724718e+000 2.46e-002 1.52e+000  -1.7 9.56e-002    -  9.39e-001 6.90e-001h  1
```

```
   6 3.0232701e+000 9.23e-003 2.65e+000  -1.7 2.98e-002    -  1.00e+000 5.86e-001h  1
   7 3.0589733e+000 3.55e-003 7.35e+000  -1.7 1.04e-002    -  1.00e+000 5.45e-001h  1
   8 3.0767044e+000 1.19e-003 1.48e+001  -1.7 4.21e-003    -  1.00e+000 6.03e-001h  1
   9 3.0766868e+000 4.57e-004 3.77e+001  -1.7 1.38e-003    -  1.00e+000 5.80e-001h  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
  10 3.0777849e+000 1.99e-004 8.89e+001  -1.7 5.51e-004    -  1.00e+000 5.89e-001h  1
  11 3.0778745e+000 7.76e-005 2.17e+002  -1.7 2.25e-004    -  1.00e+000 5.85e-001h  1
  12 3.0780555e+000 3.38e-005 5.19e+002  -1.7 9.29e-005    -  1.00e+000 5.87e-001h  1
  13 3.0780720e+000 1.31e-005 1.25e+003  -1.7 3.83e-005    -  1.00e+000 5.88e-001h  1
  14 3.0781030e+000 5.56e-006 2.96e+003  -1.7 1.58e-005    -  1.00e+000 5.91e-001h  1
  15 3.0781057e+000 2.01e-006 6.88e+003  -1.7 6.47e-006    -  1.00e+000 5.98e-001h  1
  16 3.0781111e+000 7.16e-007 1.51e+004  -1.7 2.61e-006    -  1.00e+000 6.16e-001h  1
  17 3.0781115e+000 1.07e-007 2.83e+004  -1.7 1.02e-006    -  1.00e+000 6.62e-001h  1
  18 3.0781122e+000 2.12e-008 3.21e+004  -1.7 3.84e-007    -  1.00e+000 7.85e-001h  1
  19 3.0781125e+000 0.00e+000 2.00e-007  -1.7 1.27e-007    -  1.00e+000 1.00e+000f  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
  20 3.0174392e+000 0.00e+000 7.96e+004  -5.7 7.14e-003    -  1.00e+000 3.96e-001f  1
  21 2.9725737e+000 0.00e+000 1.68e+010  -5.7 3.20e-003    -  9.24e-006 7.44e-001f  1
  22 2.9723668e+000 0.00e+000 1.56e+010  -5.7 1.92e-004    -  7.68e-001 7.30e-002h  1
  23 2.9698956e+000 0.00e+000 2.07e+008  -5.7 1.70e-004    -  8.92e-001 9.87e-001f  1
  24 2.9698972e+000 0.00e+000 6.49e-008  -5.7 2.15e-007    -  1.00e+000 1.00e+000f  1
  25 2.9698919e+000 0.00e+000 1.77e+001  -8.6 2.30e-007    -  6.53e-001 1.00e+000f  1
  26 2.9698901e+000 0.00e+000 4.55e+000  -8.6 1.07e-007    -  7.42e-001 1.00e+000f  1
  27 2.9698888e+000 0.00e+000 1.13e+000  -8.6 7.22e-008    -  7.48e-001 1.00e+000f  1
  28 2.9698873e+000 0.00e+000 2.54e-001  -8.6 8.60e-008    -  7.33e-001 1.00e+000f  1
  29 2.9698871e+000 0.00e+000 2.51e-014  -8.6 2.11e-008    -  1.00e+000 1.00e+000h  1

Number of Iterations....: 29

                                   (scaled)                 (unscaled)
Objective...............:  2.9698871242201736e+000   2.9698871242201736e+000
Dual infeasibility......:  2.5091040356528538e-014   2.5091040356528538e-014
Constraint violation....:  0.0000000000000000e+000   0.0000000000000000e+000
Complementarity.........:  3.3235049581969834e-009   3.3235049581969834e-009
Overall NLP error.......:  3.3235049581969834e-009   3.3235049581969834e-009


