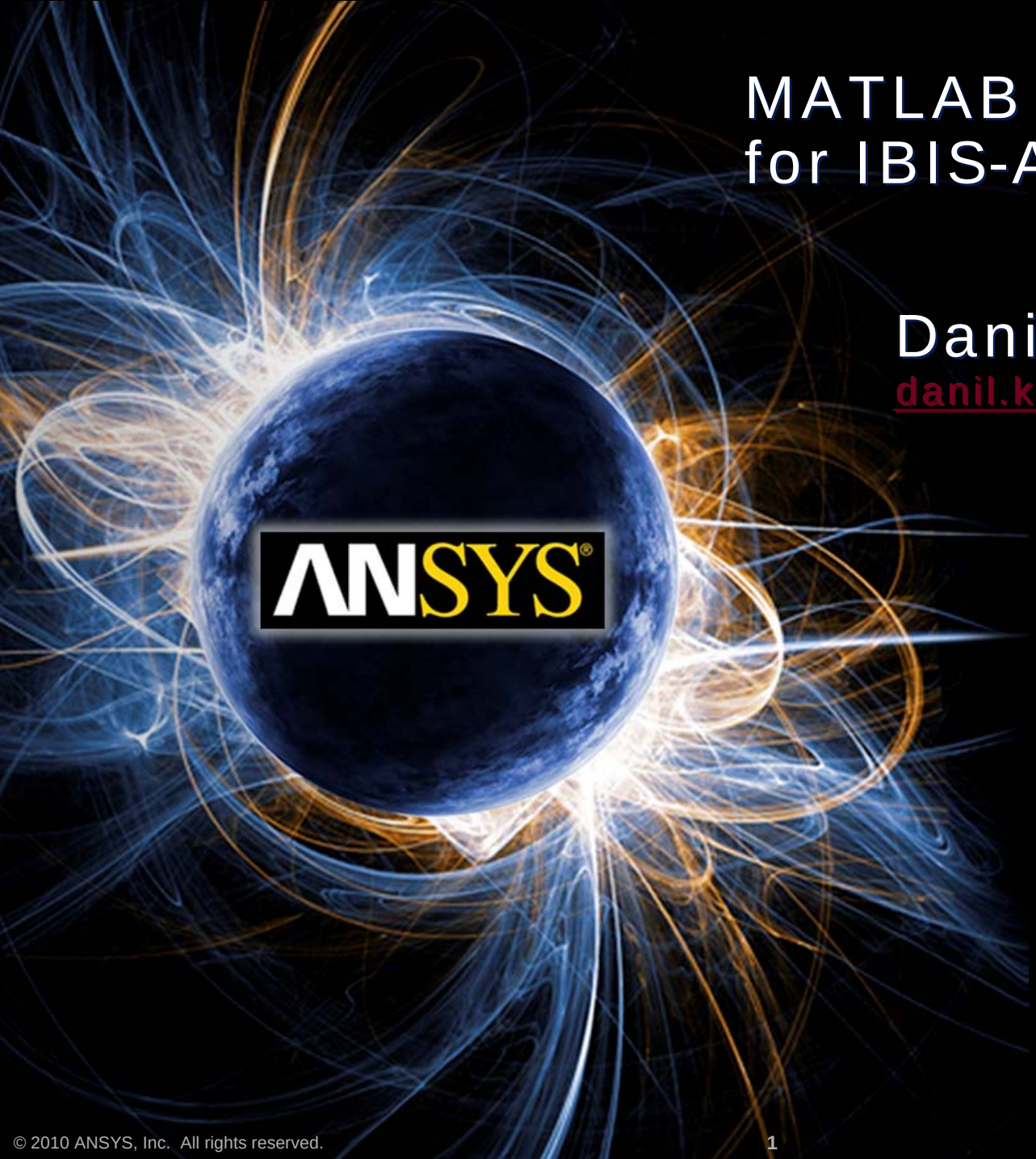


MATLAB co-simulation for IBIS-AMI models

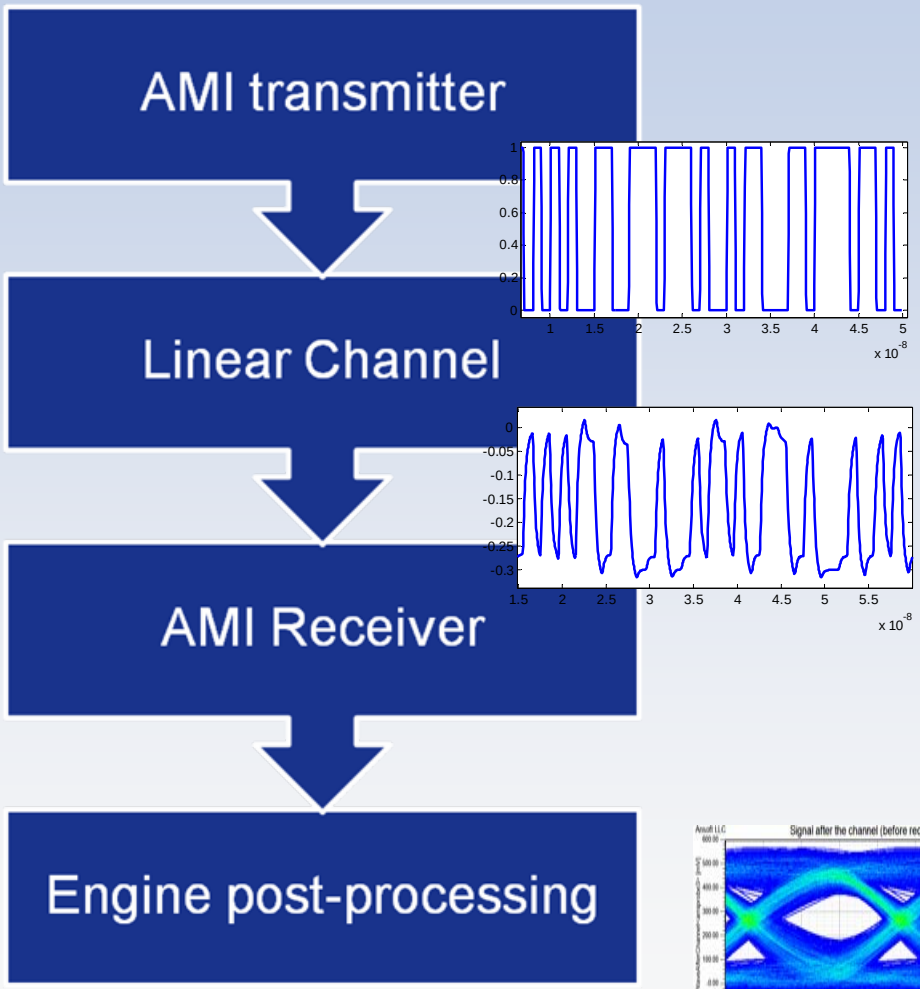
Danil Kirsanov
danil.kirsanov@ansys.com

European IBIS Summit
May 12, 2010
Hildesheim, Germany

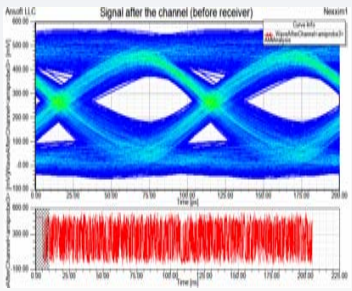
IBIS-AMI in a nutshell

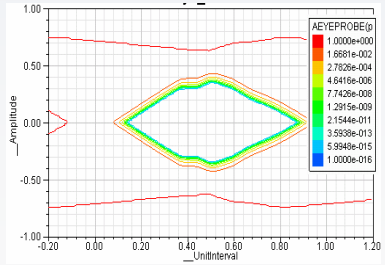
- **AMI stands for Algorithmic Modeling Interface**
- **It allows users to specify their own transmitter and receiver models as C-interface compiled libraries**
  - faster signal processing algorithms
  - intellectual property protection
- **Typically used in statistical tools and convolution (fast) transient engines for channel simulation**
  - Designed to be used with fixed time step data

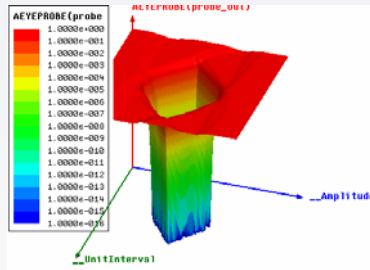
IBIS-AMI in a nutshell



- **Transmitter and receiver are based on user supplied libraries.**
- **Channel is characterized with impulse response function(s)**
- **All signals are sampled at the constant time step and handled in blocks.**







