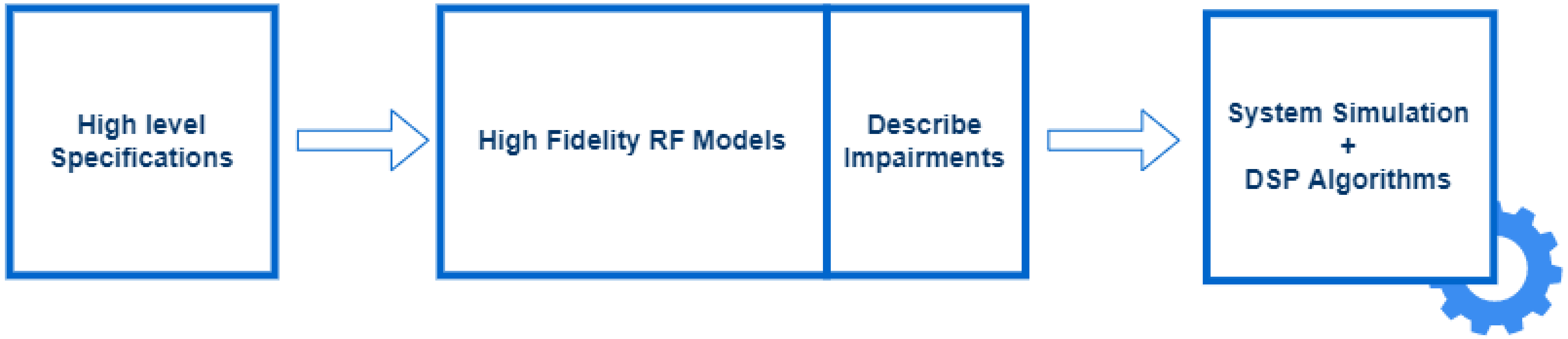
Three Steps to Simulate RF Transceivers in MATLAB

Sekhar Sekharan
Application Engineer

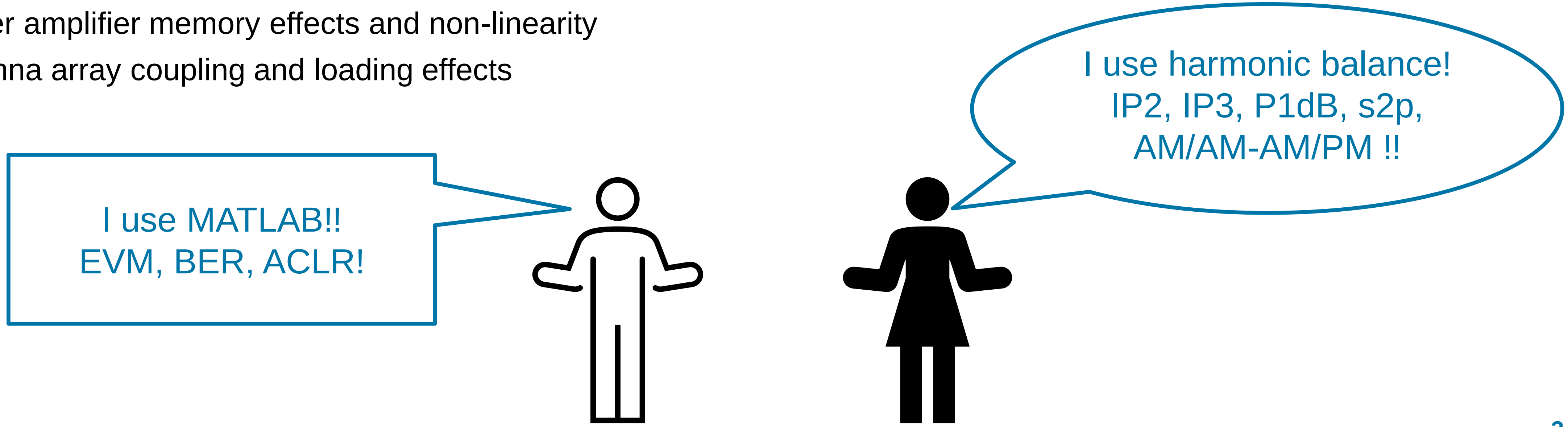
Agenda

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

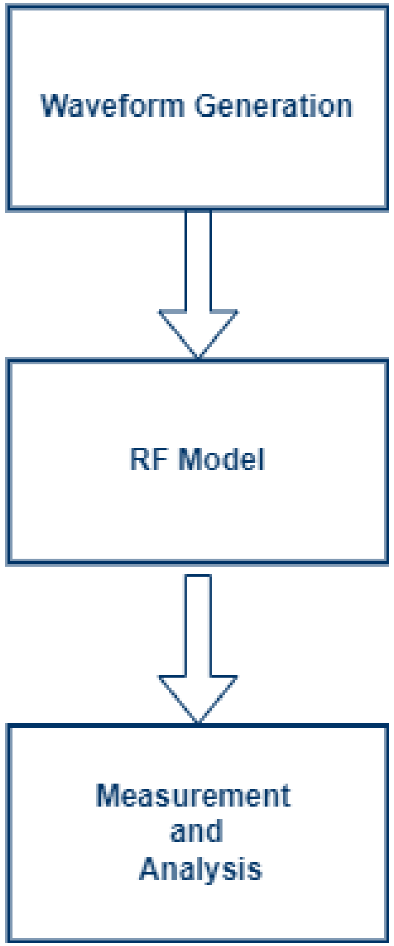


Motivation:
Connect Baseband Signal Processing with High Fidelity RF Simulation

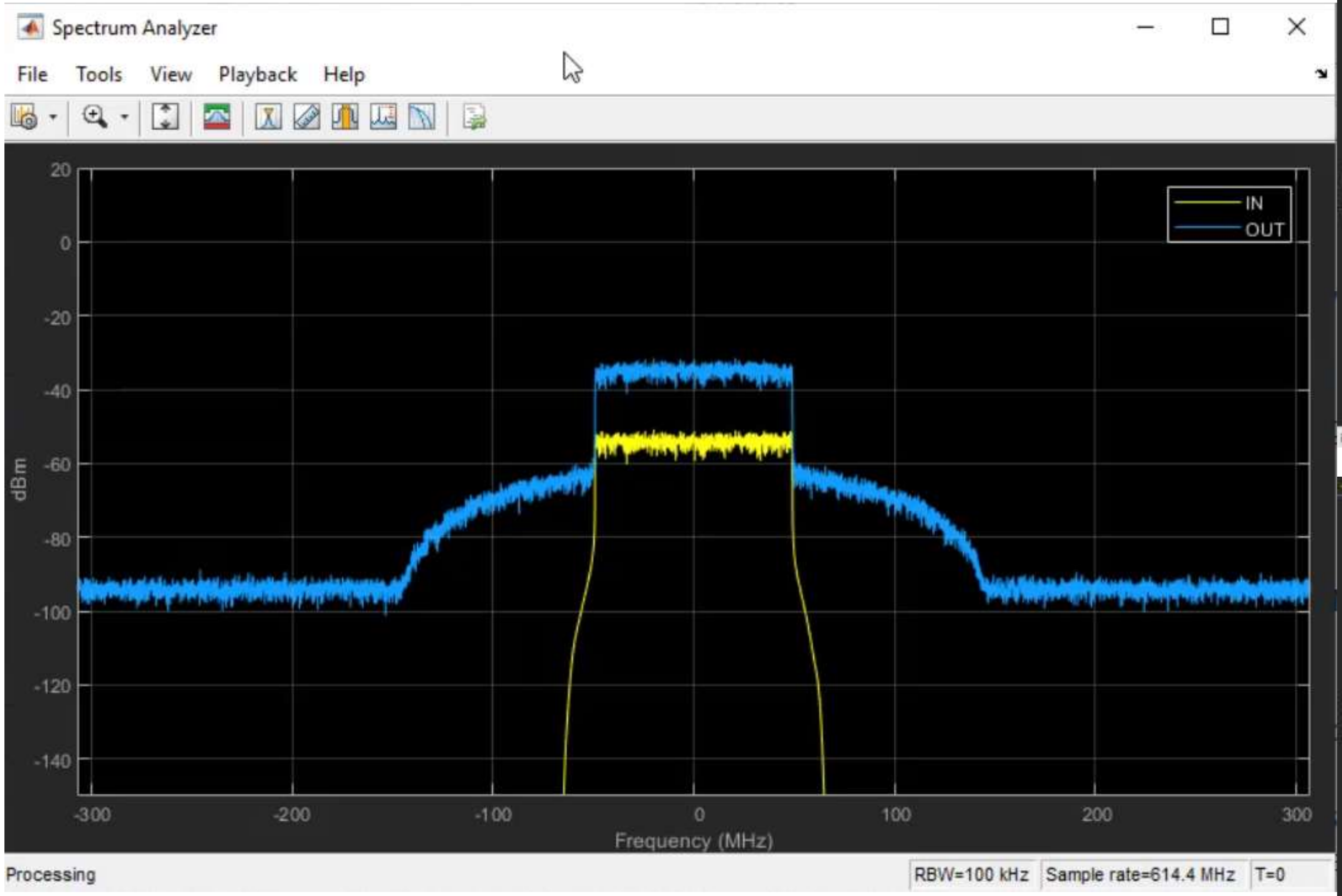
- Use MATLAB code for baseband signal processing
  - Radar and wireless communications standards
  - Control, calibration, and correction algorithms
- Increase the fidelity of the RF transceiver models, to model the following:
  - Impedance mismatches, reverse isolation, S-parameter data
  - Out of band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



