MAPP: A Platform for Prototyping Algorithms and Models Quickly and Easily

Tianshi Wang, Karthik Aadithya,
Bichen Wu and Jaijeet Roychowdhury

EECS Department
University of California at Berkeley

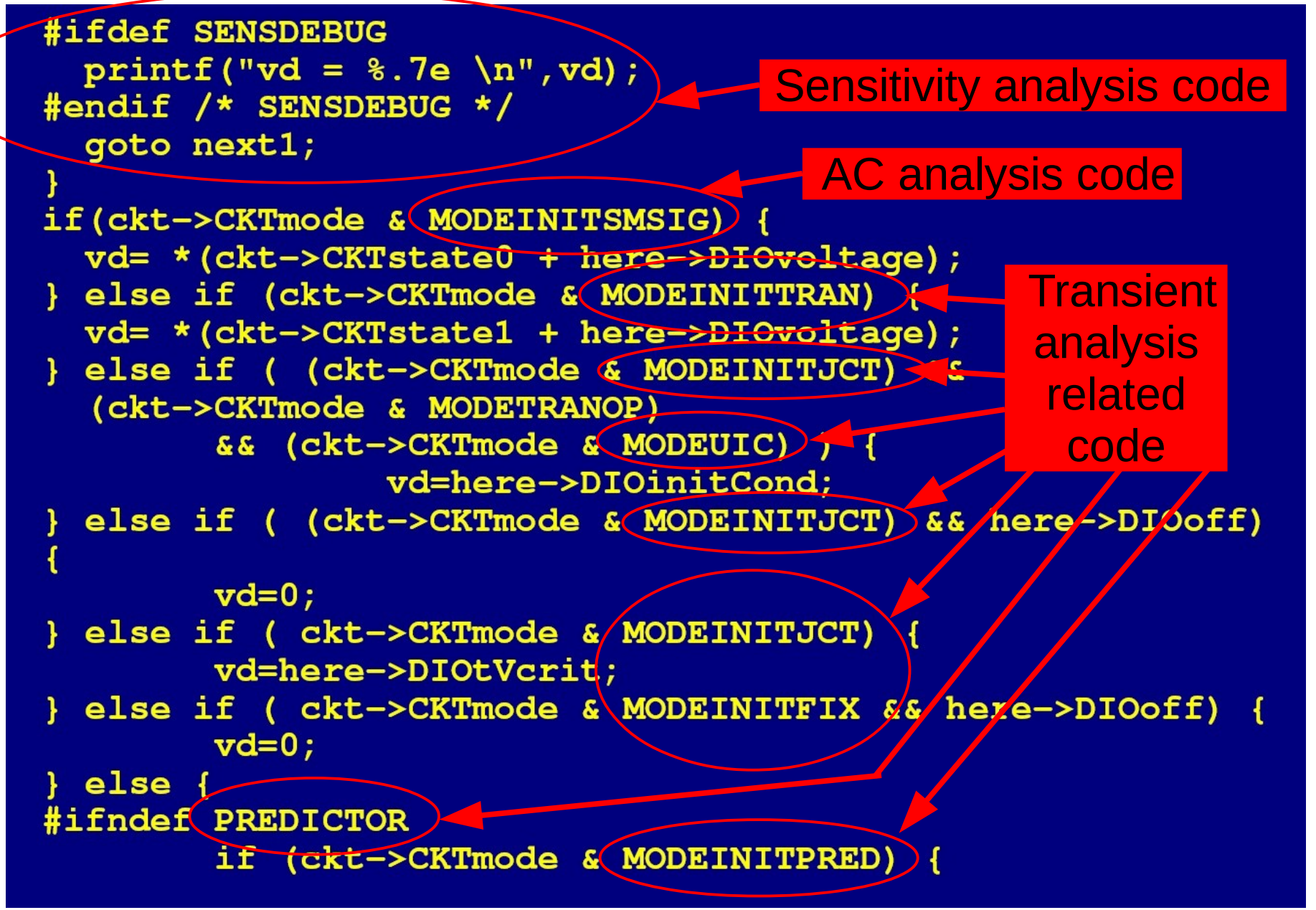
Motivation for MAPP

- **Berkeley Model and Algorithm Prototyping Platform**
- Developing **good** compact models: many pitfalls
  - » examples: discontinuities/smoothness, well-posedness
    - – problems usually discovered at deployment (ie, during simulation)
  - » problems often hard to debug and resolve
    - – compact model developer and simulator people blame each other
- Anyone working in simulation algorithms today needs
  - » device models: BSIM, MOS1, MOS2, MOS3, …
  - » base algorithms: **robust nonlinear solution**, transient, HB/shooting, …
  - » parsing, equation formulation, output, …
  - » huge (waste of) effort of re-development of basic capabilities
- One goal of MAPP: to ease these problems
  - » common, open-source simulation framework
  - » in MATLAB
    - – empowers non-programmers to debug models and algorithms

Why not use SPICE?

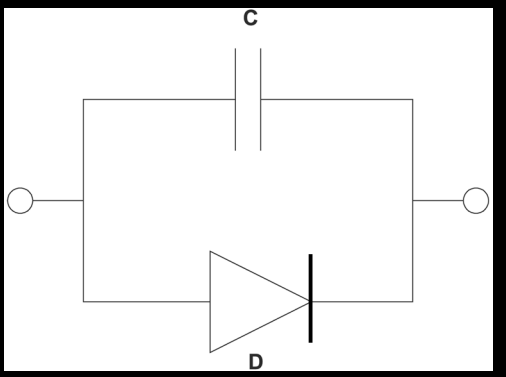
- SPICE: the original open-source simulator
  - » de-facto standard
  - » structure: all analyses in all models
  - » prototyping models & algorithms: takes months to years
  - » pain to write (even for those who can)
    - –e.g., shooting method (S-SPICE)
- To be useful: **modular, well-structured, flexible**
  - » separated models, algorithms, numerics, I/Os
  - » simple, clean interfaces
  - » short, easy to read, easy to modify

Excerpt from *dioload.c* (SPICE3)



