

# SIMULINK<sup>®</sup>

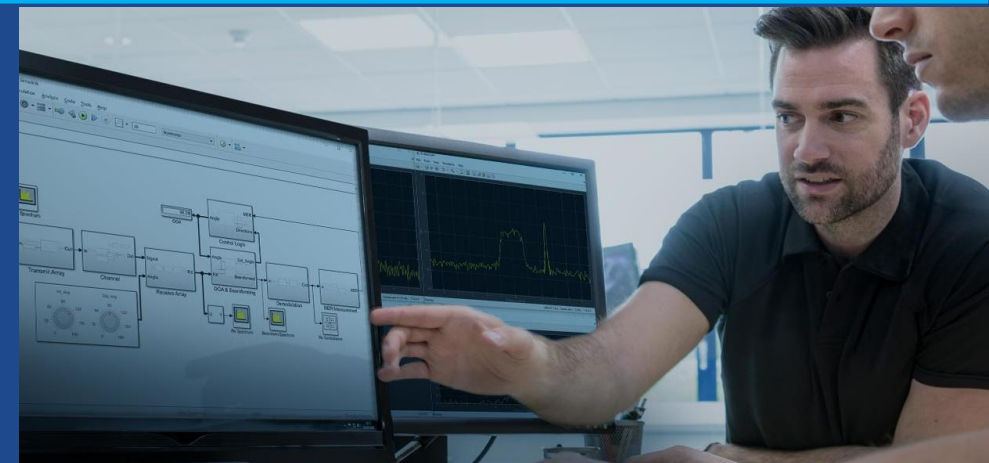
## Signal Processing with Simulink for particle accelerators and RF systems



Charbel CHERFAN

Customer Success Engineer, PhD  
[ccherfan@mathworks.com](mailto:ccherfan@mathworks.com)

April 2026



# Motivation and Context

## CERN Super Proton Synchrotron: Modeling, design optimization and performance stability

### Context:

- CERN SPS accelerates protons from from 26 to 450 GeV/c before injection into the LHC
- Beam intensity at SPS directly impacts LHC luminosity
- At high intensity, the RF and LLRF systems become limiting factor

### Key physical effects at high intensity: :

- Transient beam loading
- Phase perturbations
- Beam losses during injection/extraction
- Increased sensitivity to instabilities

[link to the paper](#)

PHYSICAL REVIEW ACCELERATORS AND BEAMS **25**, 021002 (2022)


### CERN's Super Proton Synchrotron 200 MHz cavity regulation upgrade: Modeling, design optimization, and performance estimation

T. Mastoridis<sup>✉</sup>

*California Polytechnic State University, San Luis Obispo, California 93407, USA*

P. Baudrenghien

*CERN, Geneva 1211, Switzerland*

 (Received 4 October 2021; accepted 2 February 2022; published 17 February 2022)

CERN's Super Proton Synchrotron (SPS) accelerates protons to 450 GeV/c and transfers them into the Large Hadron Collider (LHC). It is currently one of the limiting factors in increasing the beam intensity and thus the luminosity of the LHC. As part of the LHC injectors upgrade project, the SPS 200 MHz rf system has been modified during the CERN long shutdown 2 (January 2019–April 2021), resulting in a new layout containing two additional cavities. The goal is to improve longitudinal stability required for the planned doubling of the beam intensity for the high luminosity LHC. In parallel with the upgrade of the high-power rf, a new low-level rf (LLRF) system has been designed, including a new cavity field regulation system. This work presents a model of the beam-rf interaction which includes a detailed representation of the LLRF controlling the cavity. This model is used to determine the optimal LLRF design for maximum loop stability and beam loading compensation. Finally, the performance of the upgraded LLRF is estimated.

DOI: [10.1103/PhysRevAccelBeams.25.021002](https://doi.org/10.1103/PhysRevAccelBeams.25.021002)

# What changed & Why It matters

## SRS RF upgrade (200 MHz system)

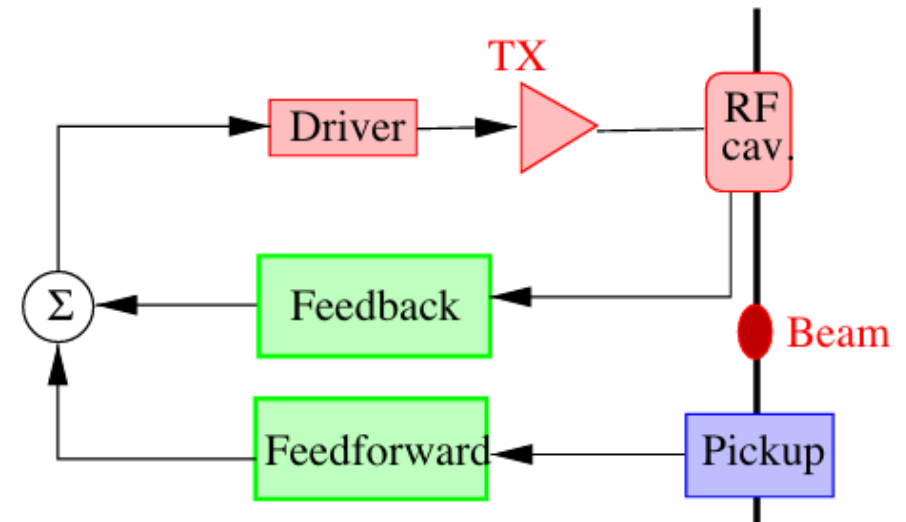
- New RF layout with additional cavities
- **Goal:** Improve longitudinal beam stability at high intensity

## LLRF Redesign:

- New cavity field regulation
- Improved beam-loading compensation
- Required because beam loading effects grow rapidly with intensity

## Critical Insight:

- System Performance is not limited by a single component hardware alone
- A system-level, closed-loop simulator is required



# DigitalGlobe Stimulates Complete Satellite-To-Ground Communication Systems

## Challenge

- Ensure rapid and reliable transmission of satellite imagery data from space to ground

## Solution

- Use Simulink to model the entire RF and digital communications system and perform simulations to calculate BER and verify link performance

## Results

- Simulation speed increased tenfold
- Distortion effects included in simulation
- Data rate increased by 50%



[DigitalGlobe Simulates Complete Satellite-to-Ground Communications Systems](#)

# Agenda

- Get started with RF data analysis
- RF system design
- Signal Processing with Simulink
- Integration custom or external code into Simulink models

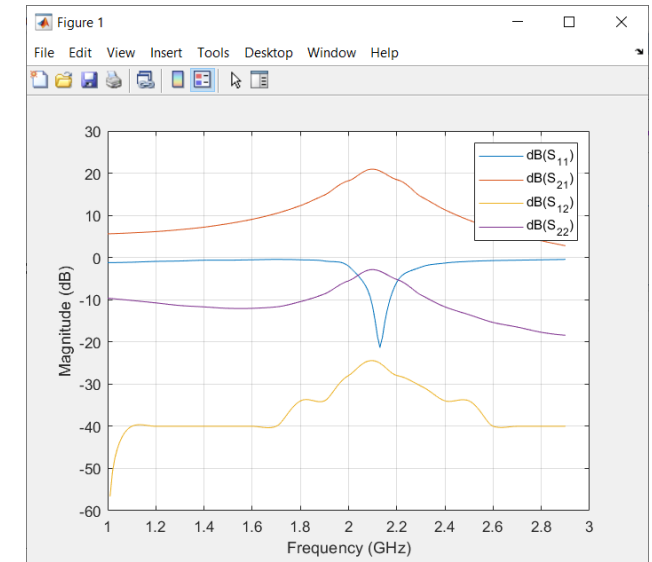
# S-Parameters Data Import and Visualization

```
help rf
```

```
s_param = sparameters('default.s2p')
```

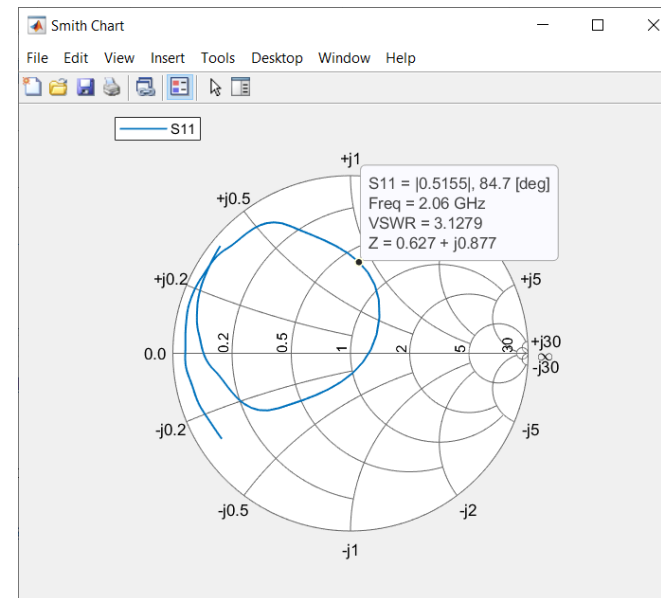
```
rfplot(s_param)
```

```
smithplot(s_param, 1, 1)
```



