Interfacing Chrono & Matlab

Chrono units for MATLAB and SIMULINK

Chrono::MATLAB

Module for interfacing Chrono and MATLAB

Chrono::MATLAB

- Chrono::MATLAB is a C++ module that enables communication with MATLAB

- Features:

  - **call Matlab commands** from your C++ program,
  - **exchange data** to/from Matlab
    - Chrono::Engine C++ matrices are converted to MATLAB ,
    - MATLAB variables and matrices are converted to Chrono::Engine C++ matrices
  - use the MATLAB **visualization** tools, to show simulation data in 2D/3D plots, etc.

- Dependencies:
  - Chrono::Engine main module (required)
  - Chrono::Matlab module
  - MATLAB license

Code organization

FOLDER	CONTENT
src/chrono_matlab	main Chrono::MATLAB library implementation
src/demos/matlab	Various demo programs

Example

- Call MATLAB command(s)

```
// This is the object that you can use to access the Matlab engine.
 ChMatlabEngine matlab_engine;

// EXAMPLE 1: execute a Matlab command

 matlab_engine.Eval(
     "z=peaks(25); \
      surf(z);  \
      colormap(jet); \
      pause(4); \
     ");
```

Example

- Pass a Chrono matrix to MATLAB

```
ChMatrixDynamic<> m_time(30, 1);
ChMatrixDynamic<> m_sine(30, 1);
for (int i = 0; i < 30; i++) {
    m_time(i, 0) = ((double)i / 30.) * 5.;
    m_sine(i, 0) = sin(m_time(i, 0) * 2.);
}
matlab_engine.PutVariable(m_time, "m_time");
matlab_engine.PutVariable(m_sine, "m_sine");
matlab_engine.Eval("figure; plot(m_time,m_sine);");
```

Example

- Pass a MATLAB matrix to Chrono

```
matlab_engine.Eval("m_matr=[0:0.1:5]';");

ChMatrixDynamic<double> m_matr;
matlab_engine.GetVariable(m_matr, "m_matr");
```

Chrono::COSIMULATION

Unit for cosimulation between Chrono and SIMULINK

Chrono::Cosimulation

- Used for interfacing to SIMULINK
- More generally, Chrono::Cosimulation is a C++ module that enables basic cosimulation via TCP/IP sockets

- Features:
  - C++ functions to **send/receive datagrams using TCP/IP** sockets from Chrono::Engine
  - Can be used to co-simulate with **SIMULINK**
  - a **CEcosimulation.mdl** block is provided, to be inserted in your Simulink models as a ready-to-use interface to Chrono::Engine

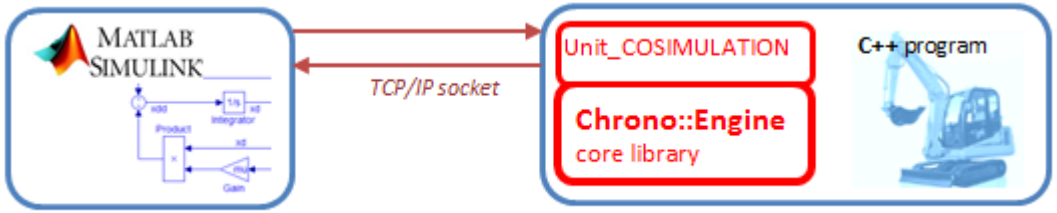
- Dependencies:
  - Chrono::Engine main module (required)
  - Chrono::Matlab and Chrono::Cosimulation
  - MATLAB & Simulink license

Code organization

FOLDER	CONTENT
src/chrono_cosimulation	main Chrono::COSIMULATION library implementation
src/demos/cosimulation	Various demo programs