Number of objective function evaluations             = 30
Number of objective gradient evaluations             = 30
Number of equality constraint evaluations            = 0
Number of inequality constraint evaluations          = 30
Number of equality constraint Jacobian evaluations   = 0
Number of inequality constraint Jacobian evaluations = 30
Number of Lagrangian Hessian evaluations             = 29
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)        30
       nlp_g  |   1.00ms ( 33.33us)   1.00ms ( 33.33us)        30
  nlp_grad_f  |         0 (       0)        0 (       0)        31
  nlp_hess_l  |         0 (       0)        0 (       0)        29
   nlp_jac_g  |   1.00ms ( 32.26us)   1.00ms ( 32.29us)        31
       total  |  33.00ms ( 33.00ms)  32.01ms ( 32.01ms)         1
```

```
zopt3 = sol3.value(z_opt3);
%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 = 2.9171 b = 4.4062
```

```
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=' num2
plot(xrand, ab(1)*xrand+ab(2), '-', LineWidth=2, Color=[.7 .0 .3], DisplayName=['||_2: a=' num2
plot(xrand, ab(1)*xrand+ab(2), '-', LineWidth=2, Color=[.7 .0 .3], DisplayName=['||_2: a=' num2


plot(xrand,zopt3(1)*xrand+zopt3(2), '-', LineWidth=1, DisplayName=['CADI ||_1: a=' num2str(zopt
plot(xrand,SolGeq(1)*xrand+SolGeq(2), '-', LineWidth=1,  DisplayName=['Geq ||_1: a=' num2str(So
plot(xrand,x_linprog(1)*xrand+x_linprog(2), '-', LineWidth=1, DisplayName=['linp ||_1: a=' num2
leg = legend('Location','northwest');
title(leg,'Data and Fits')
xlabel('x'); ylabel('y'); title('without atypical data', 'Interpreter','latex')
hold off
disp(['Residual sum: Σ si = ' num2str(sum(zopt3(3:end)))]);
```

```
Residual sum: Σ si = 2.9699
```

Atypical data

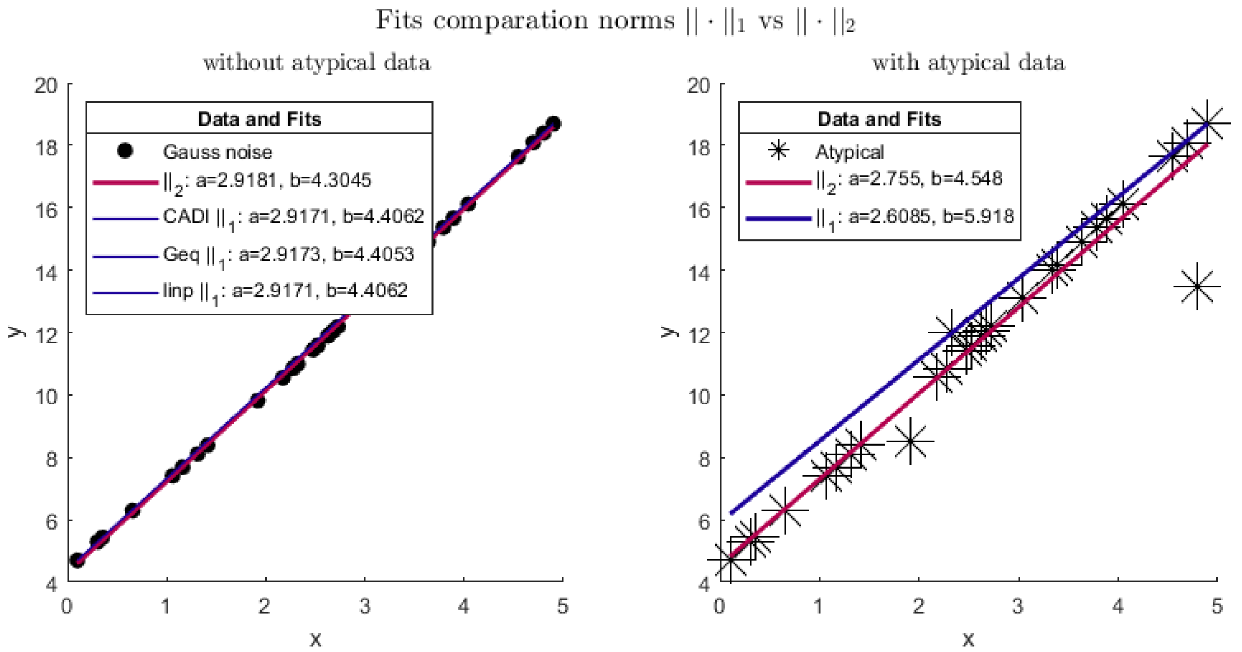
```
opti3_aty = casadi.Opti();
z_opt3_aty = opti3_aty.variable(N+2);