IBIS-AMI Model structure

- **Compiled C-interface library (.dll for windows)**
  - AMI_Init() – initialization, possible linear part of the model
  - AMI_GetWave() – non-linear part of the model
  - AMI_Close()
- **Parameter file (.ami)**
  - Parameters represented as a tree structure
  - (param1 (Usage In) (Type Boolean) (Default True) (Description “my parameter"))
- **IBIS wrapper (.ibis file)**
  - Specifies which library to use for different operating systems, etc.

Motivation

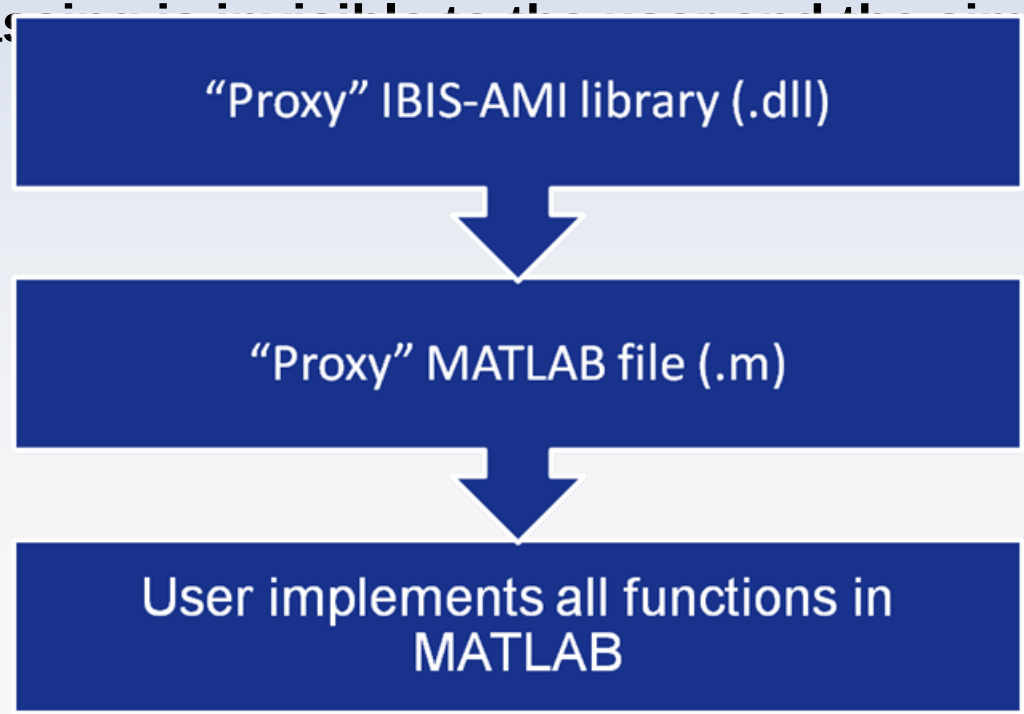
- **AMI models are implemented with high-level “signal processing” algorithms**
  - often needs least squares, fft, non-trivial matrix operations
- **MATLAB is a perfect tool for prototyping/developing/verifying such models**
  - Wide range of built-in functions and algorithms
  - Broad web community
  - Development/debugging is much faster than in C++
  - Safe range/memory handling
  - Familiar to most engineers
  - Same code for all platforms

Possible issues

- **Speed**
- **Intellectual property protection**
  - In case models are delivered to customers
  - Both code and data safety
- **Compatibility with the current IBIS-AMI standard**

General Idea

- **Instead of implementing AMI_Init, AMI_GetWave and AMI_Close function in C++, user implements them in MATLAB**
  - Interfaces are nearly identical
- **There is a proxy library that exchanges the data with MATLAB engine**
- **All data pas[illegible] ulation engine**



General Idea

- **Instead of implementing AMI_Init, AMI_GetWave and AMI_Close function in C++, user implements them in MATLAB**
  - Interfaces are nearly identical
- **There is a proxy library that exchanges the data with MATLAB engine**
- **All data passing is invisible to the user and the simulation engine**

