MAPP: The Berkeley Model and Algorithm Prototyping Platform

Jaijeet Roychowdhury **Tianshi Wang**

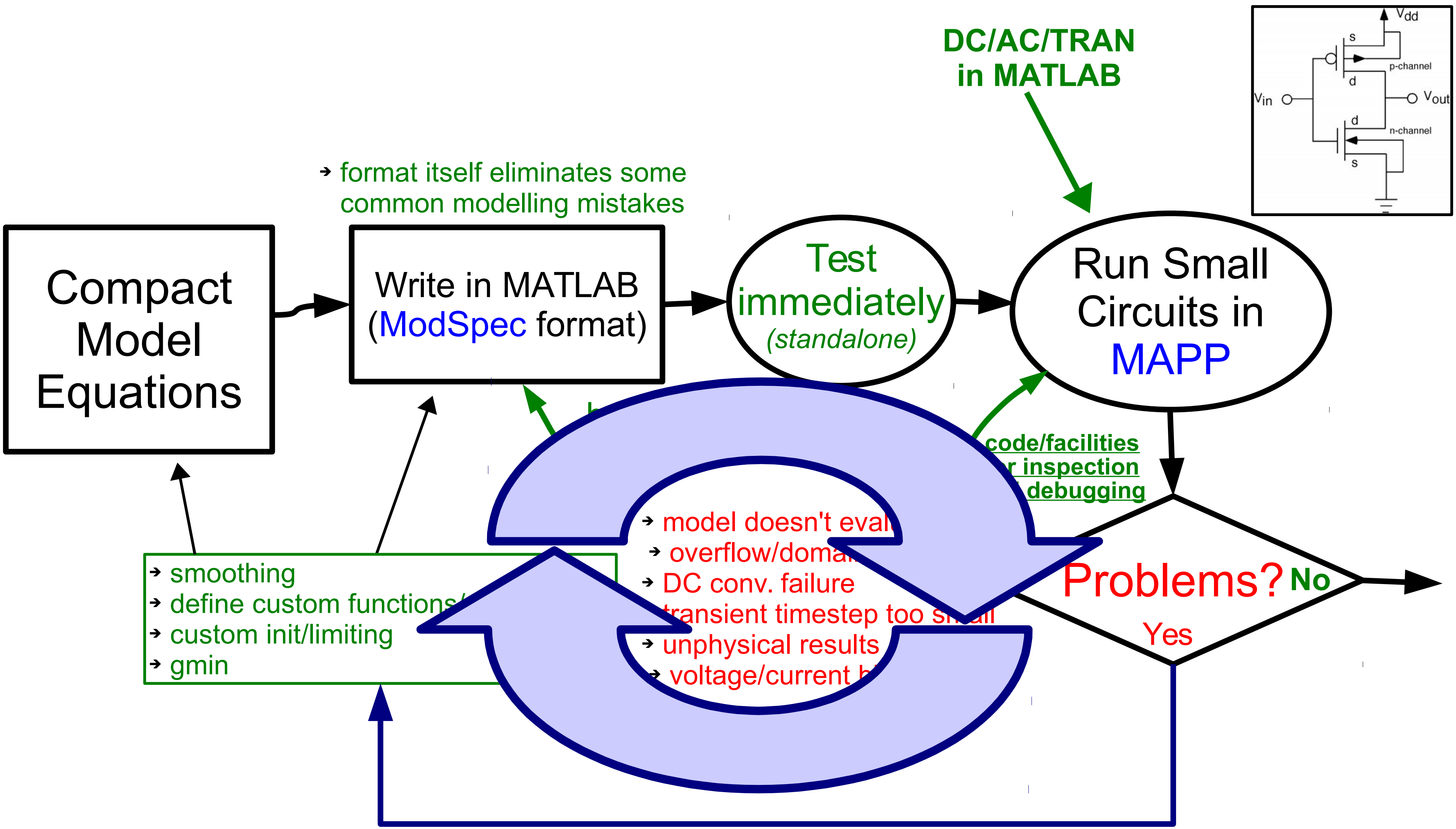
EECS Department
University of California at Berkeley

