

Three Steps to Simulate RF Transceivers in MATLAB

Sekhar Sekharan
Application Engineer

Agenda

1. Introducing RF transceivers design for wireless systems
2. From high-level specifications to Elaborating RF transceiver models
3. Enabling RF system-level simulation in MATLAB



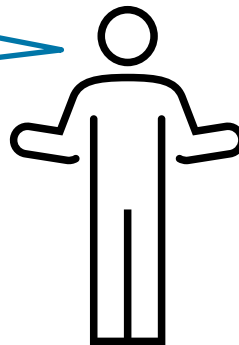
Motivation:

Connect Baseband Signal Processing with High Fidelity RF Simulation

- Use MATLAB code for baseband signal processing
 - Radar and wireless communications standards
 - Control, calibration, and correction algorithms
- Increase the fidelity of the RF transceiver models, to model the following:
 - Impedance mismatches, reverse isolation, S-parameter data
 - Out of band interference including near band signals, blockers, and jammers
 - Power amplifier memory effects and non-linearity
 - Antenna array coupling and loading effects



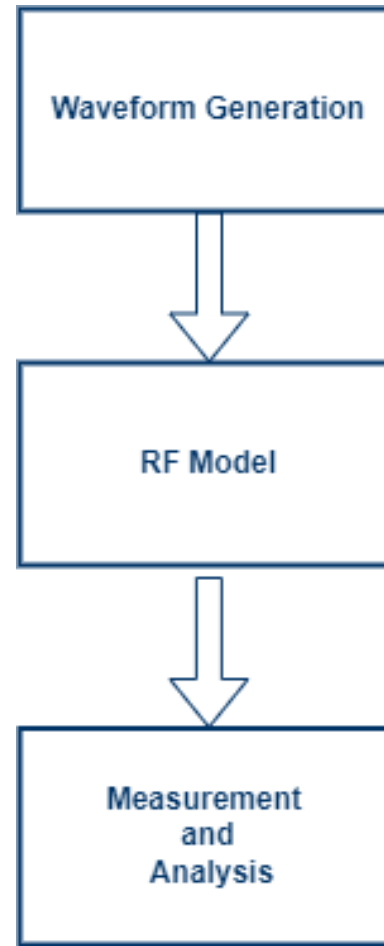
I use MATLAB!!
EVM, BER, ACLR!

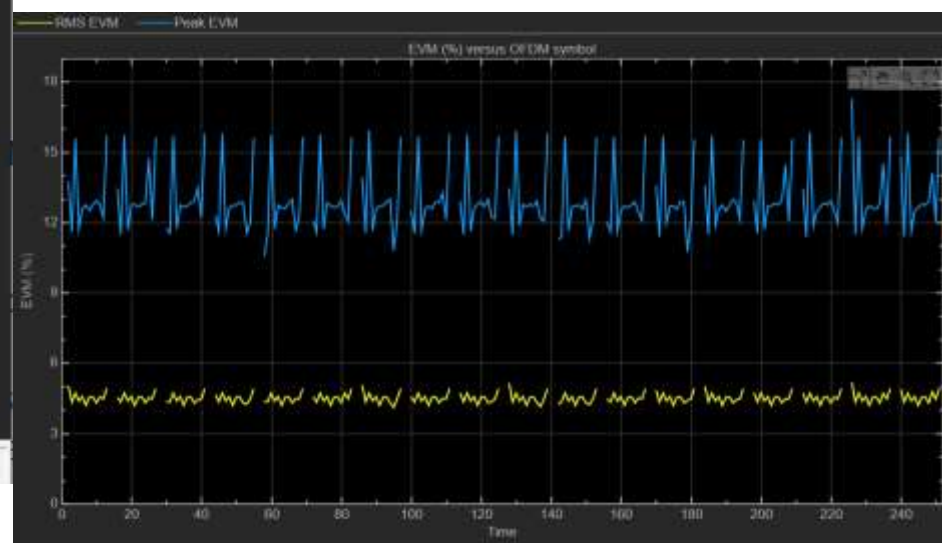
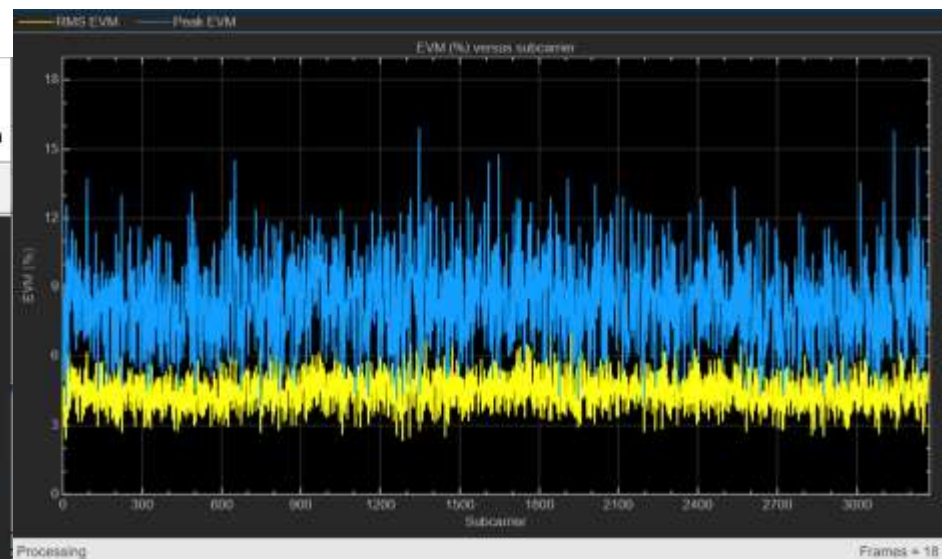
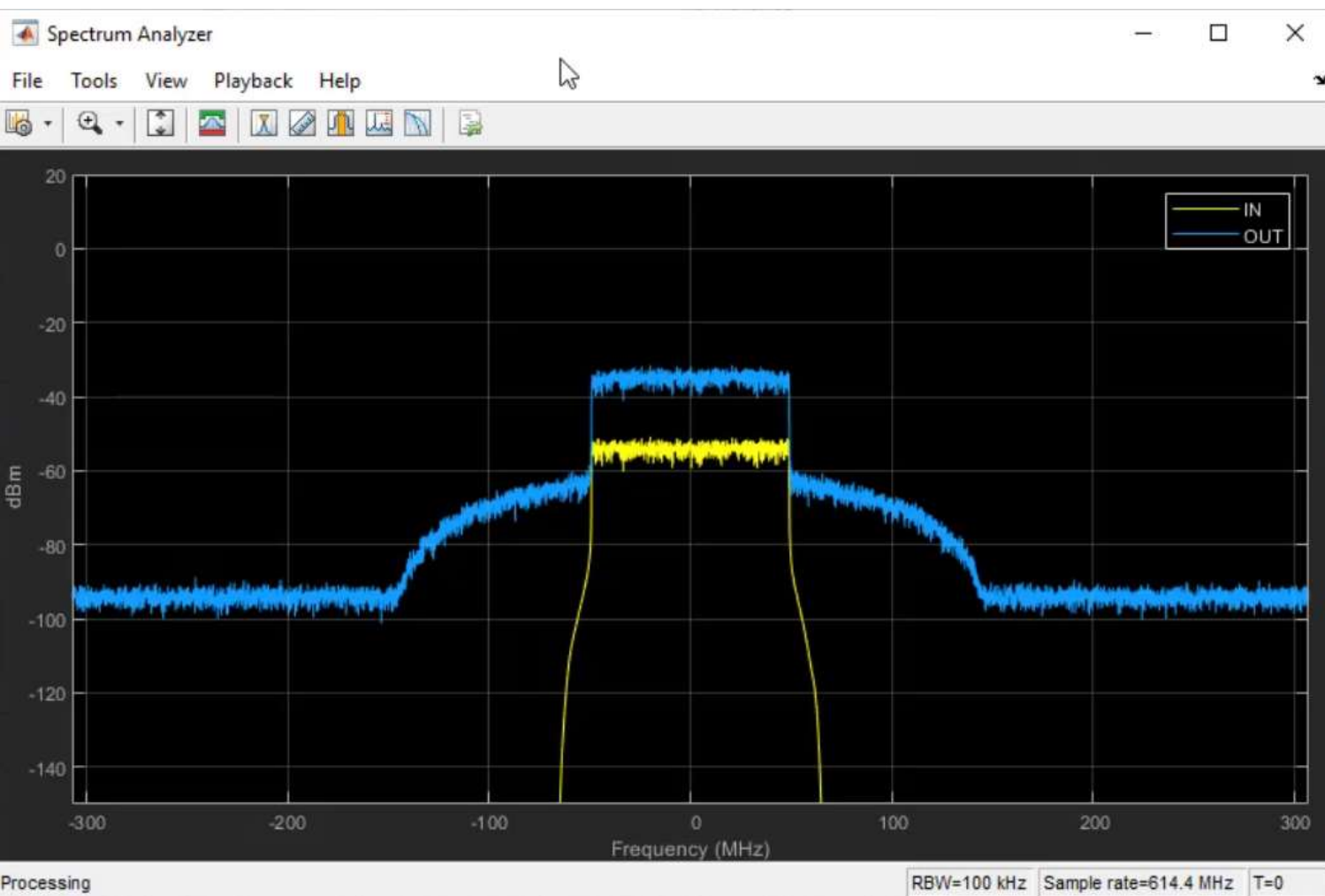


I use harmonic balance!
IP2, IP3, P1dB, s2p,
AM/AM-AM/PM !!