opti3_aty.minimize(f'*z_opt3_aty);

opti3_aty.subject_to(A*z_opt3_aty >= b_atypical);
opti3_aty.subject_to(lz <= z_opt3_aty);

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.:     1952
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...............:       92
        inequality constraints with only lower bounds:       92
   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 0.0000000e+000 1.87e+001 7.48e-002  -1.0 0.00e+000    -  0.00e+000 0.00e+000   0
   1 9.1398758e-003 1.81e+001 9.36e+000  -1.0 1.83e+001    -  3.22e-003 3.37e-002h  1
```

```
   2 2.3805715e-001 1.68e+001 8.66e+000  -1.0 1.79e+001    -  2.78e-002 7.18e-002f  1
   3 3.0426656e-001 1.65e+001 9.80e+000  -1.0 1.72e+001    -  1.08e-001 1.44e-002h  1
   4 2.9846638e+000 1.39e+001 7.24e+000  -1.0 1.74e+001    -  9.36e-002 1.52e-001f  1
   5 9.8733468e+000 8.61e+000 4.41e+000  -1.0 1.35e+001    -  4.35e-001 3.92e-001h  1   6 2.0690082e+001 1.79e+000 9
   7 2.4757289e+001 5.04e-001 3.06e-001  -1.0 1.89e+000    -  1.00e+000 6.83e-001h  1
   8 2.8670972e+001 9.25e-002 1.61e-001  -1.7 4.90e-001    -  7.86e-001 8.38e-001h  1
   9 2.9272802e+001 4.42e-002 4.92e-001  -2.5 8.93e-002    -  8.04e-001 5.41e-001h  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
  10 2.9805346e+001 1.94e-003 3.94e-002  -2.5 4.52e-002    -  1.00e+000 9.35e-001h  1
  11 2.9803661e+001 1.29e-003 7.56e+000  -2.5 2.64e-003    -  1.00e+000 2.76e-001h  1
  12 2.9817993e+001 3.52e-004 5.36e+000  -2.5 1.46e-003    -  1.00e+000 7.02e-001h  1
  13 2.9819940e+001 1.55e-004 1.91e+001  -2.5 4.90e-004    -  1.00e+000 5.52e-001h  1
  14 2.9820340e+001 1.02e-004 6.96e+001  -2.5 1.93e-004    -  1.00e+000 2.99e-001h  2
  15 2.9820719e+001 2.17e-005 3.64e+001  -2.5 1.07e-004    -  1.00e+000 8.03e-001h  1
  16 2.9820715e+001 2.13e-005 6.07e+002  -2.5 2.96e-005    -  1.00e+000 1.47e-002h  6
  17 2.9820632e+001 5.16e-006 3.00e+002  -2.5 2.37e-005    -  1.00e+000 7.48e-001h  1
  18 2.9820636e+001 4.71e-006 2.34e+003  -2.5 6.83e-006    -  1.00e+000 7.10e-002h  4
  19 2.9820662e+001 9.36e-007 9.92e+002  -2.5 5.44e-006    -  1.00e+000 7.85e-001h  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
  20 2.9820662e+001 9.23e-007 9.82e+003  -2.5 1.41e-006    -  1.00e+000 9.38e-003f  7
  21 2.9820657e+001 0.00e+000 2.48e+003  -2.5 1.19e-006    -  1.00e+000 8.57e-001h  1
  22 2.9820657e+001 0.00e+000 2.18e+004  -2.5 2.67e-007    -  1.00e+000 1.05e-001f  4
  23 2.9820655e+001 4.27e-007 2.22e+004  -2.5 1.99e-003    -  2.51e-004 3.91e-004f  1
  24 2.9820655e+001 3.52e-007 1.88e+004  -2.5 3.86e-005    -  1.53e-001 1.25e-001f  4
  25 2.9820654e+001 1.21e-006 1.62e+004  -2.5 4.37e-005    -  1.56e-001 1.12e-001F  1
  26 2.9820654e+001 1.25e-006 1.59e+004  -2.5 3.75e-005    -  9.82e-002 3.91e-003f  9
  27 2.9820654e+001 1.21e-006 1.60e+004  -2.5 4.43e-005    -  1.59e-001 1.95e-003f 10
  28 2.9820654e+001 1.21e-006 1.58e+004  -2.5 5.38e-005    -  1.13e-001 9.77e-004f 11
