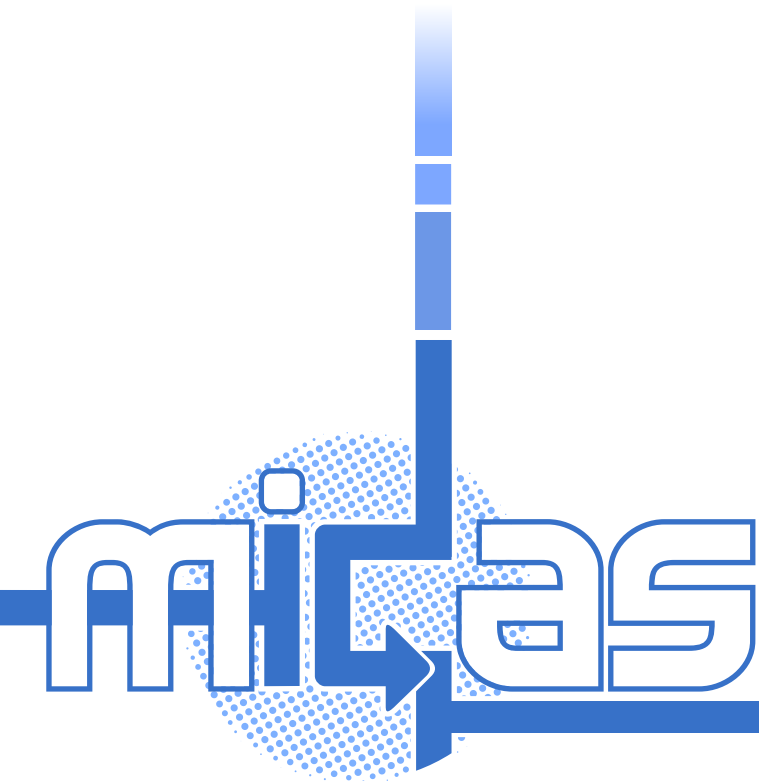

Virtual Design of Integrated Circuits

Dynamical systems,
model-order reduction and
design optimization

Dimitri De Jonghe

Georges Gielen



KATHOLIEKE UNIVERSITEIT
LEUVEN

Outline

- Virtual Design Environments
 - ❖ Application Domains
 - ❖ Modeling and Simulation of ICs
- Simulation of Large Systems
 - ❖ Model-order Reduction
 - ❖ Practical Applications
- Optimization of Complex Structures

a

VIRTUAL DESIGN ENVIRONMENTS

Computer Aided Design

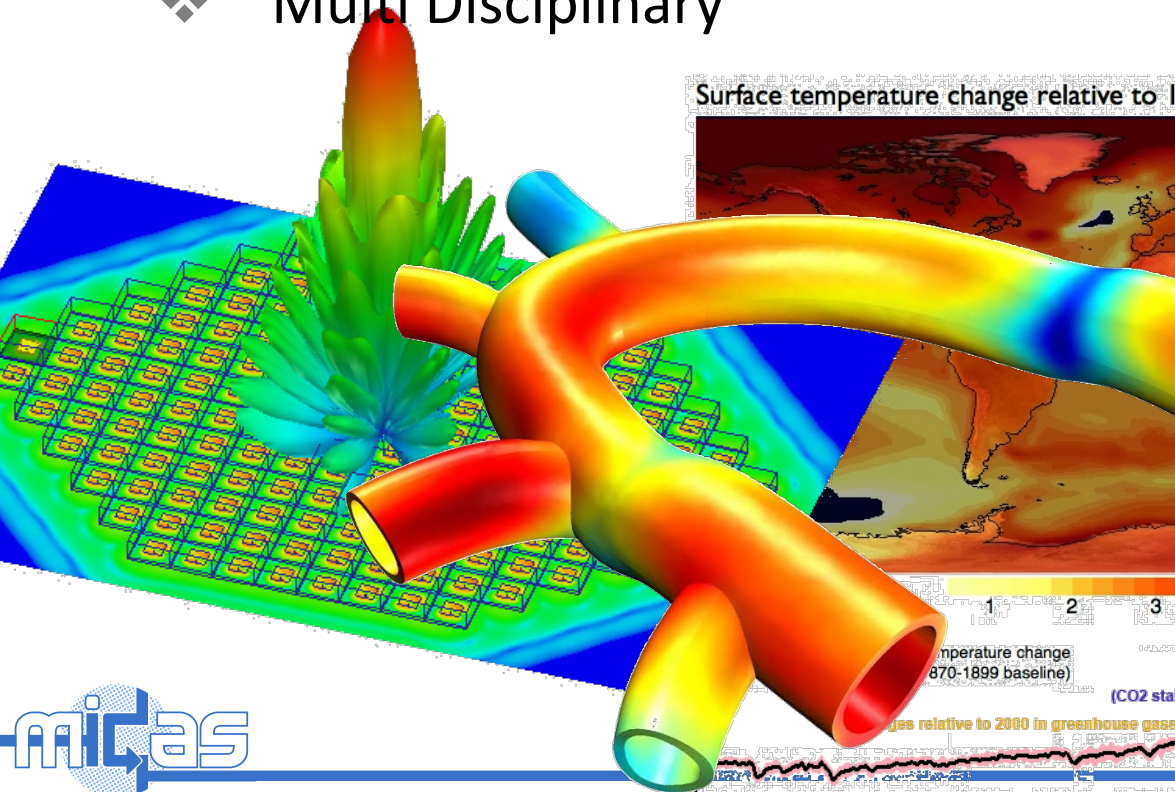
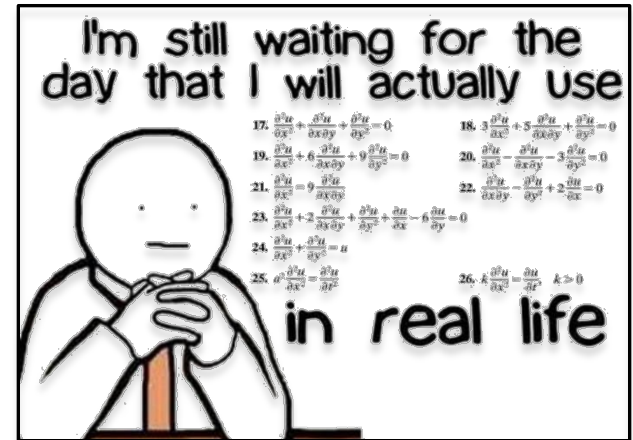
- Modeling & Simulation