Demo





Introducing RF Transceivers Design

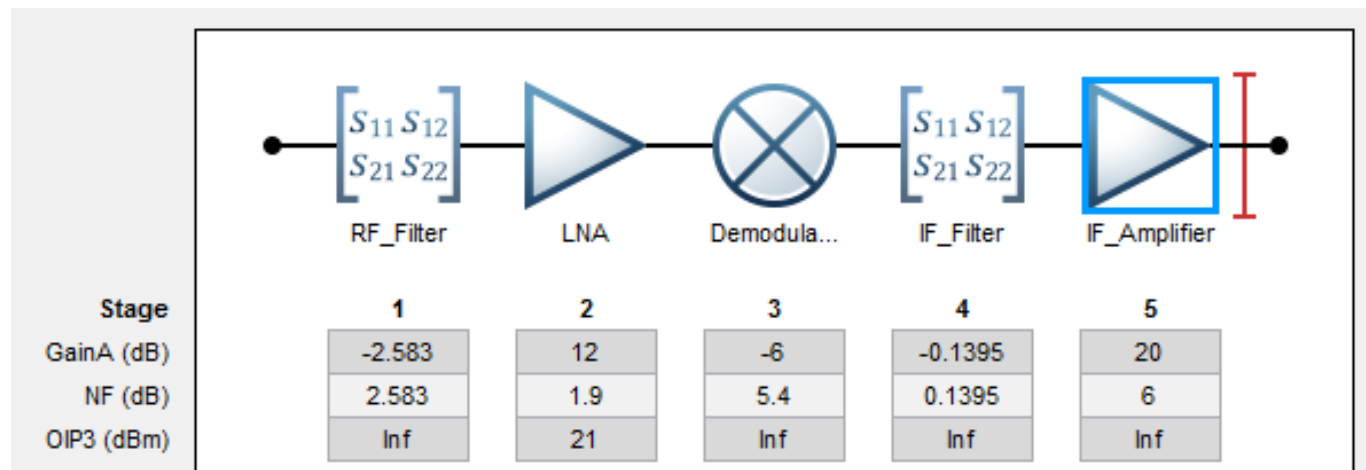
RF Transceiver Design – Where To Start?



RF Budget
Analyzer

RF Budget Analyzer App

- Computation and visualization of power/noise/IP3 RF budget
- Improvement over custom-made spreadsheets: impedance mismatches, Harmonic Balance
- Generate MATLAB scripts for automation and complex scenario analysis
- Generate Circuit Envelope models/testbenches for superheterodyne and homodyne architectures



Add RF Components **HB Analysis** **Plot Budget**

Export to RF Blockset

Component Specifications

RF Cascade

Cascade Budget Analysis

RF BUDGET ANALYZER - Results

RF BUDGET ANALYZER

FILE SYSTEM PARAMETERS ELEMENTS PLOTS VIEW

Input Frequency: 5 GHz
Available Input Power: -20 dBm
Signal Bandwidth: 12 MHz

Amplifier Modulator Demodul... Delete Element

HB-Analyze Auto-Analyze 2D Plot 3D Plot S Parameters Plot

Plot Bandwidth: 12 MHz Resolution: 51 points

Export

Element Parameters

S-parameters Element

Name: RF_Filter

Touchstone File: Jrf_filt.s2p

Apply

RFReceiverDesign

Stage

GainT (dB) NF (dB) OIP3 (dBm)

1 2 3 4 5

-2.432 12 -6 -0.1395 20

2.201 1.9 5.4 0.1395 6

Inf 21 Inf Inf Inf

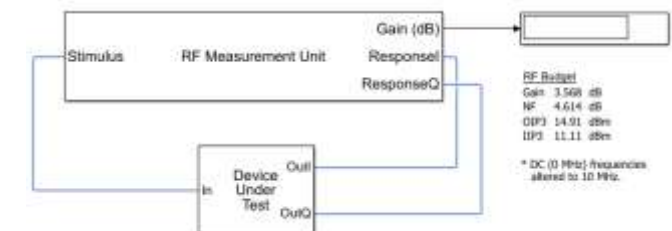
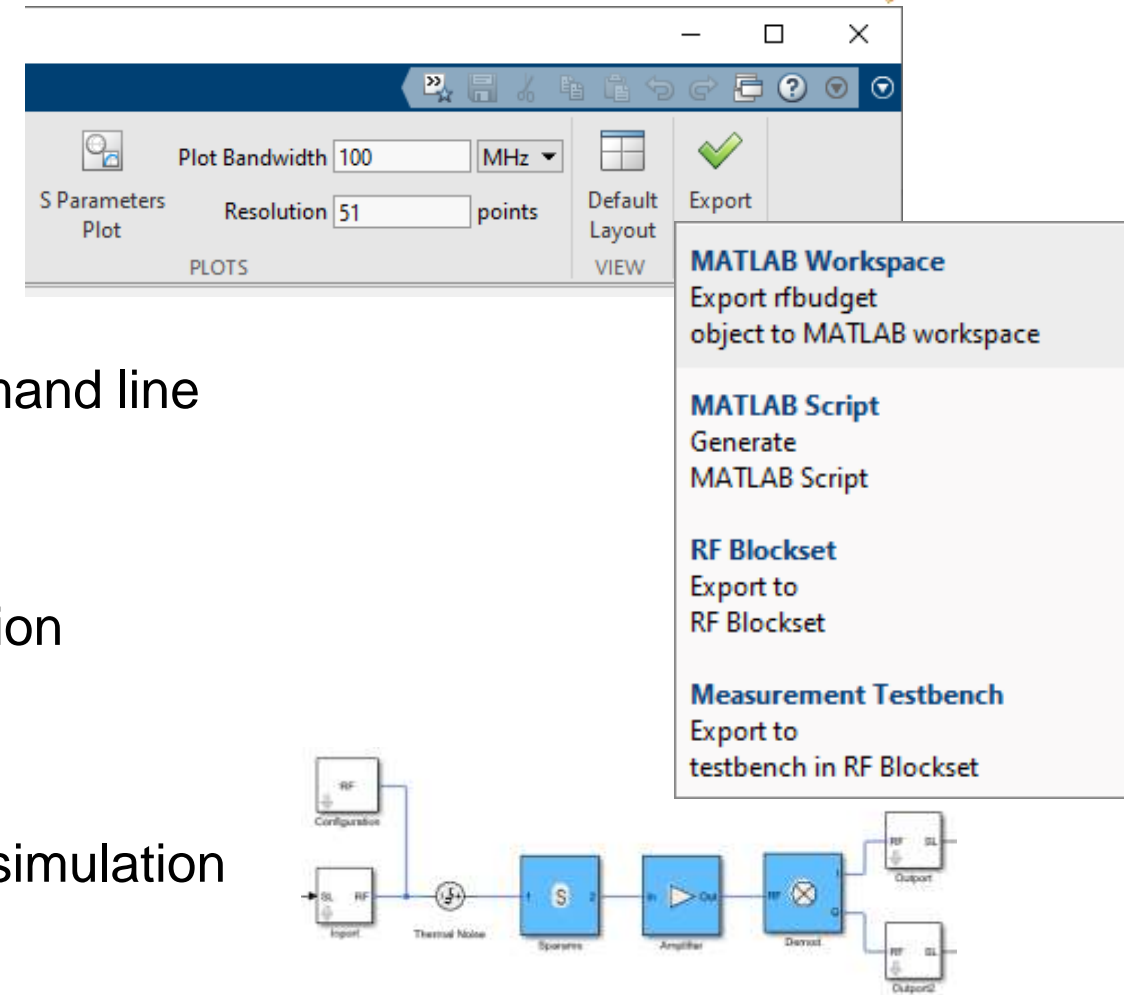
Results

