Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Federal Department of Justice and Police FDJP
Federal Office of Metrology METAS

Metas.UncLib and VNA Tools II

Agilent European Metrology Workshop Paris 22. September 2010

Michael Wollensack and Marko Zeier

Programming today is a race between software engineers striving to build bigger and better idiot-proof programs, and the Universe trying to produce bigger and better idiots.

So far, the Universe is winning.

Rich Cook

Outline

- 1. Introduction
    - 1.1 VNA Tools II Project
    - 1.2 MU according to ISO GUM
    - 1.3 Metas.UncLib
    - 1.4 Linear Uncertainty Propagation
    - 1.5 Large System Modelling
- 2. Features
- 3. MATLAB Demo
- 4. VNA Tools II

1.1 VNA Tools II Project (217.08.FP.060)

- Goal: Development of a metrology software for Vector Network Analysis (VNA), which supports uncertainty calculation compliant with the ISO-GUM.

- Duration: Feb. 2008 – Jan 2011

- METAS RF & MW laboratory:
  M. Wollensack J. Rüfenacht M. Zeier

- Partner:
  - Agilent Technologies, Santa Rosa, USA
    (Largest manufacturer of VNAs and calibration standards)

- Status: Currently work on VNA related parts and GUI

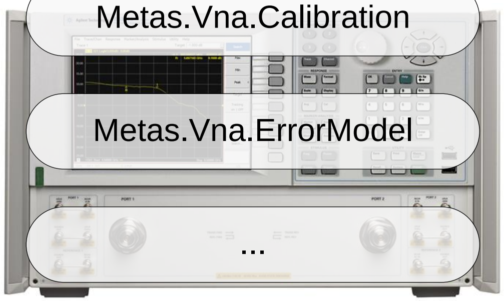
METAS

1.1 VNA Tools II Overview

Metas.UncLib

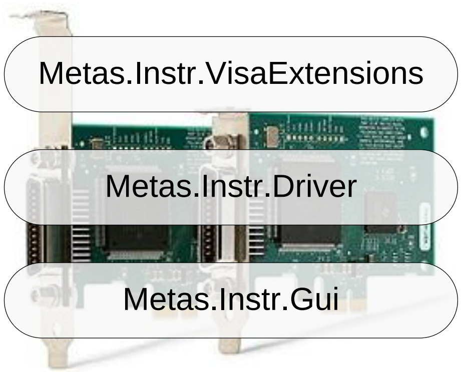
- Metas.UncLib.Core
- Metas.UncLib.LinProp
- Metas.UncLib.DistProp
- Metas.UncLib.MCProp
- Metas.UncLib.DB

Metas.Vna



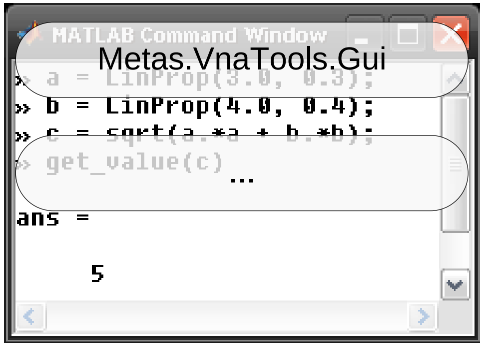
- Metas.Vna.Data
- Metas.Vna.Calibration
- Metas.Vna.ErrorModel
- ...
- ...

Metas.Instr



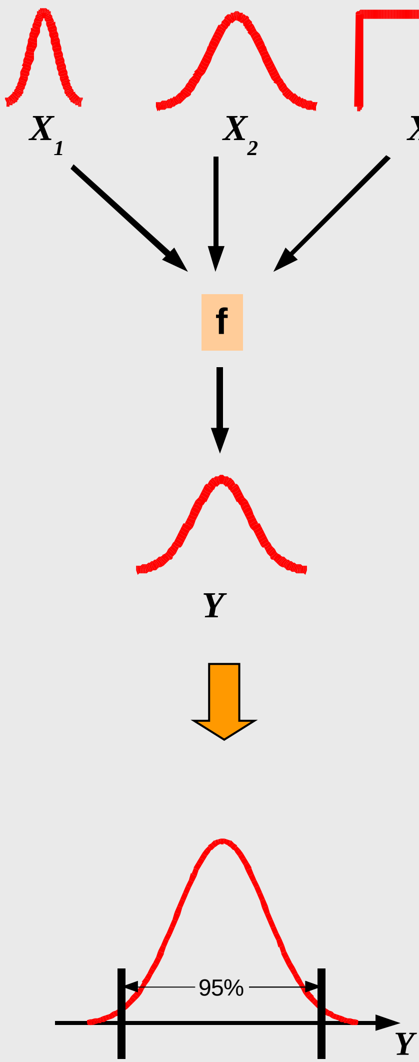
- Metas.Instr.VisaExtensions
- Metas.Instr.Driver
- Metas.Instr.Gui

Metas.VnaTools



- Metas.VnaTools.Gui
- ...

1.2 MU according to ISO-GUM (scalar)



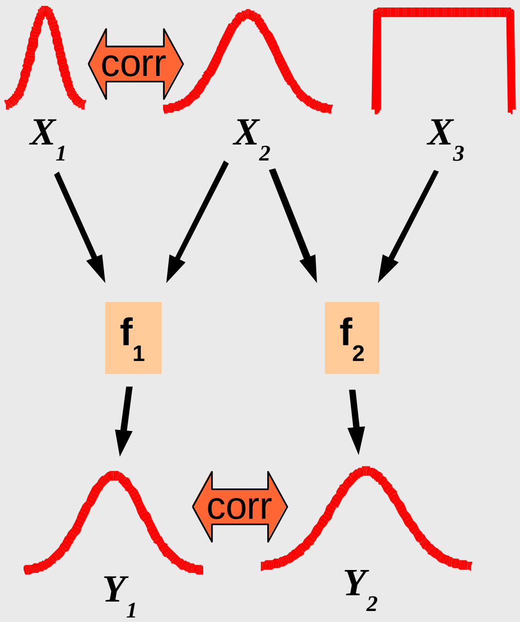
Input quantities $u(X_1), u(X_2), \ldots$

Measurement model

Output quantity $u^2(Y) = \left(\frac{\partial f}{\partial X_1}\right)^2 u^2(X_1) + \left(\frac{\partial f}{\partial X_2}\right)^2 u^2(X_2) + \ldots$

Expanded measurement uncertainty $U^2(Y) = (1.96)^2 \cdot u^2(Y)$

1.2 MU according to ISO-GUM (multivariate)



Correlation between input quantities

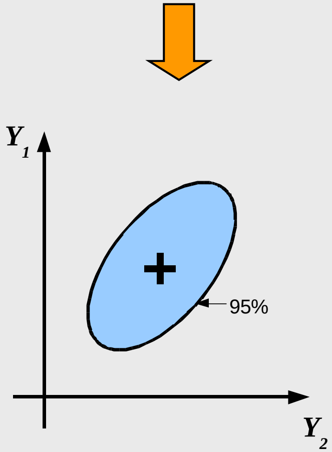
$$\mathbf{V_X}$$

Uncertainty Matrix

Several output quantities correlated!

$$\mathbf{V_Y} = \mathbf{J_{f,X}} \mathbf{V_X} \mathbf{J_{f,X}}^{\mathbf{T}}$$

Lin. Uncertainty Propagation



Expanded measurement uncertainty

$$\mathbf{U_Y} = \chi^2_{p,0.95} \cdot \mathbf{V_Y}$$

Calculations are tedious!

1.3 Metas.UncLib

General Purpose Uncertainty Library

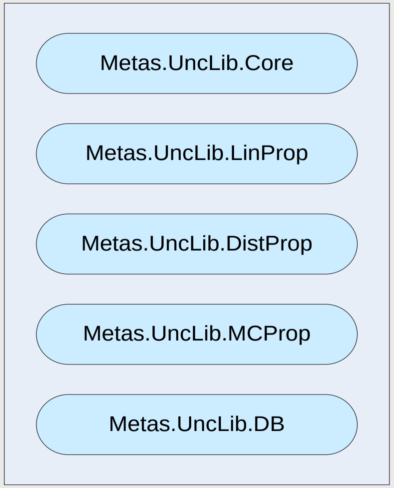
It does

- support multidimensional uncertainty calculation
- advanced math (Complex, Vector, Matrix)
- linear uncertainty propagation
- higher order taylor terms, higher moments (partially)
- Monte Carlo uncertainty propagation (preliminary)
- take care of correlations
- advanced storage / archiving
- interfacing with other applications

It does NOT

- help to build a measurement model
- have a nice graphical interface
- produce „fancy“ output

Metas.UncLib



1.4 Linear Uncertainty Propagation (LUP)

LinProp Module

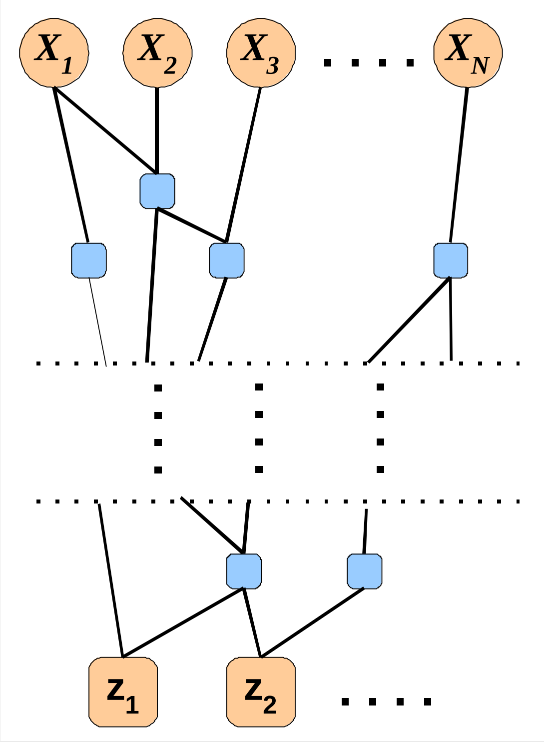
Based on *GUM Tree* concept (Blair Hall, IRL/MSL, NZ)