## Analysis

- [abcd2h](#) - Convert ABCD-parameters to hybrid H-parameters
- [abcd2s](#) - Convert ABCD-parameters to S-parameters
- [abcd2y](#) - Convert ABCD-parameters to Y-parameters
- [abcd2z](#) - Convert ABCD-parameters to Z-parameters
- [g2h](#) - Convert hybrid G-parameters to hybrid H-parameters
- [h2abcd](#) - Convert hybrid H-parameters to ABCD-parameters
- [h2g](#) - Convert hybrid H-parameters to hybrid G-parameters
- [h2s](#) - Convert hybrid H-parameters to S-parameters
- [h2y](#) - Convert hybrid H-parameters to Y-parameters
- [h2z](#) - Convert hybrid H-parameters to Z-parameters
- [snp2smp](#) - Convert N-port S-parameters to M-port S-parameters
- [s2t](#) - Convert S-parameters to T-parameters
- [t2s](#) - Convert T-parameters to S-parameters
- [rlgc2s](#) - Convert transmission line RLGC-parameters to S-parameters
- [s2rlgc](#) - Convert S-parameters to transmission line RLGC-parameters
- [s2abcd](#) - Convert S-parameters to ABCD-parameters
- [s2h](#) - Convert S-parameters to hybrid H-parameters
- [y2abcd](#) - Convert Y-parameters to ABCD-parameters
- [y2h](#) - Convert Y-parameters to hybrid H-parameters
- [z2abcd](#) - Convert Z-parameters to ABCD-parameters
- [z2h](#) - Convert Z-parameters to hybrid H-parameters
- [rationalfit](#) - Perform rational function fitting to broadband data
- [iscausal](#) - Check causality of N-port S-parameters



# Programmatically Construct an RF Network: Automate the Analysis, Explore the design space, Create Reports

Follow four steps:

- Create a **circuit**
- Add the components
  - Defines nodes and connections
  - RLC, S-parameters, PCB components
- Define the analysis ports
- Perform the analysis and extract results

## Programmatically Construct Circuit

Create a `circuit` and use the `add` function to populate the circuit with named `resistor` and `capacitor` object:

```

ckt = circuit('crosstalk');

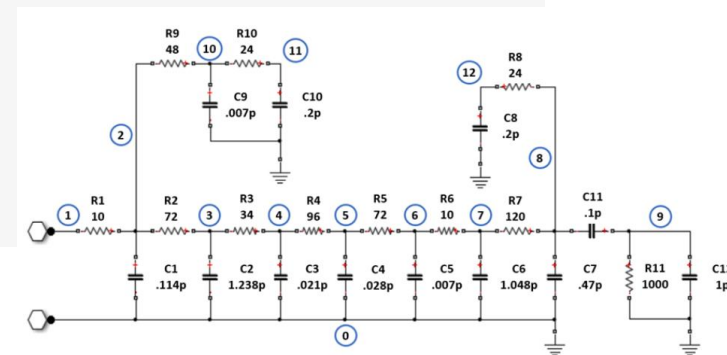
add(ckt,[2 1],resistor(10,'R1'))
add(ckt,[2 0],capacitor(0.114e-12,'C1'))
add(ckt,[3 2],resistor(72,'R2'))
add(ckt,[3 0],capacitor(1.238e-12,'C2'))
add(ckt,[4 3],resistor(34,'R3'))
add(ckt,[4 0],capacitor(0.021e-12,'C3'))
add(ckt,[5 4],resistor(96,'R4'))
add(ckt,[5 0],capacitor(0.028e-12,'C4'))
add(ckt,[6 5],resistor(72,'R5'))
add(ckt,[6 0],capacitor(0.007e-12,'C5'))
add(ckt,[7 6],resistor(10,'R6'))
add(ckt,[7 0],capacitor(1.048e-12,'C6'))
add(ckt,[8 7],resistor(120,'R7'))
add(ckt,[8 0],capacitor(0.47e-12,'C7'))

add(ckt,[12 8],resistor(24,'R8'))
add(ckt,[12 0],capacitor(0.2e-12,'C8'))

add(ckt,[10 2],resistor(48,'R9'))
add(ckt,[10 0],capacitor(0.007e-12,'C9'))
add(ckt,[11 10],resistor(24,'R10'))
add(ckt,[11 0],capacitor(0.2e-12,'C10'))

add(ckt,[9 8],capacitor(0.1e-12,'C11'))
add(ckt,[9 0],resistor(1000,'R11'))
add(ckt,[9 0],capacitor(1e-12,'C12'))

```



# Example#1: Create and Analyze a Network Using RF Measured Data

```
% Read the data
adapter1 = sparameters('coax.s1p');
elbow     = sparameters('elbow.s2p');
coupler   = sparameters('coupler.s4p');
adapter2  = sparameters('adapter.s2p');
% Build the network
ckt = circuit('circuit');
add(ckt, [1 2], adapter1);
add(ckt, [2 3], elbow);
add(ckt, [3 4 5 6], coupler);
add(ckt, [5 0], resistor(50));
add(ckt, [6 0], resistor(50));
add(ckt, [4 7], adapter2);
% Analyze the network
setports(ckt, [1 0], [7 0])
freq = linspace(347e6, 357e6, 51);
S = sparameters(ckt, freq);
%% Save to Touchstone file
rfwrite(S, 'line.s2p');
```

Import Touchstone files

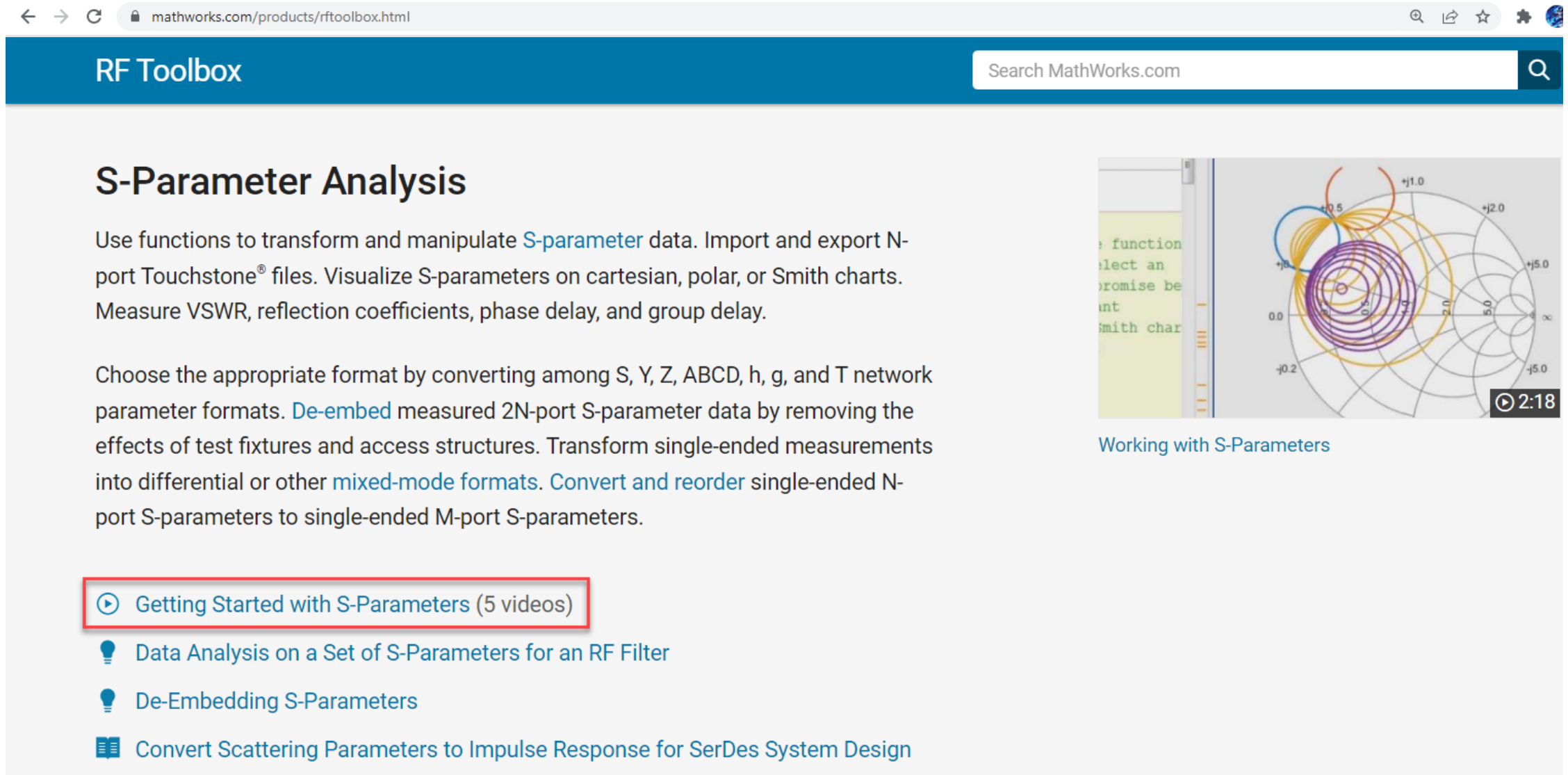
Create circuit object

Add network of components

Define ports and perform the analysis

Save and share results

# How Can I Get Started with S-Parameter Analysis in MATLAB?



The screenshot shows the MathWorks RF Toolbox website. At the top, there is a navigation bar with the "RF Toolbox" logo on the left and a search bar on the right. Below the navigation bar, the main content area features a section titled "S-Parameter Analysis". This section includes a paragraph describing the use of functions to transform and manipulate S-parameter data, and another paragraph detailing the conversion of network parameter formats. To the right of the text is a video player showing a Smith chart with a video duration of 2:18. Below the text, there is a list of video resources, with the first item, "Getting Started with S-Parameters (5 videos)", highlighted with a red border.