Select Results Compare View

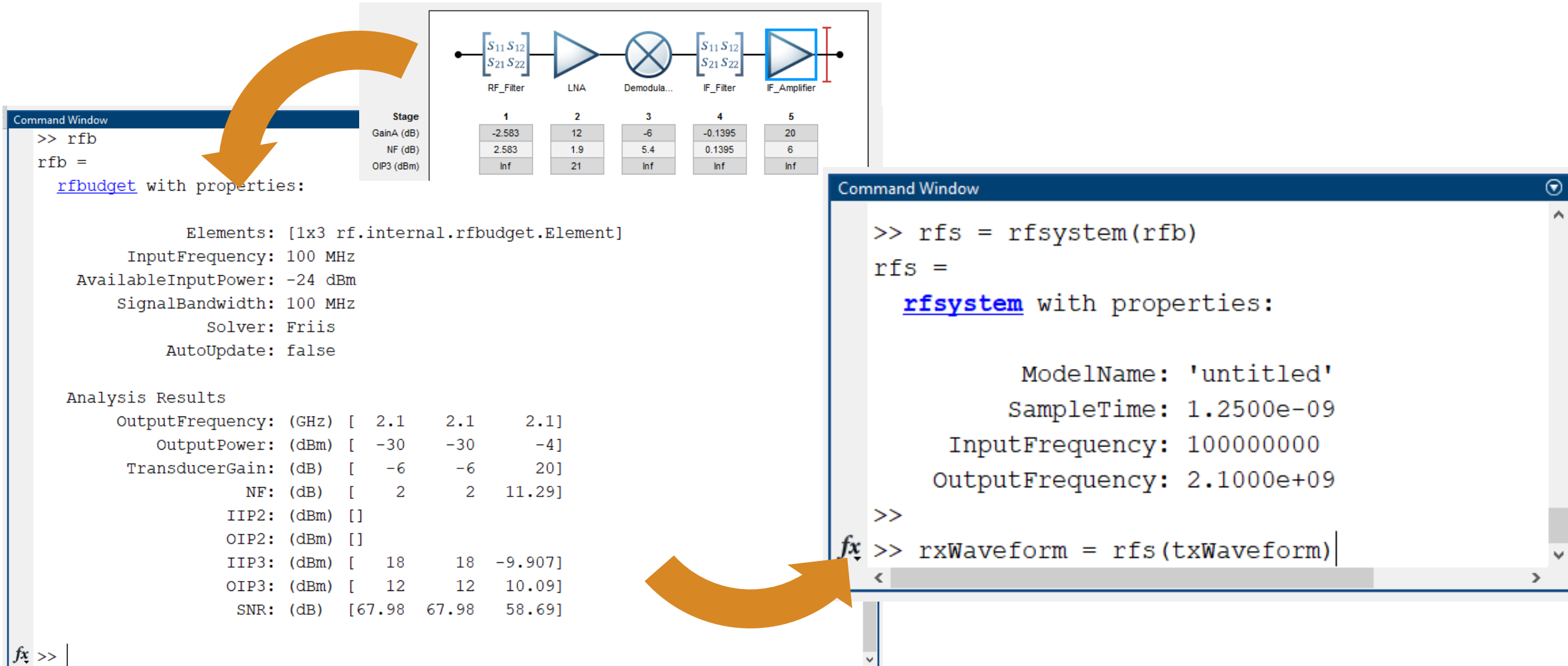
Cascade	1..1	1..2	1..3	1..4	1..5
Fout (GHz)	5	5	0.0700	0.0700	0.0700
HB-Pout (dBm)	-22.4316	-10.4379	-16.4379	-16.5774	3.4226
HB-GainT (dB)	-2.4316	9.5621	3.5621	3.4226	23.4226
HB-NF (dB)	2.2006	4.1776	4.6084	4.6300	6.2936
HB-OIP2 (dBm)	Inf	Inf	Inf	Inf	Inf
HB-OIP3 (dBm)	Inf	20.9845	14.9845	14.8495	34.8495
HB-SNR (dB)	80.9828	79.0058	78.5749	78.5534	76.8897
Friis-Pout (dBm)	-22.4316	-10.4316	-16.4316	-16.5711	3.4289
Friis-GainT (dB)	-2.4316	9.5684	3.5684	3.4289	23.4289
Friis-NF (dB)	2.2006	4.1839	4.6135	4.6350	6.2951
Friis-OIP3 (dBm)	Inf	21	15	14.8605	34.8605
Friis-SNR (dB)	80.9828	78.9995	78.5699	78.5484	76.8883

Export from RF Budget Analyzer

- Export to MATLAB workspace object
 - Bidirectional workflow between app and command line
- Export to MATLAB script
 - Automate corner analysis and chain optimization
- Export to RF Blockset
 - Generate model for HB and Circuit Envelope simulation
 - Automatic configuration of Input/output ports
- Export to measurement testbench
 - Measure gain, noise, IP3, IP2, DC offset, and image rejection
 - Validate that the RF system with Circuit Envelope simulation



From RF Budget Analysis to RF System Simulation



The diagram illustrates the process of converting an RF Budget Analysis into an RF System Simulation. It features a central block diagram of an RF system and two command windows showing the MATLAB commands and results for each step.

RF System Block Diagram:

```

graph LR
    RF_Filter[RF_Filter] --> LNA[LNA]
    LNA --> Demodulator((Demodulator))
    Demodulator --> IF_Filter[IF_Filter]
    IF_Filter --> IF_Amplifier[IF_Amplifier]
  
```

RF System Parameters Table:

Stage	1	2	3	4	5
GainA (dB)	-2.583	12	-6	-0.1395	20
NF (dB)	2.583	1.9	5.4	0.1395	6
OIP3 (dBm)	Inf	21	Inf	Inf	Inf

Command Window 1 (Left):

```

>> rfb
rfb =
    rfbudget with properties:

        Elements: [1x3 rf.internal.rfbudget.Element]
        InputFrequency: 100 MHz
        AvailableInputPower: -24 dBm
        SignalBandwidth: 100 MHz
        Solver: Friis
        AutoUpdate: false

    Analysis Results
        OutputFrequency: (GHz) [ 2.1    2.1    2.1]
        OutputPower: (dBm) [ -30   -30   -4]
        TransducerGain: (dB) [ -6    -6   20]
        NF: (dB) [ 2    2   11.29]
        IIP2: (dBm) []
        OIP2: (dBm) []
        IIP3: (dBm) [ 18    18  -9.907]
        OIP3: (dBm) [ 12    12  10.09]
        SNR: (dB) [ 67.98  67.98  58.69]
  
```

Command Window 2 (Right):

```

>> rfs = rfsystem(rfb)
rfs =
    rfsystem with properties:

        ModelName: 'untitled'
        SampleTime: 1.2500e-09
        InputFrequency: 100000000
        OutputFrequency: 2.1000e+09

>> rxWaveform = rfs(txWaveform)
  
```

Budget Calculations from the Command Line

- Further automation and design space automation
- Support for non-linear analysis with harmonic balance engine

```
a = amplifier('Gain',4);|
m = modulator('OIP3',13);
n = nport('passive.s2p');
r = rfelement('Gain',10);
b = rfbudget([a m r n],2.1e9,-30,10e6)
```

```
b =
rfbudget with properties:
    Elements: [1x4 rf.internal.rfbudget.Element]
    InputFrequency: 2.1 GHz
    AvailableInputPower: -30 dBm
    SignalBandwidth: 10 MHz
    Solver: Friis
    AutoUpdate: true

Analysis Results
OutputFrequency: (GHz) [ 2.1    3.1    3.1    3.1]
OutputPower: (dBm) [ -26   -26   -16  -20.6]
TransducerGain: (dB) [  4     4    14    9.4]
NF: (dB) [  0     0     0  0.1392]
IIP2: (dBm) []
OIP2: (dBm) []
IIP3: (dBm) [  Inf     9     9     9]
OIP3: (dBm) [  Inf    13    23   18.4]
SNR: (dB) [73.98  73.98  73.98  73.84]
```

```
m = modulator('Gain',3,'OIP2',20,'ImageReject',false,'ChannelSelect',false);
m2 = modulator('Gain',3,'OIP2',20,'ImageReject',false,'ChannelSelect',false);
b = rfbudget([m m2],2.1e9,-30,100e6,'Solver','HarmonicBalance')
```