Glimpse: Diode Model in MAPP

```
 1 function MOD = diodeCapacitor_ModSpec_wrapper()
 2 % ModSpec description of an ideal diode in parallel with a capacitor
 3     MOD = ee_model();
 4     MOD = add_to_ee_model(MOD, 'external_nodes', {'p', 'n'});
 5     MOD = add_to_ee_model(MOD, 'explicit_outs', {'ipn'});
 6     MOD = add_to_ee_model(MOD, 'parms', {'C',2e-12, 'Is',1e-12, 'VT',0.025});
 7     MOD = add_to_ee_model(MOD, 'f', @f);
 8     MOD = add_to_ee_model(MOD, 'q', @q);
 9 end
10
11 function out = f(S)
12     v2struct(S);
13     out = Is*(exp(vpn/VT)-1);
14 end
15
16 function out = q(S)
17     v2struct(S);
18     out = C*vpn;
19 end
"diodeCapacitor_ModSpec_wrapper.m" 19L, 548C written          1,1           All
```



MOD.terminals
MOD.parms
MOD.explicit_outs
MOD.f: function handle
MOD.q: function handle
…

- **executable (in Matlab)**
- **takes 10min to write**
- **works in all analyses**

Glimpse: Shooting Method in MAPP

Shooting Algorithm in MAPP (pseudo-code)

```
shootObj = shoot(DAE): // constructor
 1: shootObj.DAE = DAE;
 2: shootObj.tranObj = LMS(DAE); // transient simulation object
 3: set up member functions: .solve, .g, and .J
 4: return  shootObj;

shootObj.solve (initguess, T):
 1: x0 ← NR(@g, @J, initguess);
 2: shootSols = tranObj.solve(x0, 0, T);
 3: return  shootSols;

shootObj.g (x0):
 1: tranSols = tranObj.solve(x0, 0, T);
 2: return  gout = tranSols(:, n) - x0;

shootObj.J (x0):
 1: tranSols = tranObj.solve(x0, 0, T);
 2: Ci_pre = DAE.dq_dx(x0);
 3: M = eye(n);
 4: for i = 2:n do
 5:     x = tranSols(:, i); u = inputs(:, i);
 6:     Ci = DAE.dq_dx(x); Gi = DAE.df_dx(x, u);
 7:     M = (Ci + (tpts(i) - tpts(i-1)) * Gi) \ Ci_pre * M;
 8:     Ci_pre = Ci;
 9: end for
10: return  Jout = M - eye(n);
```