mathworks.com/products/rftoolbox.html

RF Toolbox

Search MathWorks.com

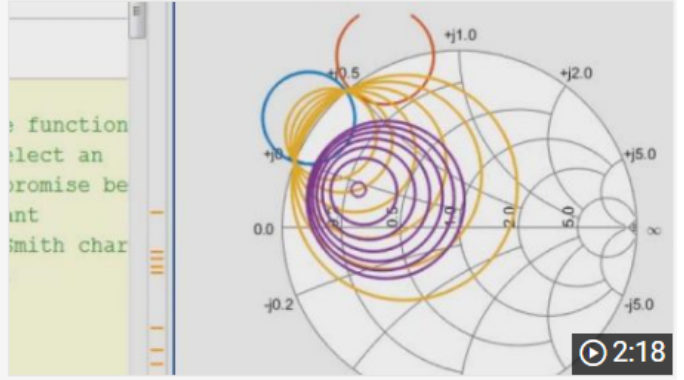
## S-Parameter Analysis

Use functions to transform and manipulate [S-parameter](#) data. Import and export N-port Touchstone® files. Visualize S-parameters on cartesian, polar, or Smith charts. Measure VSWR, reflection coefficients, phase delay, and group delay.

Choose the appropriate format by converting among S, Y, Z, ABCD, h, g, and T network parameter formats. [De-embed](#) measured 2N-port S-parameter data by removing the effects of test fixtures and access structures. Transform single-ended measurements into differential or other [mixed-mode formats](#). [Convert and reorder](#) single-ended N-port S-parameters to single-ended M-port S-parameters.

▶ Getting Started with S-Parameters (5 videos)

- 💡 Data Analysis on a Set of S-Parameters for an RF Filter
- 💡 De-Embedding S-Parameters
- 📄 Convert Scattering Parameters to Impulse Response for SerDes System Design

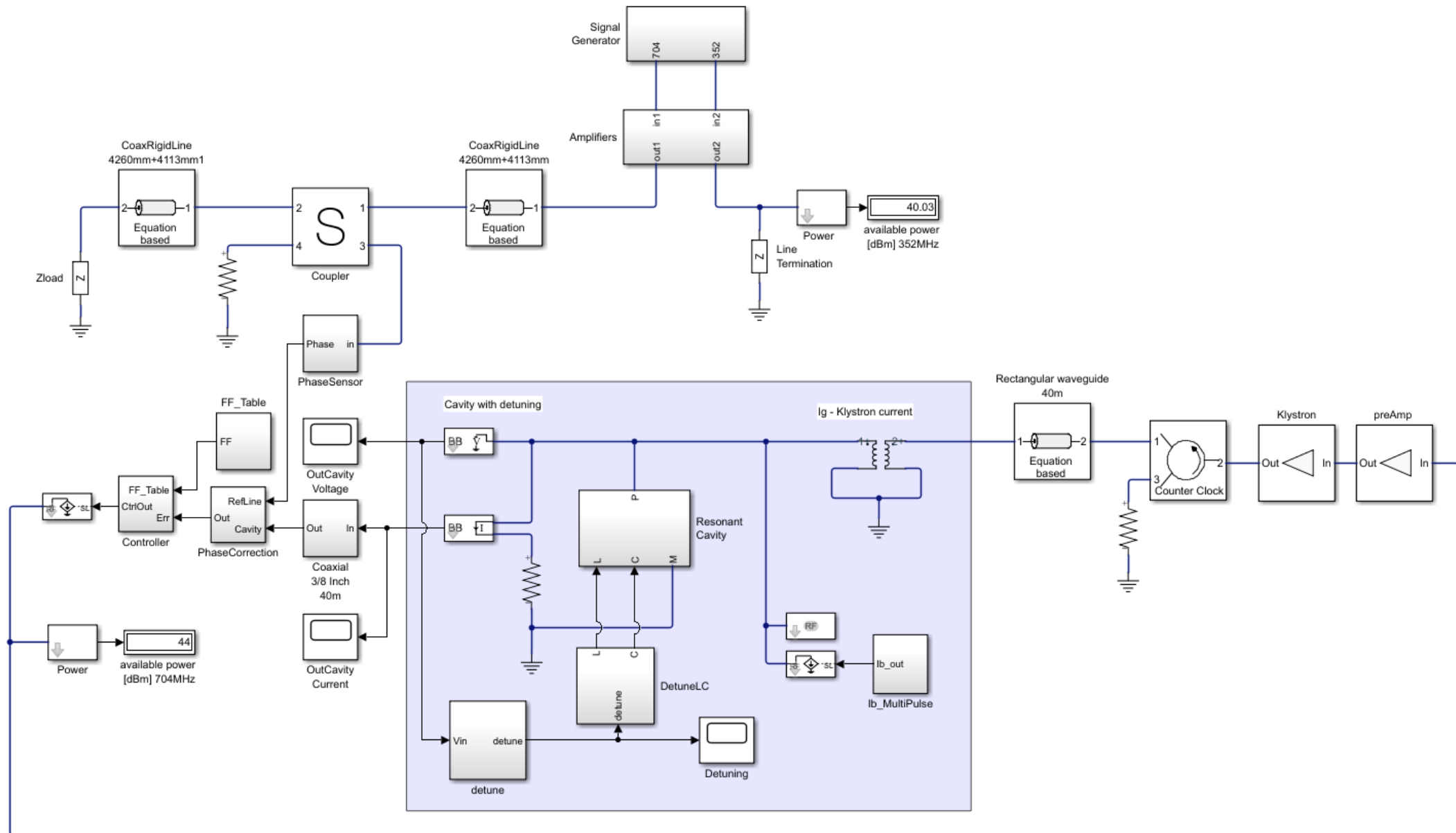


Working with S-Parameters

# Agenda

- Get started with RF data analysis
- **RF system design**
- Signal Processing with Simulink
- Integration custom or external code into Simulink models

# Inspiration: Cavity Resonator Model for Linear Particle Accelerator



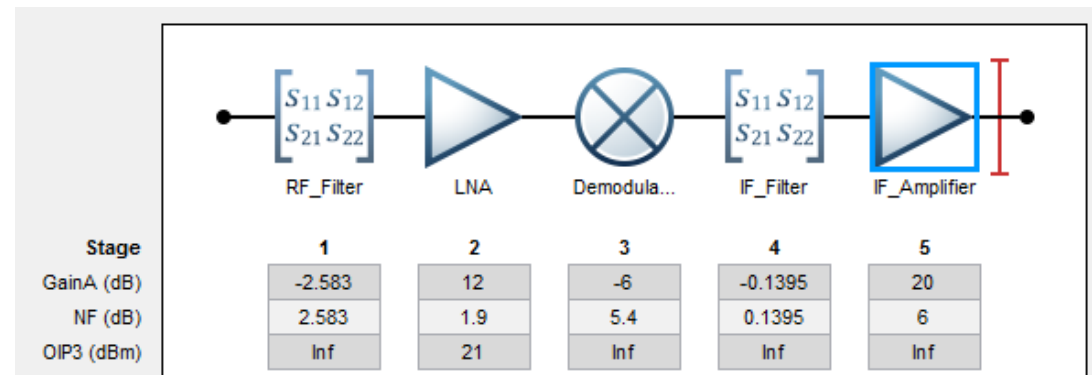
# RF Design – Where To Start?

## RF Budget Analyzer App



RF Budget  
Analyzer

- Analytical computation and visualization of power/noise/IP3 RF budget
- Improvement over custom-made spreadsheets - taking into account mismatches
- Generates Circuit Envelope models/testbenches for super-heterodyne and homodyne architectures
- Generates MATLAB scripts for automation and complex scenario analysis
- Delivers consistent results between analytical equations and simulation



