

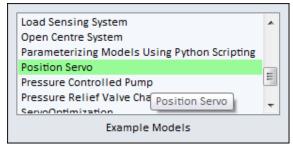
# Tutorial - Running an Optimization

# Using the optimization dialog

Numerical optimization is an important tool for automatically finding a desired parameter set for a model. Hopsan contains an optimization dialog from where optimizations can be controlled. This guide shows how to optimize control parameters for a simple hydraulic position servo using the Complex-RF algorithm. Optimization in Hopsan can either be performed from a dialog or by loading previously saved (or manually created) script files. This tutorial demonstrates both methods.

#### 1. Open the model

The Welcome tab has a box with example models. Open the Position Servo example model.



This model describes of a hydraulic position servo, where a PI-controller is used to control the position. This consists of a proportional gain  $(K_p)$  and an integrator gain  $(K_i)$ . The values of these gains will decide the speed and stability of the servo. It is thus desirable to find as good controller parameters as possible.

#### 2. Simulate

Simulate the model by clicking the simulate icon.

#### Simulate current project (Ctrl-Shift-S)

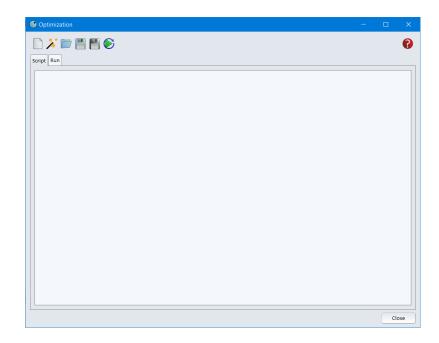
Now double click on the "Scope" component on top of the model. Two curves will be plotted, the reference signal and the actual position. The reference signal is a step, and the position shows the corresponding step response. As you can see, it is quite oscillative. We want to adjust the control parameters to reduce its rise time, overshoot and settling time by using the *optimization* feature.

#### 3. Open the dialog

Click the optimization icon to open the dialog.







The first page should look like above. You can either write an optimization script by hand, load a script file or start the script wizard. Let us begin with the latter. Click on the wizard icon.



#### 4. Optimization settings

The first wizard page will open. From here you can choose the general optimization settings.

Optimization		
Please choose general settings for optimization algorith	hm.	
Optimiation algorithm:	Simplex (Nelder-Mead)	
Number of iterations:	100	
Number of particles:	4	
Reflection coefficient:	1.3	
Expansion Factor	0.3	
Contraction factor:	-0.5	
Reduction factor:	0.5	
Number of models:	4	
Tolerance for parameter convergence:	0.0001	
Plot objective function values		
Plot particles		
Plot entropy		
Export trace data to CSV file		
Evaluate all points again after optimization (remove	es forgetting factor artifacts)	
	Next Abort	0

- Set Algorithm to Complex-RF
- Set Number of iterations to 1000
- Set Number of points to 4

Leave all other settings unchanged for now.



## 5. Choose optimization parameters

Next page controls the parameters in the model that shall be optimized.

				?	×
🗧 🕐 Optimization					
• Optimization					
Choose ontimization paramet	ers, and specify their minimum ar	nd maximum values			
Use logarithmic parameter		iu maximum values.			
Ose logarithmic parameter	scaling				
1					^
_System Parameters					
> 4_3_Servo_Valve					
> Add					
> Cylinder_C					
> Fixed_Position_Attachme	nt				
> Gainl					
> GainP					
> Hydraulic_Volume_Multi	Port				
> Integrator					
> Mass					
> Position_Sensor					
> Pressure_Relief_Valve					
> Q_type_Fixed_Displacem	ent_Pump_1				
> Scope					
> Sink					
> Step					
> Subtract					~
> Tank_C					*
Min Value	Parameter Name	Max Value			
			Next	<u>A</u> bort	0

Check the following parameters in the list:

- $\bullet \ \ \mathsf{Gainl} \to \mathsf{k} \# \mathsf{Value}$
- $\bullet \ \ \mathsf{Gain}\mathsf{P} \to \mathsf{k}\#\mathsf{Value}$

Then give them the following min and max values:

Parameter	Min	Max
GainI, k#Value	0	0.001
GainP, k#Value	0	0.04

#### 6. Choose objective functions

The optimization algorithms use objective functions to calculate how good a certain point in the parameter space is. From the third page this can be chosen. In this model we want the actual position to be as close as possible to the reference signal. To do this, check the following variables in the list:

- $\bullet \ \ \mathsf{Position\_Sensor} \to \mathbf{out} \to \mathsf{Value}$
- $\bullet \ \mathsf{Step} \to \mathbf{out} \to \mathsf{Value}$

Then choose "Minimize" and "Absolute Average Difference" in the boxes. Click "Add Function".