object-oriented

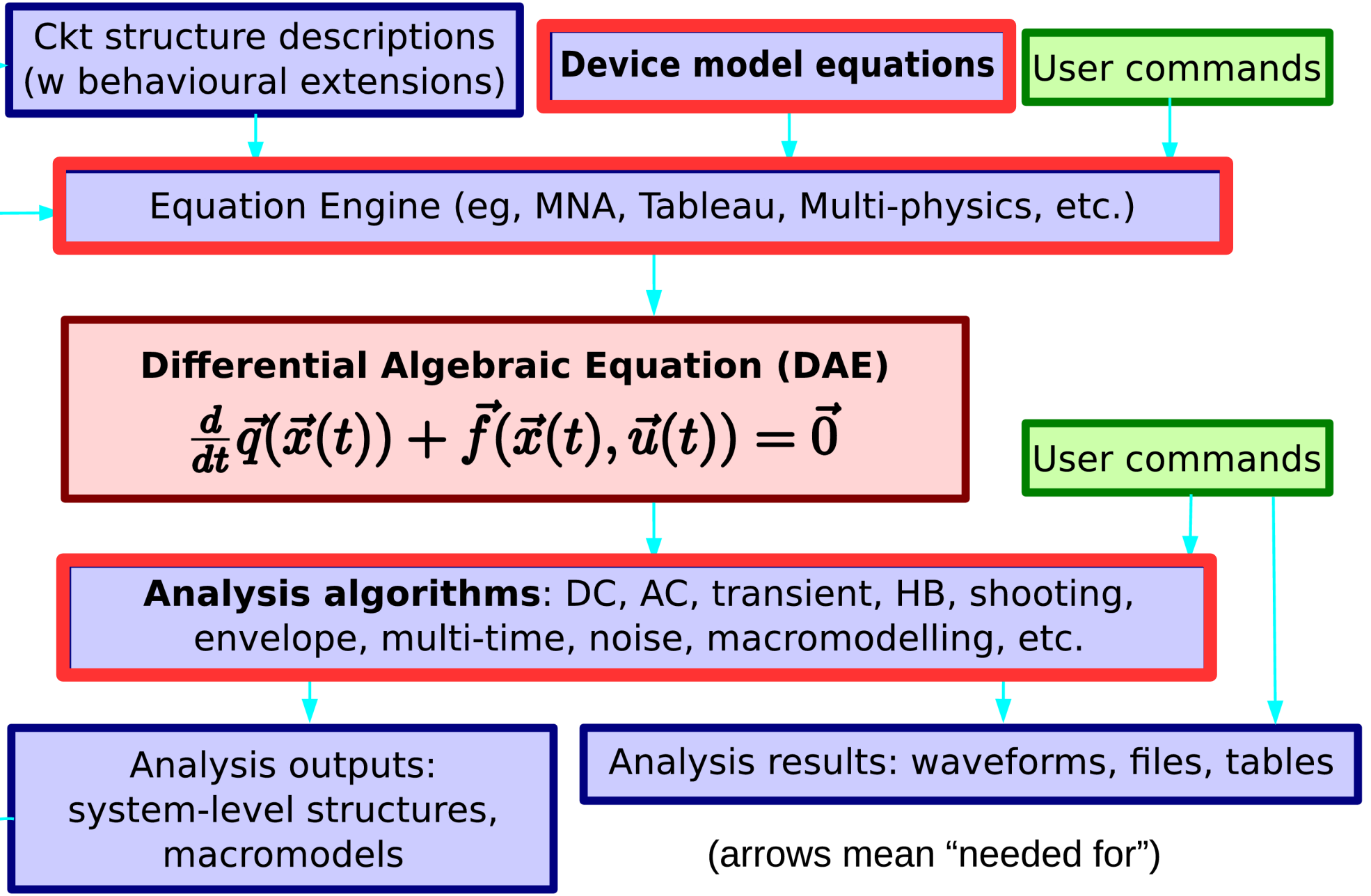
reuses LMS (transient) code

150 lines of code

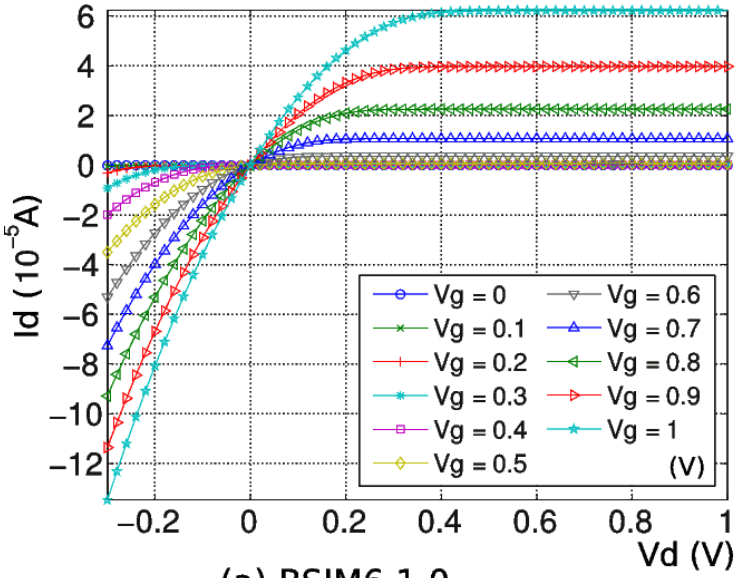
works with all devices, circuits, domains

a pleasure to write (you too can do it)