# Analyze, Understand, and Debug RF Data and RF Budget

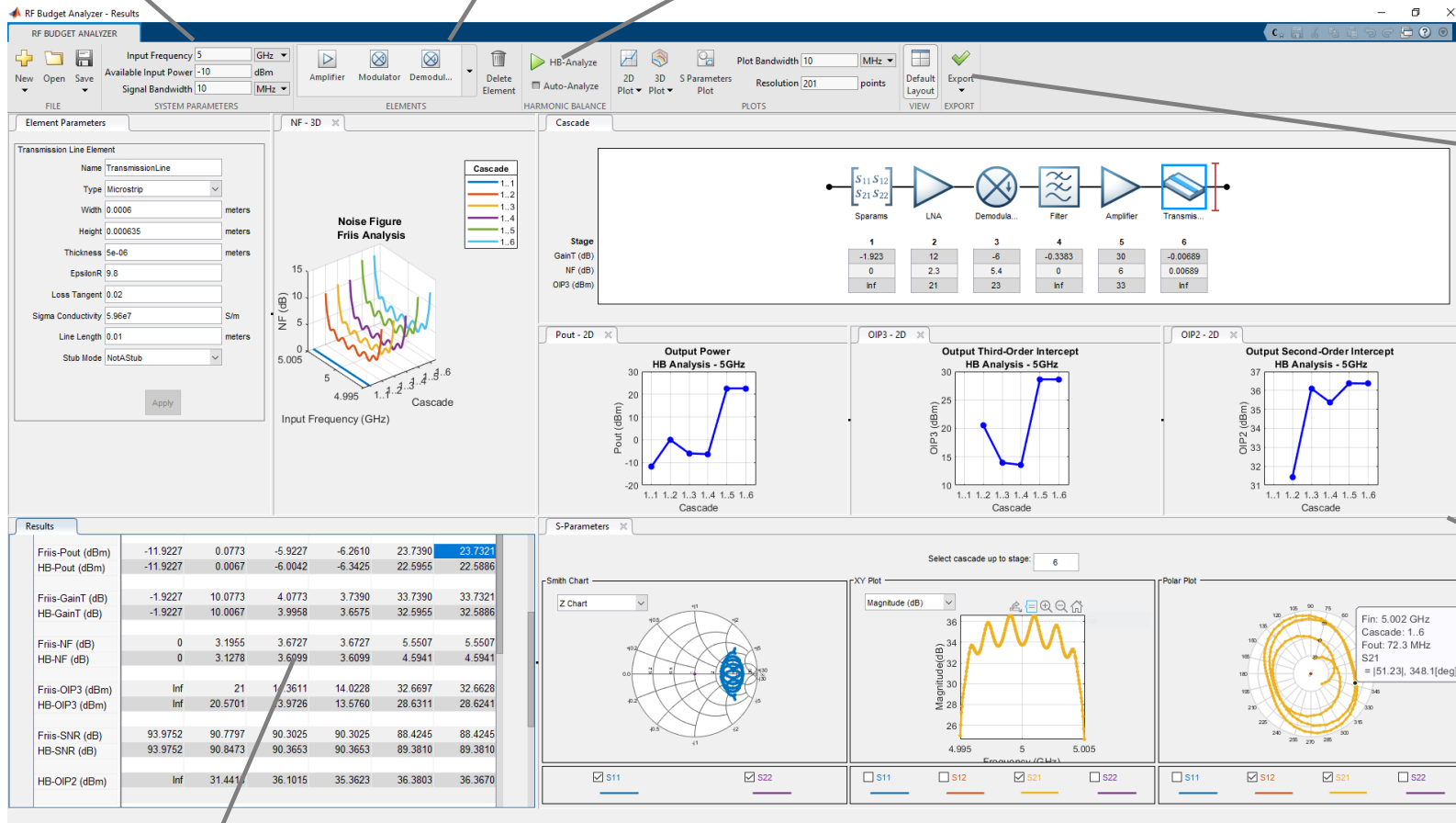
System specifications

RF components

Harmonic Balance (non-linear) analysis

Export to MATLAB / RF Blockset

New visualization



Cascaded Budget Analysis (Friis and Harmonic Balance)

# Export the RF Receiver to RF Blockset and add to PHY layer model.

RF BUDGET ANALYZER - simrfv2\_mimo\_frceiver

Input Frequency: 5 GHz  
Available Input Power: -30 dBm  
Signal Bandwidth: 10 MHz

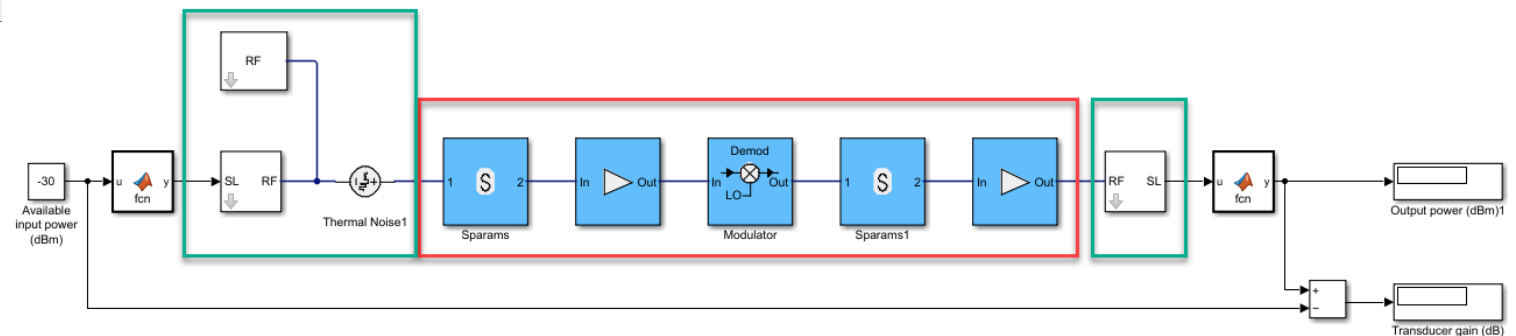
Export options:  
 MATLAB Workspace: Export rfbudget object to MATLAB workspace  
 MATLAB Script: Generate MATLAB Script  
**RF Blockset: Export to RF Blockset**  
 Measurement Testbench: Export to testbench in RF Blockset

Stage	1	2	3	4	5
GainT (dB)	-2.432	12	-6	-0.1395	20
NF (dB)	2.201	1.3	5.4	0.1395	6
OIP3 (dBm)	Inf	21	32	Inf	Inf

Output Power Friis Analysis

Cascade	1..1	1..2	1..3	1..4	1..5
Fout (GHz)	5	5	0.0700	0.0700	0.0700
Friis-Pout (dBm)	-32.4316	-20.4316	-26.4316	-26.5711	-6.5711
Friis-GainT (dB)	-2.4316	9.5684	3.5684	3.4289	23.4289
Friis-NF (dB)	2.2006	3.5615	4.0538	4.0782	5.9230
Friis-OIP3 (dBm)	Inf	21	14.9142	14.7747	34.7747
Friis-SNR (dB)	71.7746	70.4137	69.9214	69.8970	68.0522

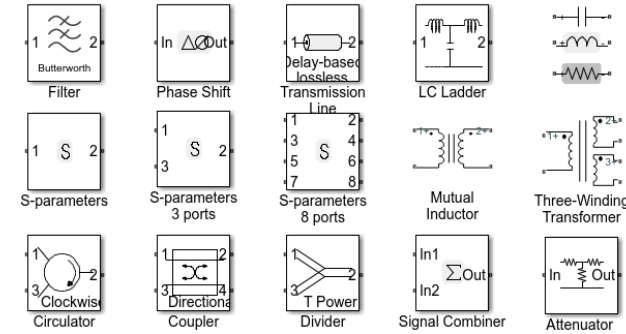
**May I do this in MATLAB?**



# Circuit Envelope Available Blocks

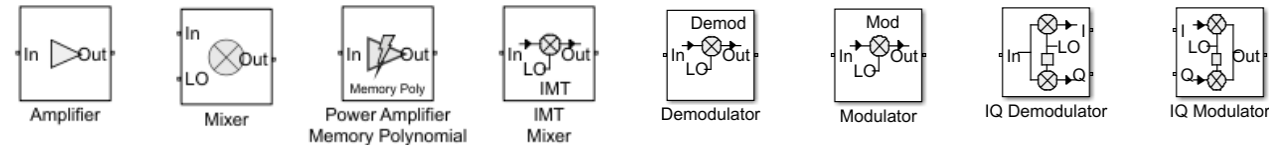
- Frequency dependent (linear) elements

- S-parameters
- RLC, transmission lines
- Filter, attenuator, coupler, circulator, divider/combiner, phase shifter
- Antenna and antenna array



- Nonlinear elements

- Amplifier
- Power Amplifier (Memory polynomial and AM/AM-AM/PM)
- Mixer (IM table)
- System blocks: Modulator, Demodulator, IQ Modulator, IQ Demodulator