- Automatic Differentiation
  - Calculate Derivatives (sensitivity coefficients) with machine precision

- Object oriented implementation
  - Hide complexity from user

1.4 LUP: Automatic Differentiation

To calculate sensitivity coefficients



Measurement model $\mathbf{z} = \mathbf{f}(\mathbf{x})$

Decomposition in elementary functions

$$\mathbf{z} = \mathbf{f}_1(\mathbf{f}_2 \ldots \mathbf{f}_M(\mathbf{x}))$$

Derivatives of elementary functions
are explicitly programmed

Chain rule $\mathbf{J}_{\mathbf{f}_i,\mathbf{x}} = \mathbf{J}_{\mathbf{f}_i,\mathbf{f}_j} \cdot \mathbf{J}_{\mathbf{f}_j,\mathbf{x}}$

Naturally supported by computers because compiler
decomposes equations into elementary functions

1.4 LUP: Metas.UncLib Object

New Data Type (contains value, dependencies and sensitivities)

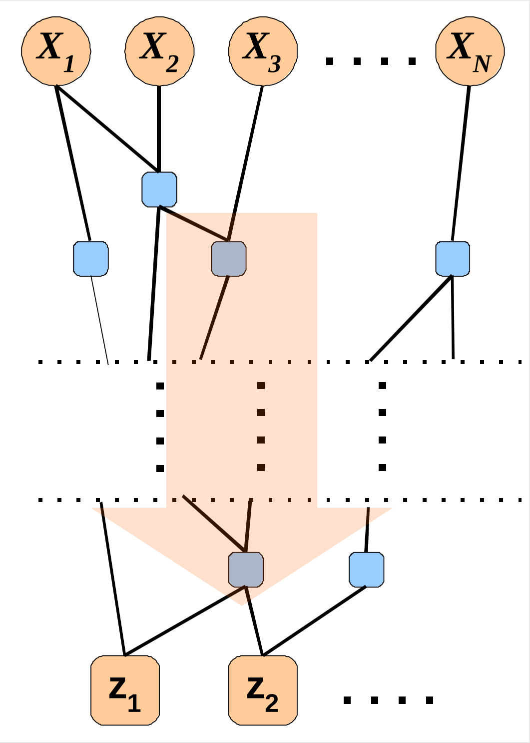
Metas.UncLib Object

value z_i

depends on $(x_1, x_3, x_4, \ldots)$

Jacobian (sensitivities) $\left(\frac{\partial z_i}{\partial x_1}, \frac{\partial z_i}{\partial x_3}, \frac{\partial z_i}{\partial x_4}, \ldots\right)$

updated at each computational step



Overloaded Operators hide complexity
→Objects can be used like ordinary numbers

Uncertainty information on demand

1.4 LUP: Correlations

Caused by common influences

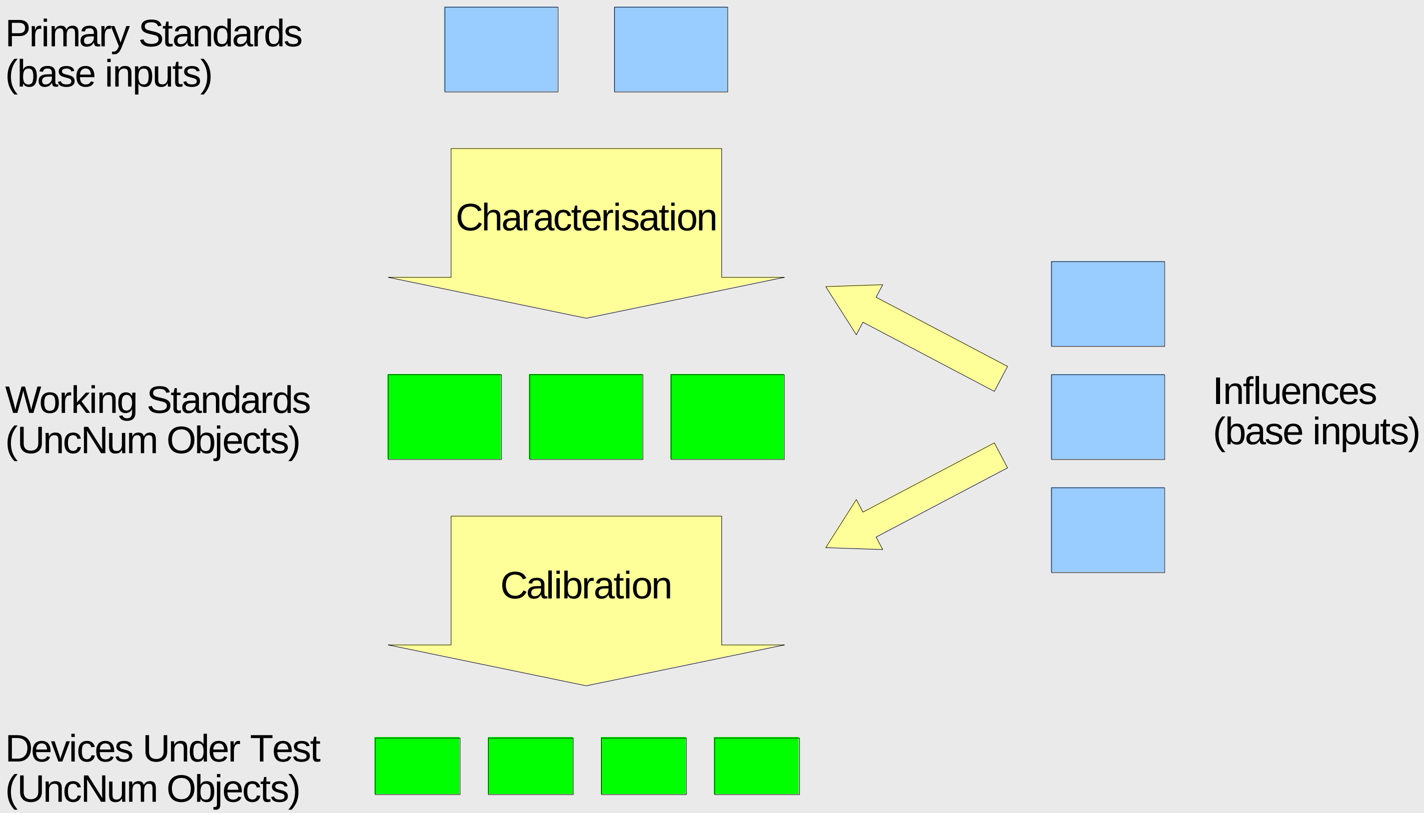
The system takes automatically care of correlations by recognizing common influences

value	z_1	Metas.UncLib Object
depends on	(x_2, x_3)	
Jacobian (sensitivities)	$\left(\frac{\partial z_1}{\partial x_2}, \frac{\partial z_1}{\partial x_3}\right)$	

value	z_2	Metas.UncLib Object
depends on	(x_1, x_2)	
Jacobian (sensitivities)	$\left(\frac{\partial z_2}{\partial x_1}, \frac{\partial z_2}{\partial x_2}\right)$	