Code Structuring of MAPP

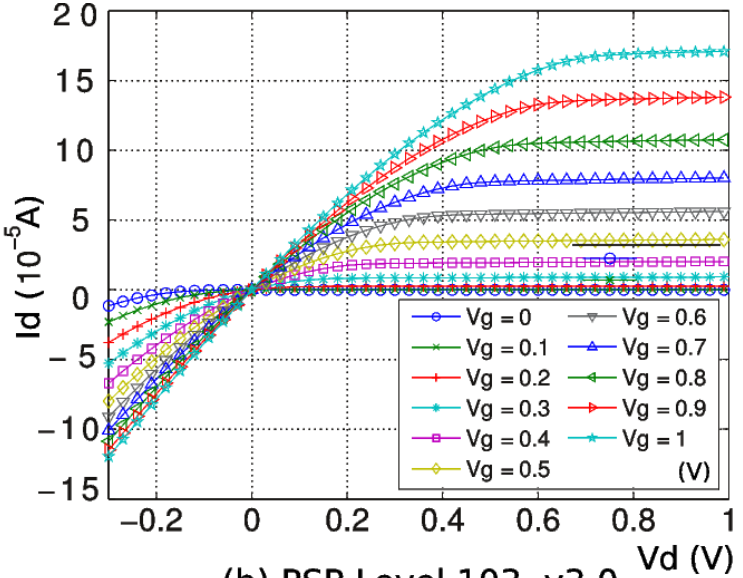


MAPP: Compact Model Prototyping



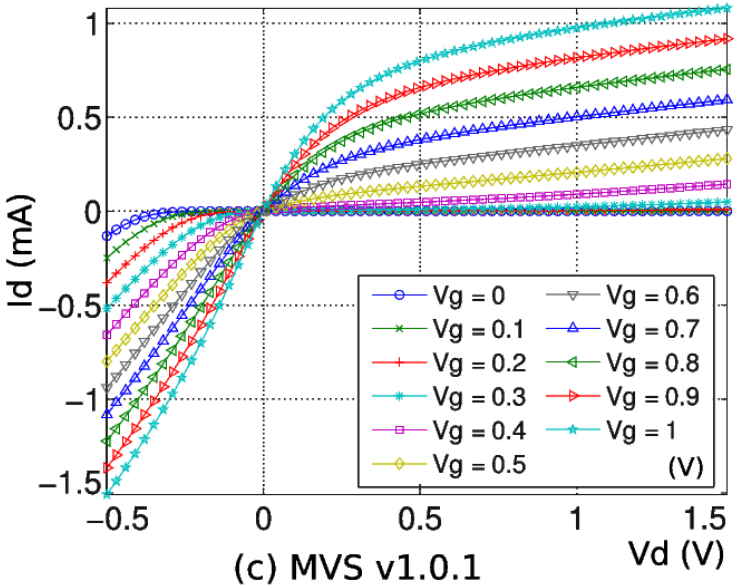
(a) BSIM6.1.0

default: L=10μm, W=10 μm



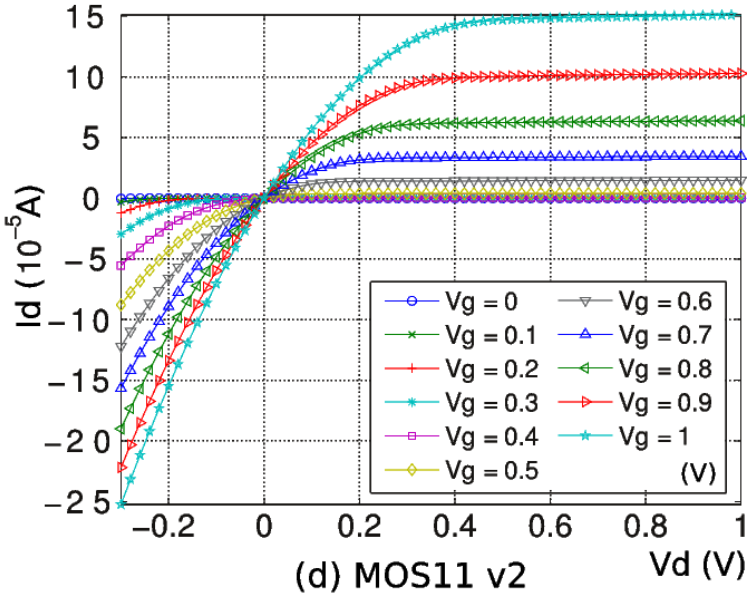
(b) PSP Level 103 v3.0

default: L=10μm, W=10 μm



(c) MVS v1.0.1

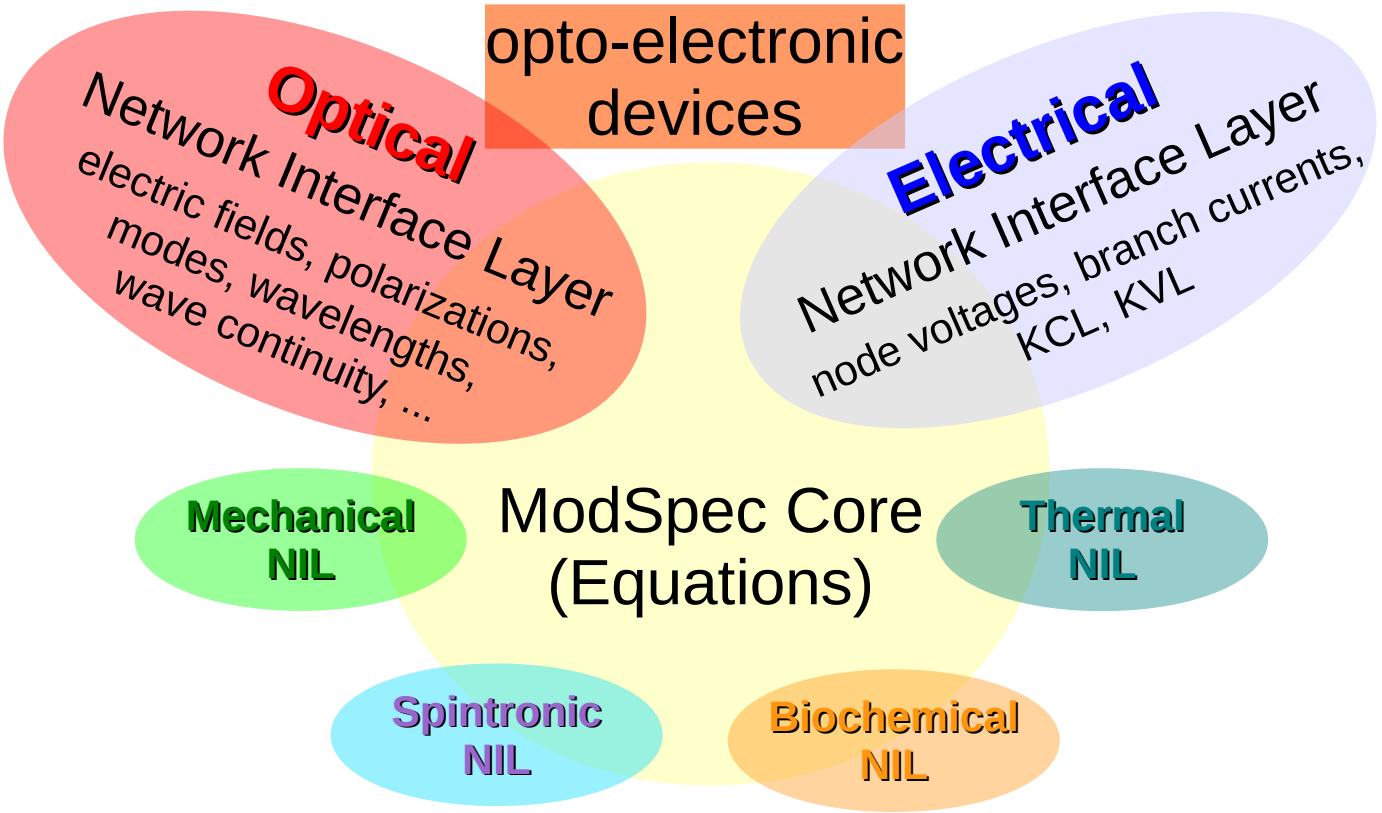