  29 2.9820655e+001 5.68e-007 5.02e+004  -2.5 4.37e-005    -  7.41e-003 2.10e-002f  2
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
  30 2.9820654e+001 2.30e-006 7.48e+004  -2.5 3.49e-005    -  1.00e+000 7.18e-002h  1
  31 2.9820655e+001 1.94e-006 4.87e+004  -2.5 6.01e-006    -  1.00e+000 8.67e-002f  4
  32 2.9820658e+001 2.59e-007 1.21e+004  -2.5 4.48e-006    -  1.00e+000 8.15e-001h  1
  33 2.9820659e+001 2.12e-007 2.83e-008  -2.5 1.23e-006    -  1.00e+000 1.00e+000f  1
  34 2.9817132e+001 2.56e-006 3.11e+004  -5.7 1.05e-003    -  2.55e-001 6.04e-001f  1
  35 2.9814958e+001 3.52e-006 2.53e+004  -5.7 3.95e-004    -  1.08e-001 9.68e-001h  1
  36 2.9814919e+001 9.31e-007 1.00e+004  -5.7 1.09e-005    -  4.05e-001 8.81e-001H  1
  37 2.9814918e+001 7.99e-007 5.62e+003  -5.7 6.92e-006    -  1.00e+000 9.80e-002f  2
  38 2.9814915e+001 8.15e-007 2.85e-011  -5.7 9.70e-007    -  1.00e+000 1.00e+000h  1
  39 2.9814912e+001 0.00e+000 1.33e+003  -8.6 2.91e-006    -  4.91e-001 1.00e+000h  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
  40 2.9814912e+001 0.00e+000 6.20e+002  -8.6 2.27e-007    -  2.89e-001 7.46e-003h  8
  41 2.9814911e+001 2.17e-008 2.55e+003  -8.6 4.08e-007    -  8.65e-002 1.00e+000h  1
  42 2.9814911e+001 5.08e-009 4.52e+003  -8.6 1.29e-007    -  9.67e-002 1.00e+000h  1
  43 2.9814911e+001 1.04e-007 5.61e+003  -8.6 2.71e-007    -  6.21e-002 1.00e+000h  1
  44 2.9814908e+001 2.88e-007 7.34e+003  -8.6 7.17e-007    -  2.04e-001 1.00e+000H  1
  45 2.9814910e+001 4.75e-007 8.19e+003  -8.6 1.31e-006    -  1.20e-001 1.00e+000h  1
  46 2.9814891e+001 6.91e-007 7.15e+003  -8.6 1.38e-006    -  2.97e-001 1.00e+000h  1
  47 2.9814902e+001 2.91e-007 4.52e+003  -8.6 1.62e-006    -  3.96e-001 5.00e-001h  2
  48 2.9814910e+001 2.41e-007 5.02e+003  -8.6 5.70e-007    -  2.26e-001 1.00e+000h  1
  49 2.9814910e+001 3.61e-007 5.26e+003  -8.6 1.02e-006    -  1.14e-001 1.00e+000h  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
  50 2.9814910e+001 2.64e-007 5.52e+003  -8.6 7.61e-007    -  2.04e-001 1.00e+000h  1
  51 2.9814896e+001 3.94e-007 3.89e+002  -8.6 1.03e-006    -  9.17e-001 1.00e+000h  1
  52 2.9814897e+001 3.81e-007 4.10e+008  -8.6 7.98e-007    -  1.88e-006 3.12e-002h  6
  53r2.9814897e+001 3.81e-007 1.00e+003  -6.4 0.00e+000    -  0.00e+000 3.73e-007R  9
  54r2.9814898e+001 3.22e-007 9.89e+002  -6.4 2.02e-005    -  4.96e-001 5.91e-003f  1
In iteration 54, 1 Slack too small, adjusting variable bound
  55 2.9814898e+001 3.22e-007 5.21e+001  -8.6 7.91e-006    -  3.18e-002 7.97e-004h  1
In iteration 55, 1 Slack too small, adjusting variable bound
  56 2.9814898e+001 3.11e-007 4.97e+001  -8.6 4.96e-006    -  6.16e-002 3.24e-002h  1
In iteration 56, 1 Slack too small, adjusting variable bound
  57 2.9814898e+001 2.80e-007 4.47e+001  -8.6 1.08e-006    -  1.01e-001 1.00e-001h  1
In iteration 57, 1 Slack too small, adjusting variable bound
```

