

Small tutorial to sword, written by O. Garpinger (150826)

This pdf-document gives a couple of examples on how to use the Matlab-based PID optimization software tool called **sword**. **sword** is an update of the previous version of the software tool, called **designpid**. This new version has been developed in Matlab 7.14 (R2012a) and it is both faster and includes more features than the previous version (e.g. PID design for process output disturbances and calculation of the H_2 -norm of the sensitivity function between noise and process input). It has, however, not been as thoroughly tested as the previous software. Therefore, please try the old version if this version does not work for you. If you have any comments, questions or bug reports regarding the software, I'll be happy to hear from you. You can reach me via the e-mail address *olof_garpinger(at)hotmail.com*, where you have to replace (at) with @. Since I no longer work at the Department of Automatic Control, Lund University, I am not sure if the software will be updated again in the future.

At the end of this document follows a list of publications where this software and its benefits have been documented.

1 Examples

Before you go through the tutorial examples, you should read the instructions related to the software, i.e. type

```
>> help sword
```

in Matlab. It also helps to read at least some of the publications listed in the end of this tutorial.

1.1 Example 1

Assume that we want to find a PID controller that gives optimal load disturbance attenuation at the process input, for the first order process model with time delay

$$P(s) = \frac{1}{s+1} e^{-s}.$$

This is a standard model used in e.g. the process which can easily be derived through for instance a step response test on the process. Let us keep all settings to default values (i.e. maximum sensitivities $M_s = M_t = 1.4$, filter time constant $T_f = 0.0001$, etc.). In Matlab, start out by defining the process:

```
>> s = tf('s')
>> P = 1/(s+1);
>> P.iodelay = 1
```

```
P =
      1
exp(-1*s) * -----
      s + 1
```

Continuous-time transfer function.

Then you can choose to run the Matlab function **sword.m** without use of the menu (default settings are used):

```
>> design = sword(P,0);
```

Nelder Mead iteration number: 10

Nelder Mead iteration number: 20
Final number of Nelder Mead iterations: 28

PID Parameters:
K-value: 0.6555
Ti-value: 0.9619
Td-value: 0.3971
Tf-value: 0.0001

IAE-value: 1.7297
H2-norm of Sc: 82904.2068

Total time for the algorithm to finish: 6.50 seconds

After just 6.50 seconds, we have our optimal PID controller and the struct called **design** holds all relevant information about it. Note that this controller gives a very noise sensitive closed-loop system (high H_2 -norm of the sensitivity function between noise and process input, i.e. $\|S_c\|_2$).

1.2 Example 2

Let us now change the default settings a bit, but still use the same process and cost function. We choose a less robust and less noise sensitive closed-loop system, i.e.:

$$\begin{aligned}M_s &= 1.8, \\M_t &= 1.8, \\T_f &= 0.1.\end{aligned}$$

We will also derive a PI controller instead of a PID. Then we can e.g. go through the menu:

```
>> design = sword(P,1);
-----
1. Change the optimization settings
2. Change the robustness settings
3. Set the frequency span
4. Change the low-pass filter
5. Graphic settings
6. Choose to design a PI or PID
7. Choose to design for input or output disturbances
8. Do not change anything else
Choose what settings to change: 2
-----
Robustness settings (default Ms=Mt=1.4, press return):
Set maximum gain of the sensitivity function, Ms>1: 1.8
Set maximum gain of the complementary sensitivity function, Mt>1: 1.8
-----
1. Change the optimization settings
2. Change the robustness settings
3. Set the frequency span
4. Change the low-pass filter
5. Graphic settings
6. Choose to design a PI or PID
7. Choose to design for input or output disturbances
8. Do not change anything else
Choose what settings to change: 4
```

```

-----
Low-pass filter (1/(0.5*(s*Tf)^2+s*Tf+1)) settings (default Tf=0.0001),
(1/(Tf*s+1) if PI controller):
Set low-pass filter time constant Tf: 0.1
-----
1. Change the optimization settings
2. Change the robustness settings
3. Set the frequency span
4. Change the low-pass filter
5. Graphic settings
6. Choose to design a PI or PID
7. Choose to design for input or output disturbances
8. Do not change anything else
Choose what settings to change: 6
-----
Choose to design a PI or PID controller (default N):
Do you wish to design a PI controller (Y/N)?: Y
-----
1. Change the optimization settings
2. Change the robustness settings
3. Set the frequency span
4. Change the low-pass filter
5. Graphic settings
6. Choose to design a PI or PID
7. Choose to design for input or output disturbances
8. Do not change anything else
Choose what settings to change: 8

Final number of Nelder Mead iterations: 2

PID Parameters:
K-value: 0.6438
Ti-value: 1.2014
Td-value: 0.0000
Tf-value: 0.1000

IAE-value: 1.8810
H2-norm of Sc: 1.6575

Total time for the algorithm to finish: 1.15 seconds

All relevant information about the controller can be found in the design struct. For example:
design.PID.Kfinal, design.PID.Tifinal, design.PID.Tdfinal, and design.PID.Grfinal
(controller and filter transfer function).

```

1.3 Example 3

There are five m-files included in the SWORD zip-file that demonstrate batch runs using **sword** (i.e. automated runs with more than one process and/or different settings). Four of them provides unfiltered, IAE-optimal, PI and PID controllers with respect to four different robustness choices ($M_s = M_t = 1.2, 1.4, 1.6, 1.8$):

- BatchrunM1_2_SWORD_PI_and_PID.m
- BatchrunM1_4_SWORD_PI_and_PID.m
- BatchrunM1_6_SWORD_PI_and_PID.m
- BatchrunM1_8_SWORD_PI_and_PID.m

The fifth batch run derives sets of IAE-optimal PI and PID controllers with respect to $M_s = M_t = 1.4$ and several different noise sensitivities:

- ControllerSets_SWORD_MsMt_1_4.m

You can read more about the last tuning method in O. Garpinger's PhD Thesis from 2015: *Analysis and Design of Software-Based Optimal PID Controllers* (Paper IV).

Finally, I wish you good luck with the software and hope you will find it as useful as I do!

Best Regards,

Olof Garpinger
Lund, Sweden
26/8-2015

Documentation of the SWORD design method for PID control

There are several publications that describes the SWORD software and design method. Here you can read how PI and PID controllers (with noise filters) can be designed with regards to the three criteria: *performance, robustness, and noise sensitivity*. Below follows a list of publications.

Garpinger, O. (2015). "Optimal PI and PID Parameters for a Batch of Benchmark Process Models Representative for the Process Industry". *Technical Report*, Department of Automatic Control, Lund University, Sweden.

Garpinger, O. (2015). *Analysis and Design of Software-Based Optimal PID Controllers*. PhD Thesis ISRN LUTFD2/TFRT-1105-SE, Department of Automatic Control, Lund University, Sweden.

Garpinger, O. and T. Hägglund (2015). "Software-based optimal PID design with robustness and noise sensitivity constraints". *Journal of Process Control* **33**:9, pp. 90–101.

Garpinger, O. and T. Hägglund (2014). "Modeling for Optimal PID Design". In: *19th IFAC World Congress*. Cape Town, South Africa.

Garpinger, O. and T. Hägglund, and L. Cederqvist (2012). "Software for PID Design: Benefits and Pitfalls". In: *IFAC Conference on Advances in PID Control*. Brescia, Italy.

Garpinger, O. (2009). *Design of Robust PID Controllers with Constrained Control Signal Activity*. Licentiate Thesis ISRN LUTFD2/TFRT-3245-SE. Department of Automatic Control, Lund University, Sweden.

Garpinger, O. and T. Hägglund (2008). "A Software Tool for Robust PID Design". In: *17th IFAC World Congress*. Seoul, South Korea.