default: L=80nm , W=1μm



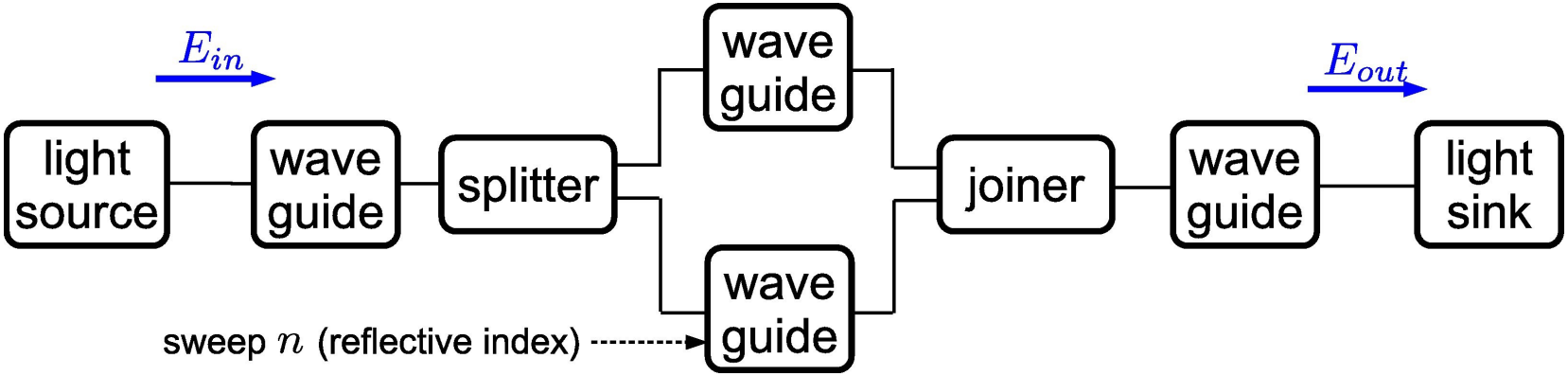
(d) MOS11 v2

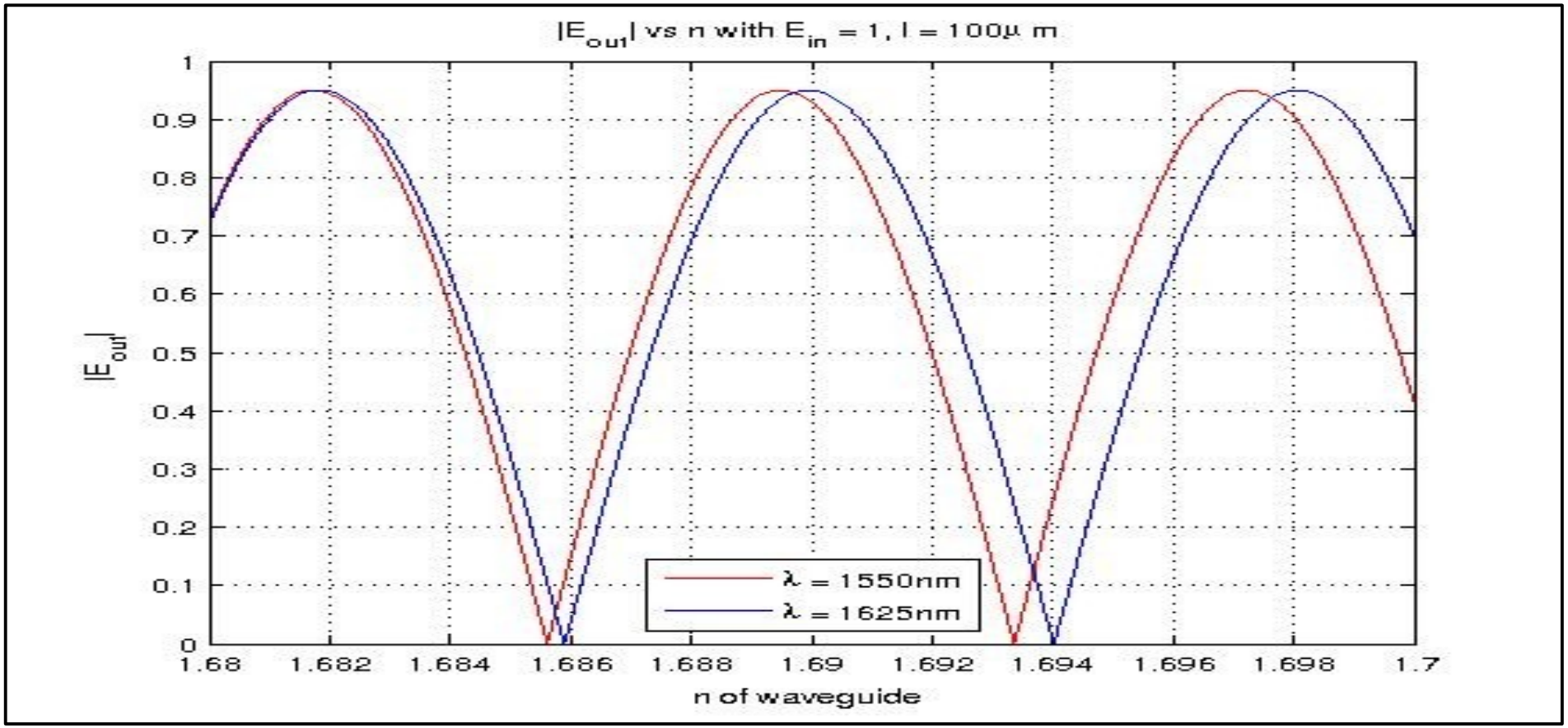
default: L=1μm, W=1μm

MAPP: Multi-Physics Support

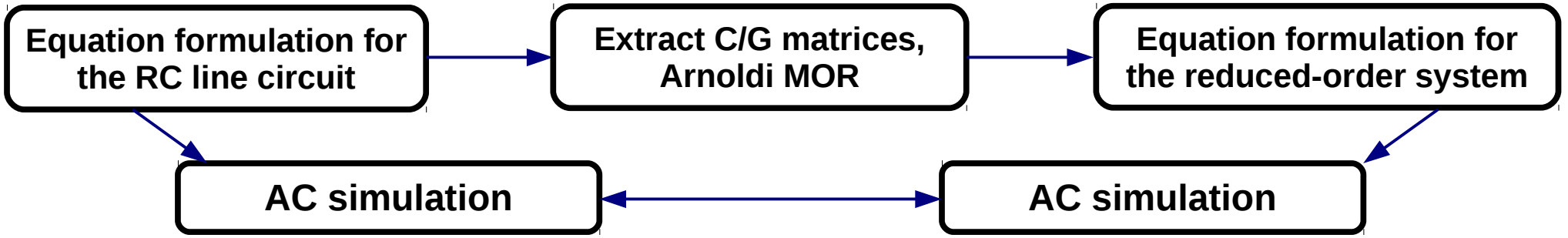


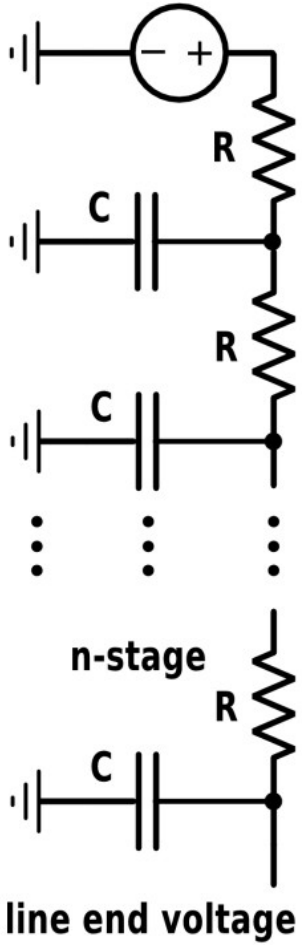
Optical System Modelling/Simulation Example