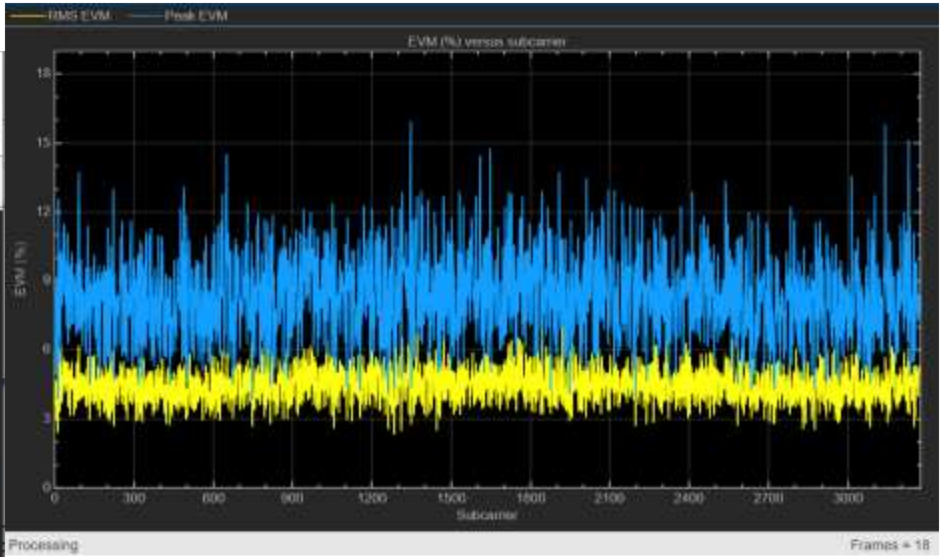
Demo



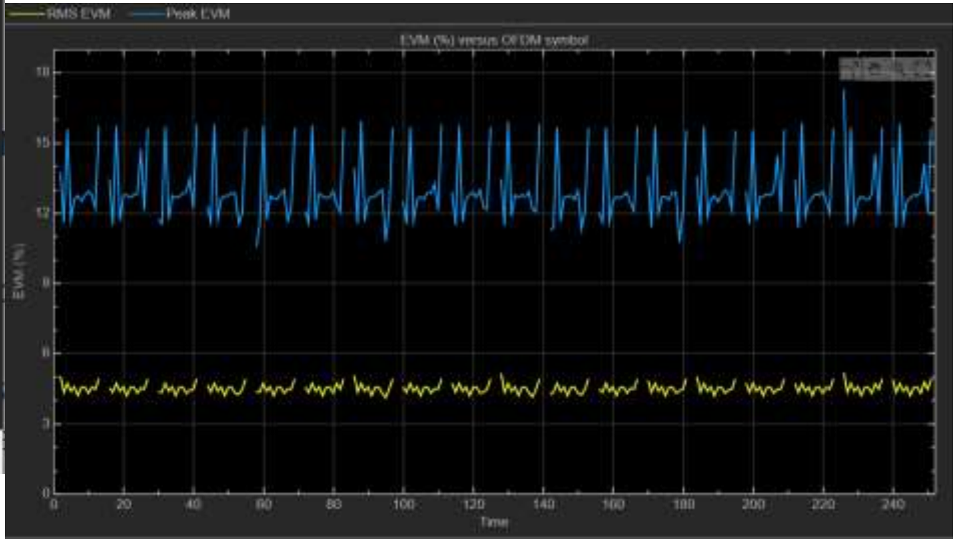
https://www.mathworks.com/help/5g/ug/evm-measurement-of-5g-nr-downlink-waveforms.html



Spectrum Analyzer
File Tools View Playback Help
IN
OUT
dBm
Frequency (MHz)
Processing
RBW=100 kHz
Sample rate=614.4 MHz
T=0



RMS EVM
Peak EVM
EVM (%) versus subcarrier
Subcarrier
Processing



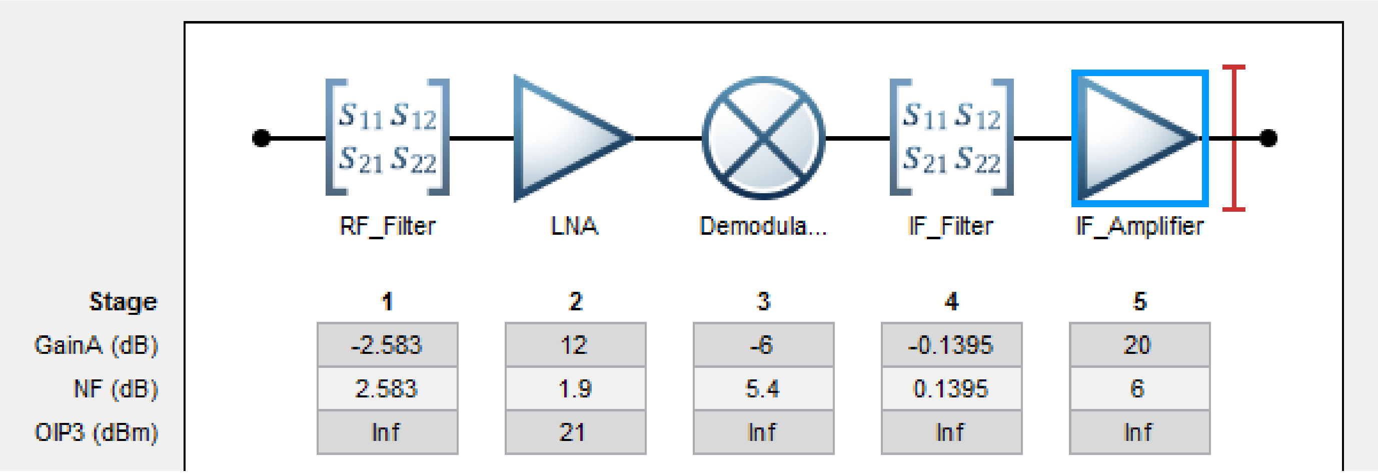
RMS EVM
Peak EVM
EVM (%) versus OFDM symbol
Time

Introducing RF Transceivers Design

RF Transceiver Design – Where To Start?

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



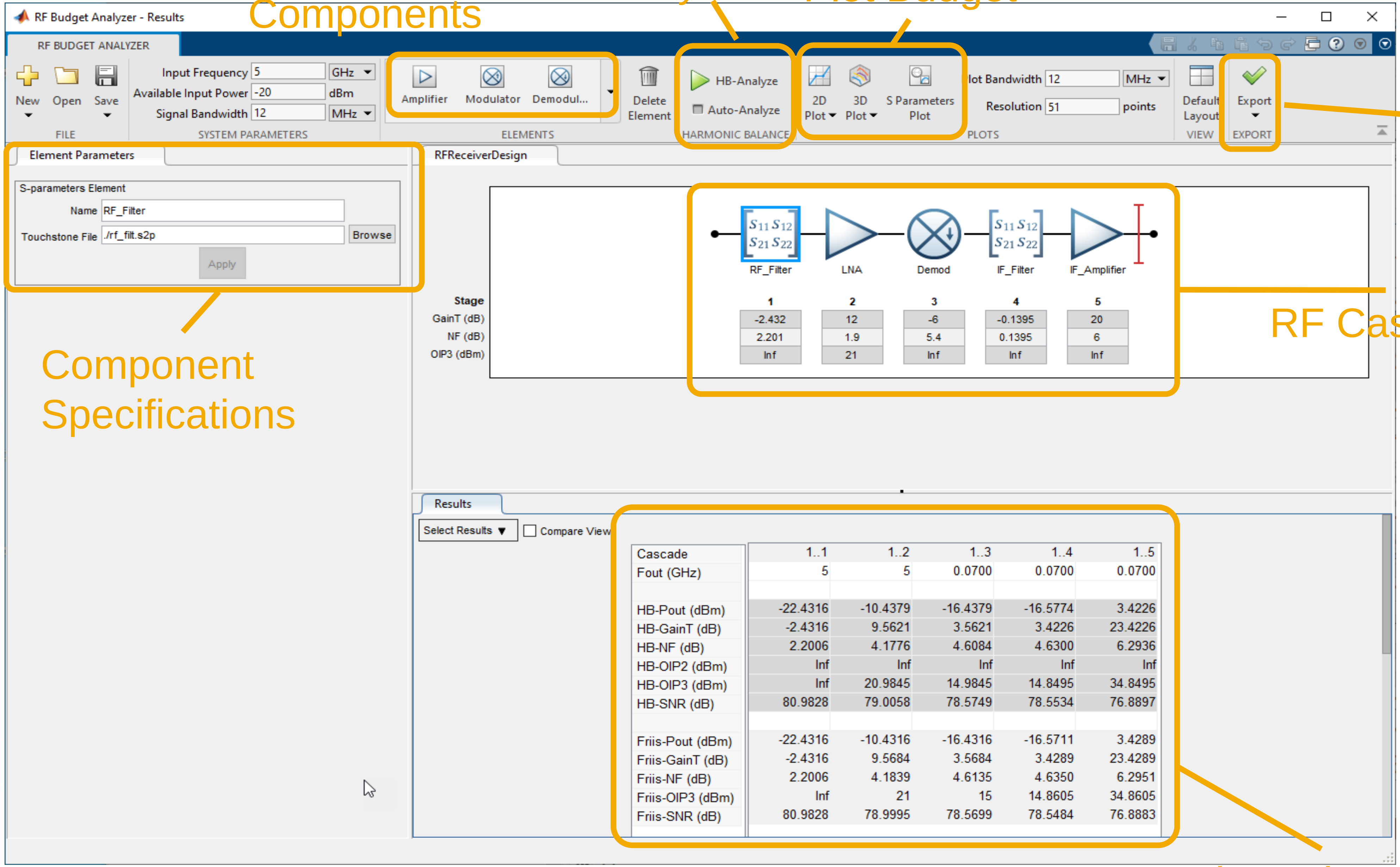
Stage	1	2	3	4	5
GainA (dB)	-2.583	12	-6	-0.1395	20
NF (dB)	2.583	1.9	5.4	0.1395	6
OIP3 (dBm)	Inf	21	Inf	Inf	Inf