🕐 Optir	nization	
	jective function by first choosing variables in the list and then choosing a	function below.
1 _Alias > 4_3_Ser	vo Valve	
> Add > Cylinde	-	
> Gainl > GainP		
	ic_Volume_Multi_Port	
Minimize	Overshoot	Add Functi
Weight	Norm. Factor Exp. Factor Description	Data

Apart from minimizing the average difference, we also want to minimize the overshoot of the step response.

• Uncheck  $\textbf{Step} \rightarrow \textbf{out} \rightarrow \textsf{Value}$ 

Now choose "Minimize" and "Overshoot", and then click "Add Function". In the box called "arg1", change the value to  $0.7.\,$ 

Now we have added a second objective function. The total objective function will the sum of these. As they may not be equally important or equally scaled, it is possible to weight the individual functions. Change "Weight" for the first function to 10. Now the page should look like this:

> Scope > Sink ~ Step	_Fixed_Displacer	ment_Pump_1			
✓ out Minimize	 Overshoot			Add Functio	on
Weight	Norm. Factor	Exp. Factor	Description Minimize Absolute Average Difference for Position_Sensor, out, Value and Step, out, Value	Data	
1	1	2	Minimize Overshoot for Position_Sensor, out, Value	arg1 0.7	



## 7. Start the optimization

Next page shows the generated script code. Modifications can be made if necessary, but leave it as it is for now.

Optimization -		×
		?
Script* Run		
# Auto-generated HCOM script for complex algorithm optimization		^
<pre>fEvaluate function define setpars     echo off     chpa Gain1.k.y optpar(optvar(evalid),0)     chpa Gain1.k.y optpar(optvar(evalid),1)     echo on enddefine</pre>		
<pre>#Objective function define obj     echo off     #Absolute Average Difference     error = Position Sensor.out.y-Step.out.y     obj1 = aver(abs(error))</pre>		
<pre>fOvershoot maxVal = max(Position_Sensor.out.y) if(maxVal &gt; 0.7)</pre>		
echo on totalObj = +10*1*exp(2)*obj1+1*1*exp(2)*obj2		~
	CI	ose

Now change to the "run" tab. From here the execution of the optimization can be monitored:

Optimization -		×
Script <sup>*</sup> Run		?
Model file: Position Servo.hmf Script File: Generated Script		
obj		
	C	lose

Click on "Start Optimization".

Start Optimization

During the optimization the objective function values and the values of the optimization parameters are shown for the different points. The rows are sorted from best to worst.



