
GECKO
SIMULATIONS

Control of a 3-Phase Rectifier in Simulink

GeckoCIRCUITS is a stand-alone circuit simulator optimized for power electronics. Standard control can be implemented easily in GeckoCIRCUITS using the built-in library. Sophisticated or complex control – e.g. as typically realized in a DSP – can be set up using the JAVA block of GeckoCIRCUITS.

Sometimes it is desired to model parts of the control or the whole control structure in simulation software optimized for control tasks.

GeckoCIRCUITS can be coupled with other simulation software, e.g. with MATLAB® / Simulink®.

In this tutorial we show how to do the control of a 3-Phase AC/DC Vienna Rectifier in a Simulink-GeckoCIRCUITS coupled simulation.

The power circuit is modelled in GeckoCIRCUITS, with the option to perform transient thermal simulation, e.g. junction temperatures of power semiconductors, there, as well.

Voltages and currents are measured in GeckoCIRCUITS, and sent to Simulink, where Simulink generates the switching signals for the power semiconductors, which are passed to GeckoCIRCUITS' gate drivers.

This is how it works:

- Java 1.6 (or higher) has to be the actual Java version in the MATLAB environment
- GeckoCIRCUITS will be defined as a Java object in MATLAB
- Simulink controls the simulation – defining the simulation duration and numerical step within the model file (.mdl) of the control
- Simulink calls GeckoCIRCUITS via an S-function. The S-function is executed as a "sampled" block, where the sampling time is determined by the simulation time step width as specified in Gecko-CIRCUITS.
- The S-function is included in the Simulink model as a masked block. The mask is configured to set the necessary parameters for accessing GeckoCIRCUITS.
- In the circuit model, blocks are defined which interface signals from Simulink to GeckoCIRCUITS and vice-versa.

This tutorial was implemented and tested on MATLAB release 2010b. Older releases have a different S-function interface and may not work with the given Simulink model.
Before proceeding make sure that Java 1.6 is the actual Java version in the MATLAB environment:

```
>> version -java

ans =
Java 1.6.0 with Sun
Microsystems Inc. Java
HotSpot(TM) Client VM mixed
mode
```

In the Help Directory of MATLAB you will find a description of how to install a newer Java version if necessary.

This tutorial assumes a Windows operating system. Nevertheless, the tutorial content also applies to Linux or Mac OS X systems, where small modifications, such as different path names (e.g. "/" path separator instead of "\") have to be used.

How it works
Is Java 1.6 available in MATLAB?
GeckoCIRCUITS as a variable

We assume that you have created a folder for GeckoCIRCUITS and a different folder for your simulation models. In this tutorial we assume the folder's name as

C:/GeckoCIRCUITS/
C:/GeckoSimulink/

GeckoCIRCUITS.jar in C:/GeckoCIRCUITS/ is the (executable) circuit simulator with its libraries in lib/ and a valid license gecko.lic. Double-click the file GeckoCIRCUITS.jar to make sure that the simulator works – a window showing the workspace should open. If you're doing this the first time, you will be asked in a license dialog to submit a valid license file, e.g. gecko.lic.

The model directory C:/GeckoSimulink/ should contain the following files

vr1_simulink.ipes
Gecko_VR1.mdl
s_GeckoSimulink.mex*
Gecko.jpg

vr1_simulink.ipes is the model of the power electronics circuit, and Gecko_VR1.mdl is MATLAB/Simulink's model of the control.

s_GeckoSimulink.mex* are the S-function libraries which are used by Simulink to communicate with GeckoCIRCUITS.

The image file Gecko.jpg is displayed as icon in the Simulink model.

First, we have to tell MATLAB the location of GeckoCIRCUITS. This is done by editing MATLAB's classpath.txt file. Open the file with >> edit classpath.txt and append the path to the GeckoCIRCUITS .jar file:

C:/GeckoCIRCUITS/GeckoCIRCUITS.jar

Please note that by default, the file permission is set to read-only. Therefore you may need administrator rights to change the file permission.