Background

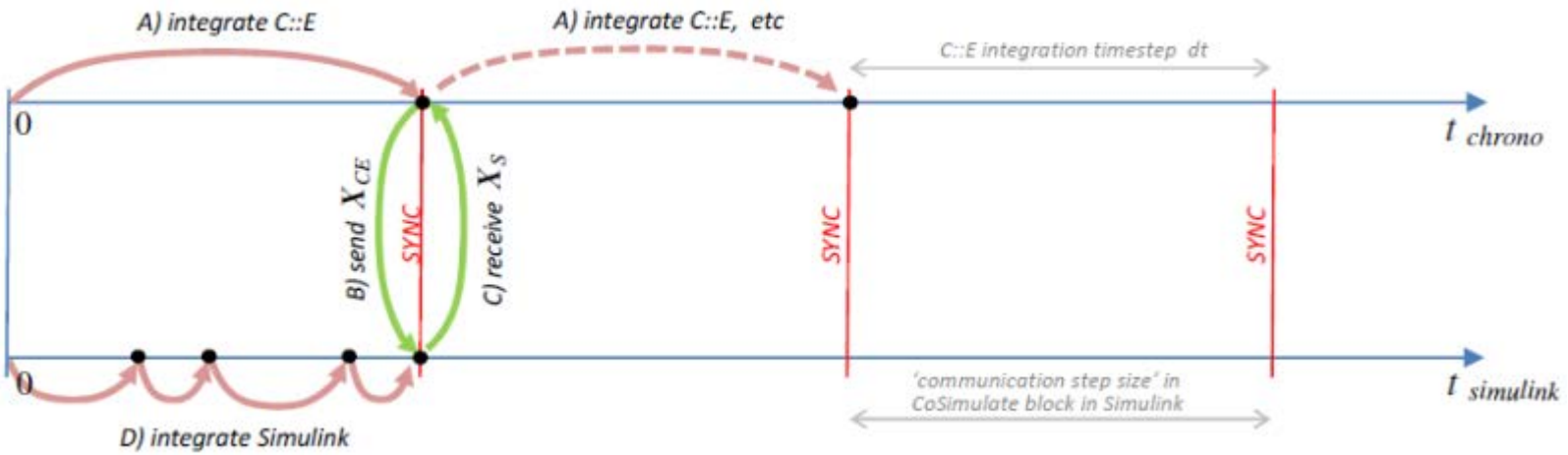
- Two way co-simulation draws on two simulation tools which simultaneously simulate (advance in time) the two subsystems in which the original system is partitioned.
- Once in a while the two solvers (simulation tools) synchronize to exchange data after which they proceed independently until the next synchronization time.
- TCP/IP sockets are used to exchange data:



- This periodic data synchronization is necessary because the subsystems are coupled. For tightly coupled subsystems the synchronization happens very often.

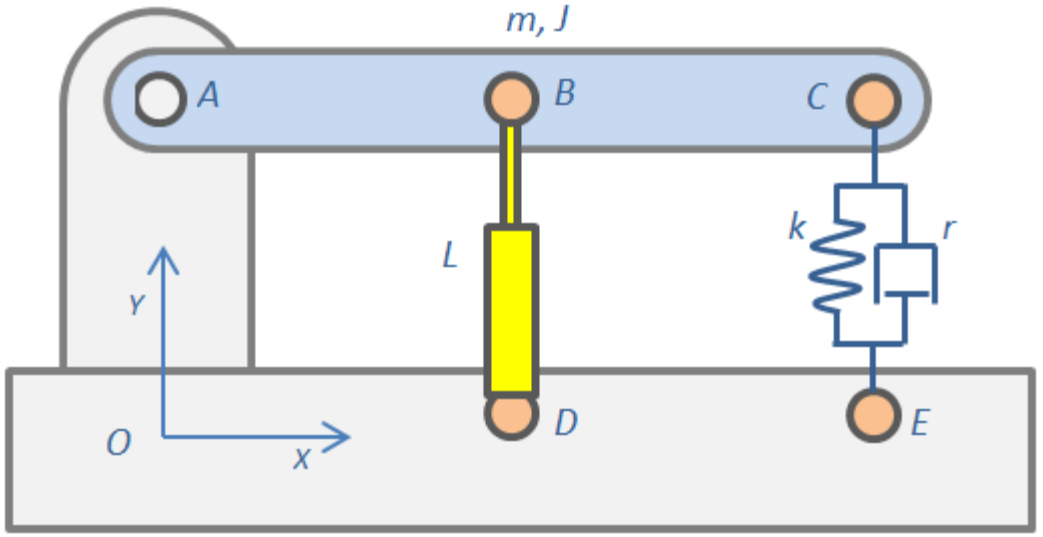
Background

- A periodic data synchronization is necessary because the subsystems are coupled.
- For tightly coupled subsystems the synchronization must happen very often.