```
  58 2.9814900e+001 2.14e-007 2.00e+001  -8.6 5.62e-007    -  1.00e+000 2.37e-001h  1
  59 2.9814905e+001 0.00e+000 8.50e-001  -8.6 4.29e-007    -  4.08e-001 1.00e+000h  1
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
  60 2.9814905e+001 0.00e+000 2.84e-014  -8.6 4.54e-010    -  1.00e+000 1.00e+000h  1

Number of Iterations....: 60

                                   (scaled)                 (unscaled)
Objective...............:   2.9814905324331626e+001    2.9814905324331626e+001
Dual infeasibility......:   2.8421709430404007e-014    2.8421709430404007e-014
Constraint violation....:   0.0000000000000000e+000    0.0000000000000000e+000
Complementarity.........:   2.5379143806015820e-009    2.5379143806015820e-009
Overall NLP error.......:   2.5379143806015820e-009    2.5379143806015820e-009


Number of objective function evaluations             = 154
Number of objective gradient evaluations             = 61
Number of equality constraint evaluations            = 0
Number of inequality constraint evaluations          = 154
Number of equality constraint Jacobian evaluations   = 0
Number of inequality constraint Jacobian evaluations = 62
Number of Lagrangian Hessian evaluations             = 60
Total CPU secs in IPOPT (w/o function evaluations)   =      0.180
Total CPU secs in NLP function evaluations           =      0.002

EXIT: Optimal Solution Found.
      solver  :   t_proc      (avg)   t_wall      (avg)    n_eval
       nlp_f  |        0 (       0)        0 (       0)       154
       nlp_g  |        0 (       0)        0 (       0)       154
  nlp_grad_f  |        0 (       0)        0 (       0)        62
  nlp_hess_l  |        0 (       0)        0 (       0)        59
   nlp_jac_g  |   2.00ms ( 31.75us)   2.00ms ( 31.78us)        63
       total  | 183.00ms (183.00ms) 183.04ms (183.04ms)         1
```

```
zopt3_aty = sol3_aty.value(z_opt3_aty);
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], DisplayNar
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');
```