NEEDS
nanoHUB.org

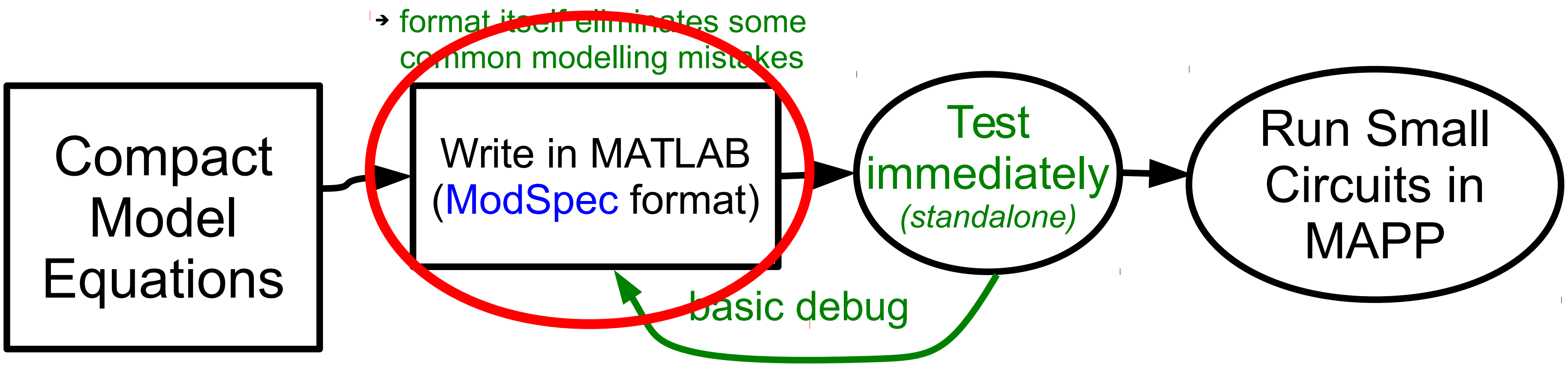
Motivation for MAPP

- Developing **good** compact models: many pitfalls
  - » examples: discontinuities/smoothness, well-posedness (has transient solution?)
    - – problems usually discovered at deployment (ie, during simulation)
  - » problems often hard to debug and resolve
    - – compact model developer blames simulator
    - – simulator people blame the model
- One goal of MAPP: to ease this problem
  - » In MATLAB
    - – empowers non-programmers to debug models
    - – case studies + knowledge base → find solution to common problems (and try in MAPP) quickly

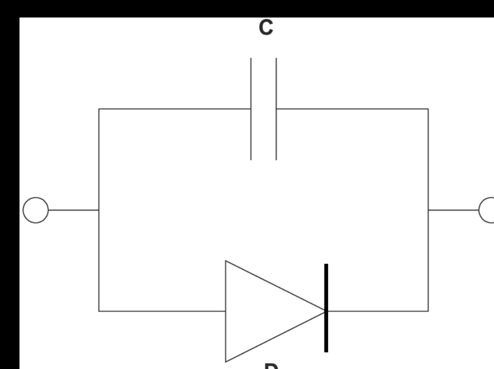
MAPP for Device Model Development



Glimpse: ModSpec Device Definition

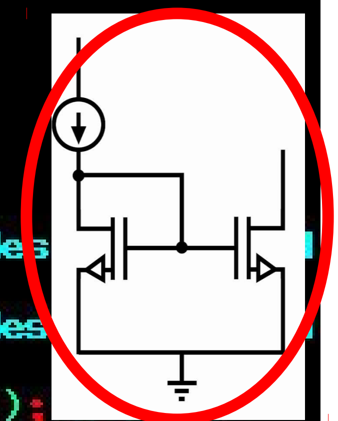



```
function MOD = diodeCapacitor_ModSpec_wrapper()
% ModSpec description of an ideal diode in parallel with a capacitor
   MOD = ee_model();
   MOD = add_to_ee_model(MOD, 'external_nodes', {'p', 'n'});
   MOD = add_to_ee_model(MOD, 'explicit_outs', {'ipn'});
   MOD = add_to_ee_model(MOD, 'parms', {'C',2e-12, 'Is',1e-12, 'VT',0.025});
   MOD = add_to_ee_model(MOD, 'f', @f);
   MOD = add_to_ee_model(MOD, 'q', @q);
end

function out = f(S)
    v2struct(S);
    out = Is*(exp(vpn/VT)-1);
end

function out = q(S)
    v2struct(S);
    out = C*vpn;
end
"diodeCapacitor_ModSpec_wrapper.m" 19L, 548C written          1,1          All
```