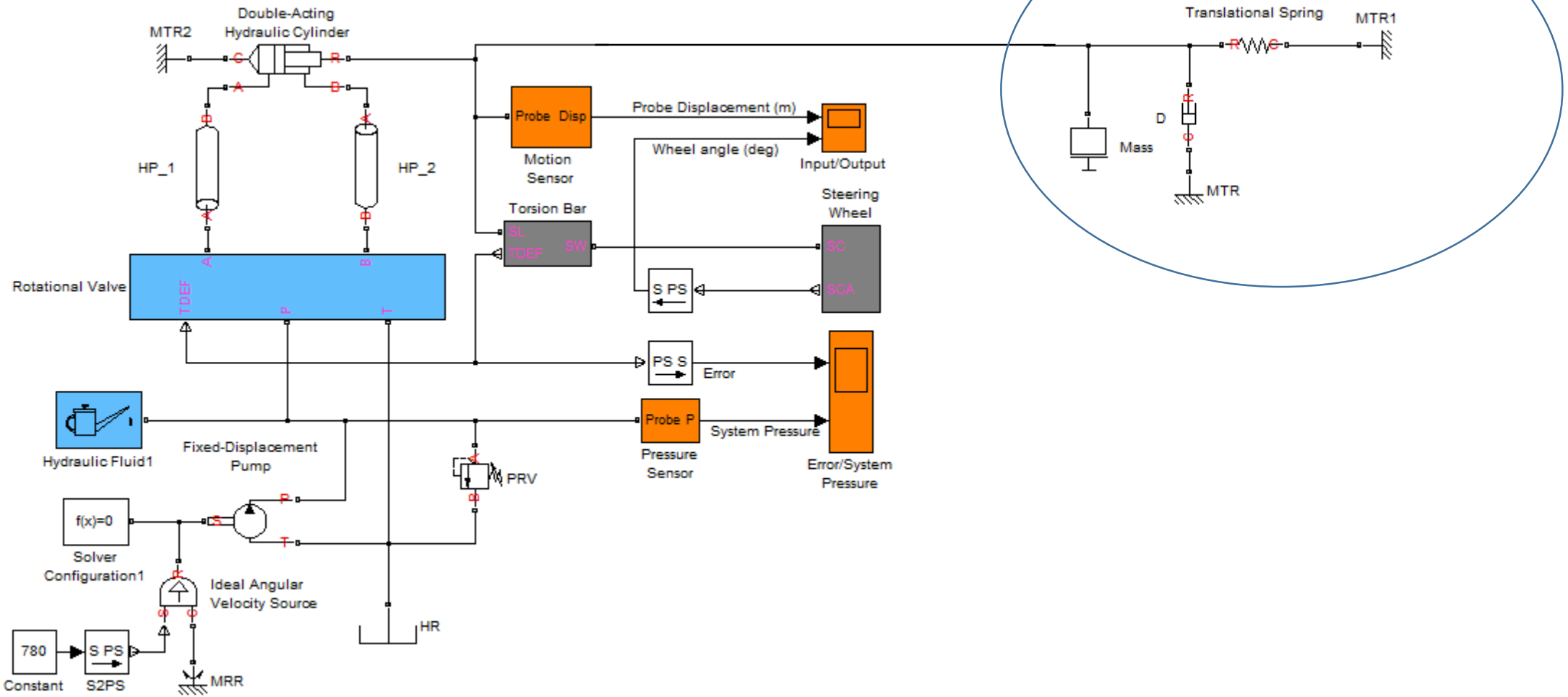
Example

- Co-simulate of a Chrono mechanism with a SIMULINK pneumatic system