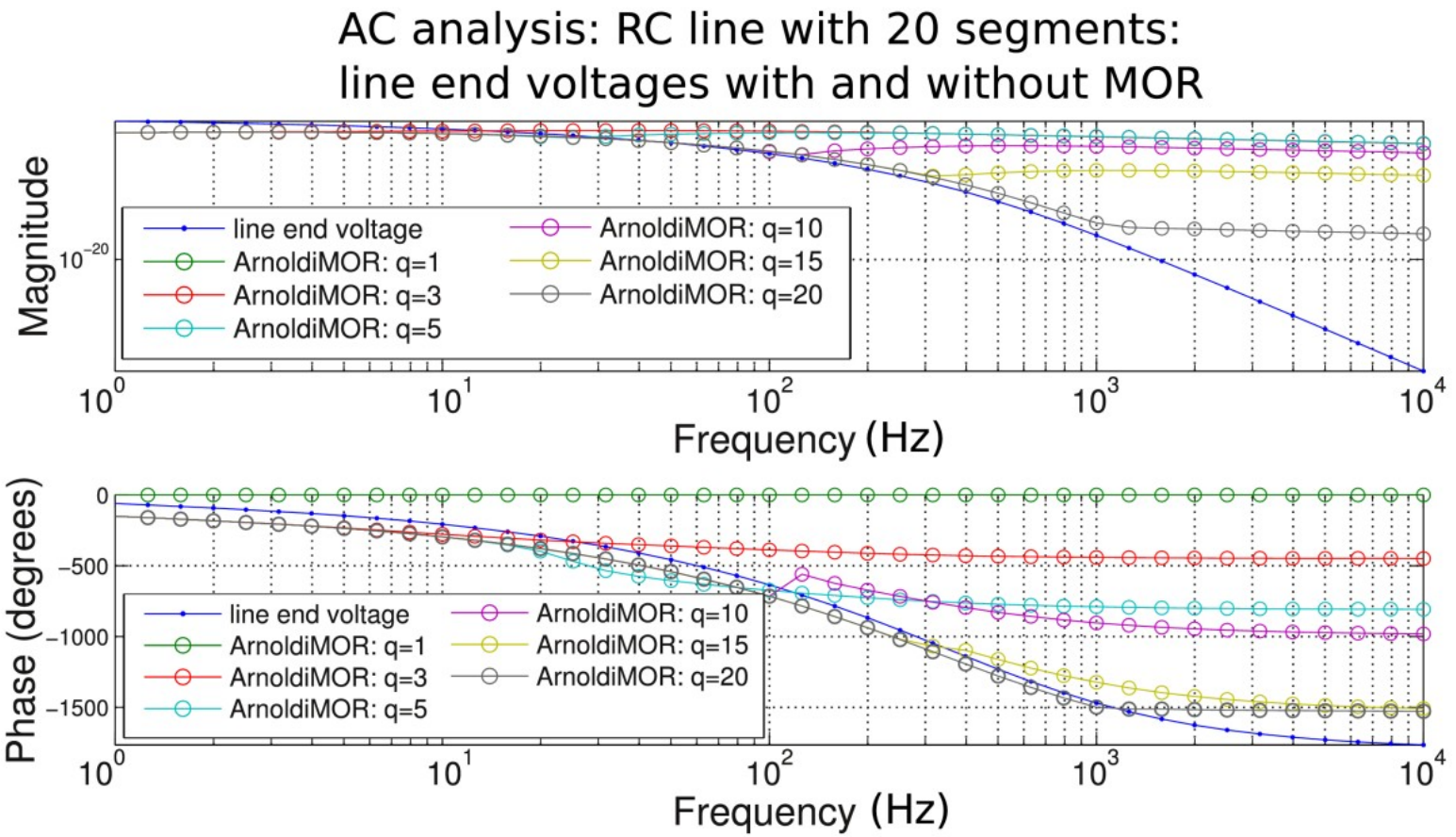




LTI MOR Example in MAPP







Phase-based Reduced-order Model in MAPP

Model osc. in MAPP

not just electrical ones

Simulate PSS: shooting or HB

Extract PPVs

Gen-Adler Analysis

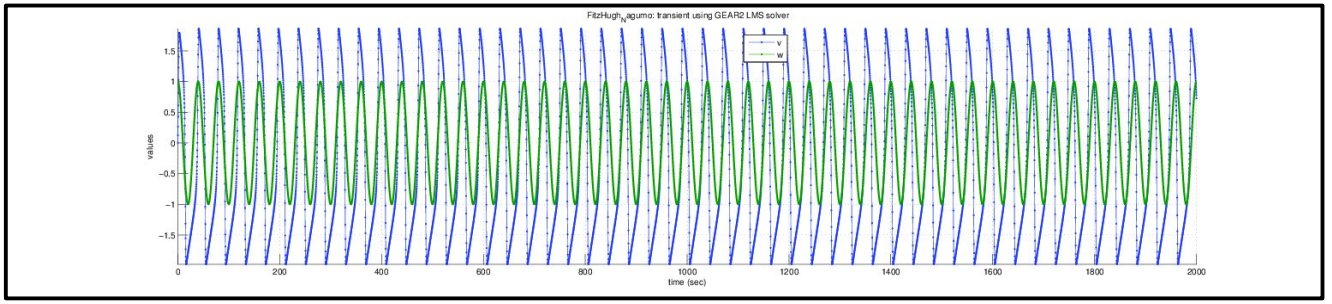
$$\frac{d}{dt}\Delta\phi(t) = f_0 - f_1 + f_0 \cdot g(\Delta\phi(t))$$

DC — locking range

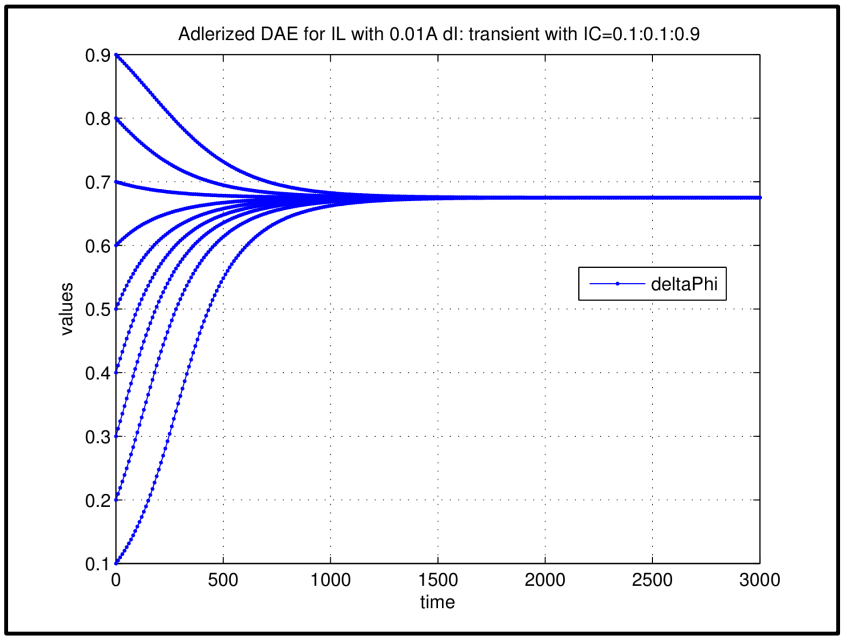
TRAN — phase dynamics

Observe Injection Locking properties

Standard TRAN simulation:



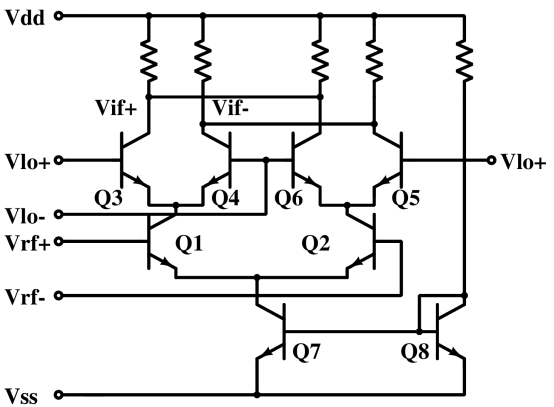
Phase-based TRAN:



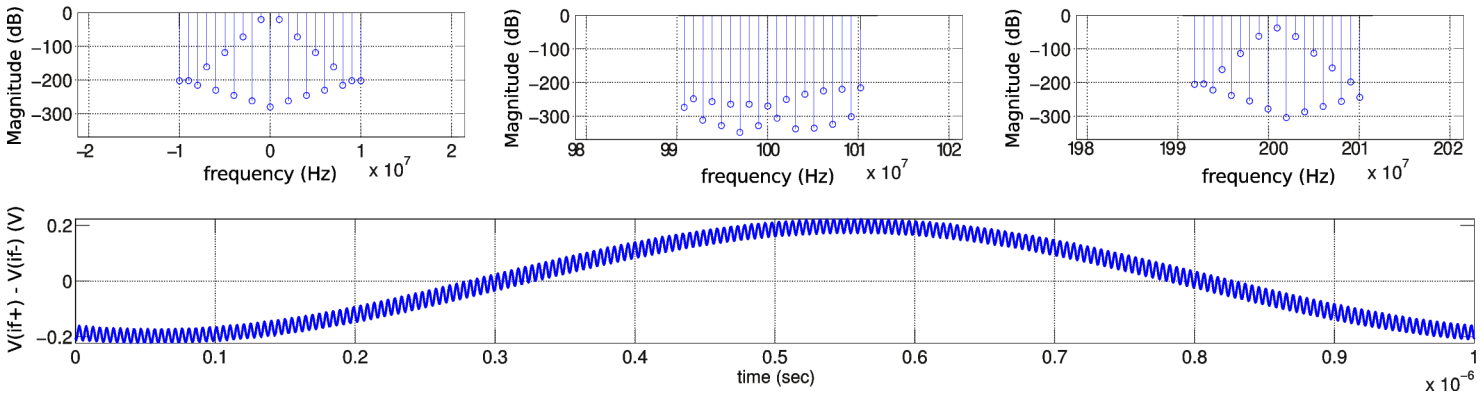
$\Delta\phi(t)$ captures phase response nicely

details: Bhansali/Roychowdhury, "Gen-Adler: the Generalized Adler's equation for injection locking analysis in oscillators". Proc. ASPDAC, 2009.

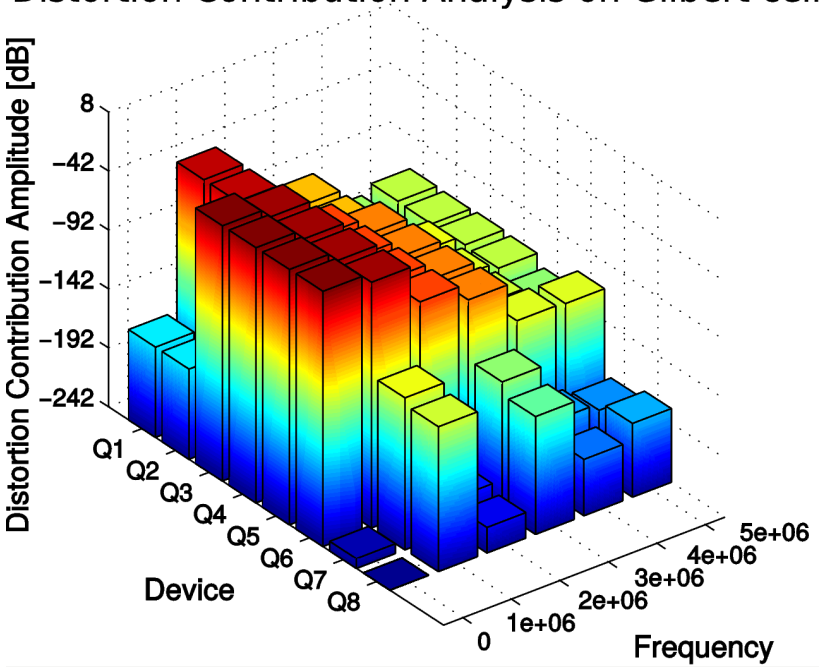
Simulation Algorithms in MAPP: More Examples



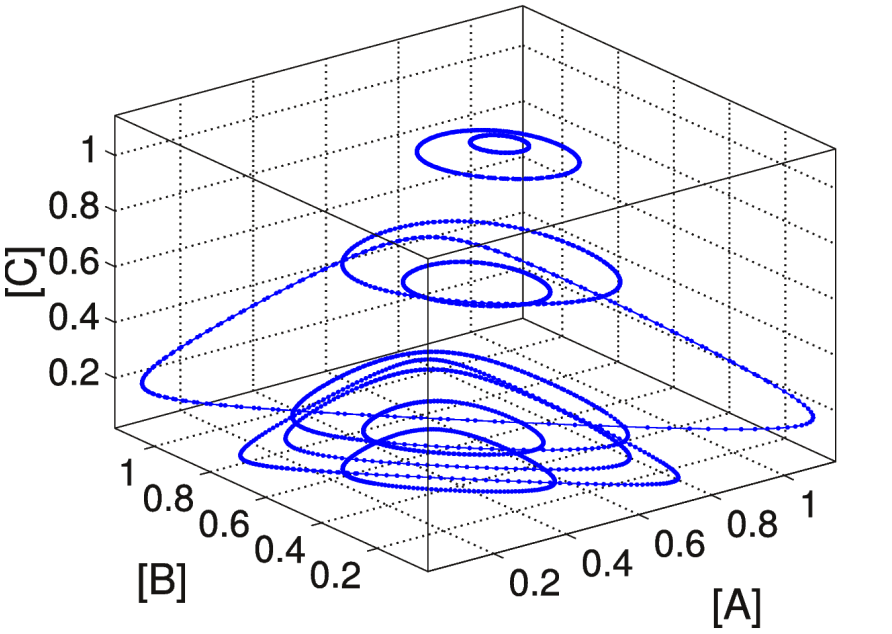
HB on MNA DAE for Gilbert cell



Distortion Contribution Analysis on Gilbert cell



3D phase plane plot of RRE
for A + B –> 2B; B + C –> 2C; C + A –> 2A



details: Wu/Roychowdhury, “Efficient per-element distortion contribution analysis via Harmonic Balance adjoints”. Proc. CICC 2014.