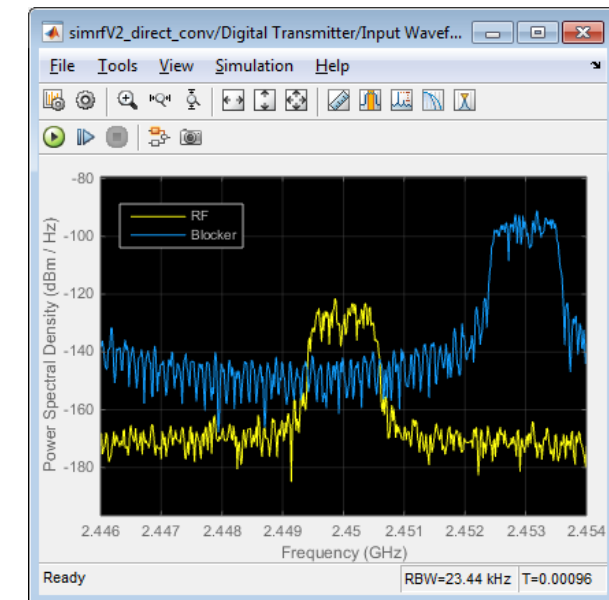
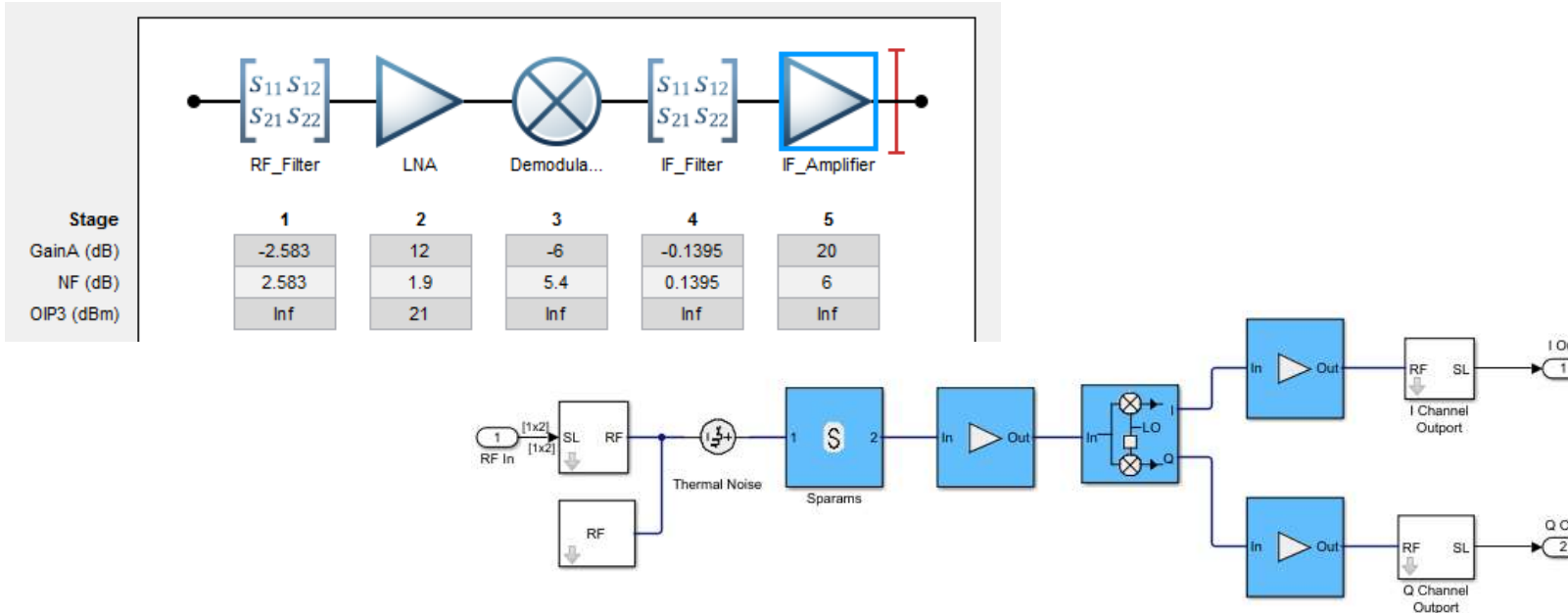
```
b =
rfbudget with properties:
    Elements: [1x2 modulator]
    InputFrequency: 2.1 GHz
    AvailableInputPower: -30 dBm
    SignalBandwidth: 100 MHz
    Solver: HarmonicBalance
    Tones: [1e+09 2.1e+09 2.1125e+09]
    Harmonics: 3
    AutoUpdate: true

Analysis Results
OutputFrequency: (GHz) [3.1    4.1]
OutputPower: (dBm) [-27   -24]
TransducerGain: (dB) [  3     6]
NF: (dB) []
IIP2: (dBm) [ 17   4.457]
OIP2: (dBm) [ 20  10.46]
IIP3: (dBm) [Inf   Inf]
OIP3: (dBm) [Inf   Inf]
SNR: (dB) []
```

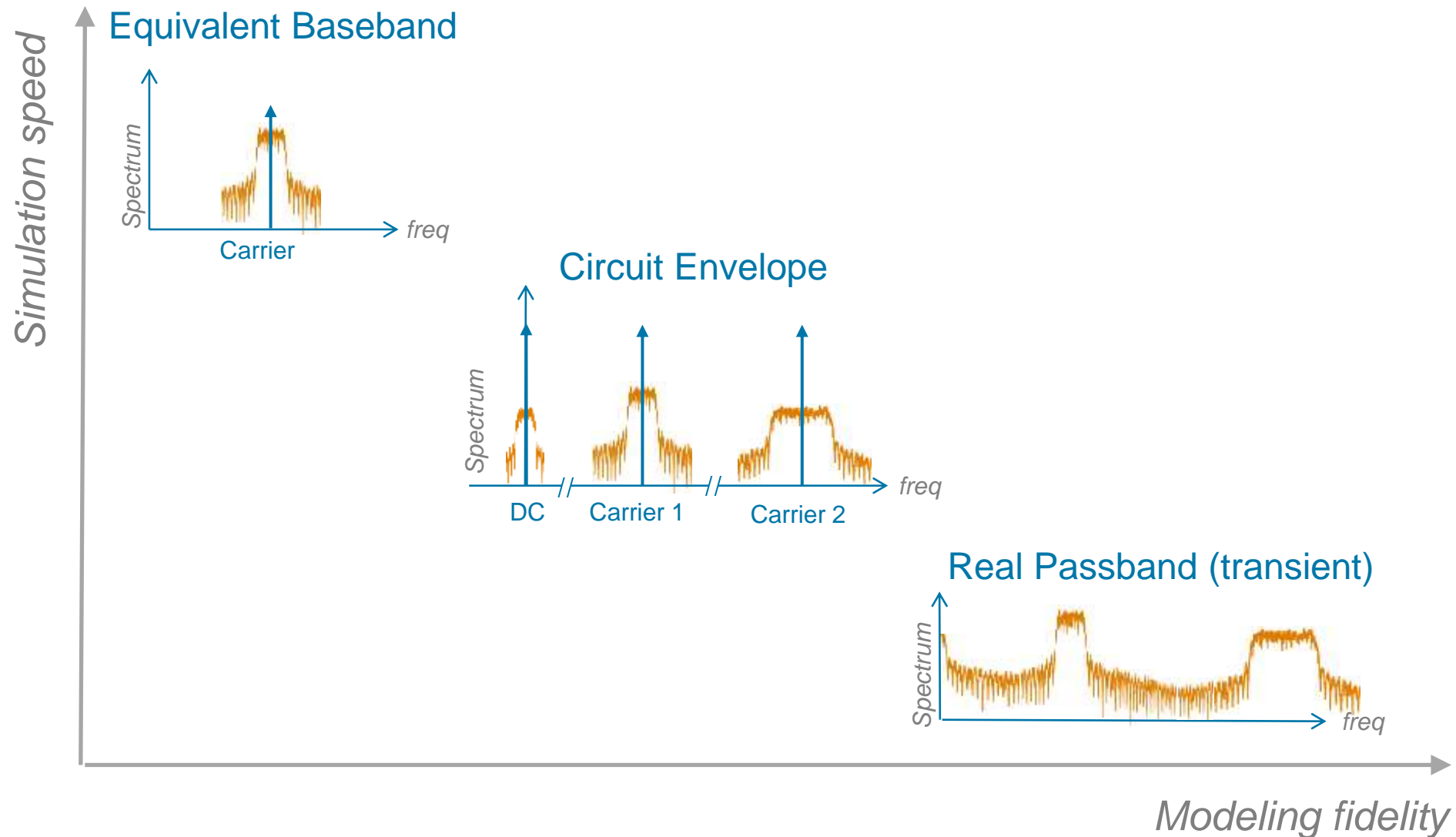
Elaborating RF Transceivers Models

RF System-Level Models Are Necessary For RF Design

- Design the architecture and define the specs of the RF components
- Integrate RF front ends with adaptive algorithms such as DPD, AGC, beamforming
- Test and debug the implementation of the transceiver before going into the lab
- Use models and measured data to gain insights in your design
- Provide a model of the RF transceiver to your colleagues and customers