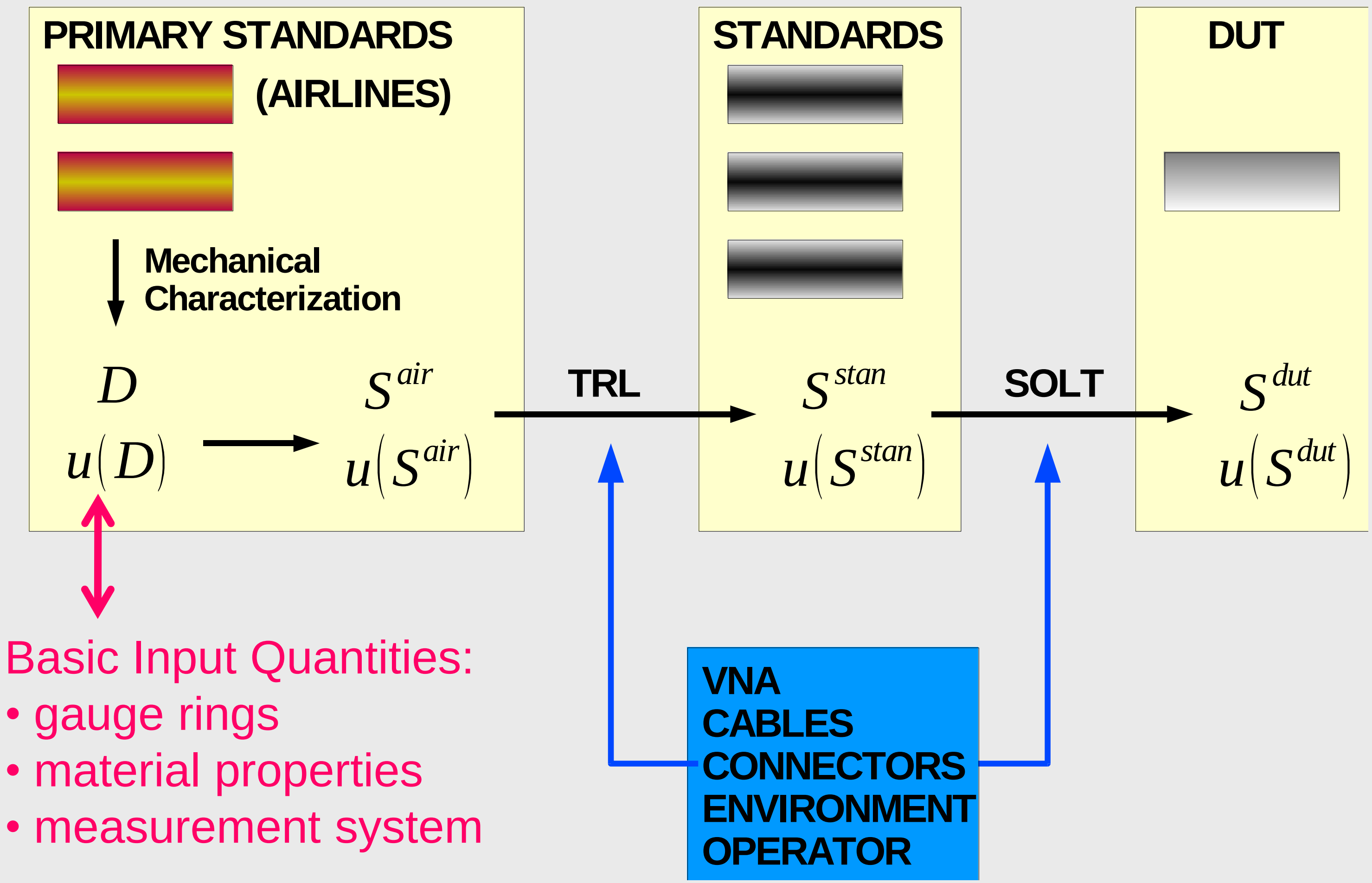
1.5 Large System Modelling

The system allows complete modelling of a multi-step process with stored intermediate results (e.g. traceability chain)



1.5 Traceability Chain in VNA

Multistep process with intermediate storage



Outline

- 1. Introduction
- 2. Features
  - 2.1 Math functions
  - 2.2 Linear Algebra / Numerical Routines
  - 2.3 Uncertainty functions
  - 2.4 Storage / Archiving
  - 2.5 Interfacing .NET / COM
- 3. MATLAB Demo
- 4. VNA Tools

METAS

2.1 Math functions

A list with the available functions and constants:

- Sqrt(x)
- Exp(x)
- Log(x)
- Log10(x)
- Log(x, y)
- Pow(x, y)
- Sign(x)
- Sin(x)
- Cos(x)
- Tan(x)
- Asin(x)
- Acos(x)
- Atan(x)
- Atan2(x, y)
- Sinh(x)
- Cosh(x)
- Tanh(x)
- Asinh(x)
- Acosh(x)
- Atanh(x)
- Real(x)
- Imag(x)
- Abs(x)
- Angle(x)
- Conj(x)
- PI
- E
- J
- One
- Zero

2.2 Linear Algebra / Numerical Routines

A list with the available functions:

▪ Dot(M1, M2)	Matrix multiplication of matrix M1 and M2
▪ Lu(M)	LU decomposition of matrix M
▪ Det(M)	Determinate of matrix M
▪ Inv(M)	Matrix inverse of M
▪ Solve(A, Y)	Solve linear equation system: A * X = Y
▪ LstSqrSolve(A, Y)	Least square solve of overdetermined linear equation system.
▪ WeightedLstSqrSolve(A, Y, W)	Weighted least square solve of over-determined linear equation system.
▪ Fft(M)	Fast Fourier transformation
▪ Ifft(M)	Inverse Fast Fourier transformation
▪ Interpolation(x, y, n, xx)	

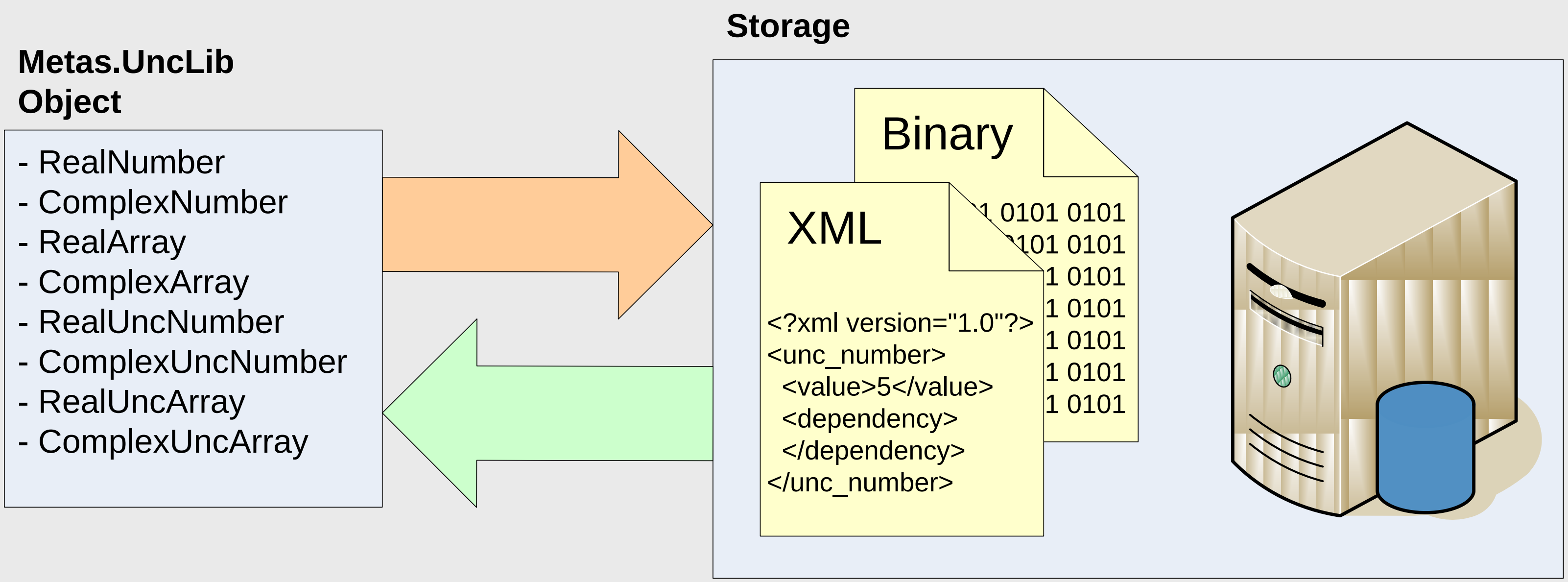
2.3 Uncertainty functions?

The available functions to obtain information from a ‘Metas.UncLib’ object:

	C# ‘Metas.UncLib’	MATLAB ‘LinProp’ or ‘MCProp’
Returns the **value**.	y.GetValue()	get_value(y)
Returns the **standard uncertainty**.	y.GetStdUnc()	get_stdunc(y)
Returns the inverse **degrees of freedom**.	y.GetIDof()	get_idof(y)
Returns the sensitivities to the virtual base inputs (with value 0 and uncertainty 1). This is equal to the uncertainties components	y.GetJacobi()	get_jacobi(y)
Returns the sensitivities to the intermediate results.	y.GetJacobi2(x)	get_jacobi2(y,x)
Returns the **uncertainty component**.	y.GetUncComponent(x)	get_unc_component(y, x)
Returns the **correlation matrix**.	[y1 y2 ...]. GetCorrelation()	get_correlation([y1 y2 ...])
Returns the **covariance matrix**.	[y1 y2 ...]. GetCovariance()	get_covariance([y1 y2 ...])

2.4 Storage / Archiving

- Save computed ‘Metas.UncLib’ objects.
- Reload stored ‘Metas.UncLib’ objects.
  After loading all information is restored.



2.5 Interfacing .NET / COM

Microsoft® .NET

Metas.UncLibCE

- Metas.UncLibCE.Core
- Metas.UncLibCE.LinProp
- Metas.UncLib.DistProp
- Metas.UncLibCE.MCProp

Platforms:

- Pocket PC
- Smart Phone
- Windows CE Device

Metas.UncLib

- Metas.UncLib.Core
- Metas.UncLib.LinProp
- Metas.UncLib.DistProp
- Metas.UncLib.MCProp
- Metas.UncLib.DB

Platforms:

- Visual Basic
- Visual C#
- Visual C++
- LabVIEW
- Mathematica
- Office 2007
- …

activex

Metas.UncLibCOM

- Metas.UncLibCOM.Core
- Metas.UncLibCOM.LinProp
- Metas.UncLibCOM.DistProp
- Metas.UncLibCOM.MCProp
- Metas.UncLibCOM.DB

Platforms:

- C++
- Matlab
- Excel 97 - 2003
- ...

Metas.UncLib.Matlab

- LinProp
- DistProp
- MCProp
- LinPropDB

Platforms:

- Use Metas.UncLib in Matlab scripts and functions.