Optin	nization											
) 🏄	i 🗁 💾											
ript*	Run											
lodel f	file: Position Se	ervo.hmf										
cript Fi	ile: Generated	Script										
		obj	par0	par1	Γ	1			2			
					L	-			2			
1	Apply	1.5233777	0.00038572181	0.0082703579	1	1 model0					10	0
2	Apply	1.524624	0.00037976183	0.00827187	1	2 total					7	%
3	Apply	1.5269086	0.00037216314	0.0082609288								
4	Apply	1.5347288	0.00038078606	0.0083716623								
<				>								
				/	L							-
		ar totalopj w			_							
						aded algorithm. Ly! Initializat	ion time: (	ms,	Simulation	time:	12	2
ms .			-	340003314.			ion time: (	) ms,	Simulation	time:	12	2
Assi		ar totalObj w:	- ith 1.53232		11	Ly! Initializat	ion time: (	) ms,	Simulation	time:	12	2
Assi [10:	45:33] Inf	o: In Position	-	single-three	11 ea	ly! Initializat. aded algorithm.						
Assi [10: [10: ms	45:33] Inf 45:33] Inf	o: In Position o: Simulated	- ith 1.53232 n_Servo; Using 'Position_Servo'	single-three	11 ea	ly! Initializat. aded algorithm.						
Assi [10: [10: ms Assi	45:33] Inf 45:33] Inf gning scal	o: In Position o: Simulated ar totalObj wi	- ith 1.53232 Servo; Using Position_Servo' ith 1.53753	single-thr 'successfui	11 ea 11	ly! Initializat ded algorithm. ly! Initializat						
Assi [10: [10: ms Assi [10:	45:33] Inf 45:33] Inf gning scal 45:33] Inf	o: In Position o: Simulated ar totalObj w: o: In Position	- ith 1.53232 h_Servo; Using 'Position_Servo' ith 1.53753 h_Servo; Using	single-thro successful single-thro	11 ea 11	Ly! Initializat uded algorithm. Ly! Initializat uded algorithm.	ion time: (	) ms,	Simulation	time:	12	2
Assi [10: [10: ms Assi [10:	45:33] Inf 45:33] Inf gning scal 45:33] Inf	o: In Position o: Simulated ar totalObj w: o: In Position	- ith 1.53232 Servo; Using Position_Servo' ith 1.53753	single-thro successful single-thro	11 ea 11	Ly! Initializat uded algorithm. Ly! Initializat uded algorithm.	ion time: (	) ms,	Simulation	time:	12	2
Assi [10: [10: ms Assi [10: [10: ms Assi	45:33] Inf 45:33] Inf gning scal 45:33] Inf 45:33] Inf 45:33] Inf	o: In Position o: Simulated ar totalObj w: o: In Position	- ith 1.53232 h_Servo; Using 'Position_Servo' ith 1.53753 h_Servo; Using 'Position_Servo'	single-thro successful single-thro	11 ea 11	ly! Initializat ded algorithm. ly! Initializat ded algorithm.	ion time: (	) ms,	Simulation	time:	12	2
Assi [10: [10: ms Assi [10: [10: ms	45:33] Inf 45:33] Inf gning scal 45:33] Inf 45:33] Inf 45:33] Inf	o: In Position o: Simulated ar totalObj w: o: In Position o: Simulated	- ith 1.53232 h_Servo; Using 'Position_Servo' ith 1.53753 h_Servo; Using 'Position_Servo'	single-thro successful single-thro	11 ea 11	ly! Initializat ded algorithm. ly! Initializat ded algorithm.	ion time: (	) ms,	Simulation	time:	12	2
Assi [10: [10: ms Assi [10: [10: ms Assi >>	45:33] Inf 45:33] Inf 45:33] Inf 45:33] Inf 45:33] Inf gning scal	o: In Position o: Simulated ar totalObj w: o: In Position o: Simulated	- ith 1.53232 1.Servo; Using 'Position_Servo' ith 1.53753 1.Servo; Using 'Position_Servo' ith 1.53473	single-thro successful single-thro	11 ea 11	y! Initializat. ded algorithm. ly! Initializat. ded algorithm. ly! Initializat.	ion time: (	) ms,	Simulation	time:	12	2
Assi [10: [10: ms Assi [10: [10: ms Assi >>	45:33] Inf 45:33] Inf 45:33] Inf 45:33] Inf 45:33] Inf gning scal	o: In Position o: Simulated ar totalObj w: o: In Position o: Simulated ar totalObj w:	- ith 1.53232 1.Servo; Using 'Position_Servo' ith 1.53753 1.Servo; Using 'Position_Servo' ith 1.53473	single-thra successful single-thra successful	11 ea 11	y! Initializat. ded algorithm. ly! Initializat. ded algorithm. ly! Initializat.	ion time: (	) ms,	Simulation	time:	12	2

#### 8. Evaluate the results

Once the optimization has finished, we can evaluate the results. Parameters from a point can be written back to the original model by using the "Apply" buttons.

	file: Position Se									
ript Fi	ile: Generated	Script	par0	par1		1		2		
1	Apply	1.4027695	0.00099941472	0.0080023889	1	model0		-		100
2	Apply	1.4027711	0.00099938628	0.0080020785	2	total				100
3	Apply	1.4027752	0.00099935215	0.0079985809						
4	Apply	1.4027694	0.00099937307	0.0079989123						
[10: ns Assi [10: [10: ns	47:06] Inf gning scal 47:06] Inf 47:06] Inf	o: Simulated ar totalObj w o: In Positio:	n_Servo; Using 'Position_Servo'	successfu	11 ea	y! Initializat ded algorithm.	tion time: 1 ms			
	47:06] Inf							, Simulation t	ime:	12

Click the "Apply" button for the point with green text (the best point). Then go back to the original model and click the Simulate button again.

## Simulate current project (Ctrl-Shift-S)

Now plot the step response by double clicking the scope component on top of the model. If a desirable result has not been found, it can be necessary to change objective functions or parameters and re-run the optimization.



# Optimizing from script files

It is often necessary to re-use the settings from the optimization more than once. This can be done by saving the optimization scripts to a file, and loading them again later.

1. Follow steps 1-7 above

#### 2. Saving a script file

Click on Save script file in the dialog.

#### 3. Loading a script file

To load a previously saved script file, click on Load script file in the dialog. You will be taken directly to the code tab. Go to the "run" tab and start the optimization as before. Note that the settings on the first three pages are not updated by the script. Running the wizard again will re-generate the script and overwrite the loaded code.

#### 4. Loading a script file from command terminal

It is also possible to load a script file from the command terminal. Just write "exec" followed by the path to the file. Hopsan will automatically understand that this is an optimization script and start the dialog. You can use "pwd" to show current directory and "cd" to change directory if necessary. Example: >> exec ...\folder\myscript.hcom