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

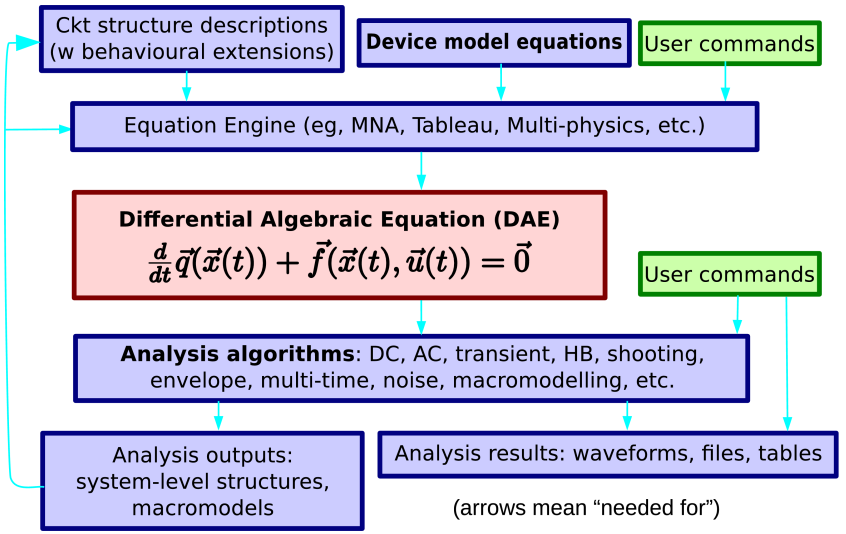
- Works entirely in MATLAB
  - » C++ version to be released
- Help system (start with `help MAPP`)
  - » quick start walk-through
- Automatic differentiation (vecvalder)
  - » `help MAPPautodiff`
- Executable device specification (ModSpec)
  - » examples, tutorial: part of help
- DC, AC, transient analyses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

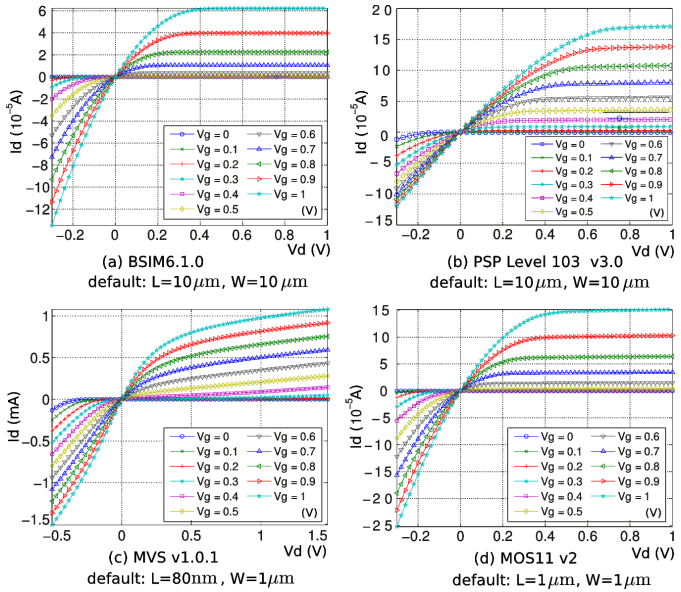
MAPP: Intended Uses

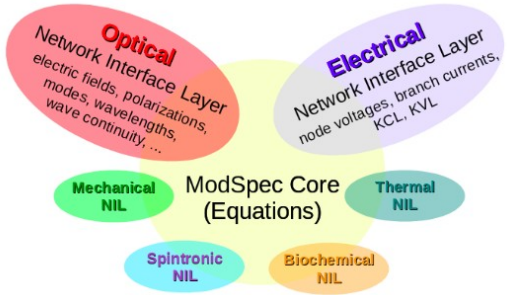
- Developing simulation-ready device models
  - » including multi-physics devices, network connectivity
- Quickly prototyping new simulation algorithms
  - » hours/days to implement a new analysis
    - – assess strengths/limitations before investing resources to implement in “real simulators”
- Learning or teaching modelling/simulation
  - » MATLAB → broadly accessible
  - » help system, tutorials, supporting resources

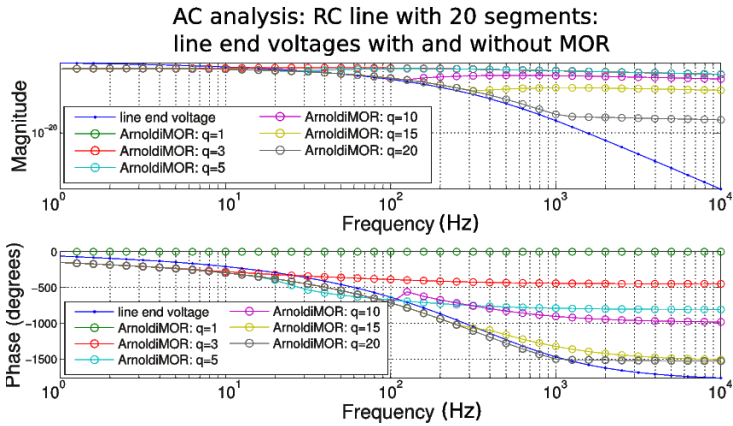
Summary

```
#ifdef SENSDEBUG
  printf("vd = %.7e \n",vd);
#endif /* SENSDEBUG */
  goto next1;
}
if(ckt->CKTmode & MODEINITSMSIG) {
  vd= *(ckt->CKTstate0 + here->DIOvoltage);
} else if (ckt->CKTmode & MODEINITTRAN) {
  vd= *(ckt->CKTstate1 + here->DIOvoltage);
} else if ( (ckt->CKTmode & MODEINITJCT) &&
  (ckt->CKTmode & MODETRANOP)
        && (ckt->CKTmode & MODEUIC) ) {
                vd=here->DIOinitCond;
} else if ( (ckt->CKTmode & MODEINITJCT) && here->DIOoff)
{
        vd=0;
} else if ( ckt->CKTmode & MODEINITJCT) {
        vd=here->DIOtVcrit;
} else if ( ckt->CKTmode & MODEINITFIX && here->DIOoff) {
        vd=0;
} else {
#ifndef PREDICTOR
        if (ckt->CKTmode & MODEINITPRED) {
```









http://MAPP.eecs.berkeley.edu