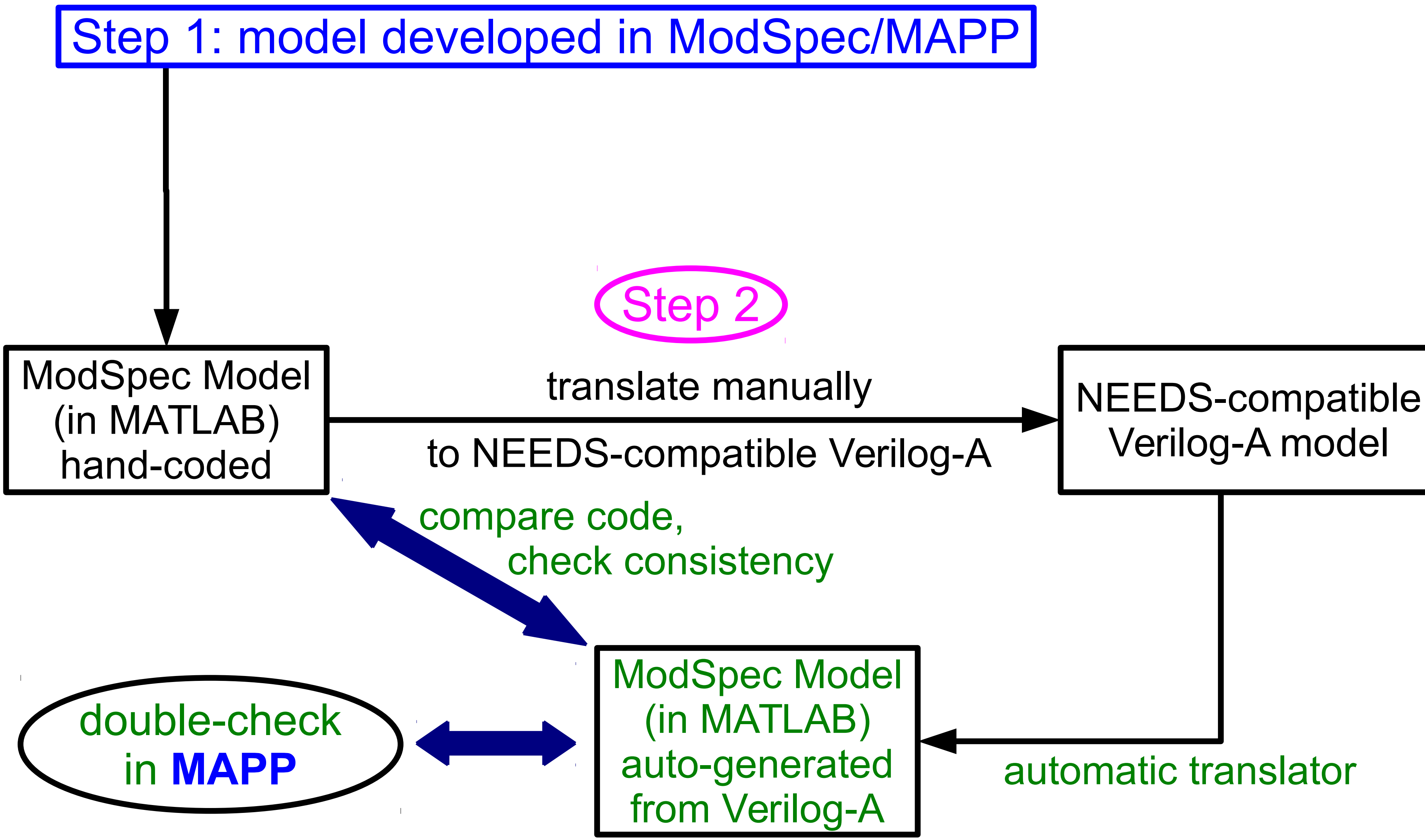


Glimpse: Circuit Netlist in MAPP

```
function cktnetlist = current_mirror_ckt()
    % MDE netlist for a simple current mirror
    % Author: Bichen Wu, 2013/10/22
    cktnetlist.cktname = 'current_mirror';
    cktnetlist.nodenames = {'D1', 'D2', 'M2D', 'G'}; % nodes in the circuit
    cktnetlist.groundnodename = 'gnd';

    Vd = 2; k = 1e-4; I0 = 10e-6; RL = 1e4; W = 270; L = 180; DSgmin = 1e-8;

    % define a time-varying function for the voltage source
    utfunc = @(t,args) args.offset + args.A * sin(2*pi*args.f*t);
    utargs.offset = I0; utargs.A = 1e-6; utargs.f = 2e7;

    % set up the netlist
    cktnetlist = add_element(cktnetlist, vsrcModSpec(), 'VD1', ...
        {'D1', 'gnd'}, {}, {{'E', {'DC', Vd}}}); % vsrc between nodes
    cktnetlist = add_element(cktnetlist, vsrcModSpec(), 'VD2', ...
        {'D2', 'gnd'}, {}, {{'E', {'DC', Vd}}}); % vsrc between nodes
    cktnetlist = add_element(cktnetlist, isrcModSpec(), 'I0', ...
        {'D1', 'G'}, {}, {{'I', {'DC', I0},{'TRAN',utfunc,utargs}}});
    cktnetlist = add_element(cktnetlist, resModSpec(), 'R', ...
        {'D2', 'M2D'}, {{'R', RL}}, {}); % resistor betwen D2 and M2D