Add RF Components

HB Analysis

Plot Budget

Export to RF Blockset

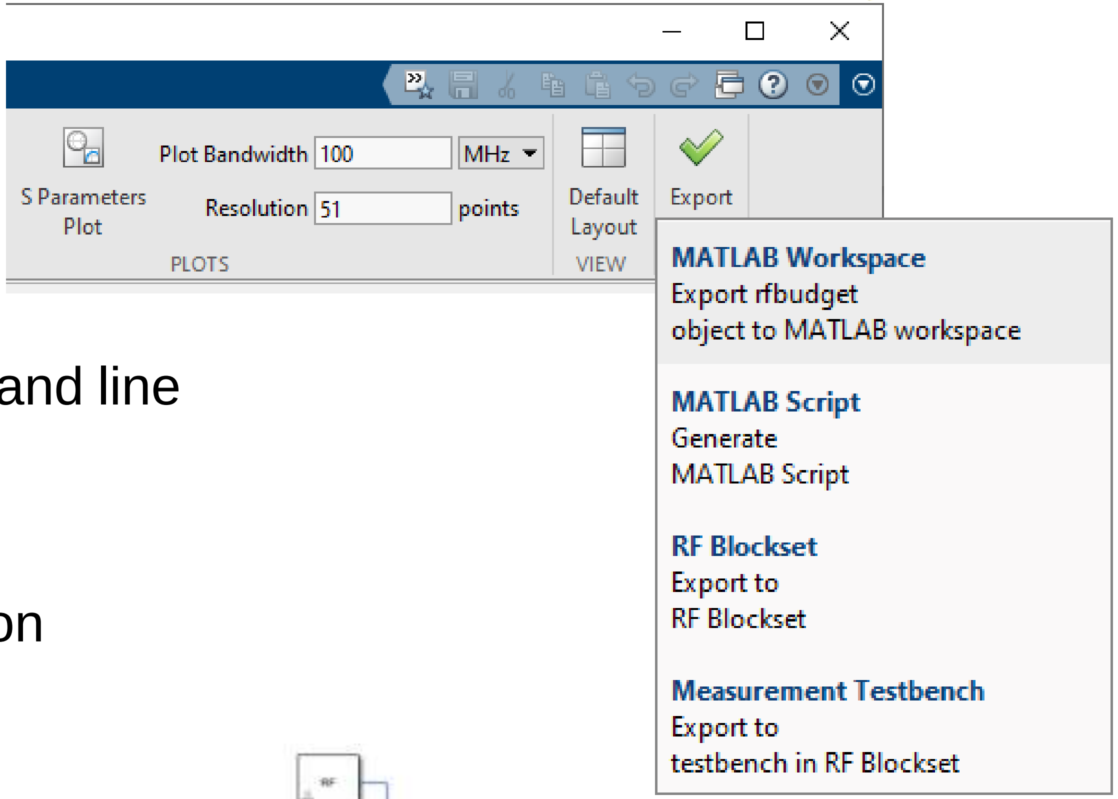


Component Specifications

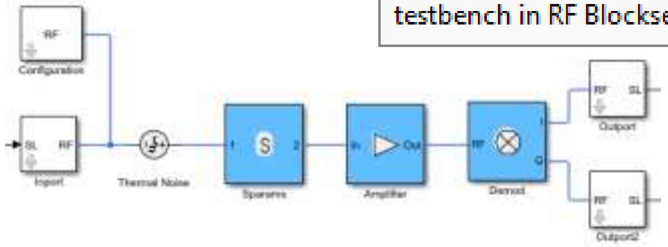
RF Cascade

Cascade Budget Analysis

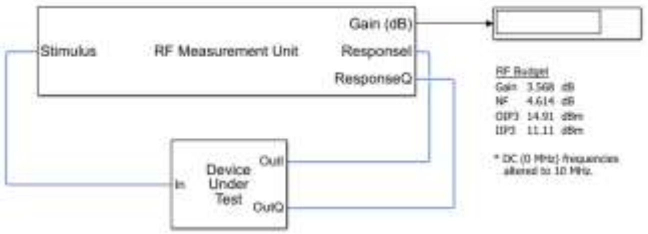
Export from RF Budget Analyzer



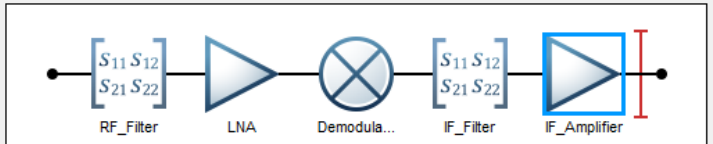
- Export to MATLAB workspace object
  - Bidirectional workflow between app and command line
- Export to MATLAB script
  - Automate corner analysis and chain optimization
- Export to RF Blockset
  - Generate model for HB and Circuit Envelope simulation
  - Automatic configuration of Input/output ports



- 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



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

```
>> 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

```
>> 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]
```

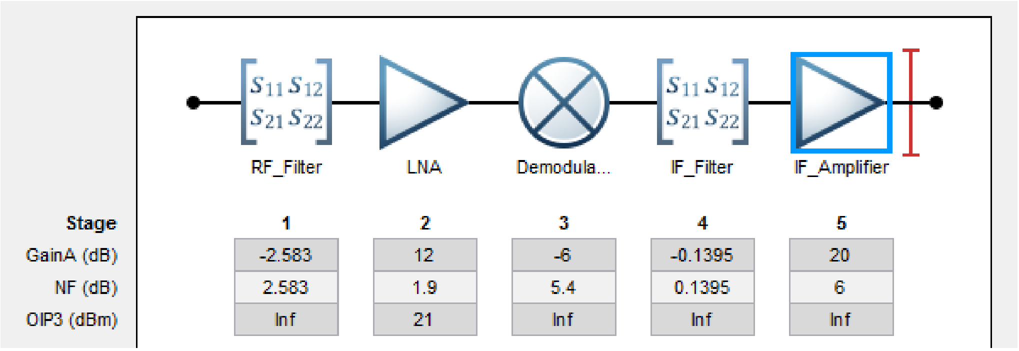
```
m = modulator('Gain',3,'OIP2',20,'ImageReject',false,'ChannelSelect',false);
m2 = modulator('Gain',3,'OIP2',20,'ImageReject',false,'ChannelSelect',false);
b = rfbudget([m m2],2.1e9,-30,100e6,'Solver','HarmonicBalance')
```