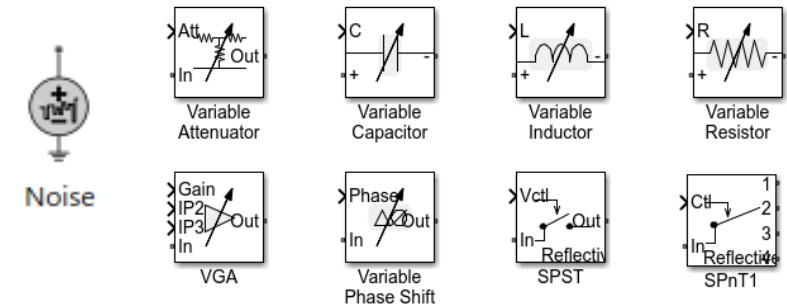


- Noise generation included for linear and non-linear elements

- Thermal Noise, Phase Noise

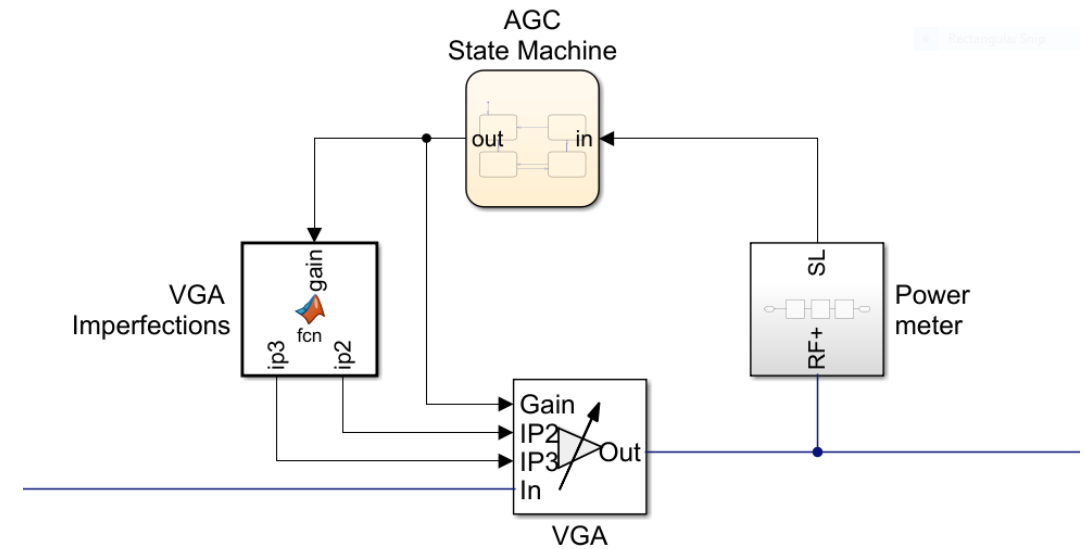
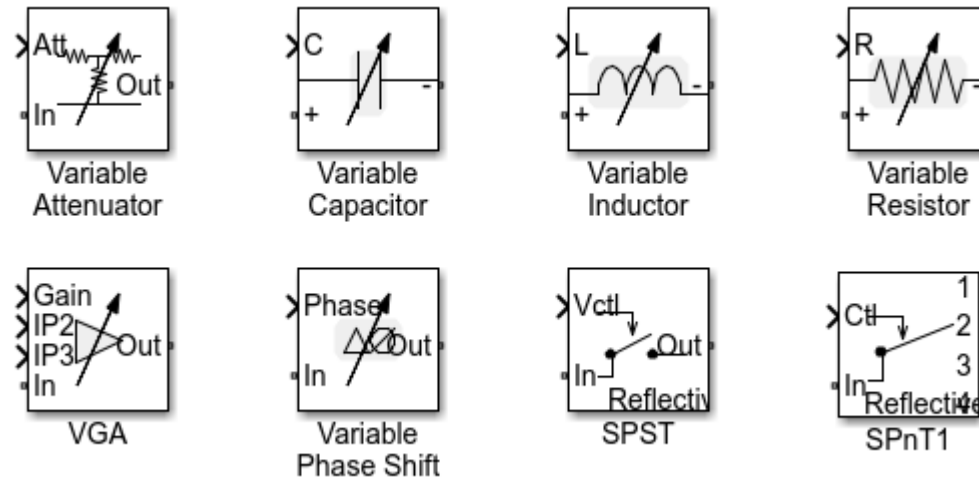
- Variable (Simulink controlled) elements for tunable networks

- Non-linear VGA, attenuator, shifter, RLC



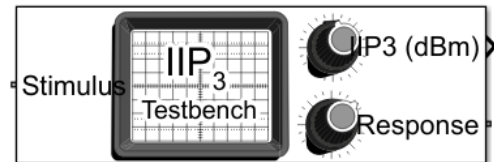
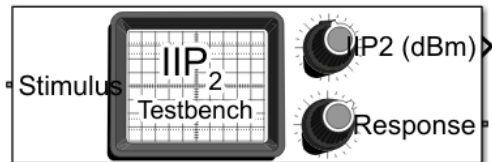
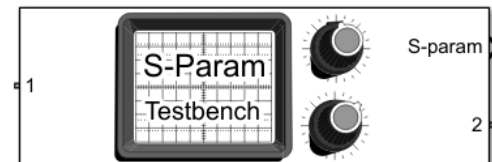
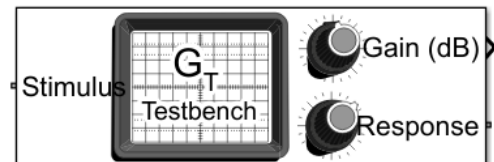
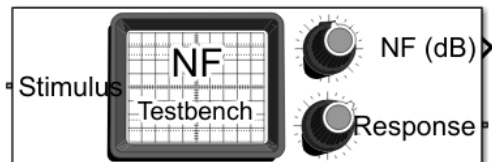
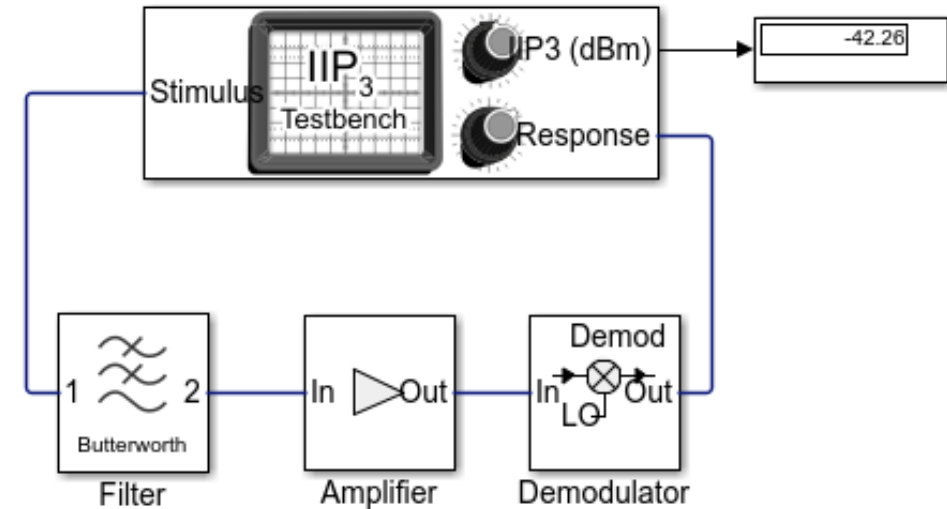
# Circuit Envelope Tunable Blocks

- RF elements with Simulink controlled characteristics
- Build tunable networks such as AGC, adaptive matching network, beamformer, butler matrix
  - Non-linear VGA, attenuator, phase shifter, RLC
  - Switches for building configurable networks



# Circuit Envelope Testbenches

- Measure and verify the performance of your DUT
  - Generate stimuli and measure response
  - Noise, gain, non-linearity, S-parameters

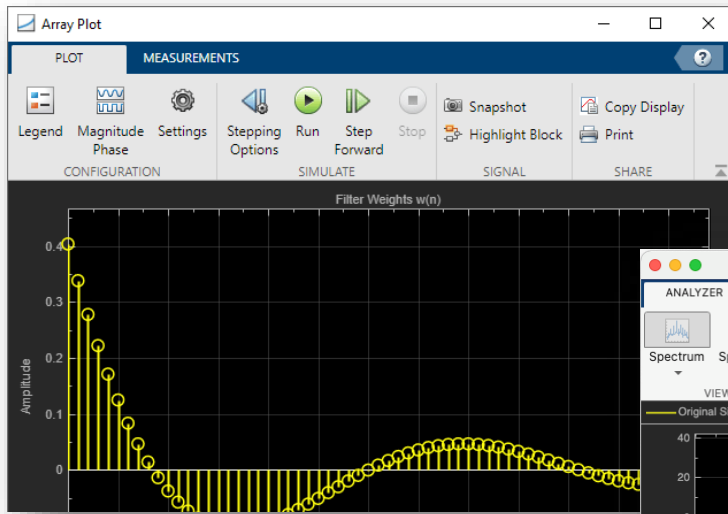


# Agenda

- Get started with RF data analysis
- RF system design
- **Signal Processing with Simulink**
- Integration custom or external code into Simulink models