    cktnetlist = add_element(cktnetlist, SH_MOS_ModSpec(), 'M1', ... % n-MOS
        {'G', 'G', 'gnd'}, {{'Type', 'N'}, {'Beta', k*W/L}, {'VT', 0.3}}, {});
    cktnetlist = add_element(cktnetlist, SH_MOS_ModSpec(), 'M2', ... % n-MOS
        {'M2D', 'G', 'gnd'}, {{'Type', 'N'}, {'Beta', k*W/L}, {'VT', 0.3}}, {});
end % function current_mirror_ckt
"current_mirror_ckt.m" 27L, 1455C written                    1,1          All
```



MAPP Model Development Flow (2)



end of Step 2: high level of confidence Verilog-A model is correct/debugged

Glimpse: Verilog-A → ModSpec Translator

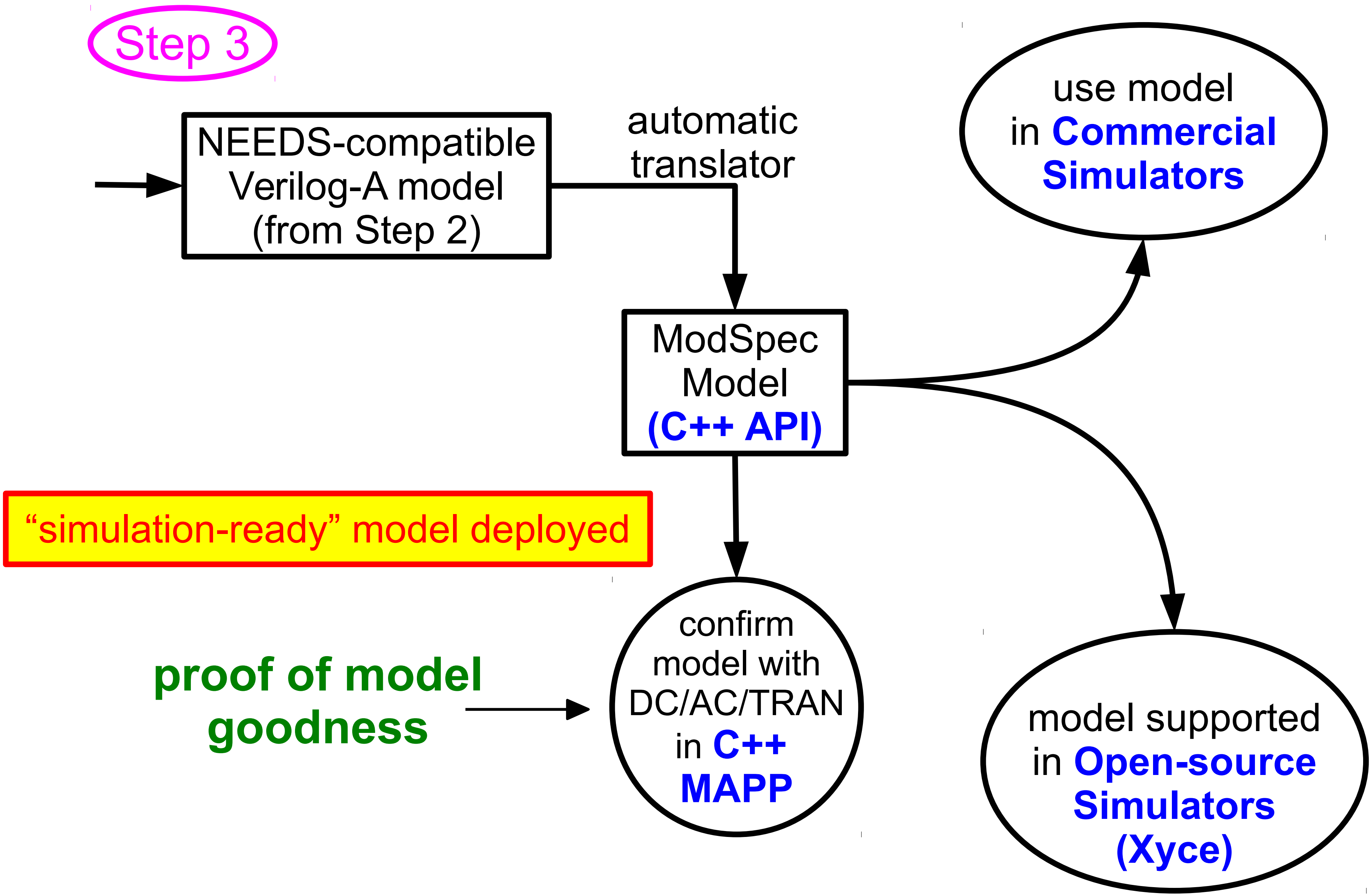
```
// Series RLC
// Version 1a, 1 June 04
// Ken Kundert
//
// Downloaded from The Designer's Guide Community
// (www.designers-guide.org).
// Taken from "The Designer's Guide to Verilog-AMS"
// by Kundert & Zinke. Chapter 3, Listing 14.

`include "disciplines.vams"

module series_rlc2 (p, n);
    parameter real r=1000;      // resistance (Ohms)
    parameter real l=1e-9;      // inductance (H)
    parameter real c=1e-6;      // capacitance (F)
    inout p, n;
    electrical p, n, i;
    branch (p, i) rl, (i, n) cap;

    analog begin
    V(rl) <+ r*I(rl);
    V(rl) <+ ddt(l*I(rl));
    I(cap) <+ ddt(c*V(cap));
    end
endmodule
```