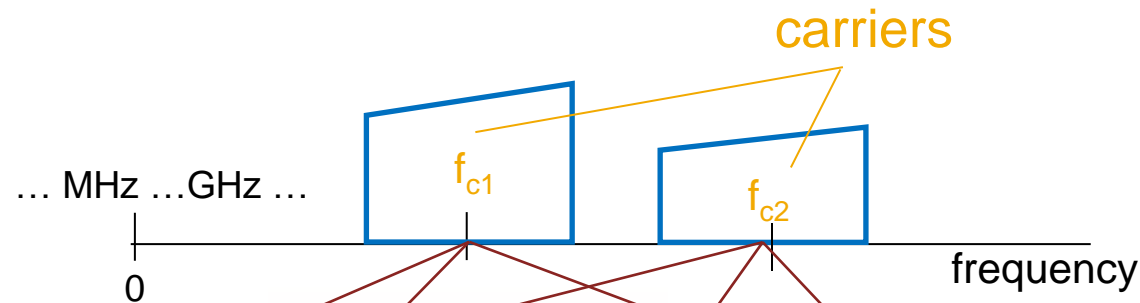


RF System Simulation Must Be Fast (and Accurate)

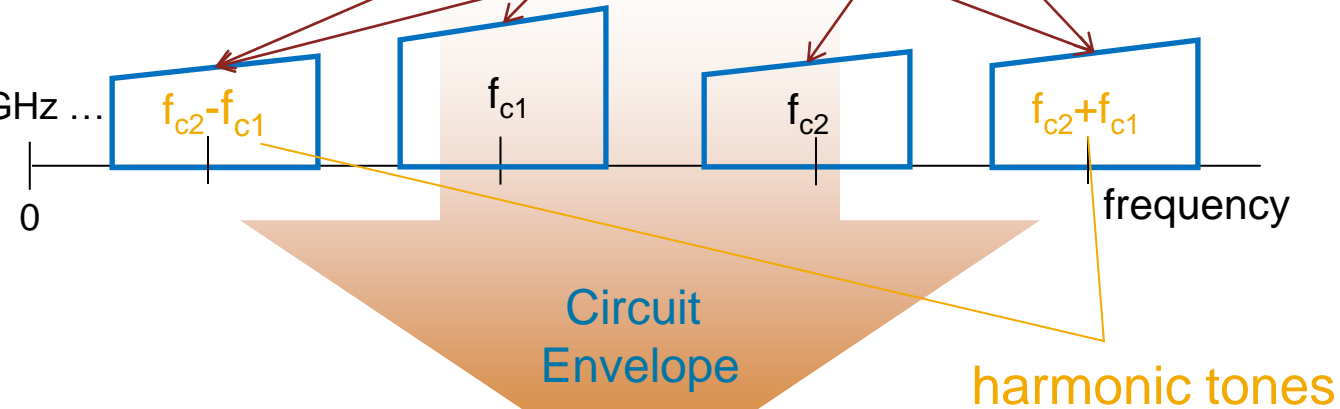


Multi-Carrier Circuit Envelope Simulation

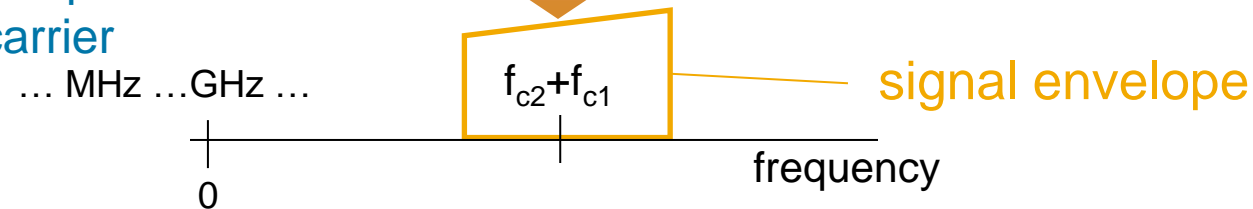
Input port: convert complex envelope modulation into RF signal



Configuration: ... MHz ... GHz ...
specify the harmonic order

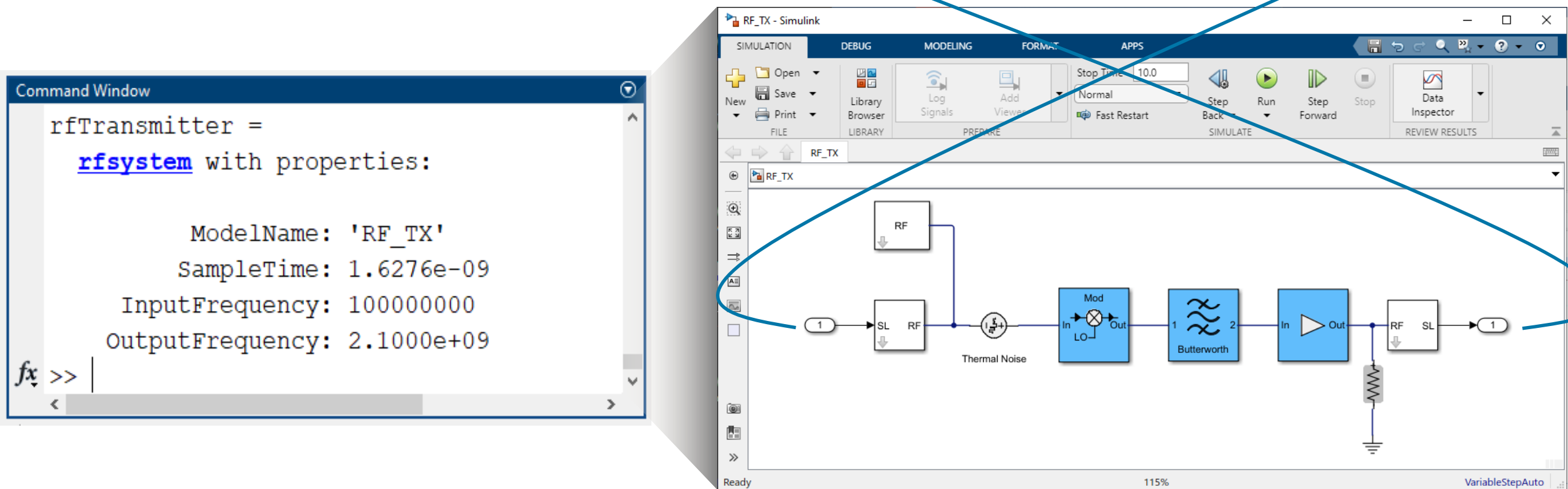


Output port: select complex envelope response around the selected carrier



Circuit Envelope: the Technology Behind RF System Objects

`rxWaveform = rfTransmitter(txWaveform)`



Elaborate the RF System Model

Command Window

```
>> rfs
rfs =
    rfsystem with properties:

        ModelName: 'untitled'
        SampleTime: 1.0000e-06
        InputFrequency: 100000000
        OutputFrequency: 2.1000e+09
fx >> open_system(rfs)
```

