Documentation of SensPlaceTool-package for Matlab

Erik Frisk
Department of Electrical Engineering
Linköpings universitet
`frisk@isy.liu.se`

September 5, 2007

1 Introduction

SensPlaceTool is a Matlab-implementation of algorithms for sensor placement analysis for fault diagnosis based on structural models. SensPlaceTool is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation.

The most recent version is always available at `http://www.fs.isy.liu.se/Software/SensPlaceTool/`.

If you have comments, bugreports or other concerns the authors may be contacted at the following email addresses:

Erik Frisk (`frisk@isy.liu.se`) and Mattias Krysander (`matkr@isy.liu.se`).

1.1 Requirements and installation

The code requires a basic Matlab installation. Installation is simple, just unpack the package archive. On a system with Gnu tar, it looks something like:

```
%> tar -xvzf sensplacetool-200x-xx-xx.tar.gz
```

The files have the following structure (the date may vary with release)

```
SensPlaceTool-200x-xx-xx/
  src/
  examples/
  doc/
```

To get started, copy the src-directory to a directory of your choice in your file system. Rename the library to e.g. sensplacetool and add the directory to Matlabs search path, e.g. using the addpath command in Matlab. This is it and now all things are installed. Try the predefined examples collected in the examples directory.

2 Functional documentation

To be done. All relevant Matlab functions are equipped with help texts. Look at and run the m-files in the examples directory to see how the package can be used.

3 Examples

This section is a brief description of the examples included in the `examples` directory.

Linear example

The Matlab file `linearsp.m` includes sensor placement analysis of the following linear ODE-model

$$\begin{aligned}
e_1 &: \quad \dot{x}_1 = -x_1 + x_2 + x_5 \\
e_2 &: \quad \dot{x}_2 = -2x_2 + x_3 + x_4 \\
e_3 &: \quad \dot{x}_3 = -3x_3 + x_5 + f_1 + f_2 \\
e_4 &: \quad \dot{x}_4 = -4x_4 + x_5 + f_3 \\
e_5 &: \quad \dot{x}_5 = -5x_5 + u + f_4
\end{aligned}$$

where x_i are the state variables, u a known control signal, and f_i the faults we want to detect and isolate.

Since, there are no specified sensors there is no redundancy in the model and the faults are not detectable. The analysis aims at finding sensor locations such that we both get detectability and full isolability among the faults f_i and also the faults in the new sensors.

DAMADICS

The file `damadicssp.m` includes a sensor placement analysis of the DAMADICS industrial valve model.

An example used to illustrate the software is an industrial valve. A schematic figure of the valve is shown in Figure 1 and consists of three main components: the control valve, a by-pass valve, and a spring-and-diaphragm pneumatic servo-motor to operate the valve plug. The figure also shows an internal control loop that is used to increase the accuracy of the valve plug positioning. Details of this model is not included in this presentation and interested readers are referred to e.g. [5] and the references therein. The structure of the model is derived in [2, 3] and is shown in Figure 2. Variables x, and x_h are valve positioning variables, P_s, P_1, P_2, P_z, P_v, Δ_p, Δ_{p-a}, are pressures, Q, Q_v, Q_{v3}, Q_c fluid flows, T_1 temperature, F_{vc} a force, and all f_i are variables indicating which equations the different faults influence.

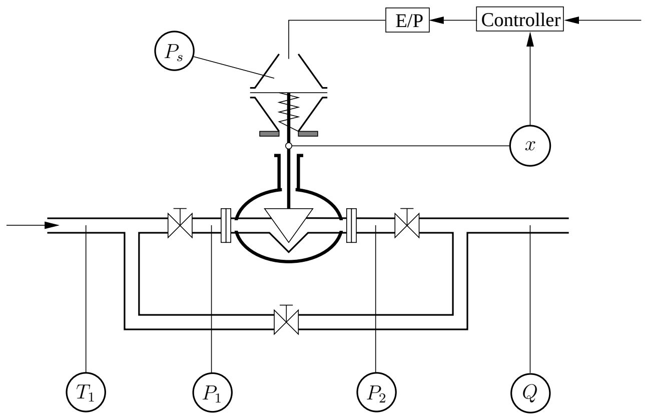

Figure 1: DAMADICS valve

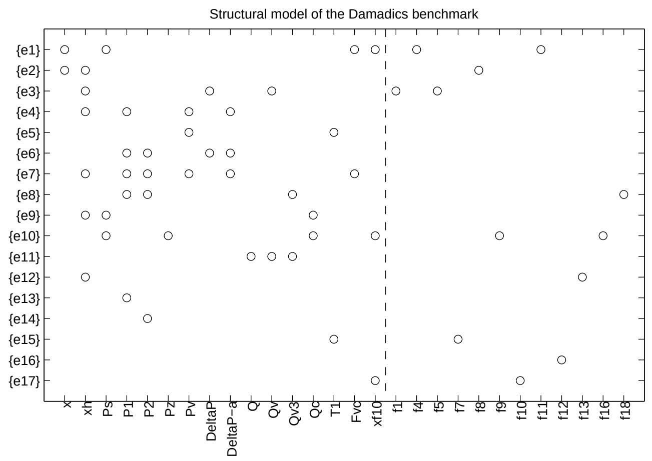

Figure 2: Structure of the DAMADICS valve model.

The original model included a specified set of sensors but since the objective here is to perform sensor placement analysis, almost all sensors has been removed. Three sensors has been kept, measurements of the two ambient pressures P_1, P_2, and the measurement of the valve position x that is used in the internal control loop. This leaves us with a model, which has no underdetermined part, consisting of 17 equations in 16 unknown variables and 12 different faults.

Commault et.al.

The paper [1] deals with a similar problem as this software. The file `commaultsp.m` performs sensor placement analysis on the example in that paper.

Raghuraj et.al.

Also the paper [4] deals with a similar problem as this software. The file `raghurajsp.m` defines both the CSTR and the reduced FCCU model and derives possible sensor locations.

4 Known bugs and misbehavior

Note that this is experimental software that has not been fully tested and there may be numerous bugs still left in the software.

At least the following things should be addressed when there is available time:

- About 70% of computing time is currently spent on doing string operations. A slight rewrite of the code avoiding string operations on cell-arrays would give a significant speed-up of the analysis.

References

[1] C. Commault, J. Dion, and S.Y. Agha. Structural analysis for the sensor location problem in fault detection and isolation. In *Proceedings of IFAC Safeprocess'06*, Beijing, China, 2006.

[2] Dilek Dustegör, Erik Frisk, Vincent Coquempot, Mattias Krysander, and Marcel Staroswiecki. Structural analysis of fault isolability in the DAMADICS benchmark. *Control Engineering Practice*, 14(6):597–608, 2006.

[3] Erik Frisk, Dilek Düştegör, Mattias Krysander, and Vincent Cocquempot. Improving fault isolability properties by structural analysis of faulty behavior models: application to the DAMADICS benchmark problem. In *Proceedings of IFAC Safeprocess 03*, Washington, USA, 2003.

[4] R. Raghuraj, M. Bhushan, and R. Rengaswamy. Locationg sensors in complex chemical plants based on fault diagnostic observability criteria. *AIChE*, 45(2):310–322, 1999.

[5] M. Syfert, M. Bartys, J. Quevedo, and R.J. Patton. Development and application of methods for actuator diagnosis in industrial control systems (damadics): A benchmark study. In *Proceedings of IFAC Safeprocess 03*, Washington, USA, 2003.