If you don't have the permission to change the file classpath.txt, you can also define the path locally for your MATLAB session to a "dynamic class path". Therefore, execute the following command:

>> javaaddpath 'C:/GeckoCIRCUITS/GeckoCIRCUITS.jar'

You can view and undo your changes to the dynamic class path employing the commands

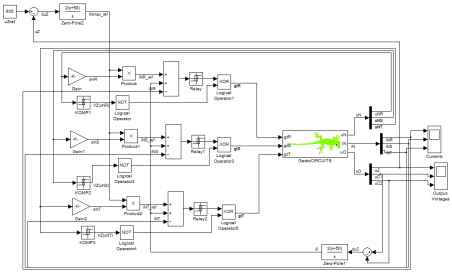
>> javaclasspath
>> javarmpath

Getting started
Files involved in this tutorial
Setting up GeckoCIRCUITS / Simulink Filepaths

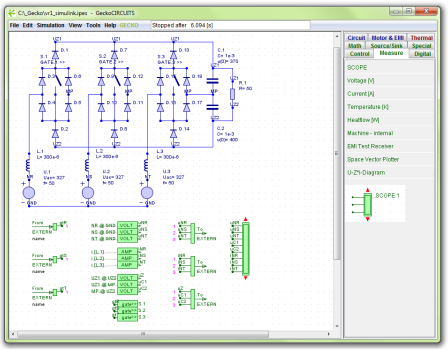
Control of a 3-Phase Rectifier in Simulink

When you have set the static class path by editing classpath.txt, then you must re-start MATLAB to apply the changes. The dynamic class path setting, however, will only persist during your current MATLAB session.

After setting up the file paths, change into the model folder and open the Simulink model Gecko_VR1.mdl.



Please note the S-function block with the green Gecko icon, which is the interface for accessing GeckoCIRCUITS. Double-clicking the Gecko-Icon will open a mask dialog where you can specify the path to the GeckoCIRCUITS model file:
C:/GeckoSimulink/vr1_simulink.ipes
This file path will be used at start-up to load the corresponding GeckoCIRCUITS model.
If GeckoCIRCUITS does not open automatically, you should check if the path-name to GeckoCIRCUITS in classpath.txt is set properly.
In case the Gecko Icon does not show up within Simulink, please double-check if the MATLAB environment contains the path to the GeckoCIRCUITS/Simulink folder.



You can view the current MATLAB path settings by typing

>> path

Adding the GeckoSimulink path to the Matlab environment is easily performed by

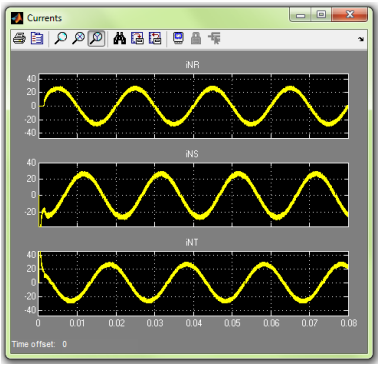
>> path('C:/GeckoSimulink', path)

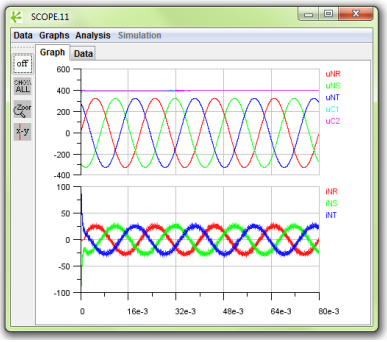
When the settings are correct, the Gecko main window will open together with the Simulink model. Closing the Gecko main window will not remove it from MATLAB. Therefore GeckoCIRCUITS never exits when it is called from Simulink, but only minimizes its main window when you attempt to close it. If you want to exit or restart GeckoCIRCUITS, you have to close MATLAB.

Getting started
Files involved in this tutorial
Setting up GeckoCIRCUITS / Simulink Filepaths

The Simulink model is now ready for simulation. Please start the simulation by choosing simulation > start in the Simulink menu.

You can visualize simulation results via the SCOPEs in GeckoCIRCUITS or, alternatively, via the Scopes in Simulink. In the shown screenshots, both scopes are active and show equal results (sinusoidal input currents).





In GeckoCIRCUITS only SCOPE input data is stored in the memory for the whole simulation. The more SCOPEs or SCOPE input ports you have in your model, the more data has to be stored. In case of long simulation periods with small numerical time steps there might be Java memory problems. In this case GeckoCIRCUITS compresses the data which might result in data loss.
In the following, we describe how to increase the Java memory size to circumvent possible memory problems.