Block Parameters: Modulator

Modulator

Model a Modulator

Main Impairments Nonlinearity IR Filter CS Filter

LO to Out isolation: 70 dB

Noise figure (dB): 2

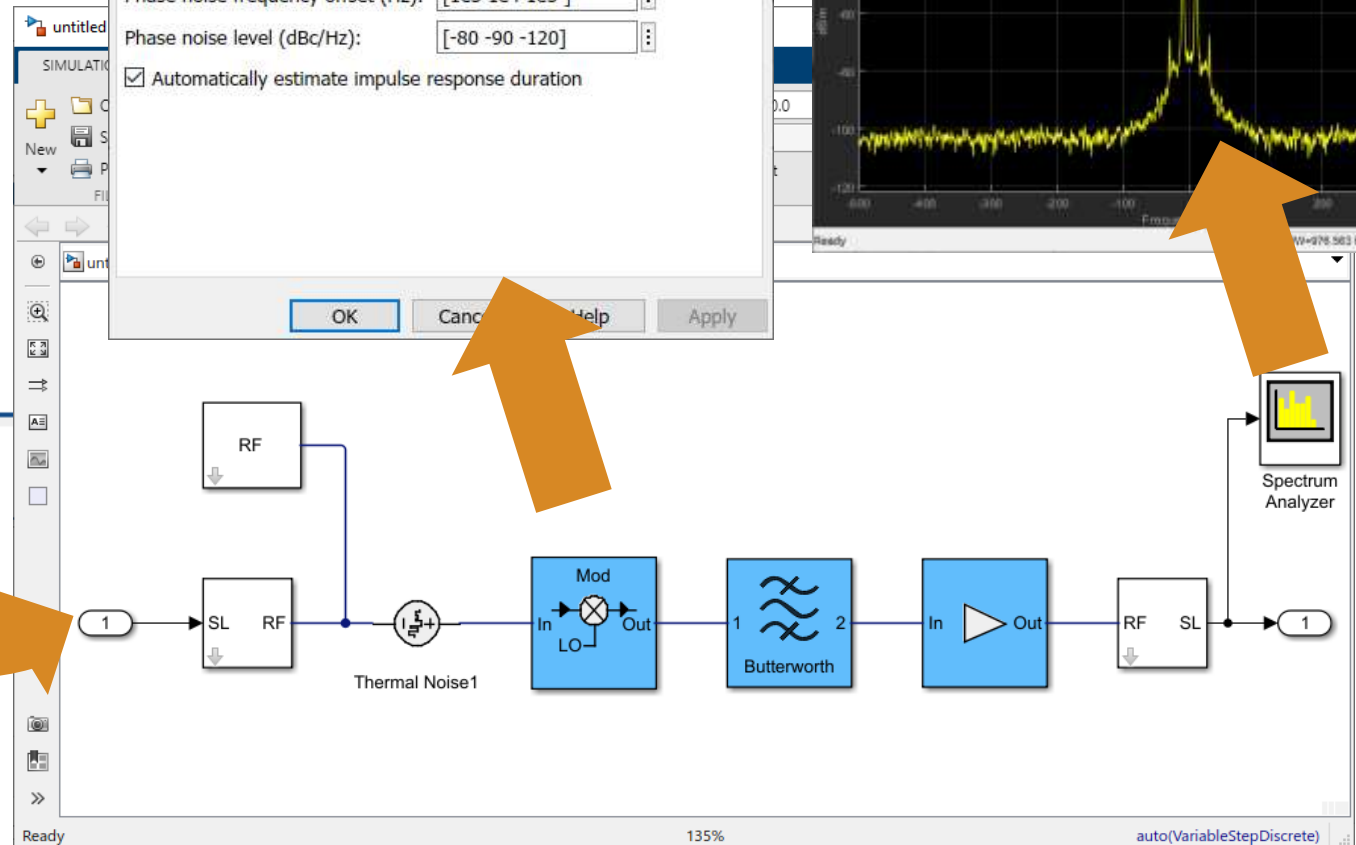
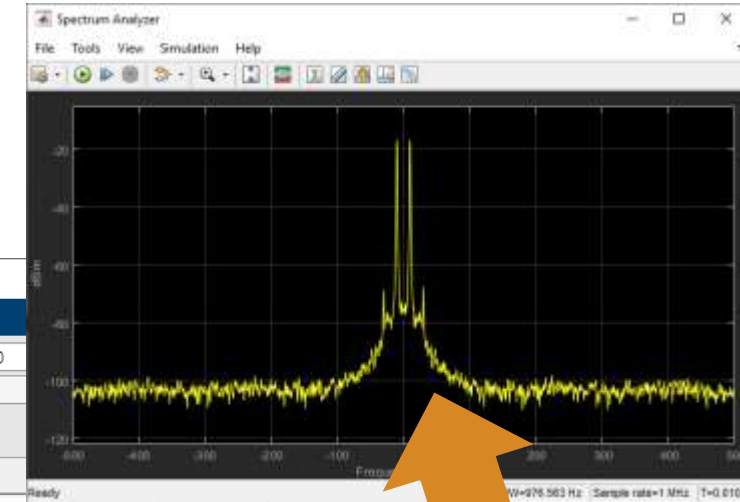
☒ Add phase noise

Phase noise frequency offset (Hz): [1e3 1e4 1e5]

Phase noise level (dBc/Hz): [-80 -90 -120]

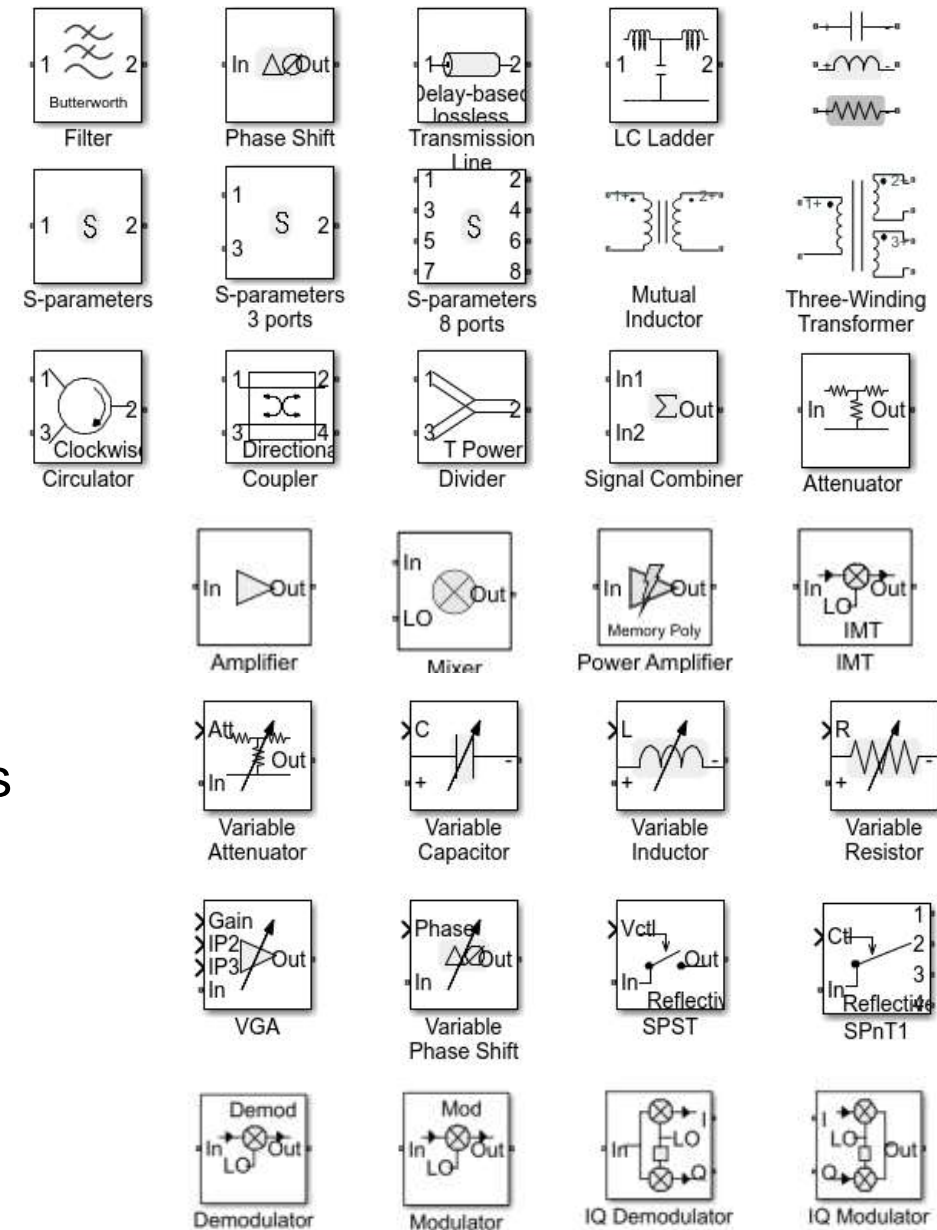
☒ Automatically estimate impulse response duration