Verilog-A

```
function MOD = series_rlc2_ModSpec_wrapper ()
    MOD = ee_model();
    MOD = add_to_ee_model (MOD, 'terminals', {'p', 'n'});
    MOD = add_to_ee_model (MOD, 'explicit_outs', {'ipn'});
    MOD = add_to_ee_model (MOD, 'internal_unks', {'Irl'});

    MOD = add_to_ee_model (MOD, 'parms', {'r', 1000});
        % resistance (Ohms)
    MOD = add_to_ee_model (MOD, 'parms', {'l', 1e-9});
        % inductance (H)
    MOD = add_to_ee_model (MOD, 'parms', {'c', 1e-6});
        % capacitance (F)
    MOD = add_to_ee_model (MOD, 'fqei_all', @fqei);
    MOD = finish_ee_model(MOD);
end

function [Vrl, Frl, Icap, Qcap] = series_rlc2(Irl, Vcap, r, l, c)
    Vrl = r*Irl;
    Frl = l*Irl;
    Icap = 0;
    Qcap = c*Vcap;
end

function [fe, qe, fi, qi] = fqei(S)
    v2struct(S);

    Vcap = vpn;
    [Vrl, Frl, Icap, Qcap] = series_rlc2(Irl, Vcap, r, l, c);

    fe(1,1) = Irl + Icap;
    qe(1,1) = 0 + Qcap;
    fi(1,1) = vpn - Vrl;
    qi(1,1) = 0 - Frl;
end
```

ModSpec (MATLAB)

MAPP Model Development Flow (3)