Getting started
Open GeckoCIRCUITS user interface from Simulink
Data visualization

With the following command, you can check the available MATLAB memory:

```
>> memory

Maximum possible array:          333 MB
Memory available for all arrays: 894 MB
Memory used by MATLAB:           509 MB
Physical Memory (RAM):          1014MB
```

Total memory available for MATLAB is in this example 1014MB RAM. The default memory for Java is just 128MB.

You can increase the memory which is available for Java / GeckoCIRCUITS by creating a file named java.opts with a text editor. The file has the single line

```
-Xmx512m
```

as content. This increased the memory for Java to 512MB but reduces accordingly the remaining memory for the MATLAB workspace.

You must save this file java.opts into the win32/ folder of MATLAB. In our example this is the directory

```
C:\Programs\MATLAB\R2008a\bin\win32
```

Now the memory available for Java (and GeckoCIRCUITS) is permanently increased. You can test this by typing

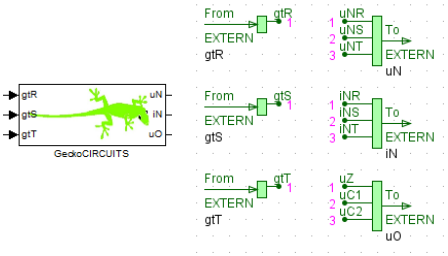
```
>>java.lang.Runtime.getRuntime.maxMemory

ans =
  523501568
```

Alternatively, you can build a Gecko-CIRCUITS model without any SCOPEs. In this case, there is no data storage in the Java environment, minimum memory consumption by GeckoCIRCUITS. All data-processing and memory issues are then handled in Simulink.

Not enough memory for Java?
How to increase Java's memory size in MATLAB

Control of a 3-Phase Rectifier in Simulink

In the GeckoCIRCUITS simulation model, the To-EXTERN / From-Extern blocks of GeckoCIRCUITS' library Source/Sink are used to communicate with Simulink.



The signal connections between Gecko-CIRCUITS and Simulink are realized via the interface port numbering, and not by the signal names. You are free to choose different signal names in both programs.

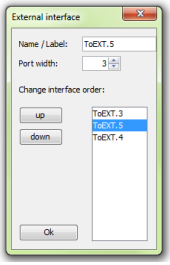
To avoid confusion on how the interface blocks are connected to Simulink, you can use signal names within Simulink. The signal names are displayed at the Gecko-CIRCUITS S-function mask, as well as below the From EXTERN and To EXTERN blocks within GeckoCIRCUITS, as shown in the screenshot above.

All input and output signals to and from GeckoCIRCUITS are of Double-type. In our example, the generated gate signals are of Boolean type. Therefore, the S-function performs an automatic type conversion for interfacing GeckoCIRCUITS. Please note that only Double or Boolean input signals are allowed to the input of the S-function. If you have different input data types in your own model, you can use the type conversion blocks that are available in the Simulink block library.

You can use scalar input signals, as well as signal busses with arbitrary width for every interface block. Please consider that all Simulink input signal variables in the same bus have to have an identical data type.

When the bus width for a specific interface port in GeckoCIRCUITS is different from the corresponding Simulink signal width, Simulink shows an error message at simulation start.

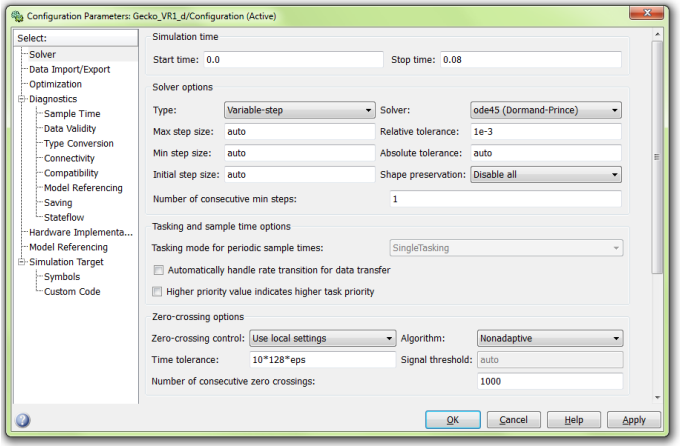
When you double-click an interface block in GeckoCIRCUITS, the following dialog will appear:



Here you can set the interface port width. Furthermore, you are able to change the interface order via the shown list, which then also represents the order of the Simulink input or output signals.