OK Cancel Help Apply



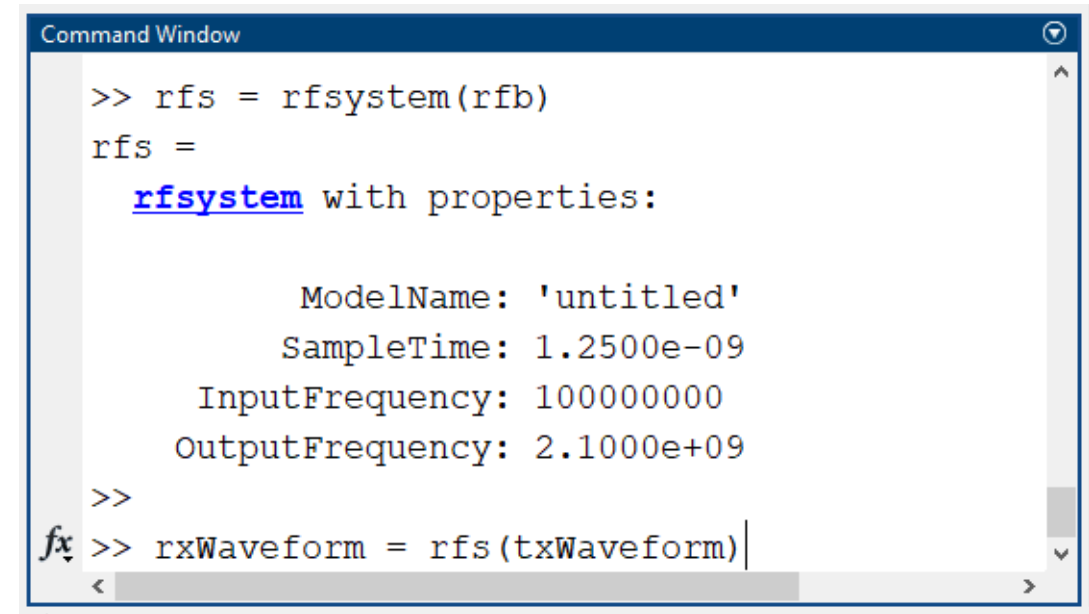
Circuit Envelope Available Blocks

- Frequency dependent (linear) elements
 - S-parameters, filter, attenuator, coupler, circulator, divider/combiner, phase shifter, RLC, transmission lines
- Nonlinear elements: Amplifier and mixer
 - Power amplifiers, mixers with IMT
- Noise generation
 - White noise, colored noise, phase noise
- Variable (Simulink controlled) elements for tunable networks
- Modulators, demodulators
 - Superhet and direct conversion
 - Including image rejection and channel selection filters
- Measurement testbenches
- Author your own component using Simscape language



Using RF System in MATLAB

- Input arguments are passed to the input ports
- Output port signals are returned as output arguments
- Impedance mismatches captured in the model
- Internal states (i.e. filters, S-parameters) are preserved across multiple simulations
- Direct conversion transceivers use 2 inputs, or 2 outputs to represent [I,Q] samples
- Multiple frequencies can be passed using vectorized input/outputs



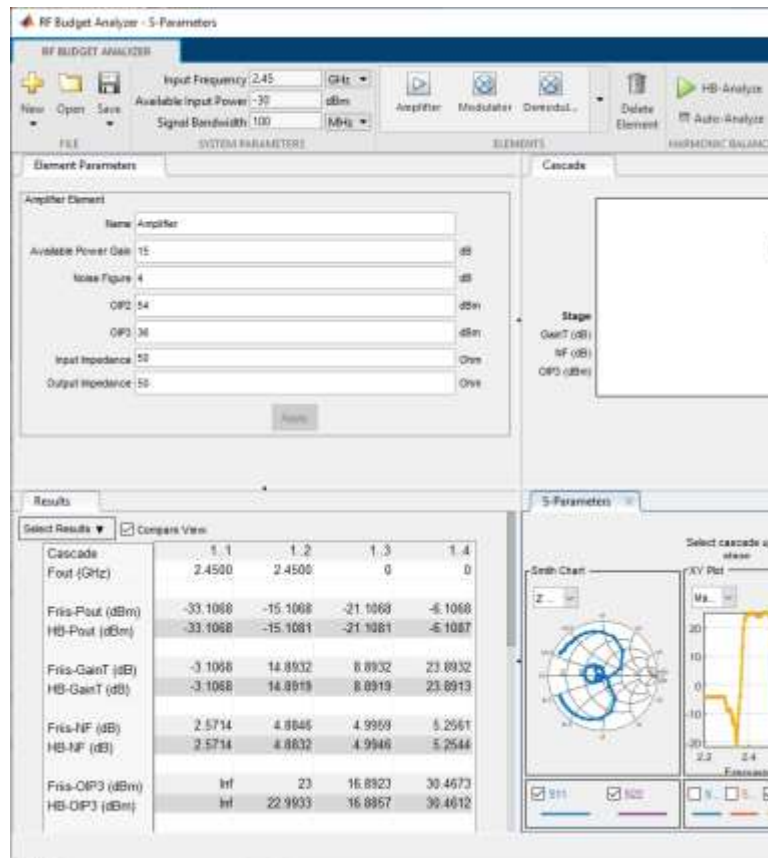
```
Command Window

>> rfs = rfsystem(rfb)
rfs =
    rfsystem with properties:

        ModelName: 'untitled'
        SampleTime: 1.2500e-09
        InputFrequency: 100000000
        OutputFrequency: 2.1000e+09

>>
fx >> rxWaveform = rfs(txWaveform)
```

Using RF System Objects in MATLAB: Direct Conversion



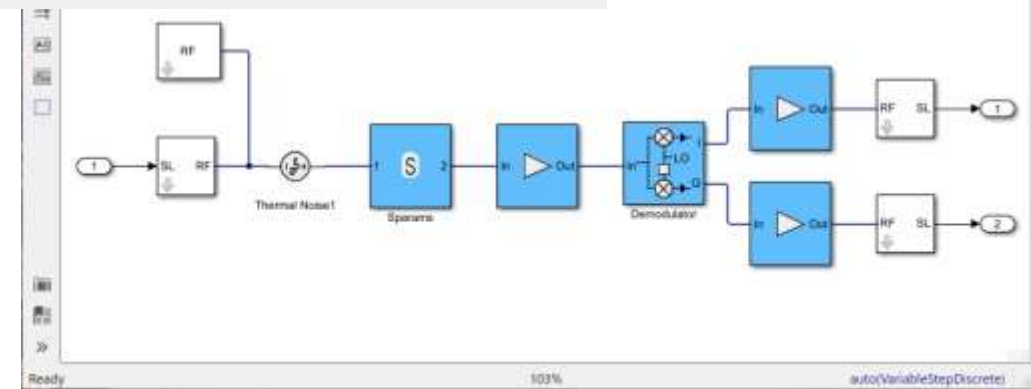
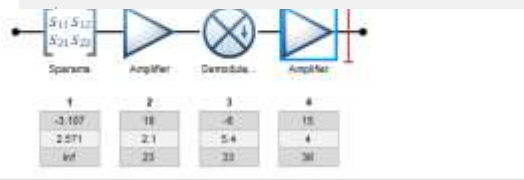
```

Command Window

>> rfs
rfs =
    rfsystem with properties:

        ModelName: 'ReceiverDirectConversion'
        SampleTime: 1.2500e-08
        InputFrequency: 2.4500e+09
        OutputFrequency: 0

fx >>
  