```
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)  []
```

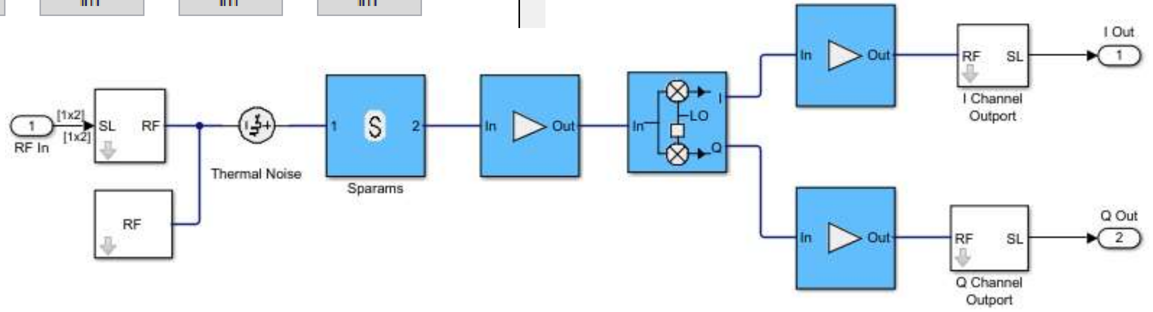
Elaborating RF Transceivers Models

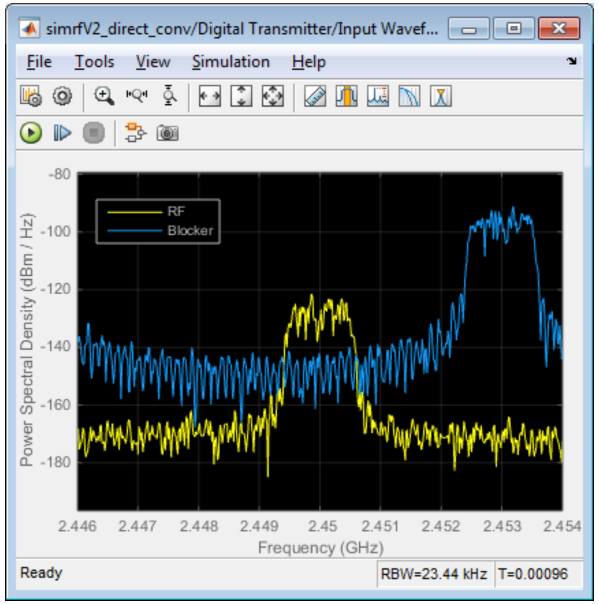
RF System-Level Models Are Necessary For RF Design

- 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

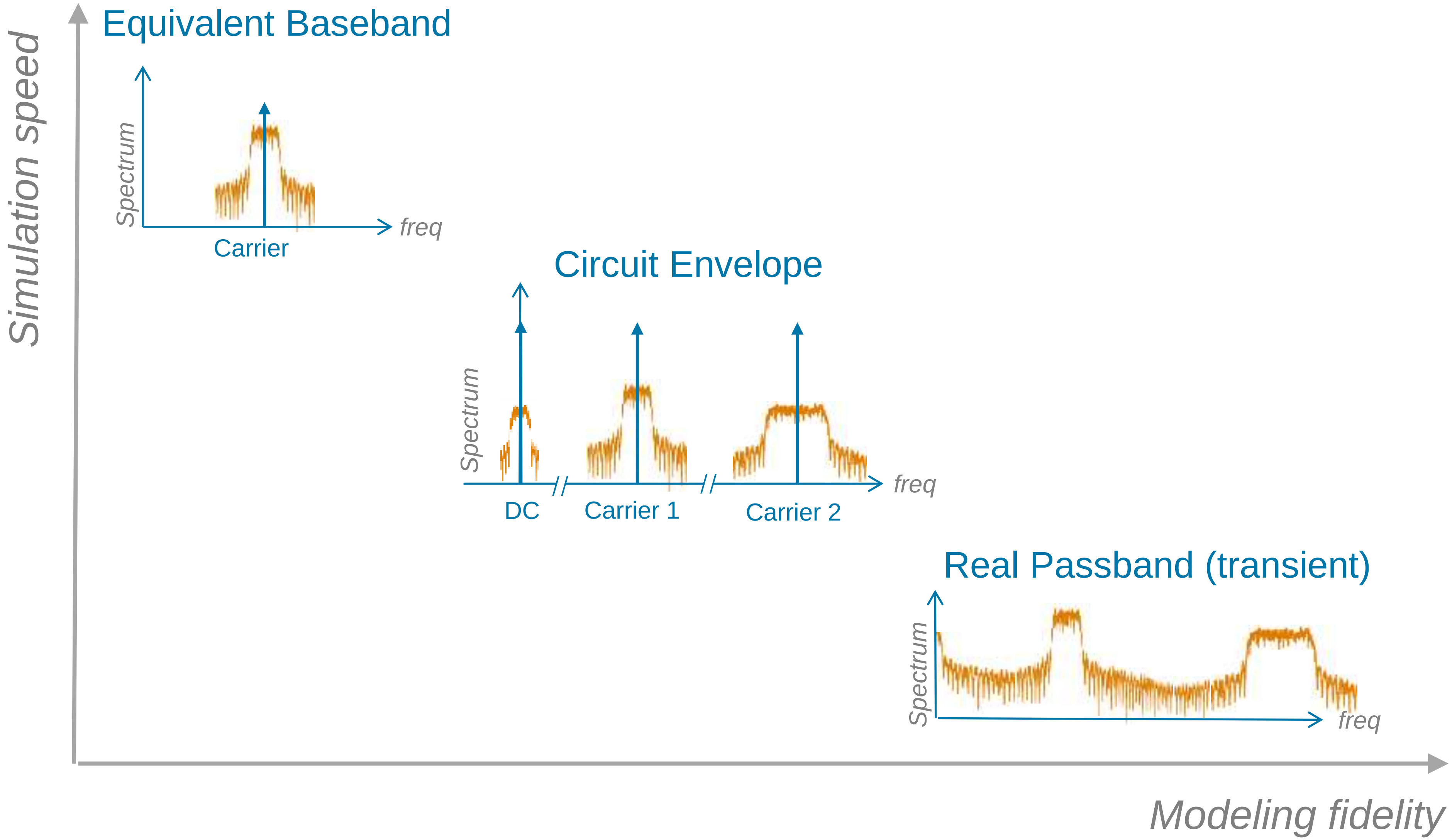


Stage	1	2	3	4	5
GainA (dB)	-2.583	12	-6	-0.1395	20
NF (dB)	2.583	1.9	5.4	0.1395	6
OIP3 (dBm)	Inf	21	Inf	Inf	Inf





