# Optimization and Control, Laboratory

# Introduction: Yalmip optimization toolbox

## 1 Installing Yalmip

The open source toolbox Yalmip [1] offers the possibility to conveniently solve numerical optimization problems. Yalmip reformulates the users input to make it independent of the used solver. Consequently, the user can specify the optimization problem without taking care of the exact syntax to call the desired solver. More information about Yalmip, supported solvers and downloads are available at `https://yalmip.github.io/`.

To install yalmip, perform the following steps (installation instructions are available at `http://www.tbxmanager.com`):

1. Create a local folder, where the files can be stored
2. Launch Matlab and navigate to the previously created folder
3. Execute the following commands:
   (a) `urlwrite('http://www.tbxmanager.com/tbxmanager.m','tbxmanager.m')`
   (b) `tbxmanager`
   (c) `savepath` (you will need admin permissions)
   (d) `tbxmanager install yalmip` (base package)
   (e) `tbxmanager install oasesmex` (solver for quadratic optimization problems)
   (f) `tbxmanager install sedumi` (solver for semidefinite optimization problems)
4. Run `yalmiptest` to verify the correct installation (available solvers are located and some optimization problems are solved)

Some older Matlab-Versions could lead to error messages, as some Matlab internal interfaces have changed. Information how to resolve these issues are provided in the Matlab help.

## 2 Example how to solve linear and quadratic optimization problems

Using a simple example, the basic functionality of yalmip is explaind, see [1].

### 2.1 Task Description

Consider that a measurement $y$ of a system

$$y = Ax \tag{1}$$

is available where $x$ denotes the vector of parameters, matrix $A$ defines how these parameters act on the output $y$. In practical applications the measurement signal is always affected by measurement noise $v$. As a result, the measured output can be written as

$$y = Ax + v \ . \tag{2}$$

The trace of a measured signal is depicted in figure 1. Using this measured signal, the

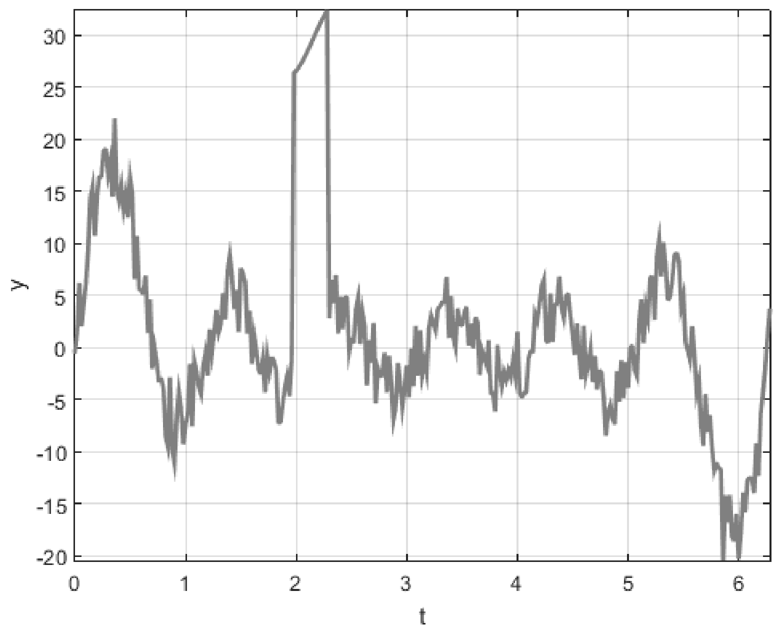

Figure 1: Data

(unknown) parameter vector $\hat{x}$ should be estimated such that the error

$$e = y - A\hat{x} \tag{3}$$

is minimized.

Matlab-Code:

```
yalmip('clear')     % delete old yalmip variables

x = [1 2 3 4 5 6]';              % Parameter
t = (0:0.02:2*pi)';              % Time Vector
A = [sin(t) sin(2*t) sin(3*t) sin(4*t) sin(5*t) sin(6*t)];
v = (-4+8*rand(length(t),1));    % Measurement noise
v(100:115) = 30;                 % Additional measurement error
y = A*x+v;
figure(1)
plot(t,y,'Color',[1 1 1]*0.5,'LineWidth',2); hold on
xlabel('t')
ylabel('y')
grid on; axis tight
```

## 2.2 Minimizing the Sum of Absolute Error Values ($L_1$)

The optimization problem can be formulated as

$$\min_{\hat{x}} \sum_{i=1}^{N} |e_i| \ , \tag{4}$$

where $e_i$ is denotes the value of (3) for $N$ different time instances (measurement values). To formulate this optimization problem with Yalmip, 6 variables are necessary. The absolute value is considered by using symmetric constraints of the error signal. Figure 2 shows the results of the $L_1$-optimization.