```



```
>> [I, Q] = rfs(input)
```

frequency dependency, non-linearity, noise, mismatches

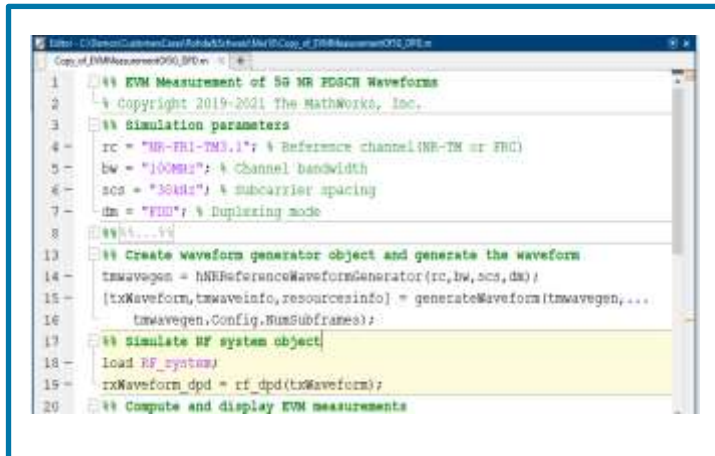
standard compliant, spectrum and time

interference, clutter, noise

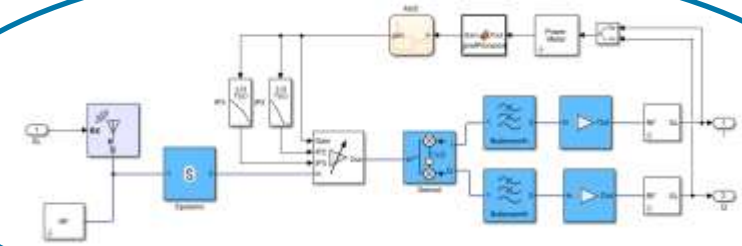
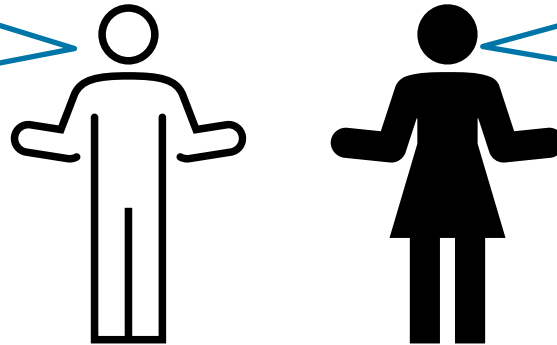


Conclusions

- Combine the power of MATLAB for signal processing with accurate RF system simulation
- Integrate RF Circuit Envelope models into MATLAB using a straightforward interface
- Use Simulink and RF Blockset to elaborate the RF transceiver models

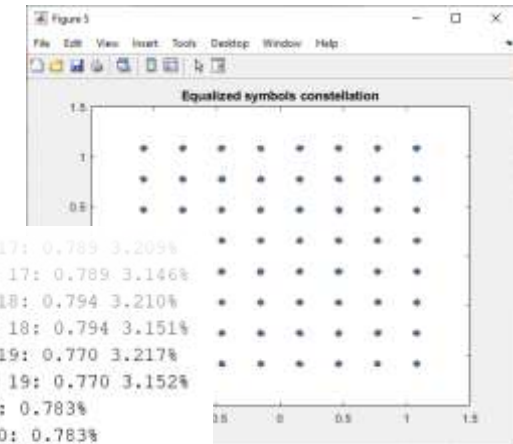
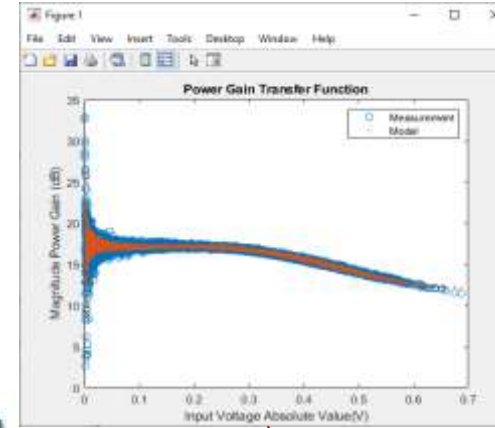
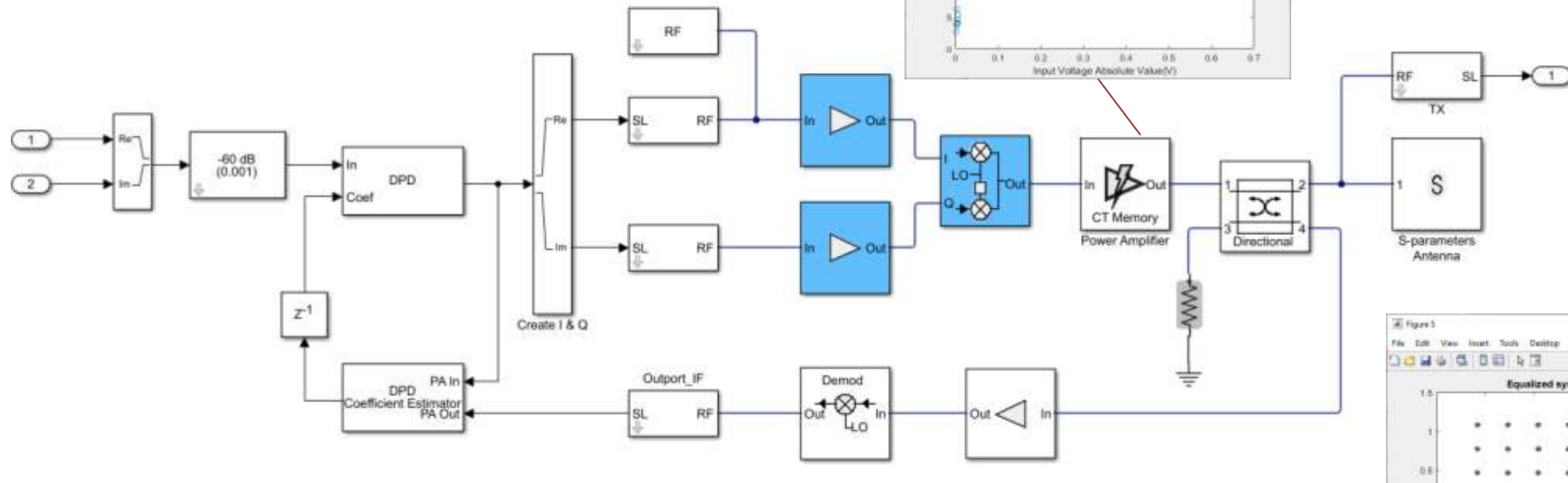


```
1 %% EVM Measurement of 5G NR FDSCN Waveforms
2 % Copyright 2019-2021 The MathWorks, Inc.
3 %% Simulation parameters
4 rc = 'NR-FR1-TM3.1'; % Reference channel (NR-TM or FDD)
5 bw = '100MHz'; % Channel bandwidth
6 scs = '30kHz'; % Subcarrier spacing
7 dm = 'FDD'; % Duplexing mode
8
9 %% ...
10
11 %% Create waveform generator object and generate the waveform
12 txwaven = hnrReferenceWaveformGenerator(rc,bw,scs,dm);
13 [txWaveform,txWaveInfo,resourceInfo] = generateWaveform(txwaven,...
14 txwaven.Config.NumSubframes);
15
16 %% Simulate RF system object
17 load RF_system;
18 rxWaveform_dpd = rf_dpd(txWaveform);
19
20 %% Compute and display EVM measurements
```



Supplementary Slides

What Else Can we Model?



```
>> out = rfDPD(I, Q)
```

```
Low edge RMS EVM, Peak EVM, slot 17: 0.789 3.209%
High edge RMS EVM, Peak EVM, slot 17: 0.789 3.146%
Low edge RMS EVM, Peak EVM, slot 18: 0.794 3.210%
High edge RMS EVM, Peak EVM, slot 18: 0.794 3.151%
Low edge RMS EVM, Peak EVM, slot 19: 0.770 3.217%
High edge RMS EVM, Peak EVM, slot 19: 0.770 3.152%
Averaged low edge RMS EVM, frame 0: 0.783%
Averaged high edge RMS EVM, frame 0: 0.783%
Averaged 3GPP EVM frame 0: 0.783%
Averaged overall RMS EVM: 0.783%
Peak EVM = 3.7347%
```


Using RF System Objects in MATLAB: Multiple Frequencies

Command Window

```
rfs =
    rfsystem with properties:

        ModelName: 'ReceiverLowIF'
        SampleTime: 2.0000e-08
        InputFrequency: [2.4000e+09 2.5000e+09]
        OutputFrequency: 50000000
fx >> |
```

RF BUDGET ANALYZER - Results

Input Frequency: 2.4 GHz
Available Input Power: -30 dBm
Signal Bandwidth: 1 MHz

Stage	1	2	3	4
GainT (dB)	-2.4	18	-6	15
MF (dB)	2.275	3.1	3.4	4
OP3 (dBm)	inf	23	33	38

Cascade	1.1	1.2	1.3	1.4
Fout (GHz)	2.4000	2.4000	0.0500	0.0500
Fris-Post (dBm)	-32.3996	-14.3996	-20.3996	-5.3996
Fris-GainT (dB)	-2.3996	15.6004	9.6004	24.6004
Fris-MF (dB)	2.2754	4.4235	4.5287	4.7753
Fris-OP3 (dBm)	inf	23	16.8923	30.4673
Fris-SNR (dB)	81.6997	79.5517	79.4464	79.1998

LO = 2.45GHz

```
>> out = rfs([input image])
```