```
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 ||_1: Σ si = 2.9699
      residual error in ||_2 r = 1.3145e-13
- with atypical data
      Residual sum  in ||_1: Σ si = 29.8149
      residual error in ||_2  r = 9.9476e-14
```

4. Integral Equation

Consider the following integral equation

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

a) Exact solution

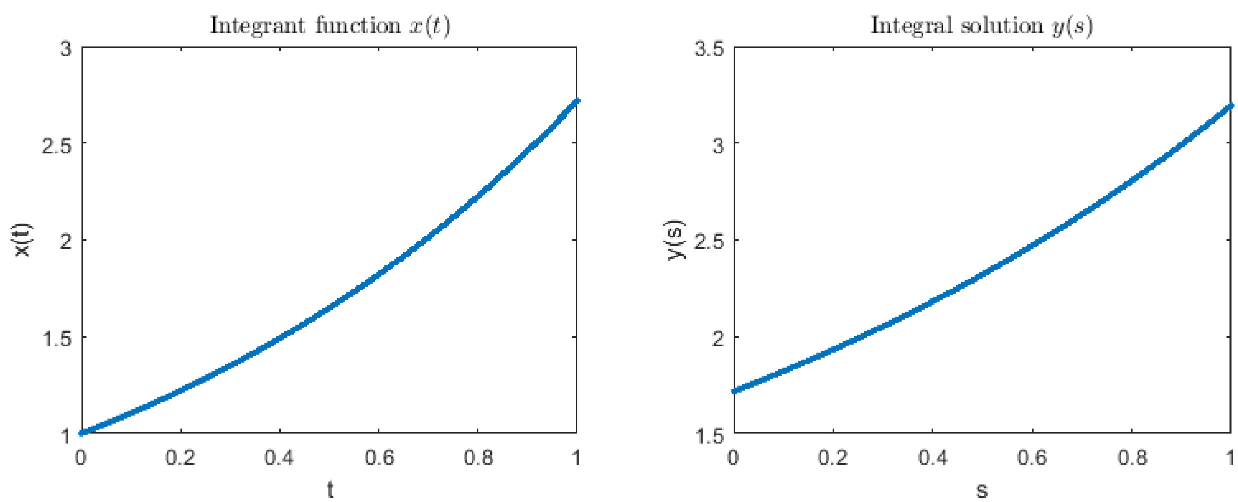
Find the exact solution (*Advice:* Interprete the operator $A : L^2(0,1) \to L^2(0,1)$ as the Laplace transform).

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('Integrant 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 $i = 1, \ldots, n$ to get a linear system of equations.

Remembering the trapezoidal rule. Let $\{x_k\}$ be a partition of $[0, 1]$ such that $0 = t_1 < t_2 < \cdots < t_{n-1} < t_n = 1$ 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

$$\int_0^1 e^{st}x(t)dt \approx \sum_{k=1}^{n-1} \frac{e^{st_{k+1}}x_{k+1} + e^{st_k}x_k}{2} \cdot \frac{1}{n} = \frac{1}{2n}\sum_{k=1}^{n-1}(e^{st_{k+1}}x_{k+1} + e^{st_k}x_k)$$

$$= \frac{1}{2n}\left[e^{st_2}x_2 + e^{st_1}x_1 + e^{st_3}x_3 + e^{st_2}x_2 + e^{st_4}x_4 + e^{st_3}x_3 + \cdots + e^{st_n}x_n + e^{skt_{n-1}}x_{n-1}\right]$$

$$= \frac{1}{2n}\left[e^{st_1}x_1 + 2e^{st_2}x_2 + 2e^{st_3}x_3 + 2e^{st_4}x_4 + \cdots + 2e^{skt_{n-1}}x_{n-1} + e^{st_n}x_n\right] = \frac{e^{s+1}-1}{s+1}$$

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

$$y(s) = \int_0^1 e^{st}x(t)dt \approx \frac{1}{2n}\begin{pmatrix} e^{st_1} & 2e^{st_2} & 2e^{st_3} & \cdots & 2e^{skt_{N-1}} & e^{st_N}\end{pmatrix} \cdot \begin{pmatrix} x1 & x2 & x_3 & \cdots & x_{n-1} & x_n\end{pmatrix} = \frac{e^{s+1}-1}{s+1}$$

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

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

$$s = h: \quad \frac{1}{2n}\begin{pmatrix} e^{ht_1} & 2e^{ht_2} & 2e^{ht_3} & \cdots & 2e^{ht_{n-1}} & e^{ht_n}\end{pmatrix} \cdot \begin{pmatrix} x_1 & x_2 & x_3 & \cdots & x_{n-1} & x_n\end{pmatrix} = \frac{e^{h+1}-1}{h+1}$$

$$s = 2h: \quad \frac{1}{2n}\begin{pmatrix} e^{2ht_1} & 2e^{2ht_2} & 2e^{2ht_3} & \cdots & 2e^{2ht_{n-1}} & e^{2ht_n}\end{pmatrix} \cdot \begin{pmatrix} x_1 & x_2 & x_3 & \cdots & x_{n-1} & x_n\end{pmatrix} = \frac{e^{2h+1}-1}{2h+1}$$

$$\vdots$$

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

$$s = nh: \quad \frac{1}{2n}\begin{pmatrix} e^{nht_1} & 2e^{nht_2} & 2e^{nht_3} & \cdots & 2e^{nht_{n-1}} & e^{nht_n}\end{pmatrix} \cdot \begin{pmatrix} x_1 & x_2 & x_3 & \cdots & x_{n-1} & x_n\end{pmatrix} = \frac{e^{nh+1}-1}{nh+1}$$

that can be write as $Ax = b$ where

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

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

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);
[time4, sol4, co4, M4, b4] = System(32,0);
[time100, sol100, co100, M100, b100] = System(128,0);
```

```
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:  ' nu
```