Glimpse: ModSpec Model in Xyce

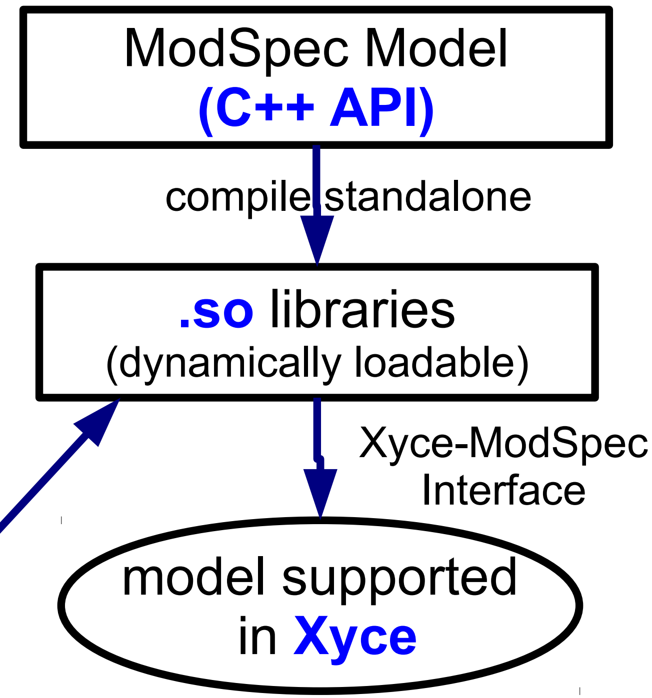
```
*** Test-bench for generting dc response of an inverter

*** Creat sub-circuit for the inverter
.subckt inverter Vin Vout Vvdd Vgnd

yModSpec_Device X1 Vvdd Vin Vout Vvdd MVSmod type=-1 W=1.0e-4
   Lgdr=32e-7 dLg=8e-7 Cg=2.57e-6 beta=1.8 alpha=3.5 Tjun=300
   Cif = 1.38e-12 Cof=1.47e-12 phib=1.2 gamma=0.1 mc=0.2
   CTM_select=1  Rs0=100 Rd0 = 100 n0=1.68 nd=0.1 vxo=7542204
   mu=165 Vt0=0.5535 delta=0.15

yModSpec_Device X0 Vout Vin Vgnd Vgnd MVSmod type=1 W=1e-4
   Lgdr=32e-7 dLg=9e-7 Cg=2.57e-6 beta=1.8 alpha=3.5 Tjun=300
   Cif=1.38e-12 Cof=1.47e-12 phib=1.2 gamma=0.1 mc=0.2
   CTM_select=1 Rs0=100 Rd0=100 n0=1.68 nd=0.1 vxo=1.2e7
   mu=200 Vt0=0.4 delta=0.15

.model MVSmod MODSPEC_DEVICE SONAME=MVS_ModSpec_Element.so

.ends

*** circuit layout
Vsup sup 0 1
Vin in 0 0
Vsource source 0 0
X2  in out sup 0 inverter

*** simulation
.dc Vin 0 1 0.01

.print dc V(in)
*** END
.end
```