Outline

- 1. Introduction
- 2. Features
- 3. MATLAB Demo
  - 3.1 Basic Features
  - 3.2 Right triangle example
  - 3.3 Resistor cube example
  - 3.4 Circle fit example
- 4. VNA Tools

METAS

3.1 Basic Features

MATLAB Command Window

```
  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

>> a = LinProp(3.0, 0.3);
>> b = LinProp(4.0, 0.4);
>> c = sqrt(a.*a + b.*b);
>> get_value(c)

ans =

     5

>> get_stdunc(c)

ans =

      0.36715

>>
```

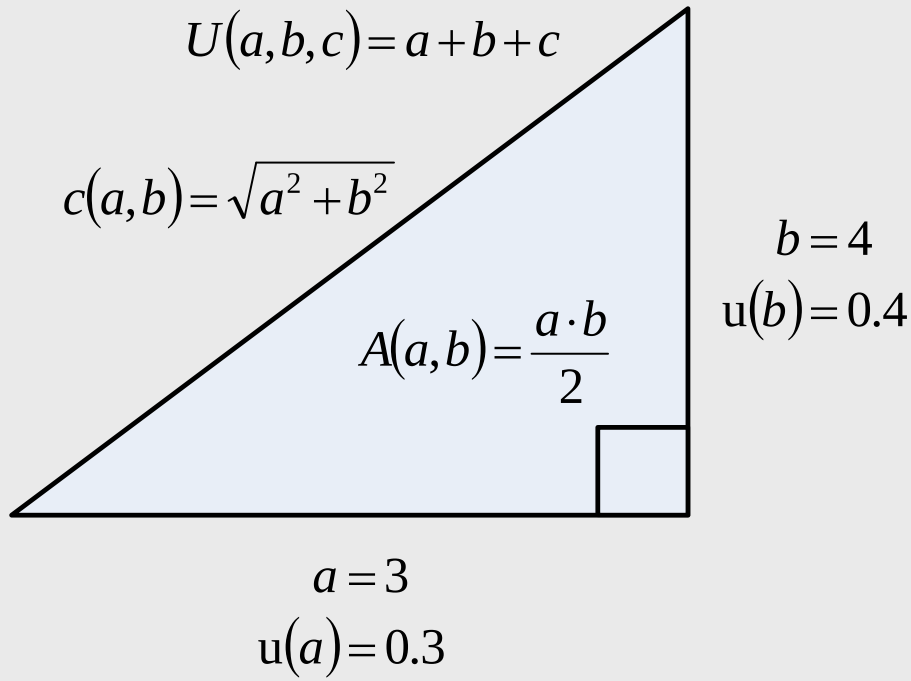

METAS

3.2 Right triangle example

- Cathetus a = 3, u(a) = 0.3
- Cathetus b = 4, u(b) = 0.4

- What‘s the value and uncertainty of the hypotenuse c?

- What‘s the value and uncertainty of the perimeter U?



- What‘s the value and uncertainty of the area A?

- What‘s the correlation between A and U?

3.2.1 Right triangle example – Matlab

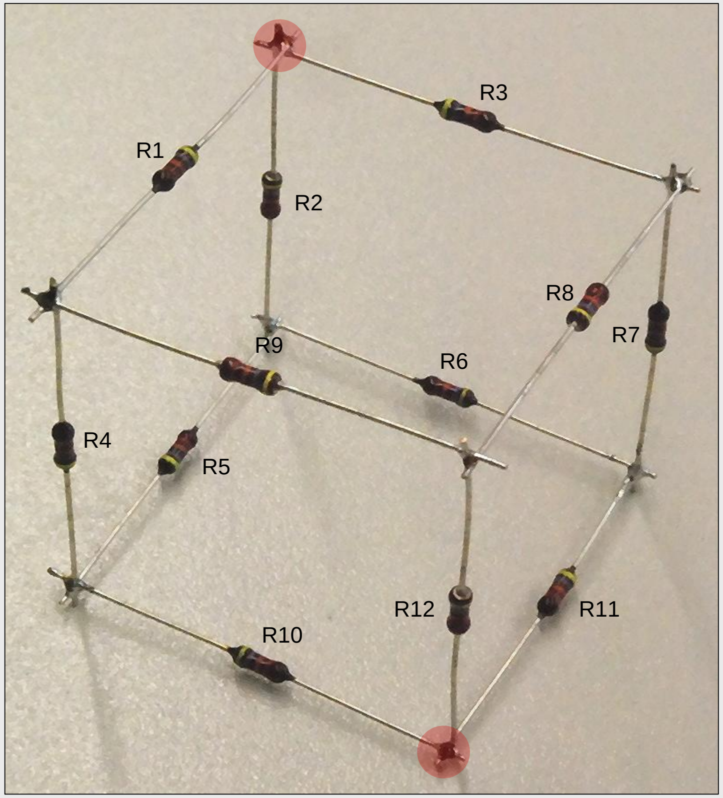
```
unc = @LinProp;

%% Definition of the inputs
a = unc(3.0, 0.3);
b = unc(4.0, 0.4);

%% Compute the outputs
c = sqrt(a.*a + b.*b);
U = a + b + c;
A = a.*b./2;

%% Display some results
disp(sprintf('c      = %7.3f', get_value(c)))
disp(sprintf('u(c)   = %7.3f', get_stdunc(c)))

disp(sprintf('U      = %7.3f', get_value(U)))
disp(sprintf('u(U)   = %7.3f', get_stdunc(U)))

disp(sprintf('A      = %7.3f', get_value(A)))
disp(sprintf('u(A)   = %7.3f', get_stdunc(A)))

temp = get_correlation([A U]);
disp(sprintf('r(A,U) = %7.3f', temp(1,2)))
```

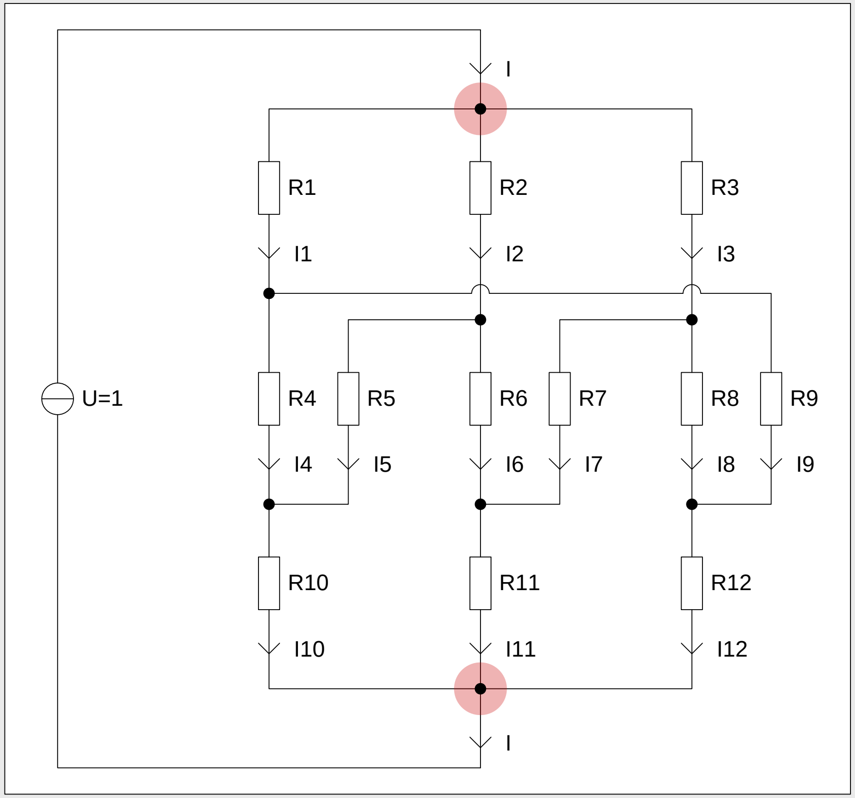
→

```
c      =   5.000
u(c)   =   0.367

U      =  12.000
u(U)   =   0.865

A      =   6.000
u(A)   =   0.849


r(A,U) =   0.981
```

3.3 Resistor cube example

- What's the equivalent resistor of the cube between the two red junctions?
- And the uncertainty of it?





$$\underset{(12\times1)}{U} = \underset{(12\times12)}{R} \times \underset{(12\times1)}{I} \rightarrow \underset{(12\times1)}{I} = \underset{(12\times12)}{R}^{-1} \times \underset{(12\times1)}{U}$$

3.3.1 Resistor cube example – Matlab

```
unc = @LinProp;
%% Definition of input uncertainty objects R01 ... R12 and U
U   = 1;
R01 = unc(50,0.1);   R02 = unc(50,0.1);   R03 = unc(50,0.1);
R04 = unc(50,0.1);   R05 = unc(50,0.1);   R06 = unc(50,0.1);
R07 = unc(50,0.1);   R08 = unc(50,0.1);   R09 = unc(50,0.1);
R10 = unc(50,0.1);   R11 = unc(50,0.1);   R12 = unc(50,0.1);
%% Kirchhoff's circuit laws --> linear equation system
Ux = [0 0 0 0 0 0 U U U U U U]';
Rx = [-1   0   0   1   0   0   0   0   1   0   0   0 ;...
       0  -1   0   0   1   1   0   0   0   0   0   0 ;...
       0   0  -1   0   0   0   1   1   0   0   0   0 ;...
       0   0   0   1   1   0   0   0   0  -1   0   0 ;...
       0   0   0   0   0   1   1   0   0   0  -1   0 ;...
       0   0   0   0   0   0   0   1   1   0   0  -1 ;...
     R01   0   0 R04   0   0   0   0   0 R10   0   0 ;...
     R01   0   0   0   0   0   0   0 R09   0   0 R12;...
       0 R02   0   0   0 R06   0   0   0   0 R11   0 ;...
       0 R02   0   0 R05   0   0   0   0 R10   0   0 ;...
       0   0 R03   0   0   0   0 R08   0   0   0 R12;...
       0   0 R03   0   0   0 R07   0   0   0 R11   0 ];
%% Solve linear equation system
Ix = Rx\Ux;
%% Compute equivalent resistor of the cube
I = Ix(1) + Ix(2) + Ix(3);
R = U/I
```

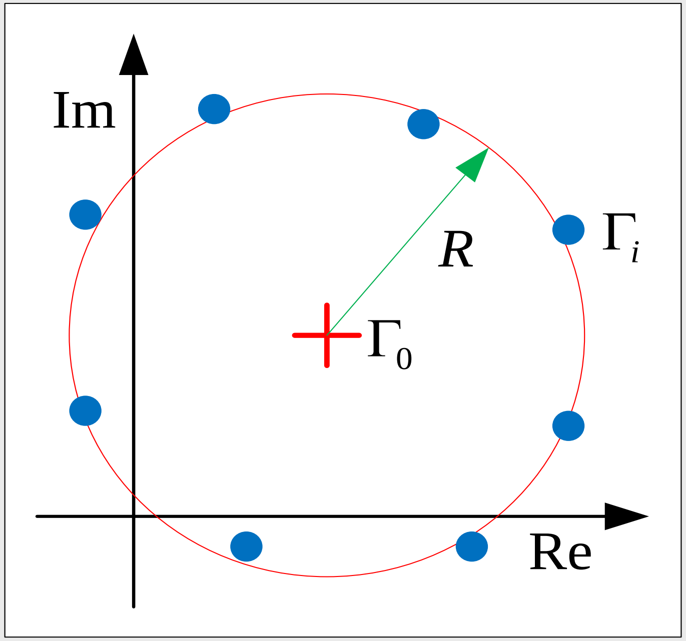
```
R =
   (41.6667 ± 0.0280542)
```

3.4 Circle fit example

- Where is the center of the circle? What is the uncertainty?