where this variable **`cond`** is the conndition 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

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

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

```
x1 = SolCasADI(4, M1, b1);
x2 = SolCasADI(8, M2, b2);
x3 = SolCasADI(16, M3, b3);
x4 = SolCasADI(32, M4, b4);
x100 = SolCasADI(128, M100, b100);
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',Mark
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',Marke
plot(time2', 100*abs((x2-exp( 0:1/7:1))/exp( 0:1/7:1)),'+-', DisplayName='CasADIeq, 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:  ' nu
np = 2.^(1:1:8);
Solutions = zeros(8,2);
for i=np
    [s_casadi, s_Gausseq] = Error(0, i);
    Solutions(floor(log2(i)),:) = [s_casadi, s_Gausseq];
end
```

```
np_small = 1:2:256;
parameters = zeros(2,2);
parameters(1,:) = [ones(length(np),1)  np'] \ Solutions(:,1);
parameters(2,:) = [ones(length(np),1)  np'] \ Solutions(:,2);
clf
figure(Position=[10 10 900 200])
hold on
```

```
plot(np, Solutions(:,1), 'k.', MarkerSize=20, DisplayName='CasADI')
plot(np, Solutions(:,2), 'b.', MarkerSize=15, DisplayName='Gauss Eq')
plot(np_small, parameters(1,1)+np_small*+parameters(1,2), 'k-', DisplayName=['m=' num2str(paran
plot(np_small, parameters(2,1)+np_small*parameters(2,2), 'b-', DisplayName=['m=' num2str(parame
hold off
leg = legend('Location','northeastoutside');
title(leg,'Number of points')

ylim([-1 10])
xlabel('n'); ylabel('error')
```