.model line

model's name

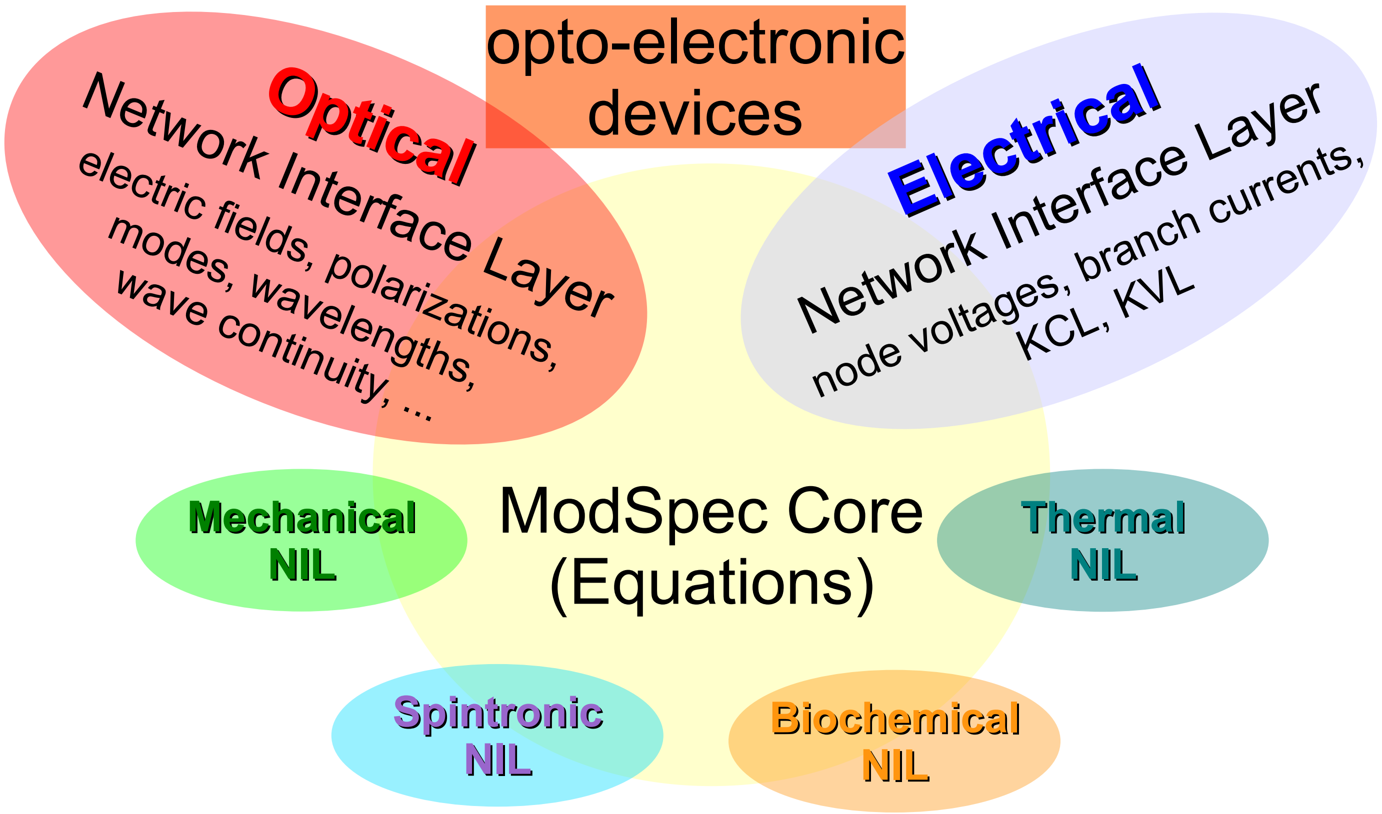
model parameter: name of .so library

Xyce netlist for inverter (using MVS ModSpec/C++ model)