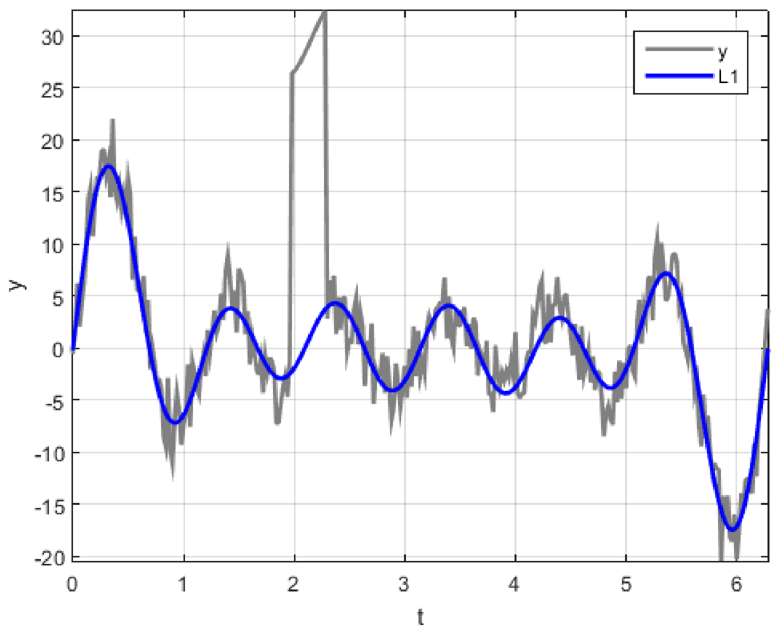

Figure 2: Minimizing the sum of absolute error values ($L_1$)

Matlab-Code:

```
xhat = sdpvar(6,1);  % Declare optimization variables
e = y-A*xhat;        % Error

bounds = sdpvar(length(e),1);
constraints = [-bounds <= e <= bounds];
options = sdpsettings('verbose',0);

diagnostics_L1 = optimize(constraints,sum(bounds),options)

if diagnostics_L1.problem > 0
 error('Error during optimization')
end
x_L1 = value(xhat);  % Convert results in double variables
y_L1 = A*x_L1;       % Compute results

plot(t,y_L1,'b','LineWidth',2);
axis tight
legend('y','L1')
```

### 2.3 Minimizing the Sum of Squared Errors ($L_2$)

Optimization problem

$$\min_{\hat{x}} \sum_{i=1}^{N} e_i^2 = \min_{\hat{x}} e^T e \tag{5}$$

Using the dot product of the error vector $e$, the sum of squared error values are introduced. The result obtained with the estimated parameter vector $\hat{x}$ is shown in figure 3.
Matlab-Code:

```
diagnostics_L2 = optimize([],e'*e,options)
if diagnostics_L2.problem > 0
 error('Error during optimization')
end
x_L2 = value(xhat);
y_L2 = A*x_L2;

plot(t,y_L2,'r','LineWidth',2);
axis tight
legend('y','L1','L2')
```

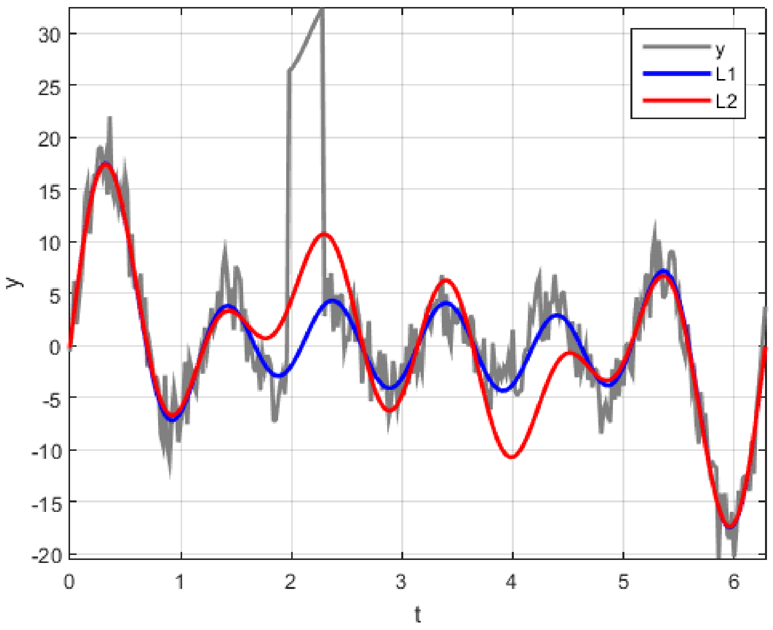

Figure 3: Minimizing the Sum of Squared Errors ($L_2$)