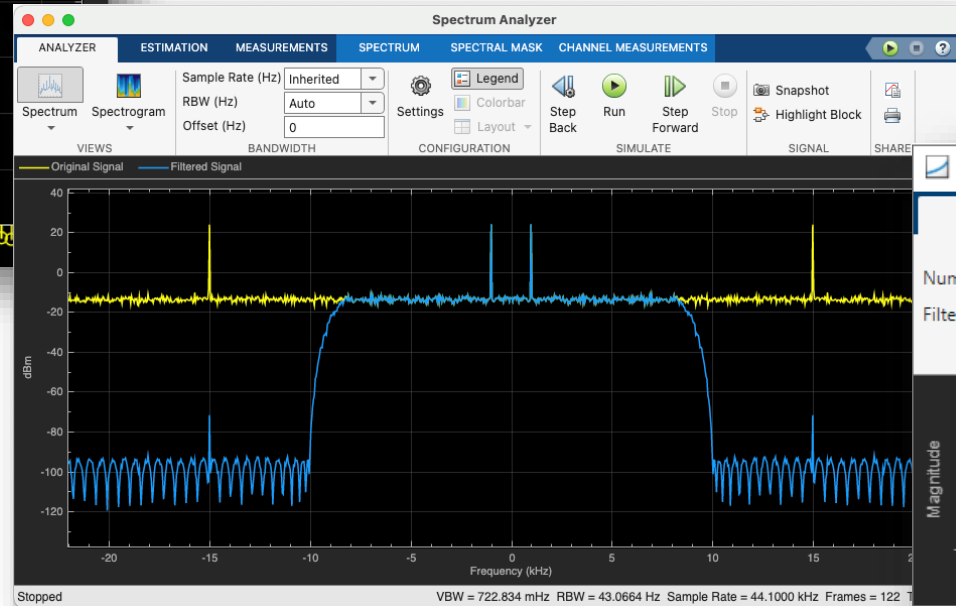
# Modern Responsive Soft Scopes

## Analyze and measure dynamic simulation signals and filters

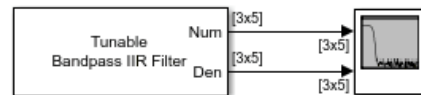
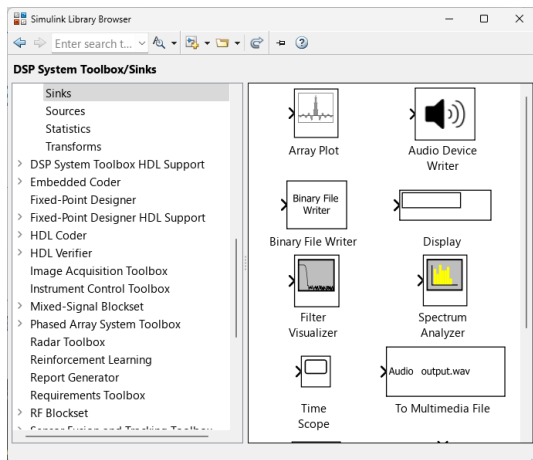
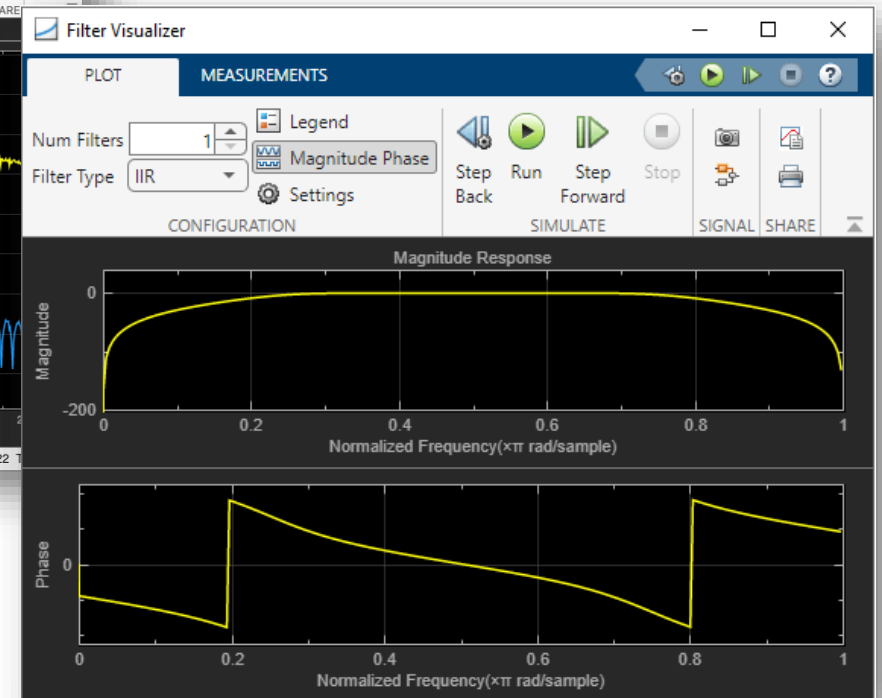


R2021a

R2023a



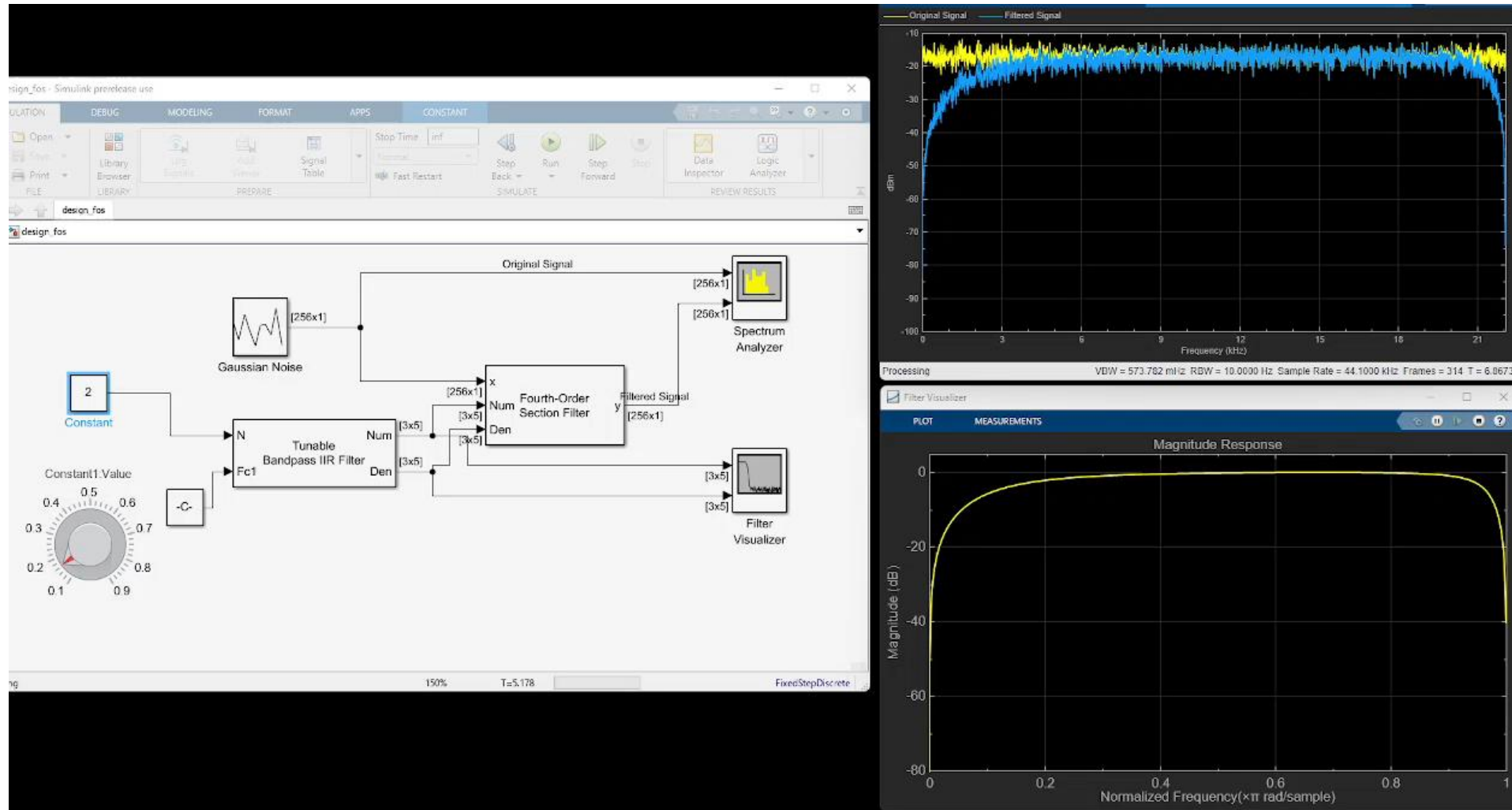
R2023a



# Dynamic Visualization of Filter Response and Signal Spectra

## Example

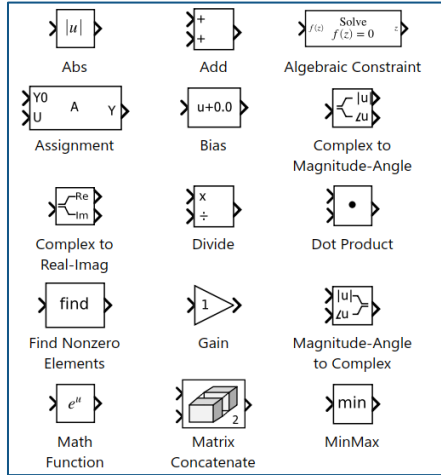
R2023a



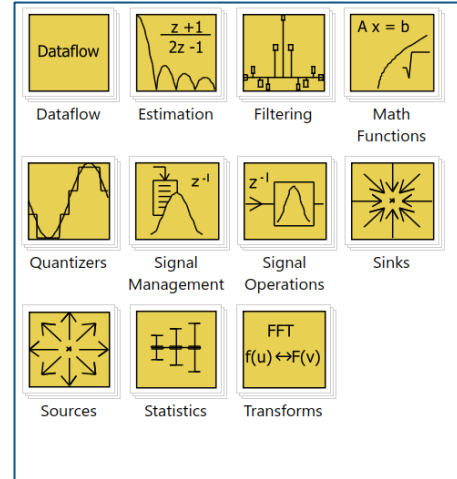
# Working with multiple domains?