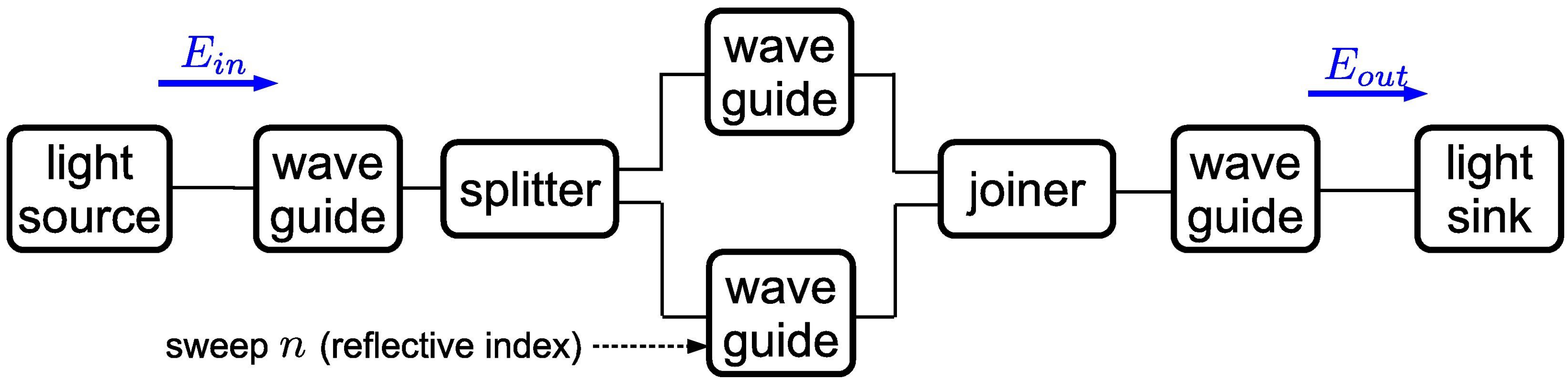


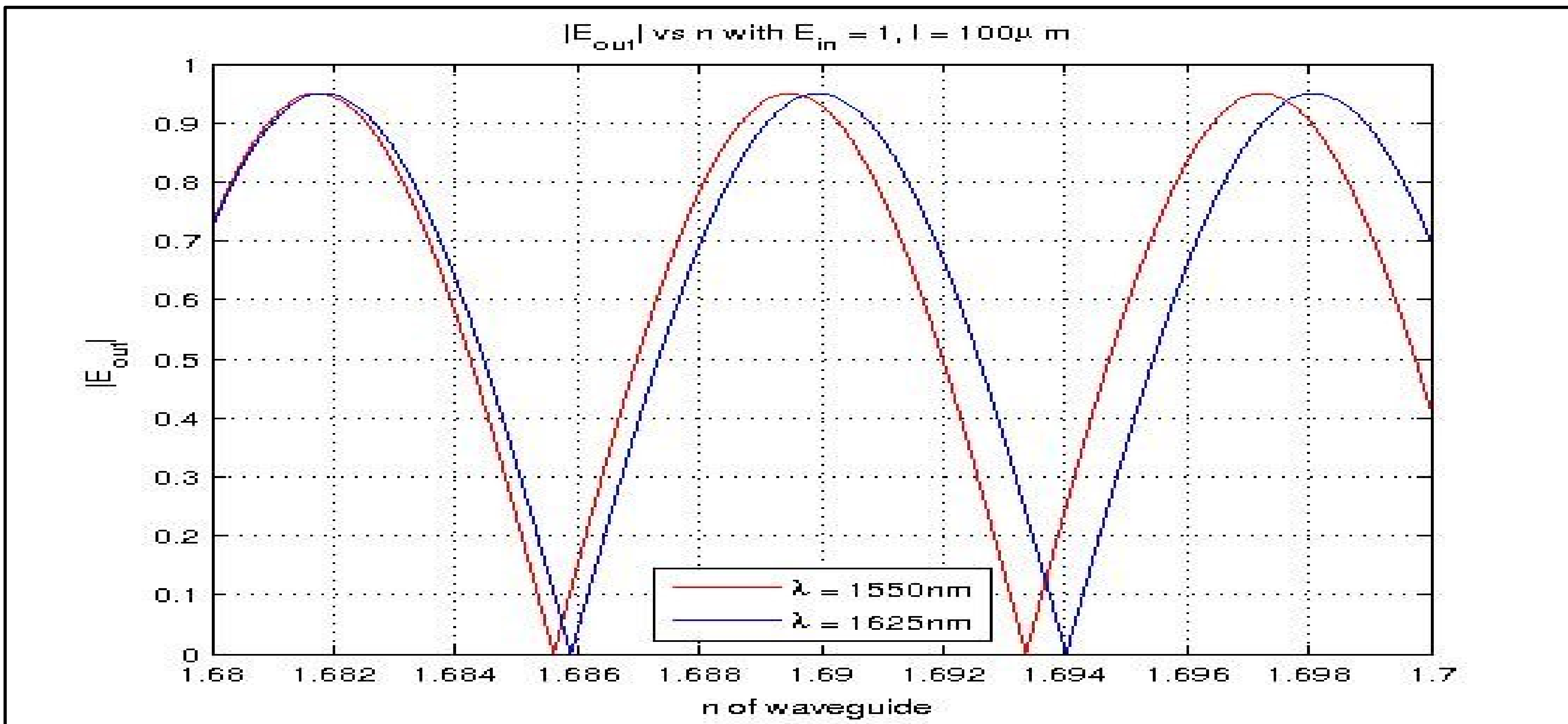
DC sweep using Xyce

ModSpec: Multi-Physics Support



Optical System Modelling/Simulation Example





MAPP: First Public Release

- Open Source download: http://mapp.eecs.berkeley.edu
  - » mailing list (MAPP announcements/discussion)
  - » bug reporting and tracking site
  - » git repository access (you can contribute)
- License
  - » primary: GPL-v3
  - » alternative licensing available
    - – eg, SRC contract terms apply for SRC company use
  - » contributors can specify their own alternative licensing terms for their contributions

MAPP: Features

- Entirely in MATLAB
- Help system (start with `help MAPP`)
  - » quick start walk-through
- Automatic differentiation (vecvalder)
  - » `help MAPPautodiff`
- Executable device specification (ModSpec)
  - » examples, tutorial: part of help
- DC, AC, transient analyses
  - » also noise, homotopy, HB, shooting, PPV, MOR, etc. (not released yet)
- Automated testing system exercising suite of tests

MAPP: Intended Uses

- Developing simulation-ready device models
  - » including multi-physics devices, network connectivity
- Quickly prototyping new simulation algorithms
  - » hours/days to implement a new analysis
    - – assess strengths/limitations before investing resources to implement in “real simulators”
  - » MAPP's structuring is different from SPICE's
    - – devices “don't know about” algorithms; and vice-versa
    - – central concept: DAEs (connect analyses and devices)
- Learning or teaching modelling/simulation
  - » MATLAB → broadly accessible
  - » help system, tutorials, supporting resources

mapp.eecs.berkeley.edu