RF System Simulation Must Be Fast (and Accurate)

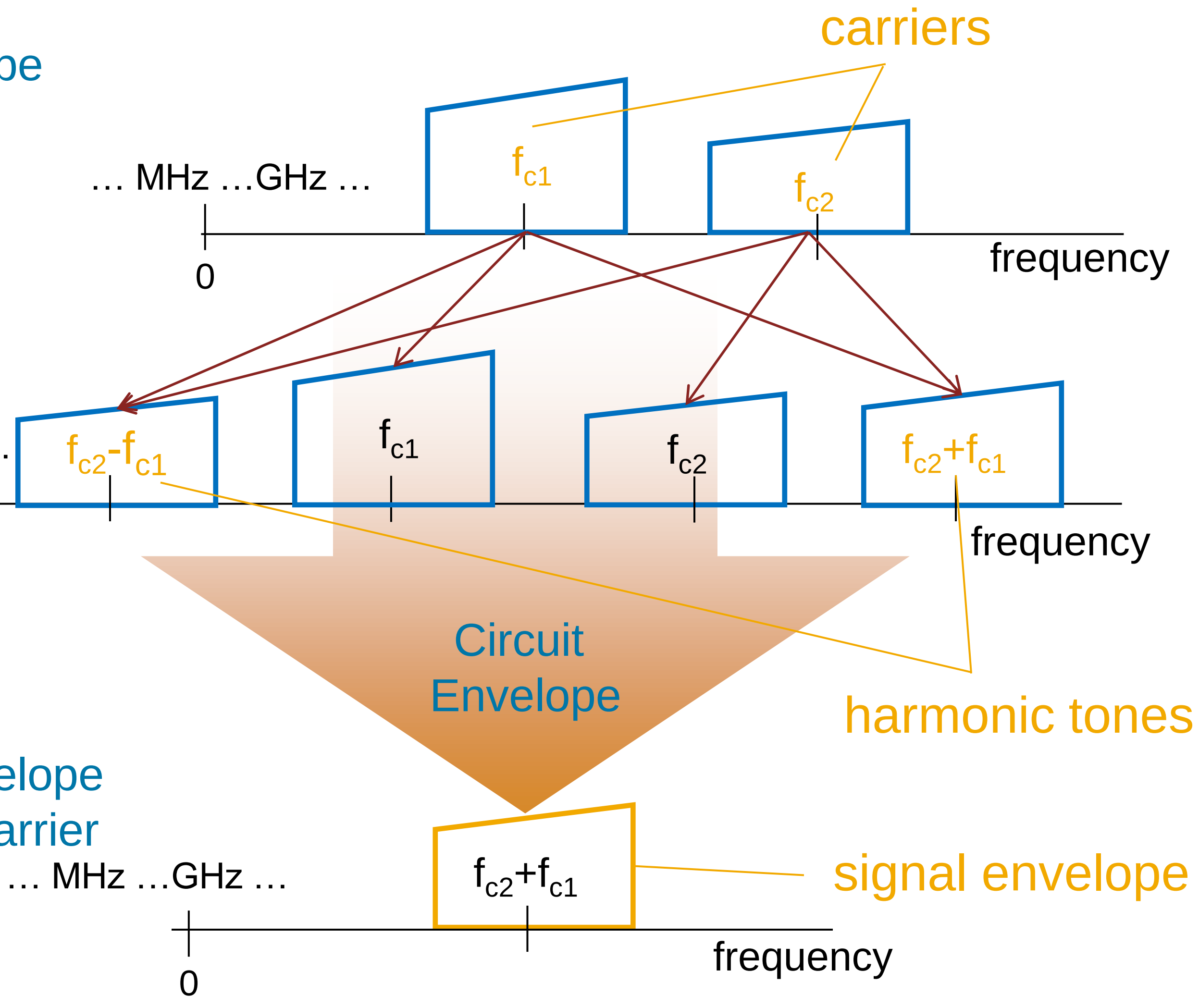


Multi-Carrier Circuit Envelope Simulation

Input port: convert complex envelope modulation into RF signal

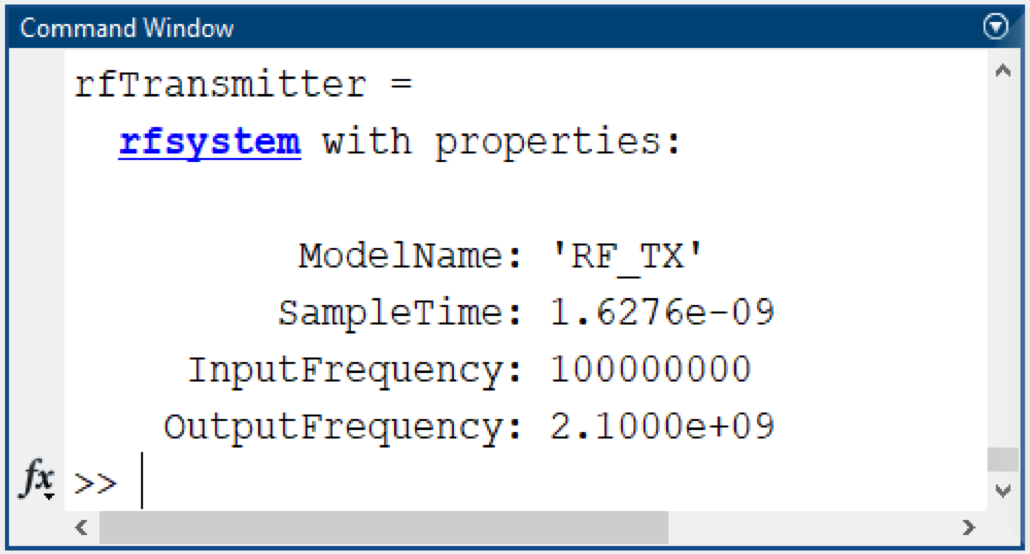
Configuration: specify the harmonic order

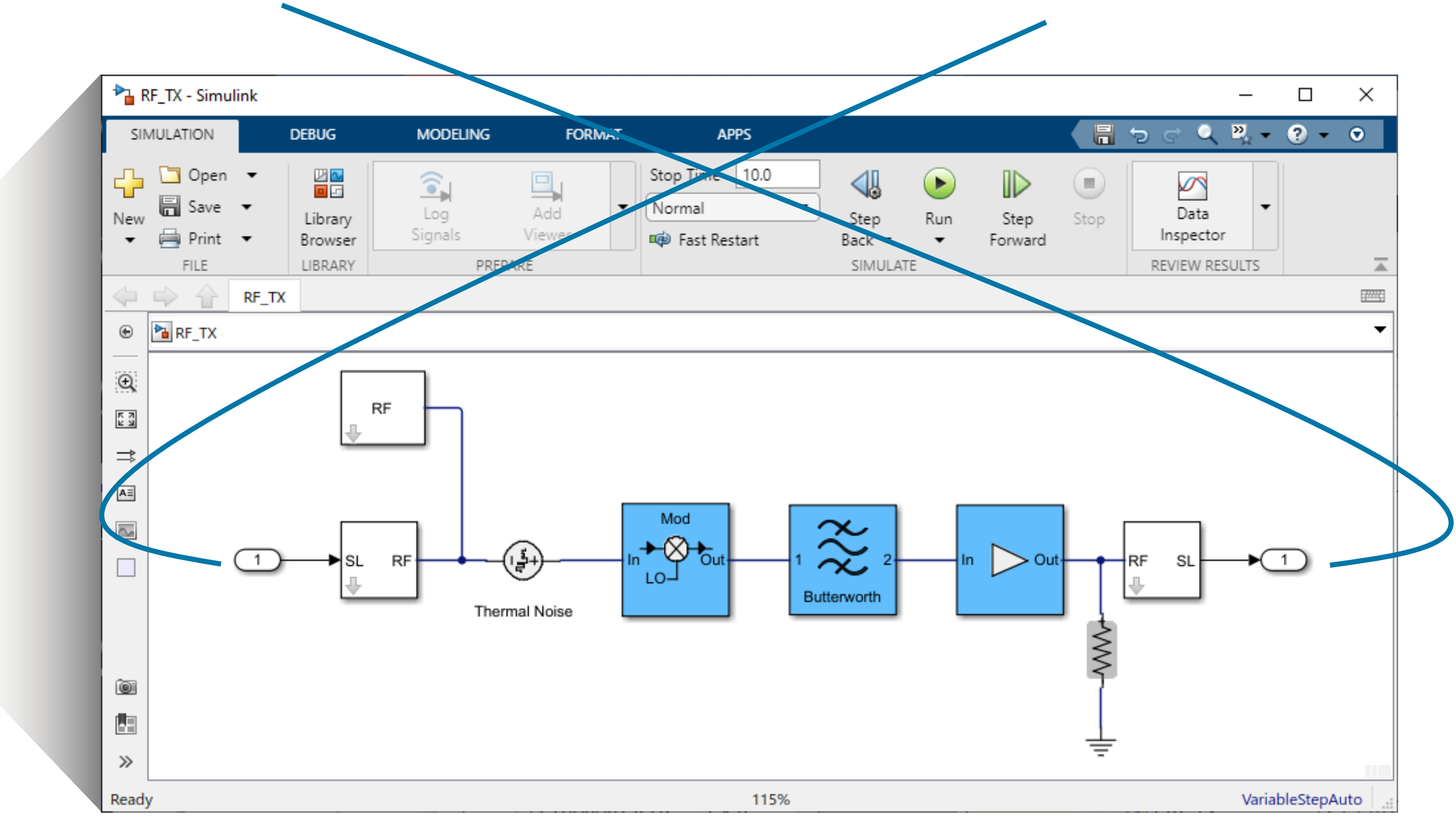
Output port: select complex envelope response around the selected carrier



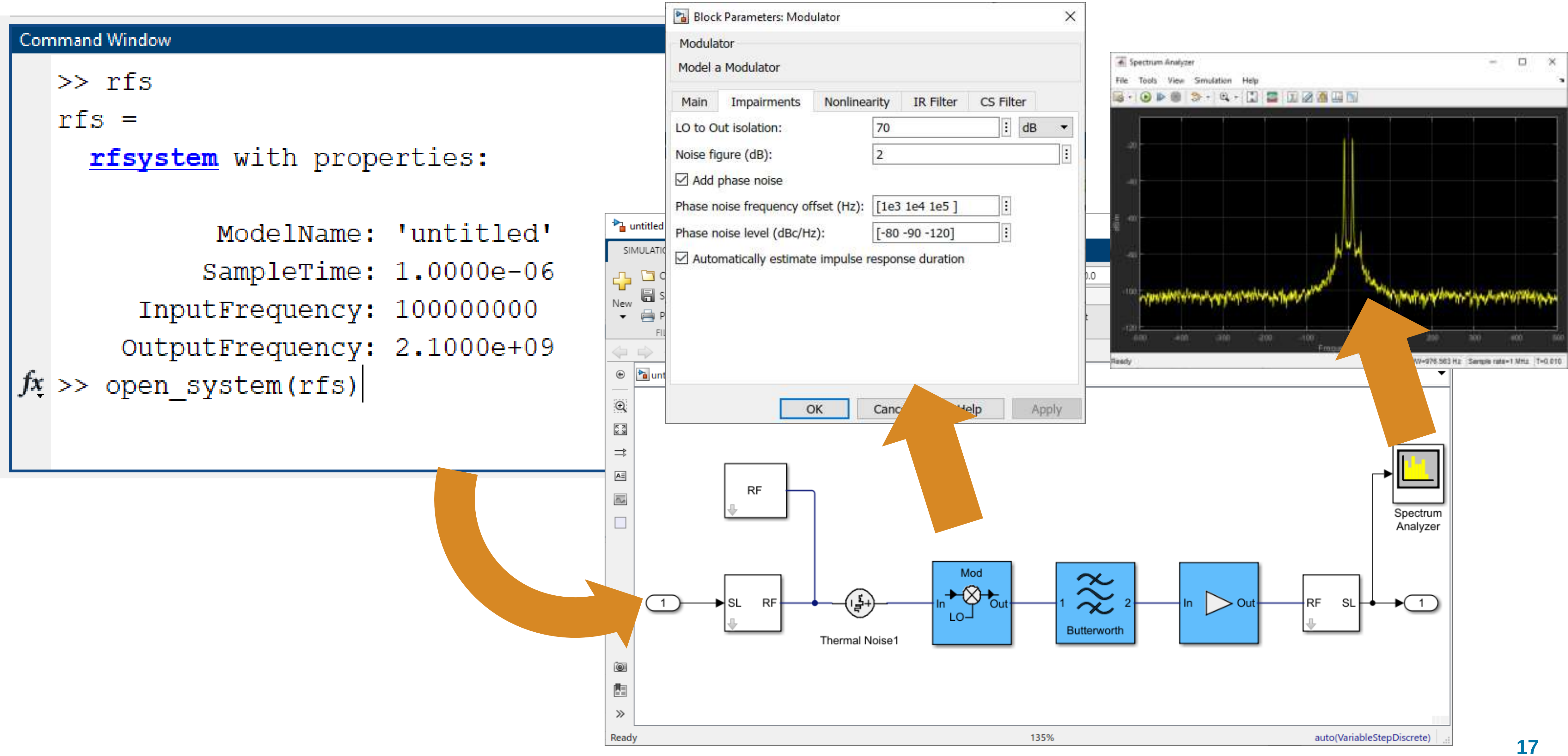
Circuit Envelope: the Technology Behind RF System Objects