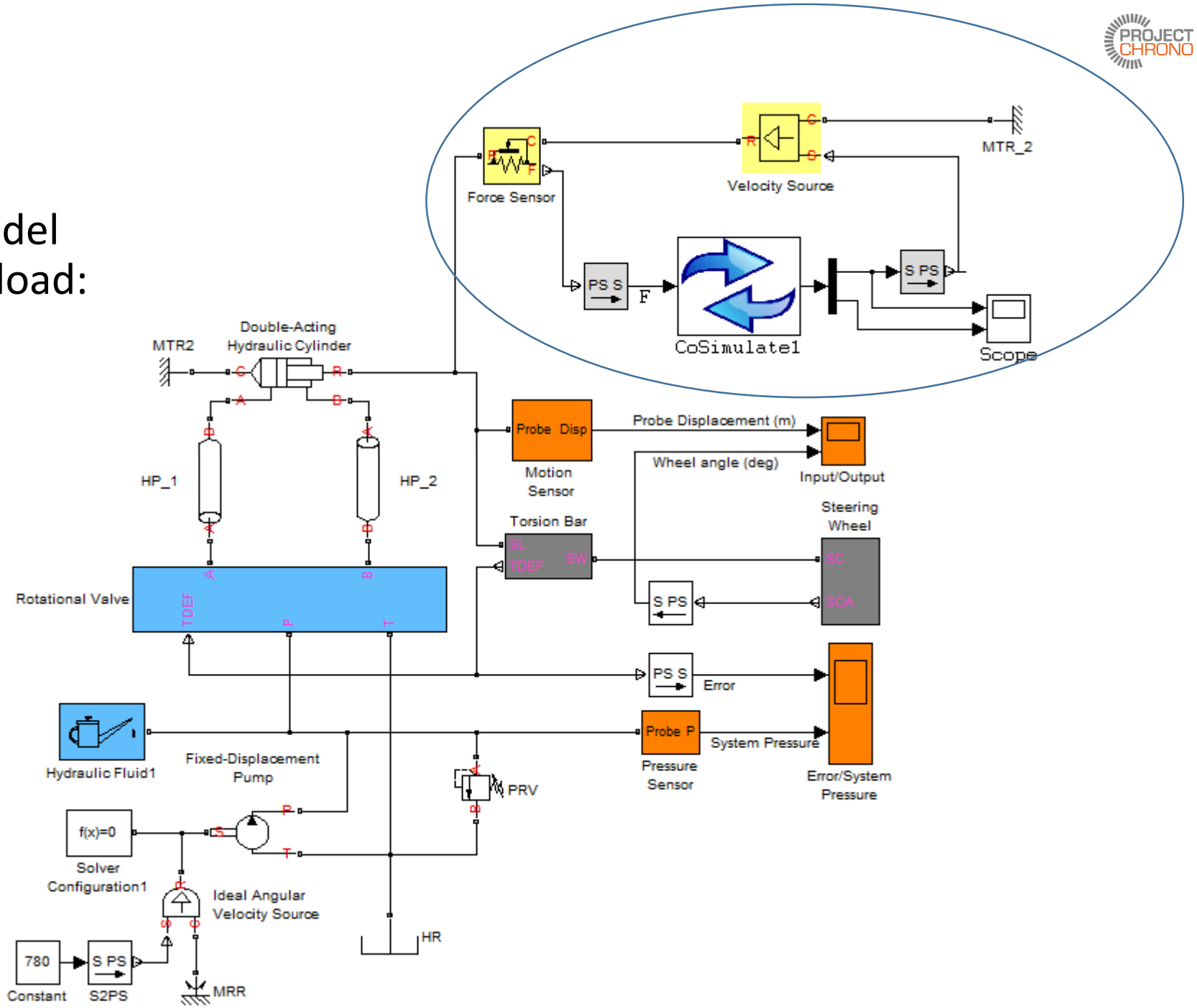
Example

- A typical SIMULINK model of a pneumatic system:



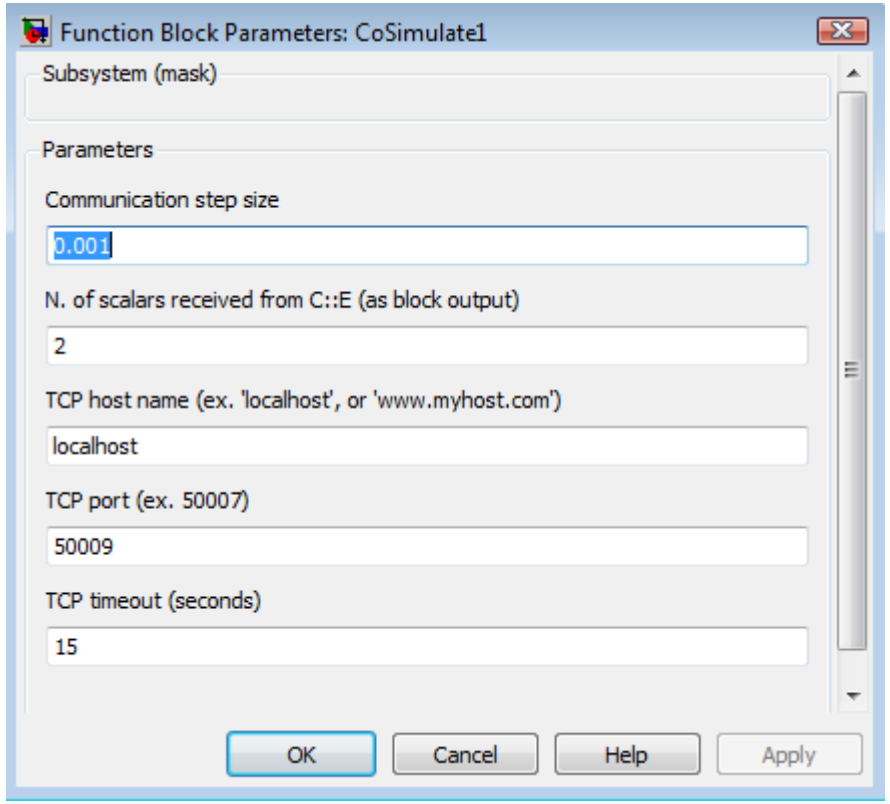
Example

- The SIMULINK model with cosimulated load:



Example

- Set parameters in the cosimulation block:



Example

- In Chrono:
  - **receive** one variable from Simulink (the hydraulic cylinder force)
  - **send** two variables to Simulink (the hydraulic cylinder velocity and displacement)

- First we must open a TCP/IP socket:

```
ChSocketFramework socket_tools;

ChCosimulation cosimul_interface(socket_tools,
                                 1,   // n.input values from Simulink
                                 2);  // n.output values to Simulink

// Prepare the two column vectors of data that will be swapped
ChMatrixDynamic<double> data_in(1, 1);
ChMatrixDynamic<double> data_out(2, 1);

// 4) Wait client (Simulink) to connect...
int PORTNUM = 50009;
cosimul_interface.WaitConnection(PORTNUM);
matlab_engine.Eval("m_matr=[0:0.1:5]';");
```

Example

```
        while (true) {
            // A) ----------------- ADVANCE THE Chrono SIMULATION
            if (dt > 0)
                my_system.DoStepDynamics(dt);

            mytime += dt;

            // B) ----------------- SYNCHRONIZATION

            // B.1) - SEND data
            //      * the velocity of the hydraulic actuator
            //      * the displacement of the hydraulic actuator
            data_out(0) = my_link_actuator->GetDist_dt();
            data_out(1) = my_link_actuator->GetDist() - my_link_actuator->Get_SpringRestLength();

            cosimul_interface.SendData(mytime, &data_out);  // --> to Simulink

            // B.2) - RECEIVE data
            //      * the force of the hydraulic actuator
            cosimul_interface.ReceiveData(histime, &data_in);  // <-- from Simulink

            // - Update the Chrono system with the force value that we received
            my_link_actuator->Set_SpringF(data_in(0));

        }
```