Further technical information

GeckoCIRCUITS "to/from EXTERN"
Input and output data types
Connections via port numbers / ordering



Open the Configuration Parameters dialog from the menu Simulation >> Parameters in the Simulink window.

In this dialog, you are able to define parameters as start- and end-times of the simulation as well as many solver options.

GeckoCIRCUITS uses its own simulation step width, which is independent from the Simulink settings. The S-function will be executed as sampled block, where the sampling rate is set by the Gecko step width.
The coupling between GeckoCIRCUITS and Simulink works properly with most settings, for instance a variable step ode45-Solver. However, when you specify a fixed step-width, the chosen value must be smaller than the step-width as within GeckoCIRCUITS.

If Simulink encounters any problems with the solver settings, it will show an error message popup-window. For debugging such errors, we recommend to start with a fixed time-step solver that has the same step width as set in GeckoCIRCUITS.
If Simulink gives an error message that it encounters an algebraic loop, the option automatically handle rate transition can help to resolve the problem.

Configuration Parameters in Simulink
Stop time
Simulation time-step
Solver options

Since version 1.4 of GeckoCIRCUITS, it is possible to access or modify all circuit parameters within GeckoCIRCUITS from MATLAB/Simulink. For details, please see the file GeckoSCRIPT.pdf in your GeckoCIRCUITS folder.

The following functions are available to receive or set circuit parameters:

```
writeOutput(String toBeWritten)
writeOutputLn(String toBeWritten)
setParameter(String elementName, String
parameterName, double value)
void setParameters(String elementName,
String[] parameterNames, double[] values)
double getOutput(String elementName, String
outputName)
double getOutput(String elementName)
String[] getCircuitElements()
String[] getControlElements()
String[] getThermalElements()
String[] getIGBTs()
String[] getDiodes()
String[] getThyristors()
String[] getIdealSwitches()
String[] getResistors()
String[] getInductors()
String[] getCapacitors()
```

Please note that these functions are available in the package gecko.GeckoExternal.*
As an example, you could execute the following command at the MATLAB prompt, to set the inductance value of the model vr1_simulink.ipes to a higher value:

```
>> gecko.GeckoExternal.setParameter('L.1','L',4e-4)
```

Please note that shown the syntax is upper case sensitive.

You can include this kind of command to M-files, Simulink S-functions, or a "MATLAB Fcn"-block within Simulink. Since writing the fully qualified function calls, including gecko.GeckoExternal, is quite cumbersome when you program more complex scripts, you can also import the class path:

```
>> import('gecko.GeckoExternal.*')
```

Then, as an example, you could print the value of the voltage measurement in your GeckoCIRCUITS model using the simplified command

```
>> getOutput('VOLT.2')
        ans = -283.1903
```

Setting and modifying simulation parameters can be performed before the simulation start, or alternatively during running simulations.
However, please notice that the change of some parameter values during a simulation run does not always make sense. For instance, to change a capacitance value at runtime will not model a nonlinear capacitor since charge conservation is violated in the simulation model (however, a correct nonlinear capacitor model is available in GeckoCIRCUITS).
The modification of passive elements (resistors), voltage source and current source values or several control parameters is possible without any problems.

Accessing Circuit Parameters from MATLAB/Simulink

Control of a 3-Phase Rectifier in Simulink

In this tutorial, a step-by-step introduction was given on how to couple the Gecko-CIRCUITS power electronics simulator with MATLAB and Simulink. The tutorial demonstrated a Vienna rectifier circuit in GeckoCIRCUITS, where the control operation was performed in Simulink.

This tutorial example gives a good starting point to the reader, who wants to implement sophisticated control in Simulink and couple it with our high performance circuit solver.

If you want to include GeckoCIRCUITS S-functions into your own model, we recommend you to simply "cut and paste" the Gecko S-function block from the given example file Gecko_VR1.mdl to your own Simulink model.

Contact Information / Feedback

Dr. Andreas Müsing

Gecko-Simulations AG
Physikstrasse 3, ETL H13
CH-8092 Zürich, Switzerland

Phone +41-44-632 4265
Fax +41-44-632 1212

www.gecko-simulations.com
contact@gecko-simulations.com

Document version: February 2014

Summary
Contact information