Equation for a circle:

$$\left|\Gamma_i - \Gamma_0\right|^2 = R^2 \quad i = 1, \ldots, n$$



Linearization:

$$2 \cdot \mathrm{Re}(\Gamma_i) \cdot \underbrace{\mathrm{Re}(\Gamma_0)}_{a} + 2 \cdot \mathrm{Im}(\Gamma_i) \cdot \underbrace{\mathrm{Im}(\Gamma_0)}_{b} + \underbrace{R^2 - \left|\Gamma_0\right|^2}_{c} = \left|\Gamma_i\right|^2$$

Circle fit equation system:

$$\mathrm{A} \times \mathrm{p} = \mathrm{d}$$

over determined

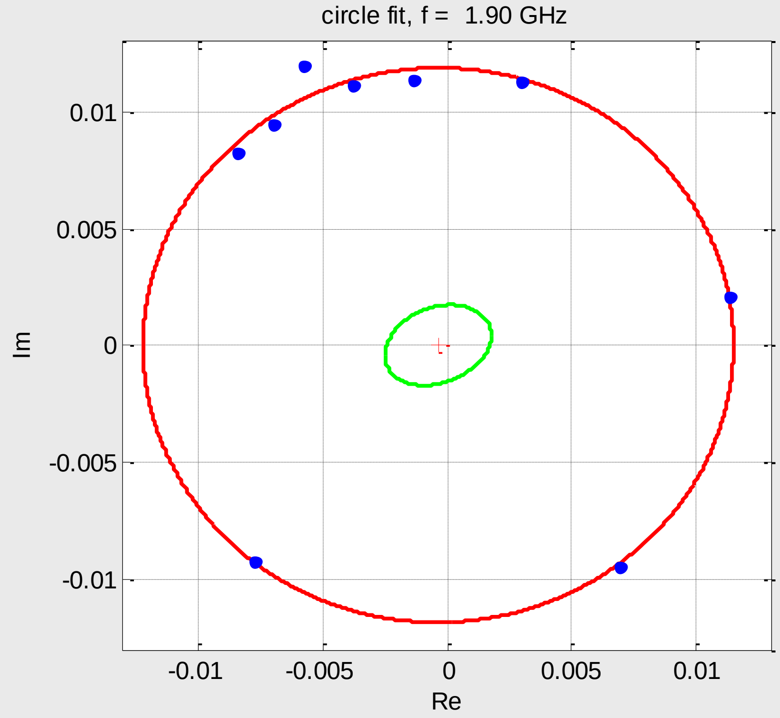
$$\mathrm{A} = \begin{bmatrix} 2 \cdot \mathrm{Re}(\Gamma_1) & 2 \cdot \mathrm{Im}(\Gamma_1) & 1 \\ 2 \cdot \mathrm{Re}(\Gamma_2) & 2 \cdot \mathrm{Im}(\Gamma_2) & 1 \\ \vdots & \vdots & \vdots \\ 2 \cdot \mathrm{Re}(\Gamma_n) & 2 \cdot \mathrm{Im}(\Gamma_n) & 1 \end{bmatrix} \quad \mathrm{p} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad \mathrm{d} = \begin{bmatrix} \left|\Gamma_1\right|^2 \\ \left|\Gamma_2\right|^2 \\ \\ \left|\Gamma_n\right|^2 \end{bmatrix}$$

$$\Gamma_0 = a + b \cdot \mathrm{i} \qquad R = \sqrt{c + a^2 + b^2}$$

- And how to compute the uncertainty? The analytic solution is too complicated. Too much data for Monte Carlo.
→ Metas.UncLib is the ideal solution!

3.4.1 Circle fit example – Matlab

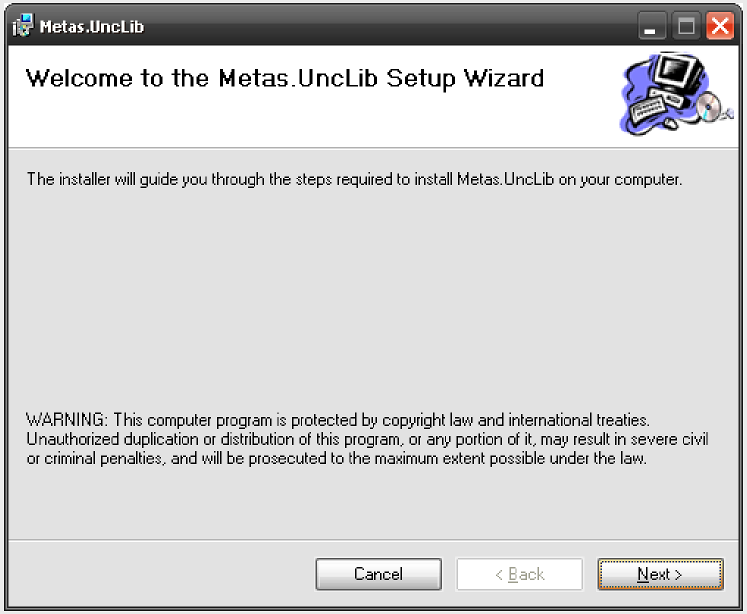
```
unc = @LinProp;
%% Circle points
f = 1.9; % (GHz)

V = diag([6.9e-4^2, 6.9e-4^2]);

s1 = unc(-8.43E-3 +8.28E-3*i, V);
s2 = unc(-6.99E-3 +9.43E-3*i, V);
s3 = unc(-3.73E-3 +1.11E-2*i, V);
s4 = unc( 3.00E-3 +1.13E-2*i, V);
s5 = unc( 1.14E-2 +2.11E-3*i, V);
s6 = unc( 6.92E-3 -9.43E-3*i, V);
s7 = unc(-7.75E-3 -9.27E-3*i, V);
s8 = unc(-5.76E-3 +1.20E-2*i, V);
s9 = unc(-1.31E-3 +1.14E-2*i, V);

s = [s1 s2 s3 s4 s5 s6 s7 s8 s9].';

%% Circle fit
A = [2.*real(s) 2.*imag(s) ones(size(s))];
d = abs(s).^2;
p = A\d;
S0 = p(1) + i*p(2);
R = sqrt(p(3) + abs(S0).^2);
```




```
S0.value =
  -0.00035669 +3.0153e-005i


S0.covariance =
  1.8235e-007   3.6297e-008
  3.6297e-008   1.1865e-007
```

Get Metas.UncLib now!

- Download for free at www.metas.ch/unclib
- License prohibits redistribution
- 2 setup files (*.msi): Metas.UncLib and Metas.UncLib.Matlab



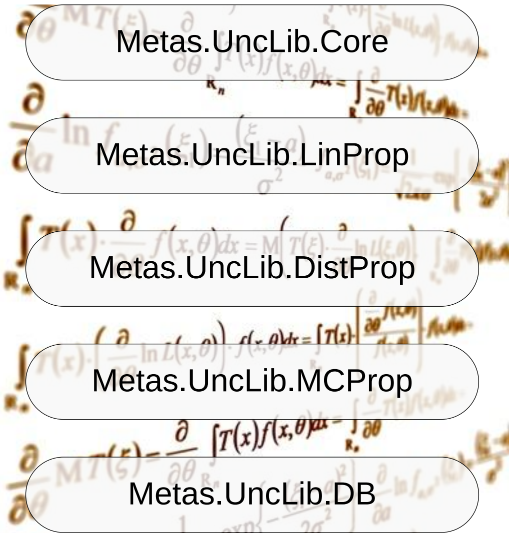
- MS Windows XP or later
- Matlab Version 2006b or later

Outline

- 1. Introduction
- 2. Features
- 3. MATLAB Demo
- 4. VNA Tools
  - 4.1 VNA Tools II Overview and Status
  - 4.2 VNA Data
  - 4.3 VNA Calibration
  - 4.4 VNA Error Model
  - 4.5 VNA Tools GUI
  - 4.6 VNA Tools Script

4.1 VNA Tools II Overview and Status

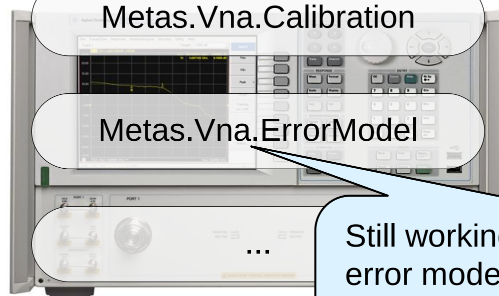
Metas.UncLib



- Metas.UncLib.Core
- Metas.UncLib.LinProp
- Metas.UncLib.DistProp
- Metas.UncLib.MCProp
- Metas.UncLib.DB

done

Metas.Vna

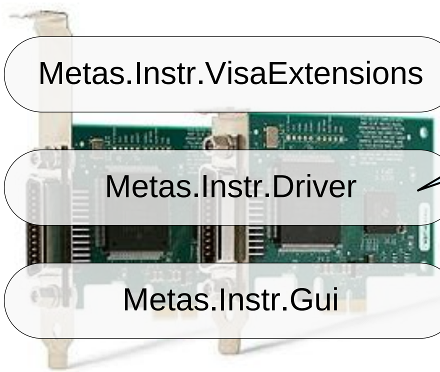


- Metas.Vna.Data
- Metas.Vna.Calibration
- Metas.Vna.ErrorModel
- ...
- ...

done

Still working on the VNA error model. Characterize the influences and bring them into the calculation.