## 2.4 Minimizing the Largest Absolute Error Value ($L_\infty$)

Optimization problem

$$\min_{\hat{x}} \max_{i=1,\ldots,N} |e_i| \tag{6}$$

To formulate this optimization problem, only one additional variable (the maximum absolute error value) is necessary. This scalar variable is used in the constraints of the optimization problem. In contrast to the $L_1$-optimization, where the sum of all absolute error values are minimized, only the largest absolute value of the error is minimized. Figure 4 illustrate the results obtained with this optimization method. For comparison reasons, the results obtained with the other methods are also shown in this figure.
Matlab-Code:

```
bound = sdpvar(1,1);
constraints  = [-bound <= e <= bound];

diagnostics_Linf = optimize(constraints,bound,options)
if diagnostics_Linf.problem > 0
 error('Error during optimization')
end
x_Linf = value(xhat);
y_Linf = A*x_Linf;

plot(t,y_Linf,'g','LineWidth',2);
axis tight
legend('y','L1','L2','Linf')
```

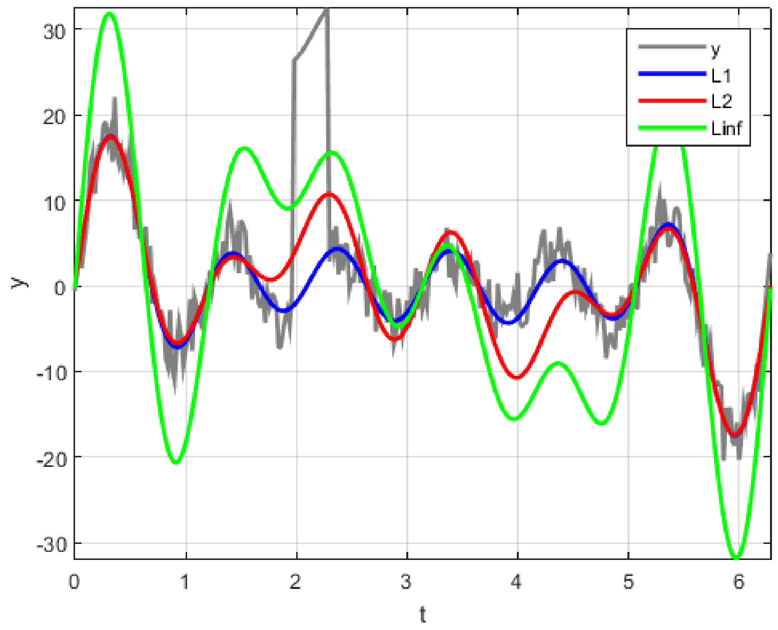

Figure 4: Minimizing the Largest Absolute Error Value ($L_\infty$)

## 2.5 Comparison

Before comparing the results obtained with the three cost functions, it is not known which one leads to the best results. All methods are optimal with respect to their cost function. Comparing the results in figure 4 makes it clear, that the $L_1$-optimization leads to the best results. The results obtained with the $L_2$- and $L_\infty$-optimization are affected by a disturbance which acts between $2.0\,s$ and $2.3\,s$. Table 1 shows the resulting values obtained with the different cost functions and the resulting cost function values.

| | $L_1$ | $L_2$ | $L_\infty$ |
|---|---|---|---|
| $\sum \lvert e \rvert$ | $1.0134\,10^3$ | $1.2054\,10^3$ | $2.8157\,10^3$ |
| $\sum e^2$ | $1.4424\,10^4$ | $1.1590\,10^4$ | $3.3376\,10^4$ |
| $\max \lvert e \rvert$ | 28.6348 | 22.5206 | 16.9614 |

Table 1: Comparison of the results obtained with the different optimization methods.

## References

[1] J. Löfberg: *YALMIP : A Toolbox for Modeling and Optimization in MATLAB*, Proceedings of the CACSD Conference, 2004, Taipei, Taiwan