- Incorporate multiple domains all in one model. Choose from hundreds of blocks in different domains:

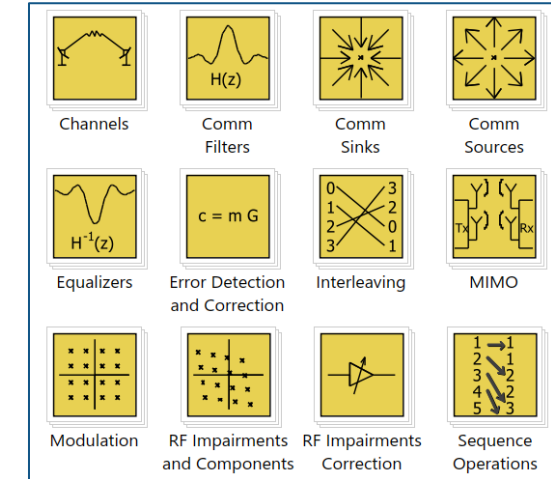
## Math Operations



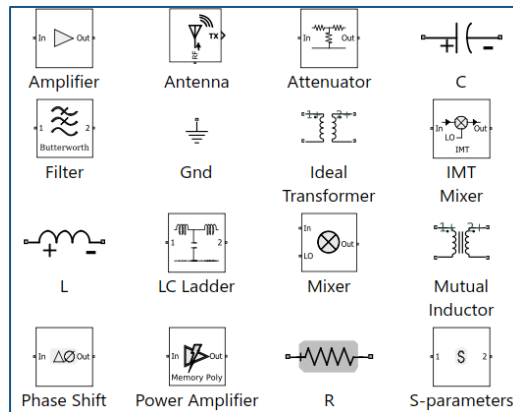
## Signal Processing



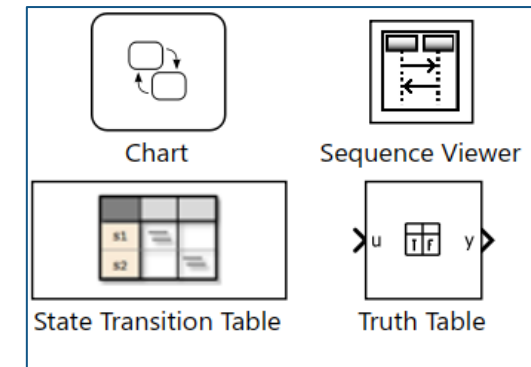
## Communications Systems




## RF




## State Machines




# How to learn MATLAB Academy




**MATLAB Onramp**  
14 modules | 2 heures | Langues  
Apprenez rapidement les bases de MATLAB.




**Simulink Onramp**  
14 modules | 2 heures | Langues  
Apprenez rapidement les bases de Simulink.



**Simulink Fundamentals**  
10 modules | 8 heures | Langues  
Apprenez comment utiliser Simulink, un outil de simulation graphique pour la modélisation de systèmes physiques dynamiques.



**Stateflow Onramp**  
12 modules | 2 heures | Langues  
Découvrez comment créer, modifier et simuler des machines à états dans Stateflow.



**Control Design Onramp with Simulink**  
7 modules | 1 heure | Langues  
Maîtrisez rapidement les concepts fondamentaux du design de systèmes de contrôle en boucle fermée dans Simulink.

## Traitement d'images et du signal



**Image Processing Onramp**  
6 modules | 2 heures | Langues  
Maîtrisez les concepts fondamentaux des techniques de traitement d'images dans MATLAB.




**Image Processing with MATLAB**  
11 modules | 11 heures | Langues  
Découvrez des workflows efficaces de traitement d'images dans MATLAB.




**Signal Processing with MATLAB**  
8 modules | 7.5 heures | Langues  
Apprenez comment effectuer le traitement du signal dans MATLAB.


## IA, Machine Learning et Deep Learning




**Machine Learning Onramp**  
6 modules | 2 heures | Langues  
Apprenez les bases des méthodes efficaces de Machine Learning pour résoudre des problèmes de classification.




**Machine Learning with MATLAB**  
6 modules | 12 heures | Langues  
Explorez les données et construisez des modèles prédictifs.



**Deep Learning Onramp**  
5 modules | 2 heures | Langues  
Familiarisez-vous rapidement avec l'utilisation de méthodes de Deep Learning pour la reconnaissance d'images.




**Deep Learning with MATLAB**  
11 modules | 7 heures | Langues  
Apprenez la théorie et la pratique pour développer des réseaux de neurones profonds avec des données séquentielles et des images.




**Reinforcement Learning Onramp**  
5 modules | 3 heures | Langues  
Maîtrisez les bases de la création de contrôleurs intelligents qui apprennent par expérience.


## Mathématiques et optimisation



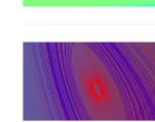
**Optimization Onramp**  
5 modules | 1 heure | Langues  
Apprenez les fondamentaux de la résolution de problèmes d'optimisation dans MATLAB à l'aide de l'approche par problèmes.



**Introduction to Symbolic Math with MATLAB**  
13 modules | 2 heures | Langues  
Démarrez rapidement avec une introduction au calcul symbolique.



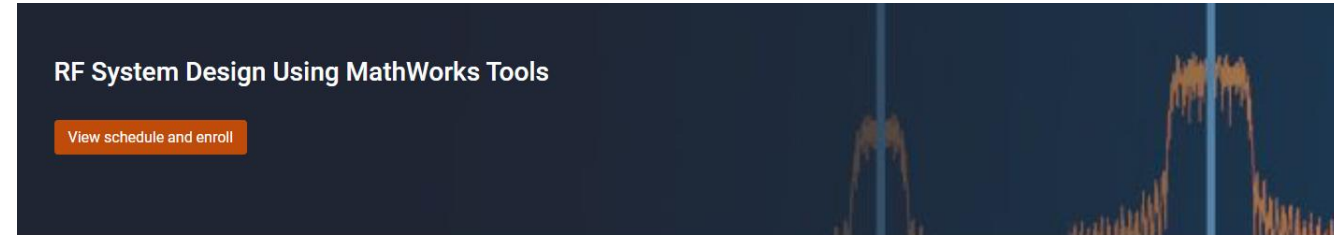
**Solving Nonlinear Equations with MATLAB**  
6 modules | 3 heures | Langues  
Utilisez des méthodes de recherche de racines pour résoudre des équations non linéaires.



**Solving Ordinary Differential Equations with MATLAB**  
6 modules | 4 heures | Langues  
Résolvez numériquement des équations différentielles ordinaires avec les solveurs ODE de MATLAB.

# Instructor led Training: RF system Design

- Build RF system models
- Import and Simulate S-parameters
- Model Linear and non-Linear RF components
- Analyze the impact of noise and nonlinearities



## RF System Design Using MathWorks Tools

[View schedule and enroll](#)

### Course Details

This two-day course shows how to use RF Blockset™ and RF Toolbox™ for modeling wireless front ends. You will learn when to use two different modeling paradigms to speed up the simulation of RF signals: Equivalent Baseband and Circuit Envelope. The fundamentals of the simulation techniques will be discussed, and best modeling practices will be highlighted.

Topics include:

- Importing S-parameters and modeling linear elements
- Simulating thermal and phase noise
- Modeling amplifiers and mixers operating in nonlinear conditions
- Developing custom models
- Integrating antennas and arrays in RF transceivers

### Day 1 of 2

What is RF Blockset?

**Objective:** Give an overview of RF simulation using MathWorks tools; the goal of this section is to familiarize the audience with RF Blockset and its terminology. Attendees who never used RF Blockset before will be able to recognize which applications can be modeled with RF Blockset and which approach is most suitable.

- Describe what can be done with RF Blockset
- Describe the benefits of using RF simulation techniques
- Articulate why and when RF Blockset is needed

Fundamentals of RF Simulation

**Objective:** Describe the fundamentals of Equivalent Baseband and Circuit Envelope simulation techniques. We will

**Level:** Intermediate

**Prerequisites:**

- [Signal Processing with Simulink](#) Linear Elements
- A good understanding of RF theory

**Duration:** 2 days

**Languages:** English

[View schedule and enroll](#)

# Agenda

- Get started with RF data analysis
- Simulating RF controlled components at the system-level
- Signal Processing with Simulink
- **Integration custom or external code into Simulink models**