C++:

```
long AMI_GetWave(double *wave_in,
                 long wave_size,…)
{
    double mean_wave = 0.0;
    for(int i=0; i<wave_size; ++i)
    {
        mean_wave += wave[i];
    }
    mean_wave /= (double) wave_size;
    …
}
```

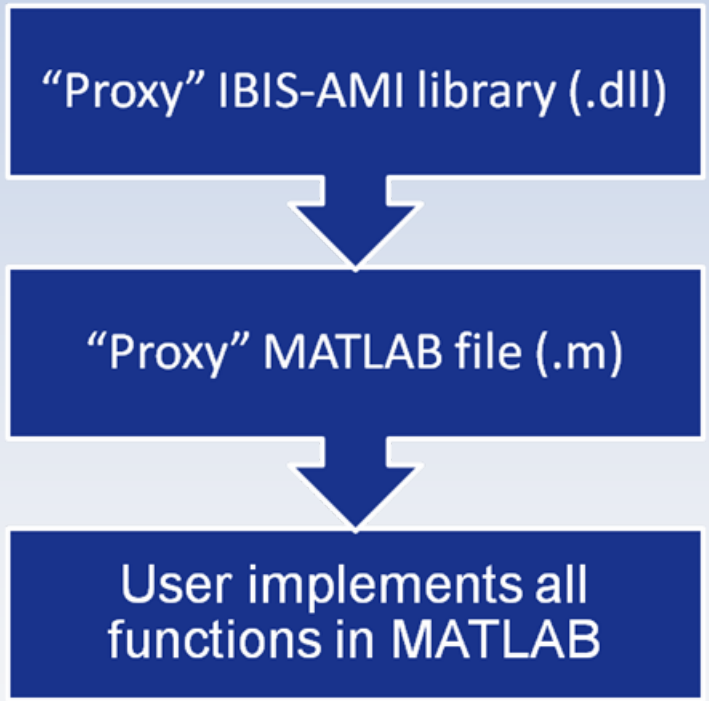
MATLAB:

```
function [error_code,…]
        =AMI_GetWave(wave_in,…)

mean_wave = mean(wave_in);
…
```

Possible issues: Compatibility

- **Fully compatible with the current IBIS-AMI standard**
  - Simulator believes it deals with the regular AMI library
  - Proxy MATLAB file takes care of the model data storage
  - Variables *matlab_proxy_path* and *matlab_proxy_function* are added to the model's .ami file



Possible issues: Speed

- **No speed impact for statistical simulators**
  - AMI_Init() function is called only once
- **Transient "convolution" simulation**
  - Depends on the complexity of AMI_GetWave
  - Most of the time in AMI simulation is usually spent not in AMI functions
    - channel convolution
    - eye diagram post-processing
  - In our tests Matlab-based simulation is about 10%-15% slower than C++-based simulation

Possible issues: Intellectual Property

- **Intellectual property**
  - Convert user's files to pseudocode (see *pcode* command)
    - In theory, the data that the model returns to the "proxy" MATLAB function can be accessible
  - MATLAB compiler

Details: AMI_Init()

- **Simulation engine calls AMI_Init() in proxy dll, passes all the data**
  - channel impulse response, parameters string, etc
- **Proxy dll passes the data to the proxy MATLAB function**
- **Proxy MATLAB function:**
  - generates new unique ID for this model
  - calls user's MATLAB file with AMI_Init function
  - stores model's internal data in MATLAB
  - returns the results to the proxy dll
    - Returns model ID instead of the model's internal data

Details: AMI_GetWave()

- **Simulation engine calls AMI_GetWave() in proxy dll**
- **Proxy dll**
  - extracts model's ID from the data stored by engine
  - passes ID and rest of the input to the proxy MATLAB function
- **Proxy MATLAB function:**
  - extracts the model's internal data, using ID
  - calls user's MATLAB file with AMI_GetWave function
  - stores model's internal data in MATLAB
  - returns the results to the proxy dll

Conlcusion

- **It is possible to run IBIS-AMI models written in MATLAB**
  - Compatible with IBIS-AMI specs
  - Does not need additional support from the simulation engines
  - Single proxy libraries fits all vendors
  - Reasonable speed impact
  - Data/code security comments are welcome
- **Proxy libraries are currently in testing**
- **Questions, comments, beta-testing**
  - danil.kirsanov@ansys.com