Metas.Instr

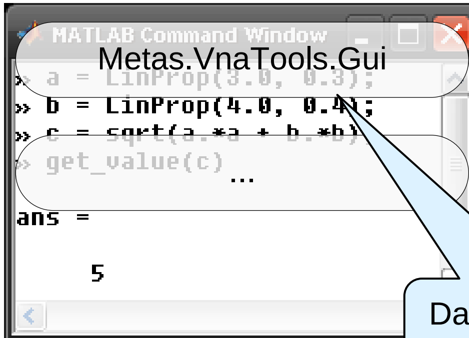


- Metas.Instr.VisaExtensions
- Metas.Instr.Driver
- Metas.Instr.Gui

done

Agilent PNA
HP 8510C
HP 8753D
HP 8751A
Agilent ECal
Attn. Switch Drv.

Metas.VnaTools



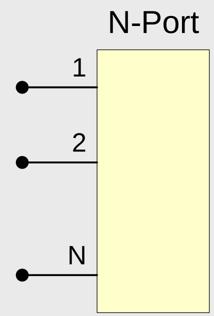
- Metas.VnaTools.Gui
- ...

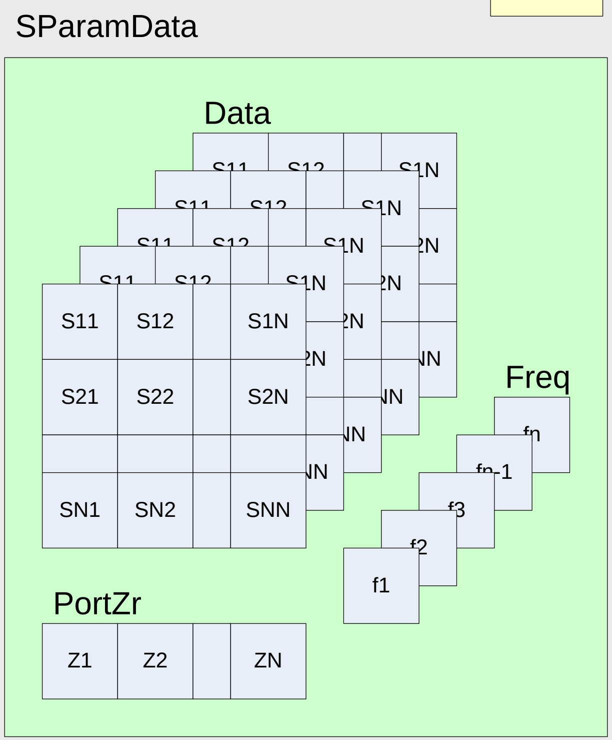
done

Data visualization done.

4.2 VNA Data

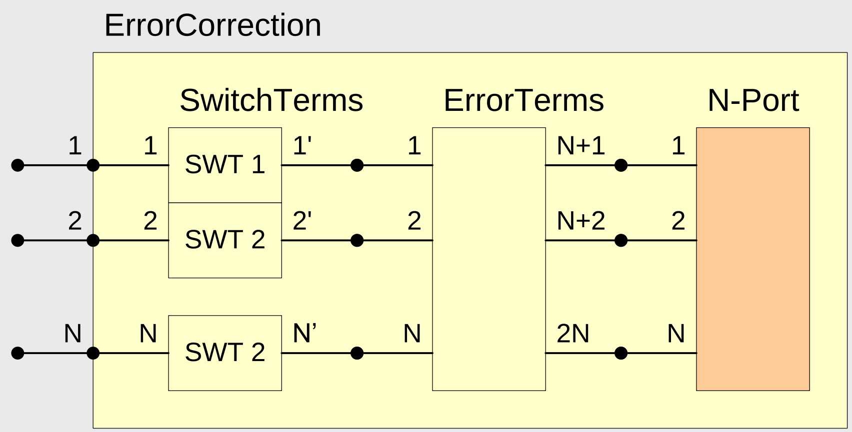
- SParamData is class for N-Port S-Parameter Data.
- Which properties need to be stored?
  - Frequency List (1d Array)
  - Data (3d Array)
  - Port Zr (1d Array)
- Supported file formats:
  - CITI:
    no uncertainties, no Port Zr
  - Touchstone (*.s*p):
    no uncertainties
  - SParamData XML (*.sdata)
    new, supports uncertainties
- SParamTools is a tool class for computing with SParamData objects. For example: Cascading two SParamData objects





4.3 VNA Calibration

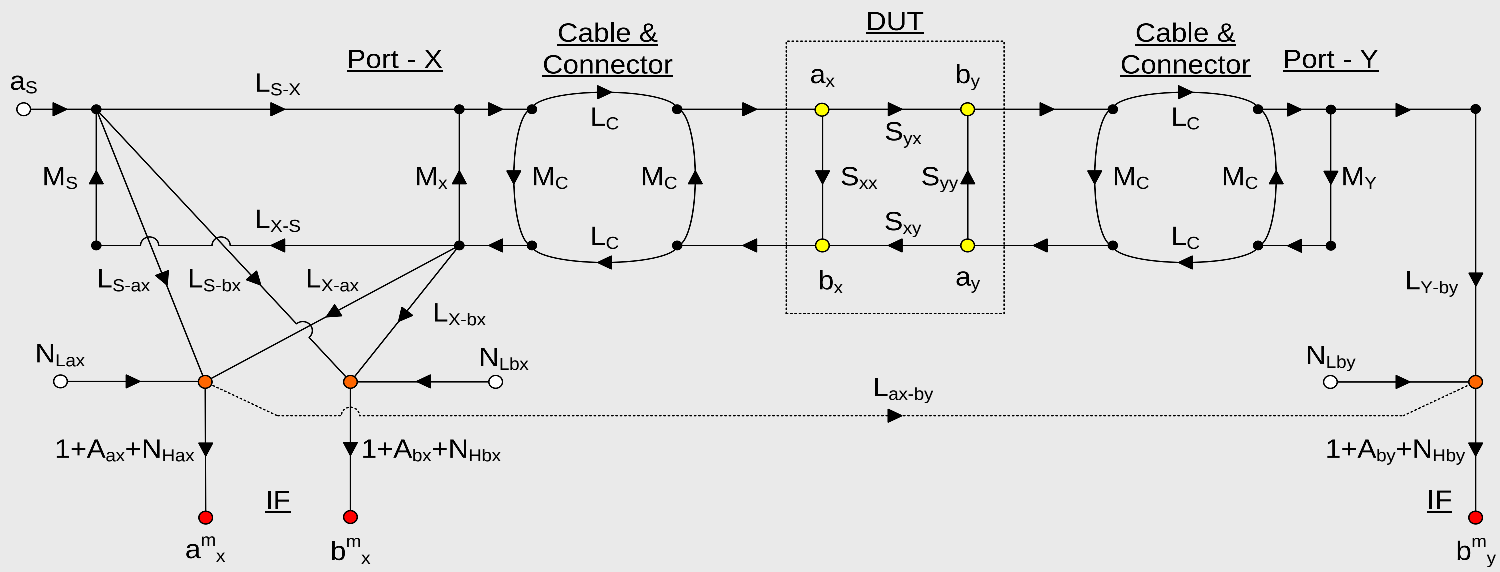
- The error terms of a N-Port VNA Calibration can be described in 2N-Port SParamData object.
- The error terms can describe a full leaky VNA model.
- SParamTools can be used for switch terms and error terms correction.



- Metas.Vna.Calibration includes various algorithms for calibration (SOLT, offset shorts, unknown thru, …).
- The Error Correction is independent of the used calibration method.

4.4 VNA Error Model

- Characterize the Influences:
  - Linearity, Drift, Noise Floor, Trace Noise
  - Cable stability, Connector repeatability
- Combine with uncertainty calculation.