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

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

It is also important to mention that the result given by solving the Gauss equations is counterintuitive, when the number of points n increse the function starts to behave chaotically, the solution starts to oscillate drastically, then the bets 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) $n = 100,\ \alpha = 0.2$, i) $n = 100,\ \alpha = 0.1$, and i) $n = 100,\ \alpha = 10^{-3}$. 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);
[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 α')
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
plot(time_iii, 100*abs((sol_iii-exp(time_iii)')./exp(time_iii)'), '-', DisplayName='0.001', Lin
leg = legend('Location','northeastoutside');
title(leg,'Regularization α')
xlabel('time'); ylabel('error (%)'); ylim([0 20])
hold off
disp(['The higher relative error for α=0.2 is   ' num2str(max(100*abs((sol_i-exp(time_i)')./exp
      'The higher relative error for α=0.1 is   ' num2str(max(100*abs((sol_ii-exp(time_ii)')./e
      'The higher relative error for α=0.001 is ' num2str(max(100*abs((sol_iii-exp(time_iii)').
disp(['cond n=100, α=0:     ' num2str(round(co_0,2)) newline 'cond n=100, α=0.2:    ' num2str(co
      'cond n=100, α=0.01:  ' num2str(co_ii) newline 'cond n=100, α=0.001: ' num2str(co_iii)  ]
```

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 regularizaion 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);
x_casadi_i = SolCasADI(100, M_i, b_i);
x_casadi_ii = SolCasADI(100, M_ii, b_ii);
x_casadi_iii = SolCasADI(100, M_iii, b_iii);
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 α')
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
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
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 α')
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))'
```

The relative error is ploted 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.

```
Solutions_regu = zeros(8,6);
for i=2.^(1:1:8)
    [s_casadi_i,   s_Gausseq_i]   = Error(0.2,   i);
    [s_casadi_ii,  s_Gausseq_ii]  = Error(0.1,   i);
    [s_casadi_iii, s_Gausseq_iii] = Error(0.001, i);
    Solutions_regu(floor(log2(i)),:) = [s_casadi_i, s_Gausseq_i, s_casadi_ii, s_Gausseq_ii, s_c
end
```

```
np_small = 1:2:256;
parameters = zeros(6,2);
for i=1:6
    parameters(i,:) = [ones(length(np),1)  np'] \ Solutions_regu(:,i);
end

figure(Position=[10 10 900 300])
hold on
plot(2.^(1:1:8), Solutions_regu(:,1), 'k.', MarkerSize=15, DisplayName='CasADI α=0.2')
plot(2.^(1:1:8), Solutions_regu(:,3), 'r.', MarkerSize=15, DisplayName='CasADI α=0.1')
plot(2.^(1:1:8), Solutions_regu(:,5), 'b.', MarkerSize=15, DisplayName='CasADI α=0.001')
plot(2.^(1:1:8), Solutions_regu(:,2), 'k+', MarkerSize=20, DisplayName='Gauss Eq α=0.2')
plot(2.^(1:1:8), Solutions_regu(:,4), 'r+', MarkerSize=20, DisplayName='Gauss Eq α=0.1')
plot(2.^(1:1:8), Solutions_regu(:,6), 'b+', MarkerSize=20, DisplayName='Gauss Eq α=0.001')

plot(np_small, parameters(1,1)+np_small*+parameters(1,2), 'k-', DisplayName=['CADI m=' num2str(
plot(np_small, parameters(3,1)+np_small*+parameters(3,2), 'r-', DisplayName=['CADI m=' num2str(
plot(np_small, parameters(5,1)+np_small*+parameters(5,2), 'b-', DisplayName=['CADI m=' num2str(
plot(np_small, parameters(2,1)+np_small*+parameters(2,2), 'k--', DisplayName=['Geq m=' num2str(
plot(np_small, parameters(4,1)+np_small*+parameters(4,2), 'r--', DisplayName=['Geq m=' num2str(
plot(np_small, parameters(6,1)+np_small*+parameters(6,2), 'b--', DisplayName=['Geq m=' num2str(

hold off
leg = legend('Location','northeastoutside');
title(leg,'Number of points')
```

```
%ylim([-1 10])
xlabel('n'); ylabel('error'); title('Error vs Number of Points')
parameters
```

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

function [s_casadi, s_Gausseq] = Error(alpha, n)
    [time, sol, co, M, b] = System(n,alpha);
    x= SolCasADI(n, M, b);
    s_casadi  = sum((x'-exp(0:1/(n-1):1)).^2);
    s_Gausseq = sum((sol'-exp(0:1/(n-1):1)).^2);
end
```