# Implement MATLAB Functions in Simulink

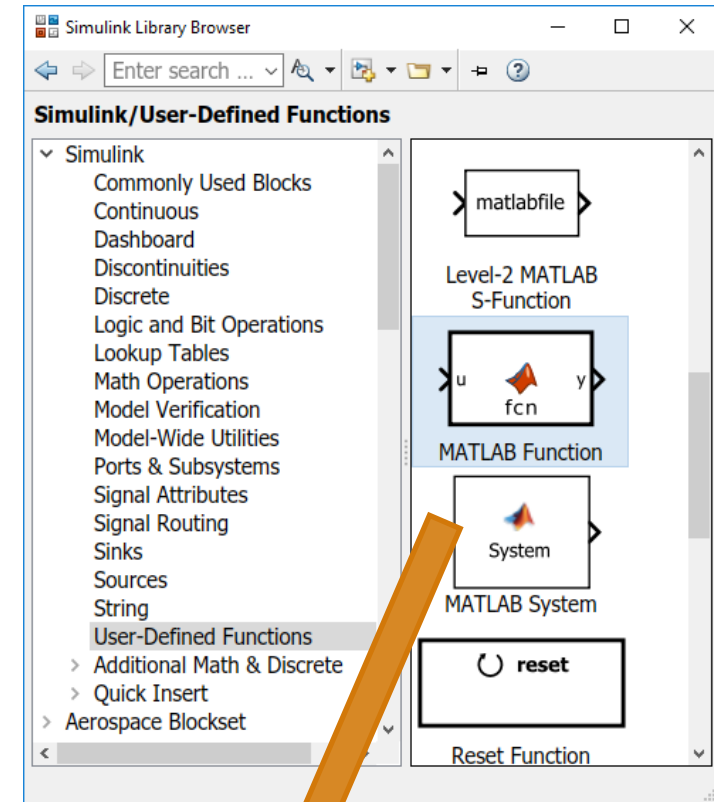
Write a MATLAB function that runs in a Simulink model

Can accept multiple input and reproduce multiple output signals

## Why use it?:

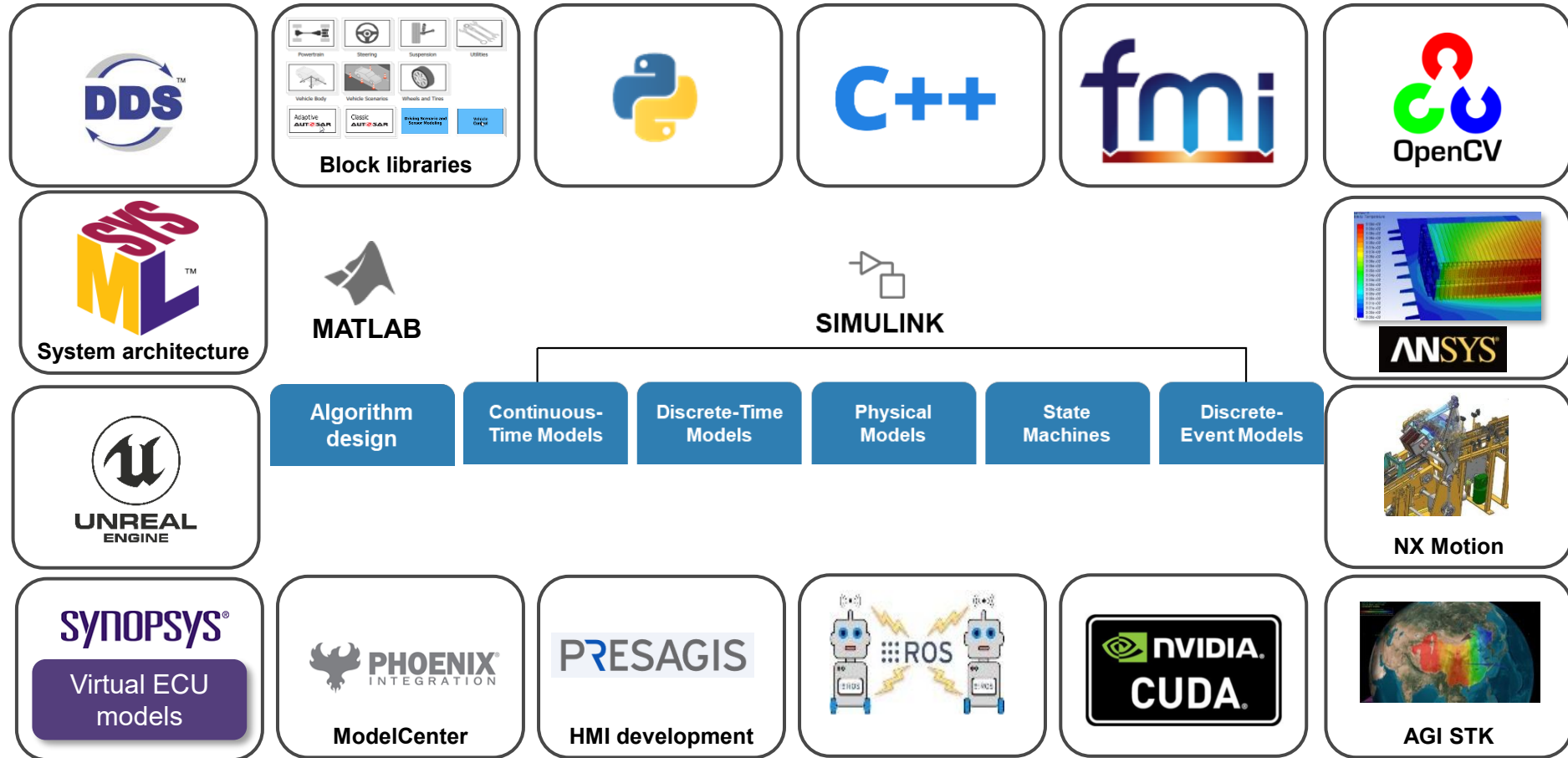
Include existing MATLAB algorithms into Simulink

Write simple algorithms without the graphical overhead



```
1 function y = fcn(u)
2
3 y = u;
4
```

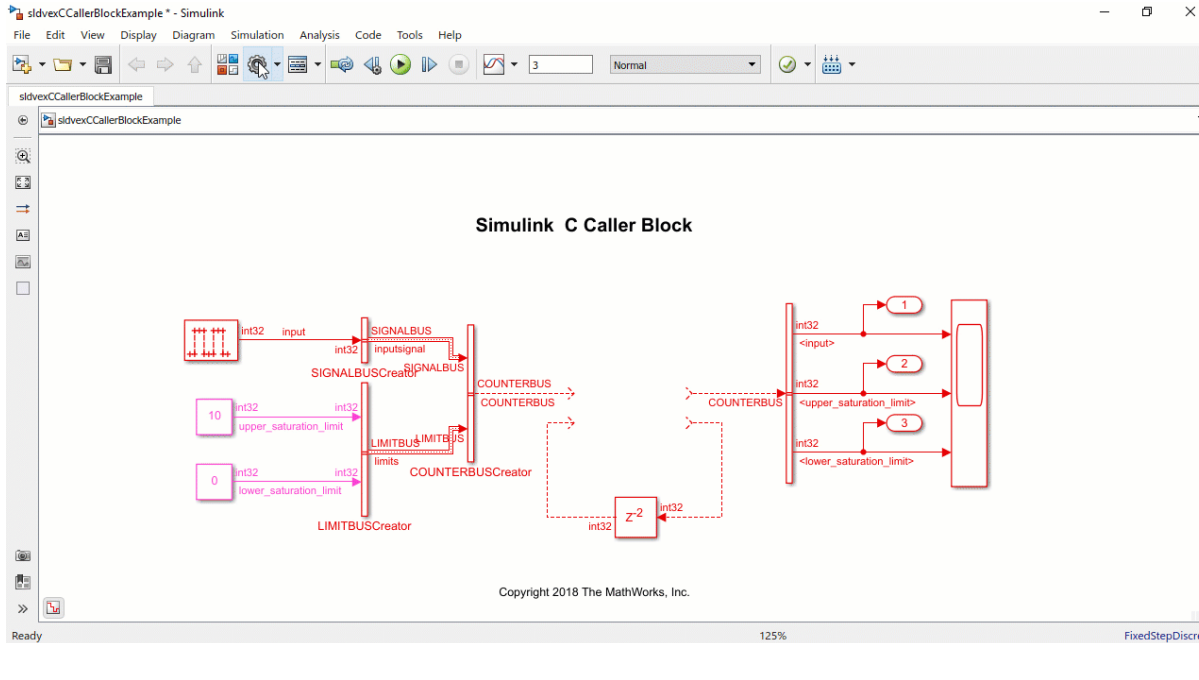
# Integrate external components & code into your Simulink models





# Integrating your C code into Simulink is simple

## C Caller Block: Call external C functions directly from the model



Supports simulation and code generation

The diagram shows a block labeled '<FunctionName>' and 'C Function Caller'. A dialog box titled 'Block Parameters: C Function Caller' is open, showing the following settings:

- Function name:
- Array layout:
- Port specification: (expanded)

Buttons for 'OK', 'Cancel', 'Help', and 'Apply' are visible at the bottom of the dialog.

<https://www.mathworks.com/help/simulink/ug/integrate-c-code-caller.html>



Learn which approach will work for your application

## Python Importer

*Use to bring an existing Python function into Simulink.*

- + Graphical interface with minimal code
- + Creates reusable blocks from Python functions
- Cannot edit Python code once the block is created

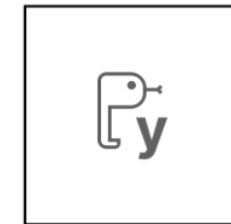


Python

## Python Code Block

*Use to write native Python code in Simulink.*

- + Easily edit code from the block dialog
- + Supports all simulation modes
- No debugging capabilities in the block dialog



Python Code

*For releases prior to R2023a, you can use MATLAB Function or System blocks to use Python code in Simulink.*

## Key Takeaways

- Closed-loop, time-domain simulation is essential
- Start with RF data analysis using RF Toolbox
- Reuse validated RF models at the system level
- Simulink enables live observation of system behavior

Merci !