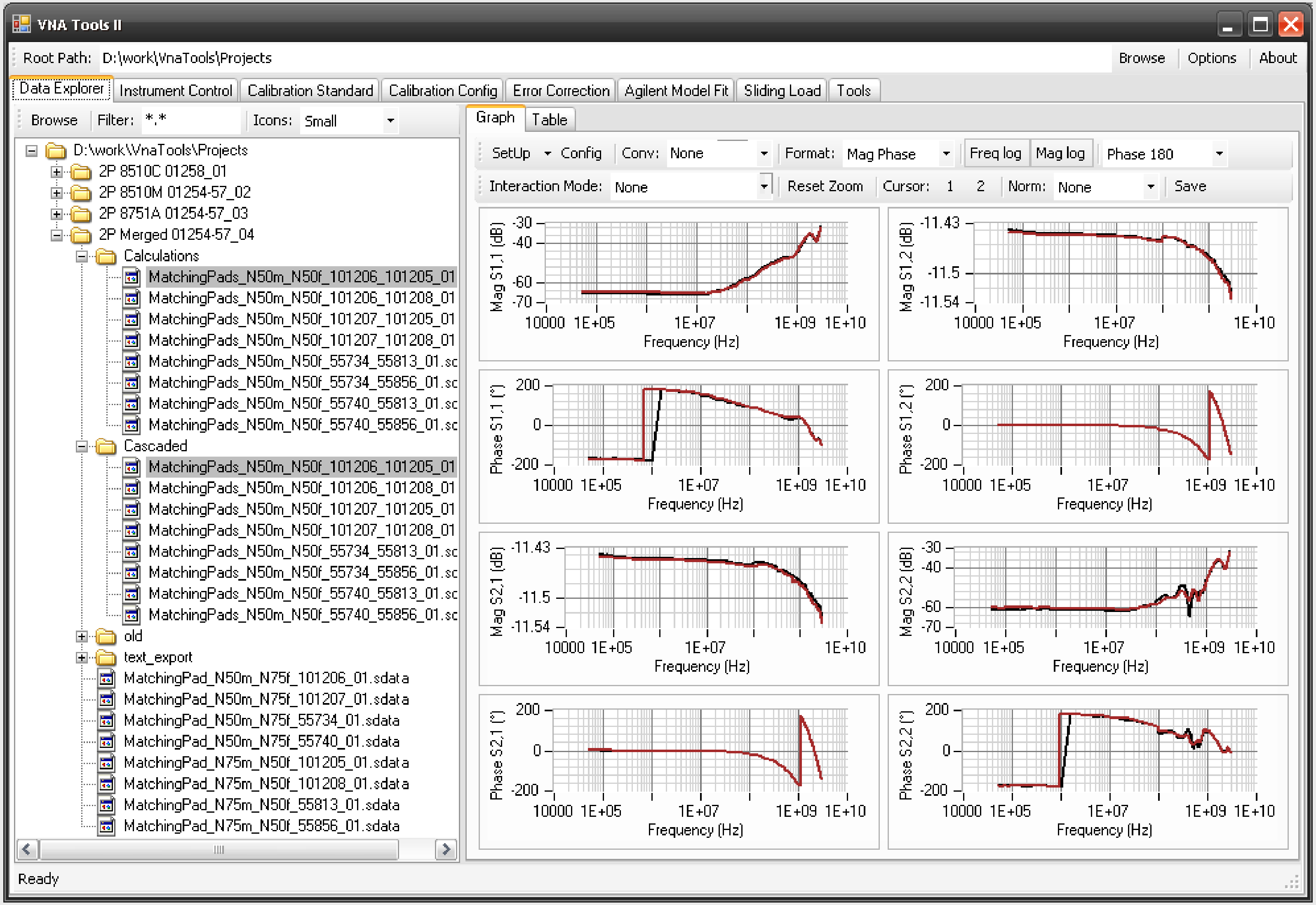


4.5. VNA Tools GUI

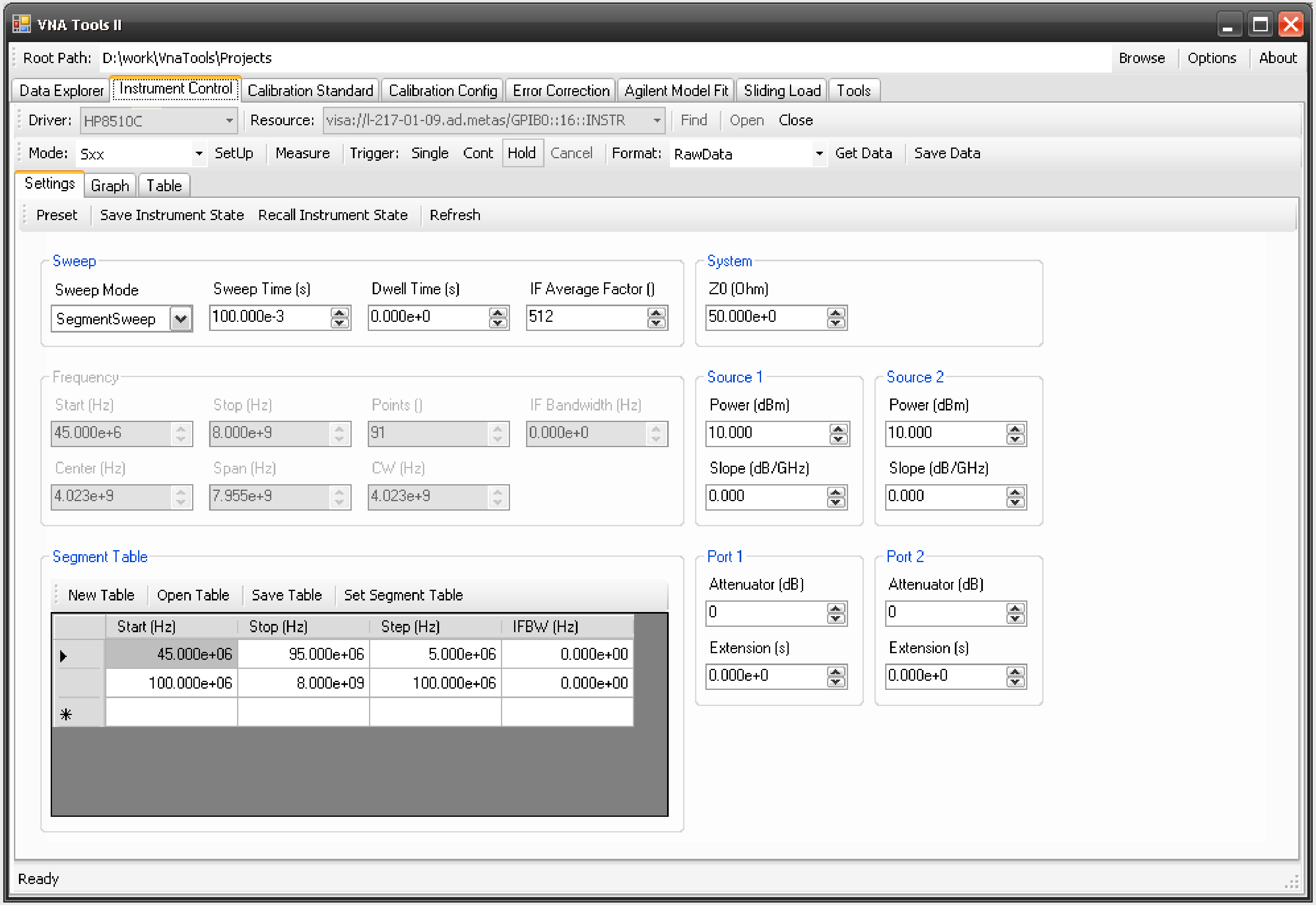
- VNA Tools II has a graphical user interface ☺
- There a several modules:

• Data Explorer	Data visualization
• Instrument Control	VNA control & measurements
• Calibration Standard	Cal Std Definitions Database
• Calibration Config	Assign measurements & definitions
• Error Correction	Error correction of raw measurements
• Agilent Model Fit	Fit calibration standard parameters
• Sliding Load	Circle fit and merge measurements
• Tools	Cascade, Decascade, Reverse, Subset, Mean, Merge, Change Zr, …