```
rxWaveform = rfTransmitter(txWaveform)
```



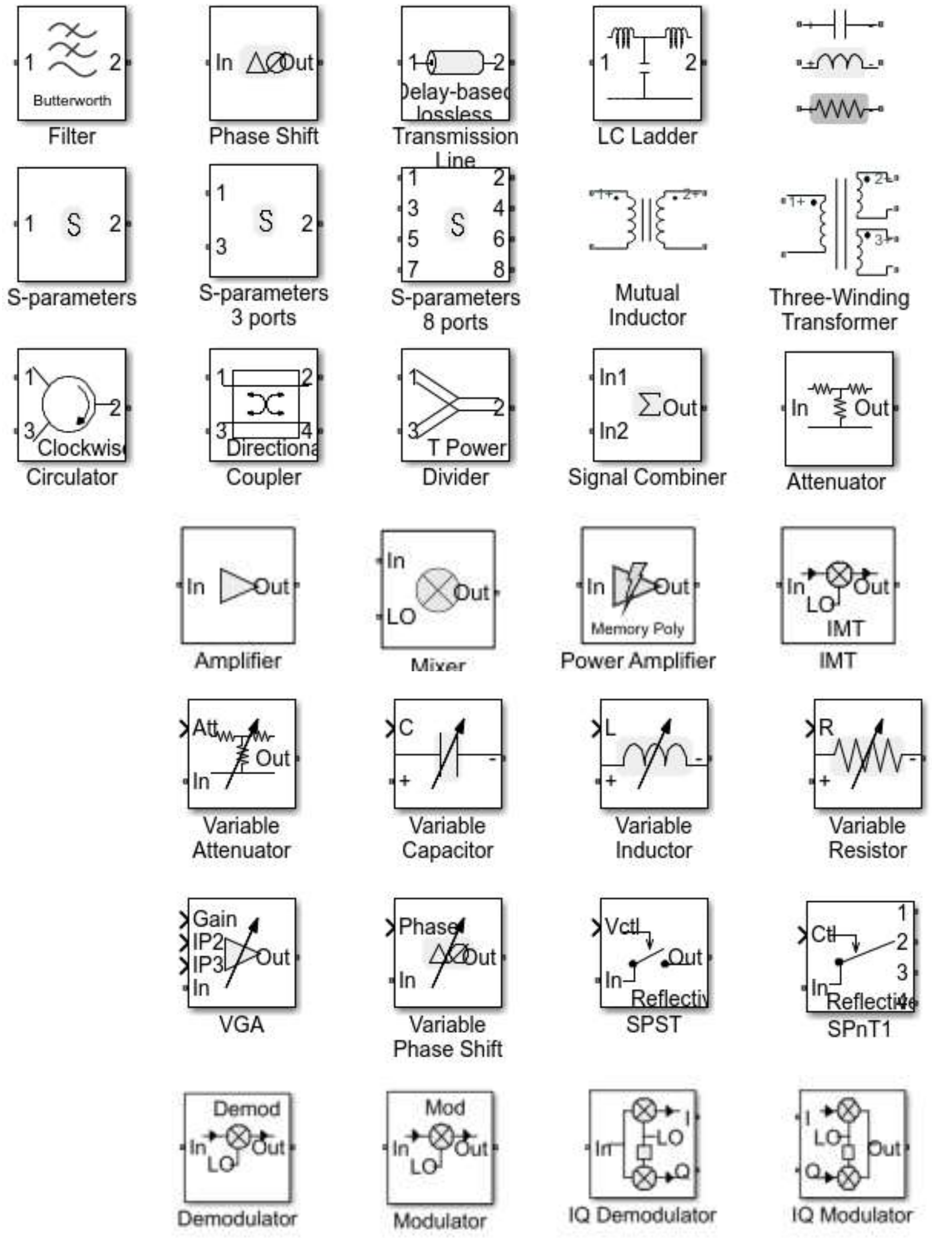


Elaborate the RF System Model



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

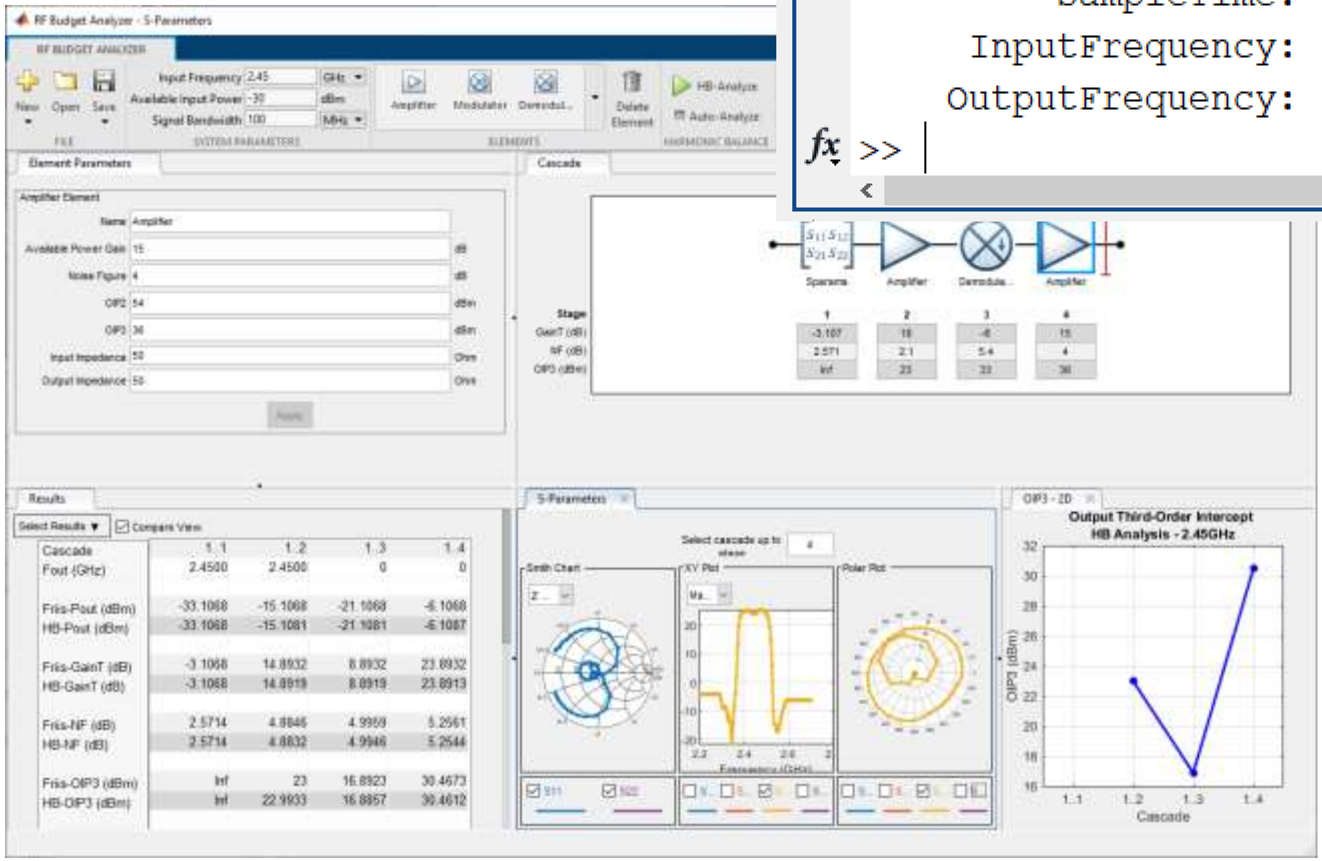
Command Window

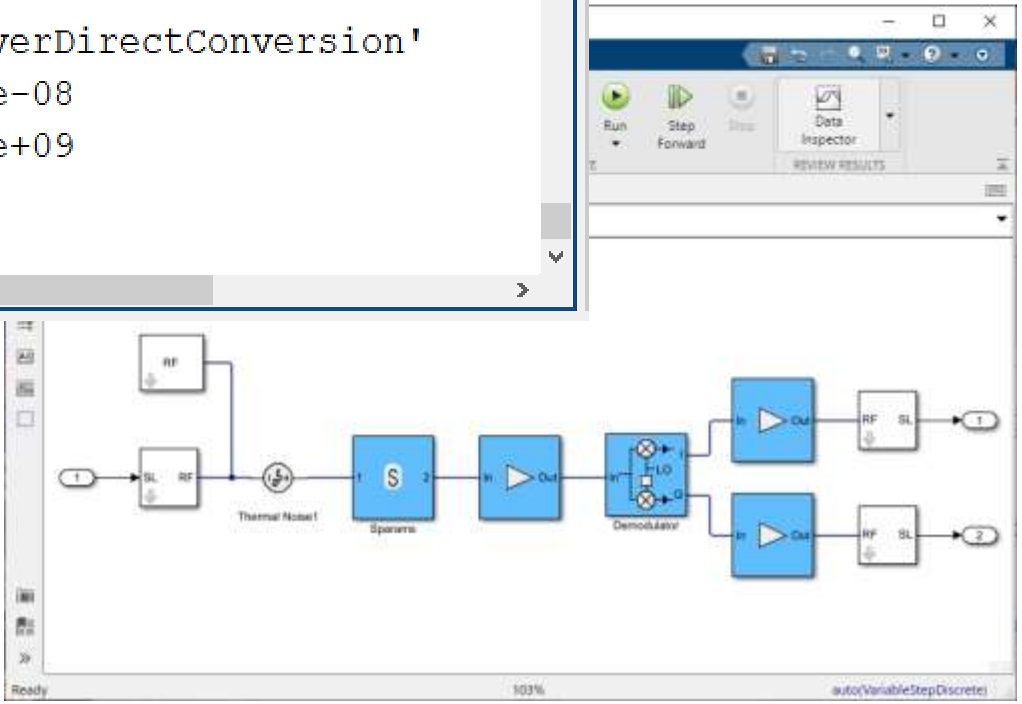
```
>> rfs = rfsystem(rfb)
rfs =
  rfsystem with properties:

          ModelName: 'untitled'
         SampleTime: 1.2500e-09
     InputFrequency: 100000000
    OutputFrequency: 2.1000e+09
>>
>> rxWaveform = rfs(txWaveform)
```

Using RF System Objects in MATLAB: Direct Conversion

Command Window

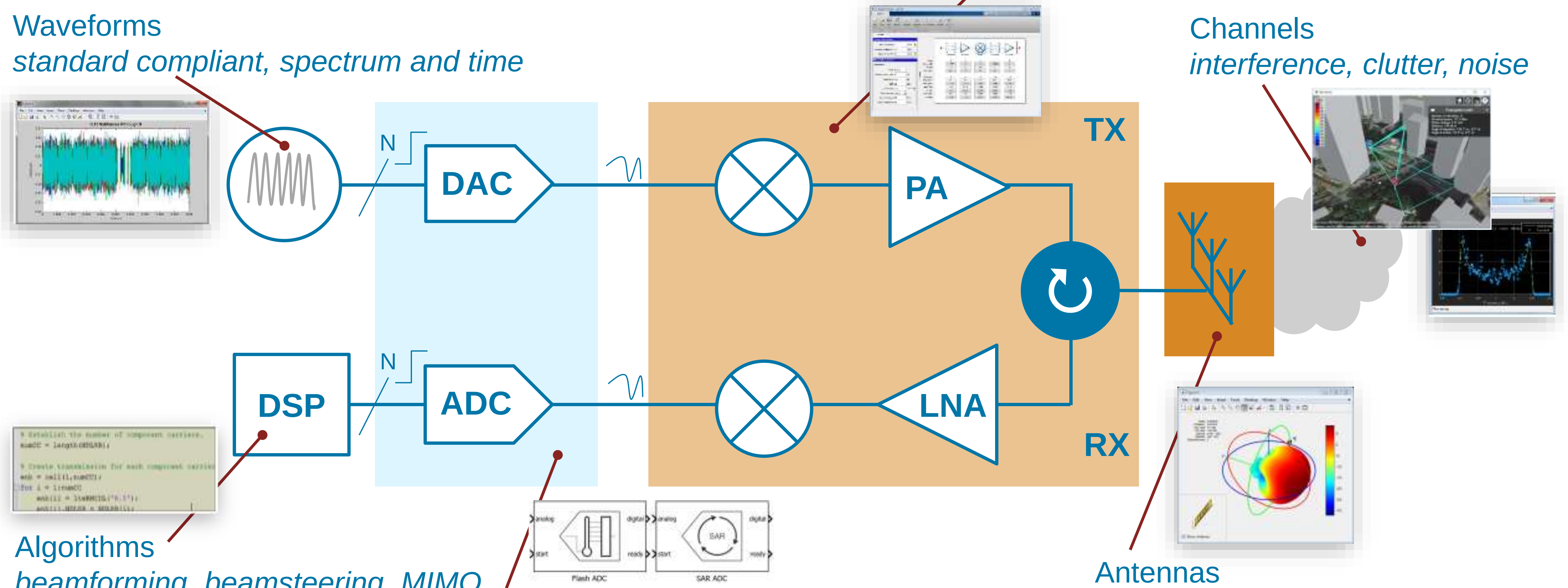
```
>> rfs
rfs =
  rfsystem with properties:

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

>>
```





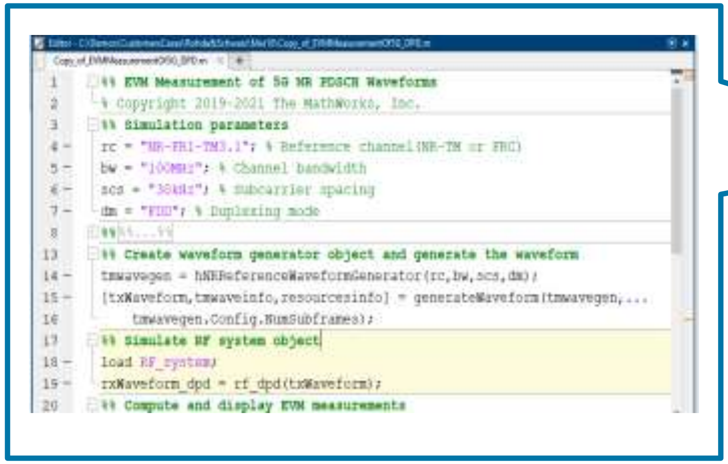
```
>> [I, Q] = rfs(input)
```

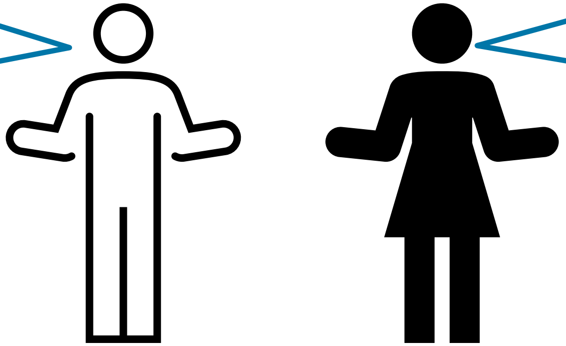
Vision:
Model and Simulate Wireless Systems

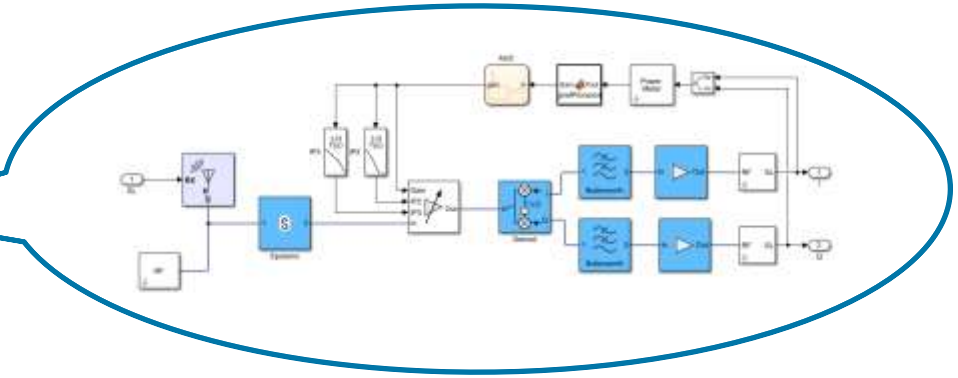


Conclusions

- 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

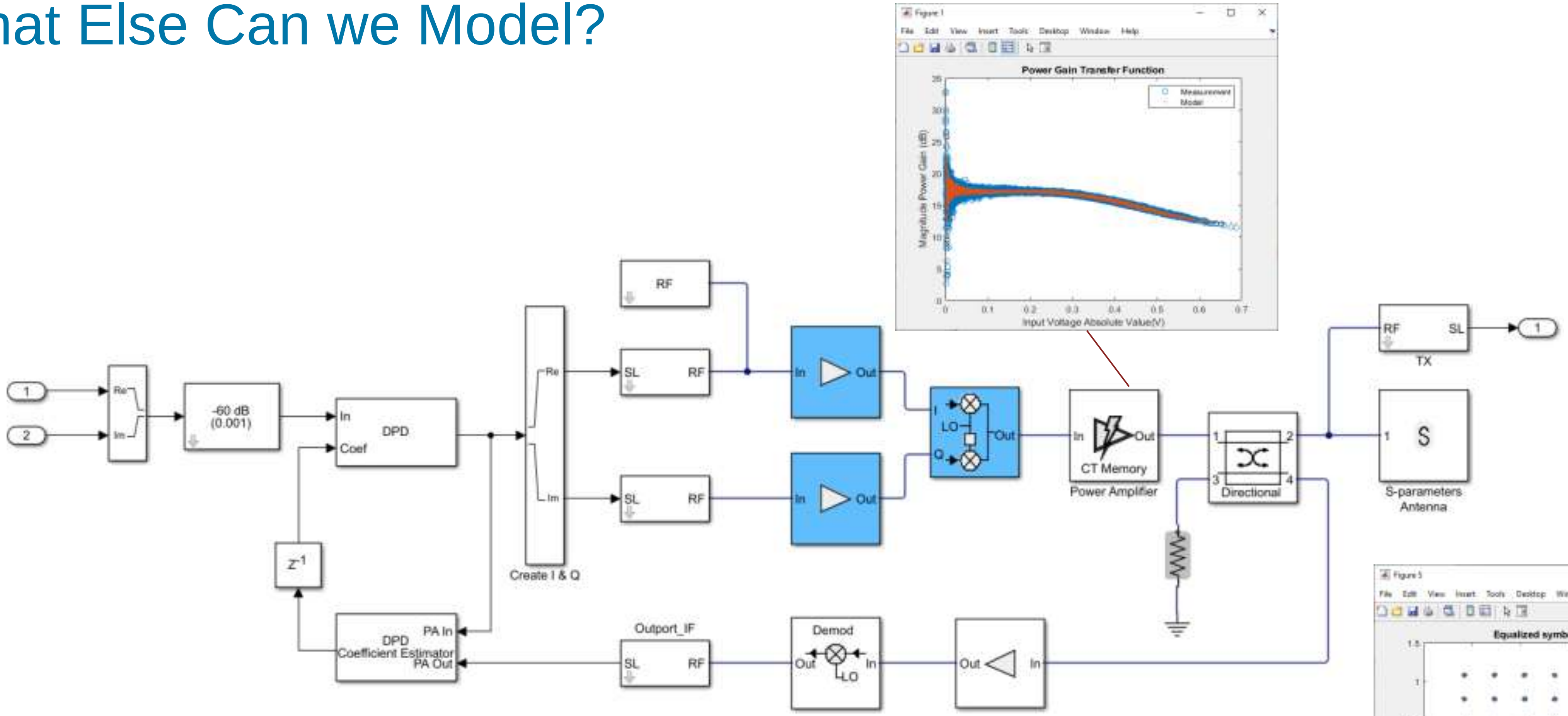


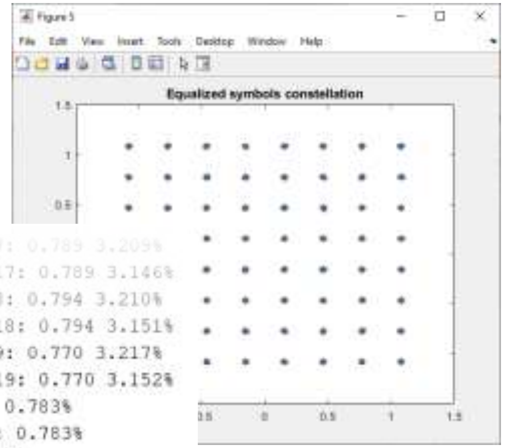




Supplementary Slides

What Else Can we Model?



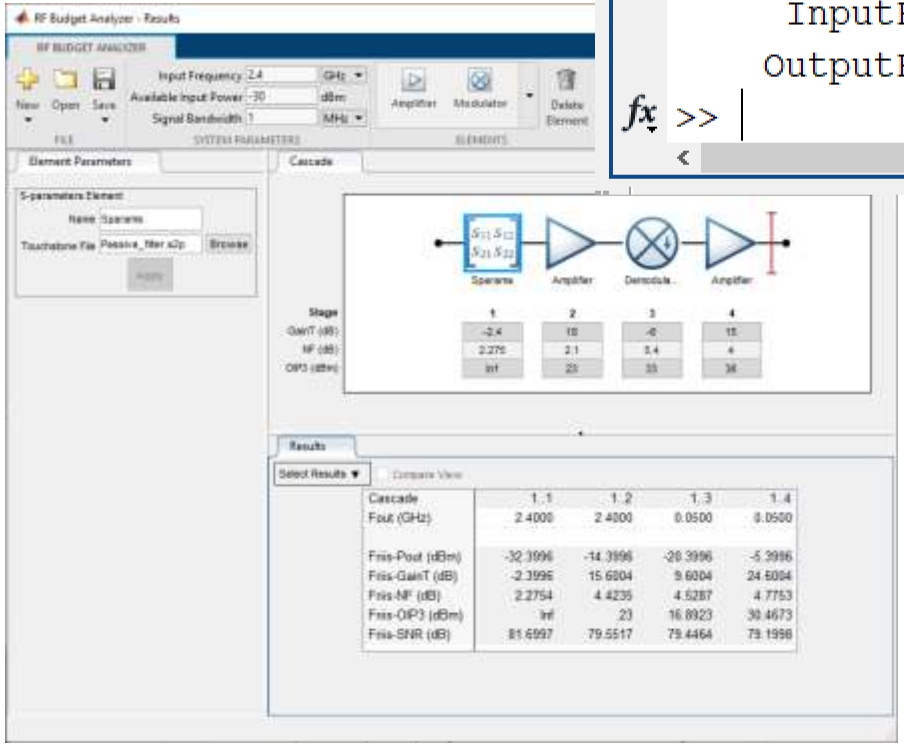



```
>> out = rfDPD(I, Q)
```

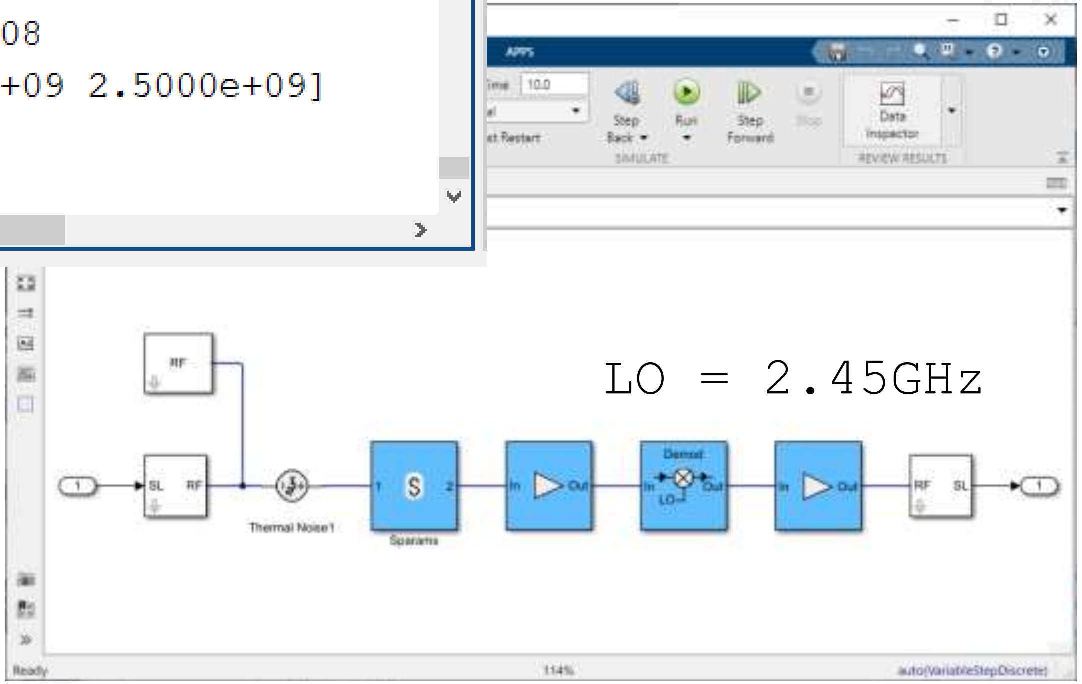
```
Low edge RMS EVM, Peak EVM, slot 17: 0.789 3.209%
High edge RMS EVM, Peak EVM, slot 17: 0.789 3.146%
Low edge RMS EVM, Peak EVM, slot 18: 0.794 3.210%
High edge RMS EVM, Peak EVM, slot 18: 0.794 3.151%
Low edge RMS EVM, Peak EVM, slot 19: 0.770 3.217%
High edge RMS EVM, Peak EVM, slot 19: 0.770 3.152%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

https://www.mathworks.com/help/simrf/ug/power-amplifier-characterization-with-dpd-for-reduced-signal-distortion.html

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 >>
```



LO = 2.45GHz

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