4.5.1 VNA Tools GUI – Data Explorer

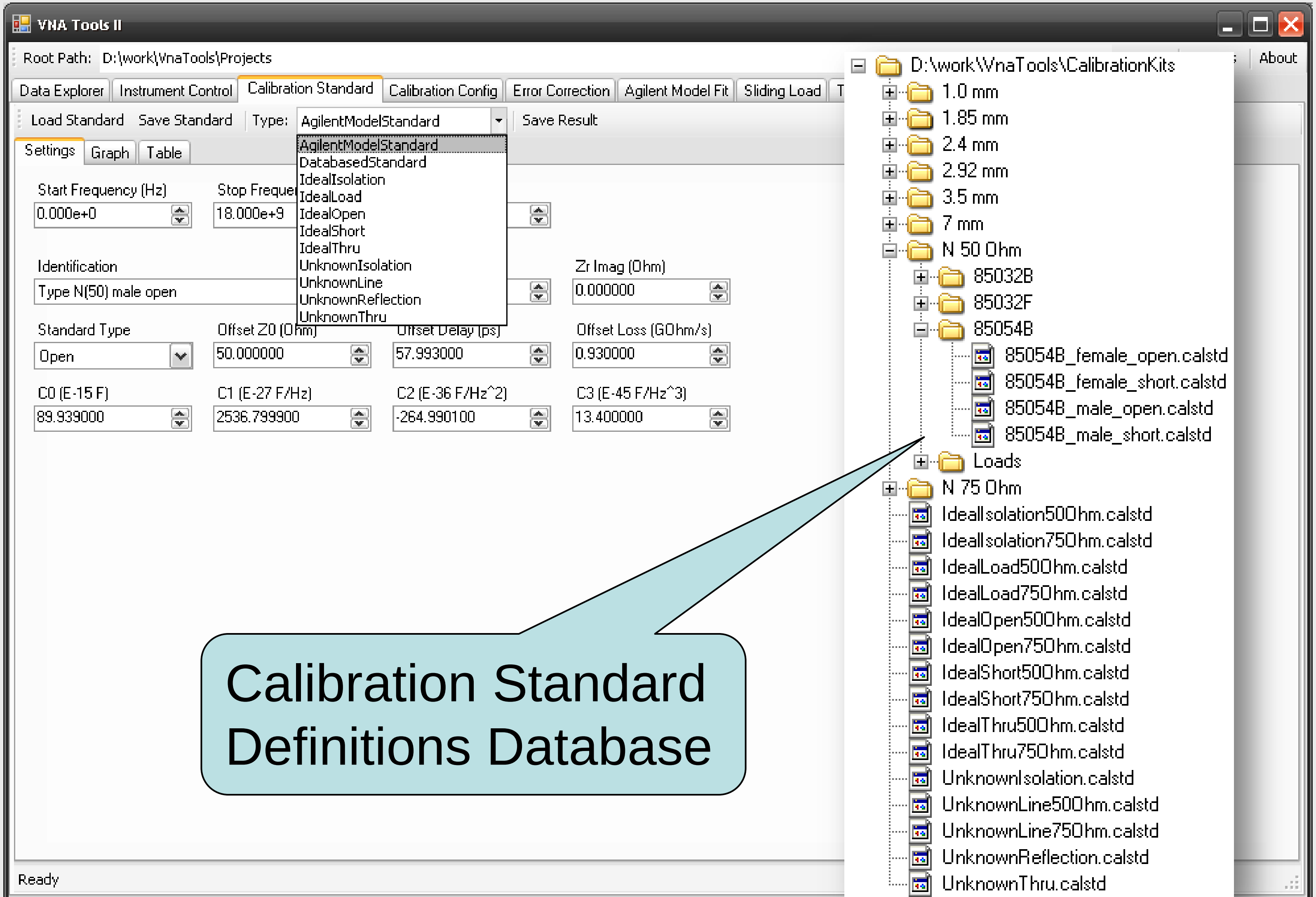


4.5.2 VNA Tools II – Instrument Control

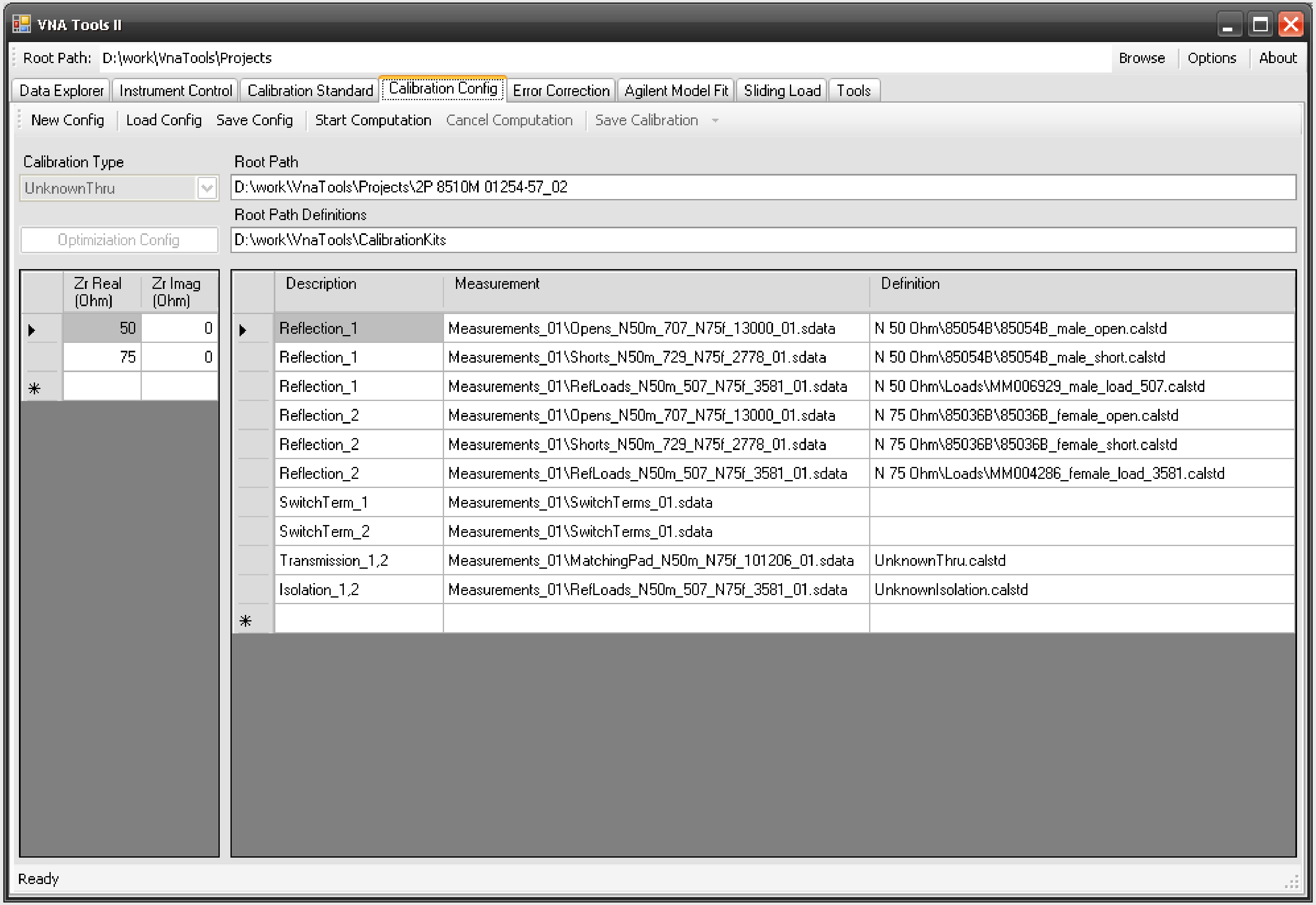


Start (Hz)	Stop (Hz)	Step (Hz)	IFBW (Hz)
45.000e+06	95.000e+06	5.000e+06	0.000e+00
100.000e+06	8.000e+09	100.000e+06	0.000e+00

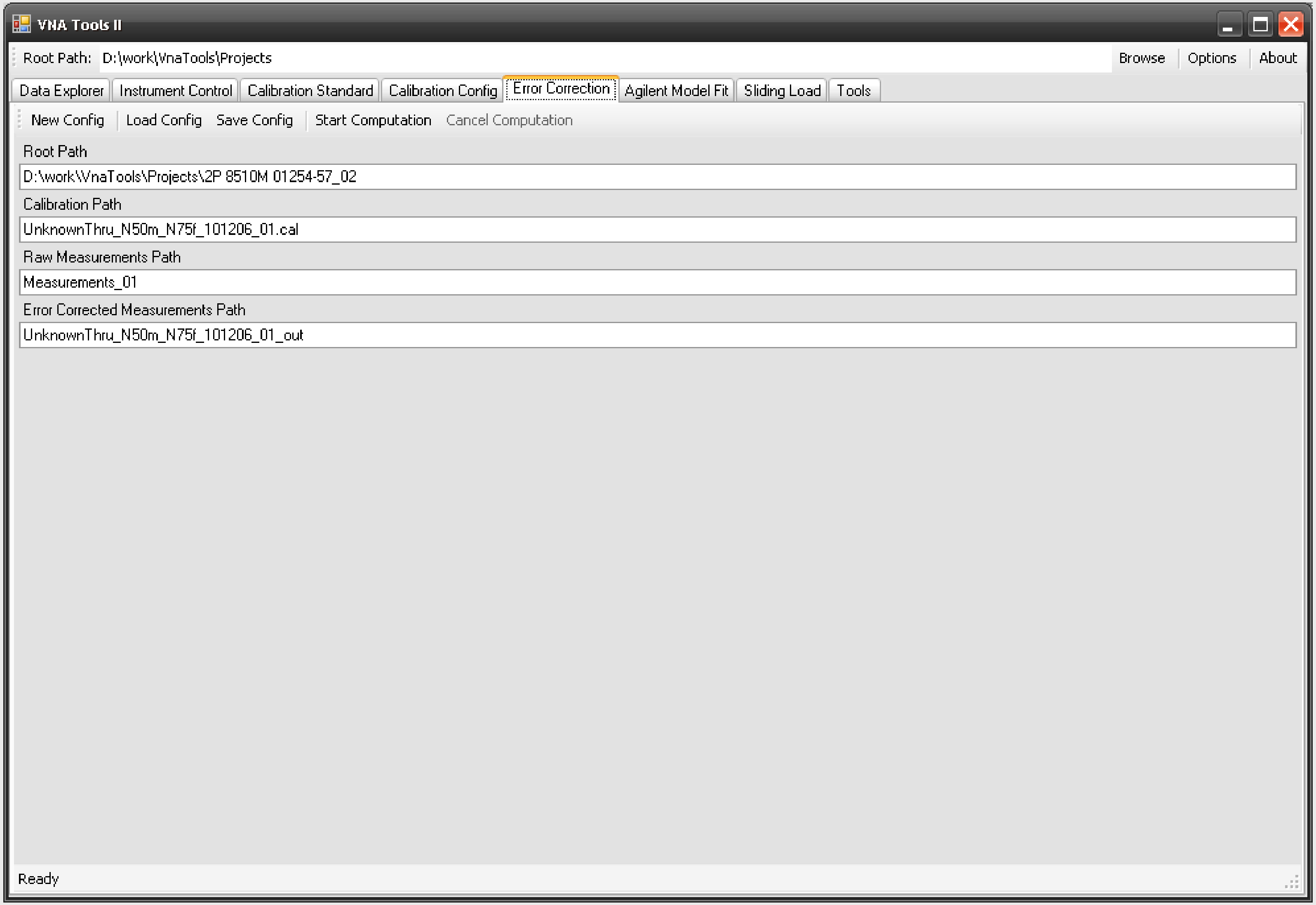
4.5.3 VNA Tools II – Calibration Standard



4.5.4 VNA Tools II – Calibration Config



4.5.5 VNA Tools II – Error Correction



4.6. VNA Tools Script

- VNA Tools II has a scripting possibility ☺
- The end user can script VNA Tools II from any scripting language which supports the .net framework.
  For Example: IronPhyton, MATLAB, Mathematica …
- VNA Tools II provides a high level script class.
  Metas.Vna.Tools.Script
- The user can control instruments and make measurements.
  For example: Step attenuator measurements, drift measurements, …
- Compute a calibration (error and switch terms) and compute the error correction of raw measurements.
- Other script tools: Cascade, Decascade, Reverse, Subset, Mean, Merge, Change Zr, Sliding Load, …

Thank you for your attention …

The VNA Tools II Team in action …

J. Rüfenacht M. Wollensack M. Zeier

Questions