- ❖ Applied mathematics

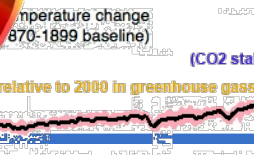
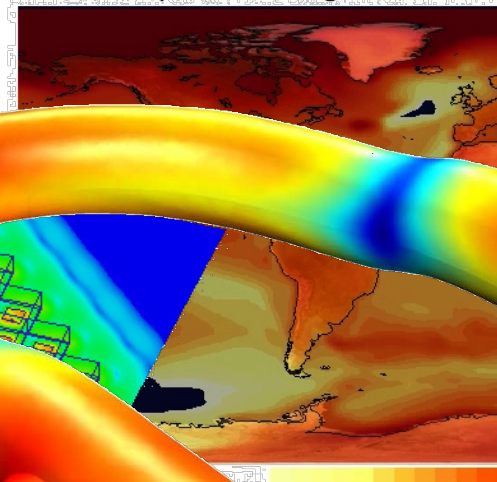
'50s – '90s: *Systems Theory, Finite Elements*

> '90: + *Machine Learning, + AI*

- ❖ Multi Disciplinary

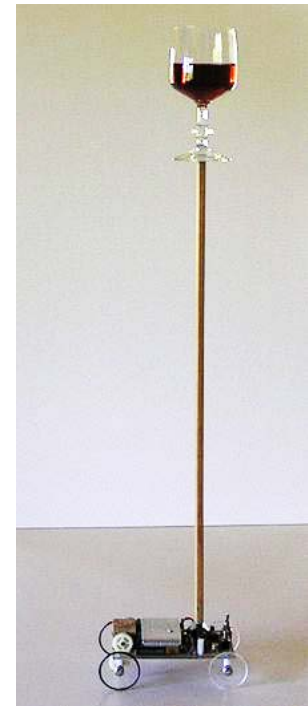
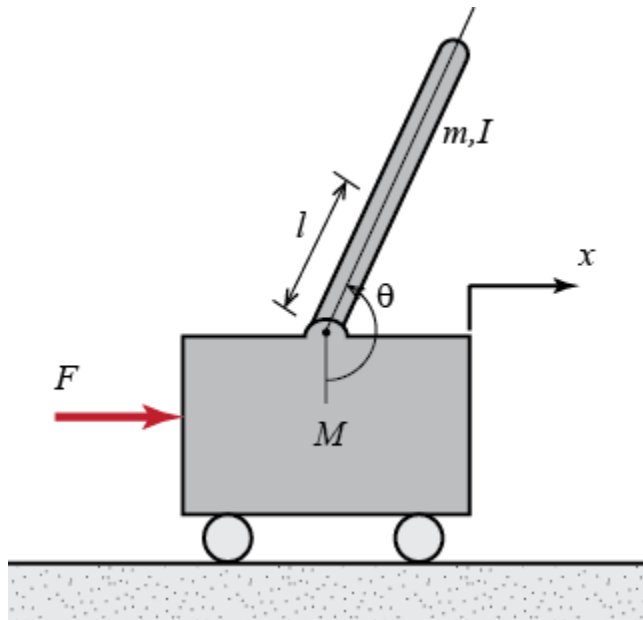


Surface temperature change relative to 1870-1899 baseline **CCSM3 IPCC AR4**



Some Examples

- Inverted pendulum



Inverted Pendulum

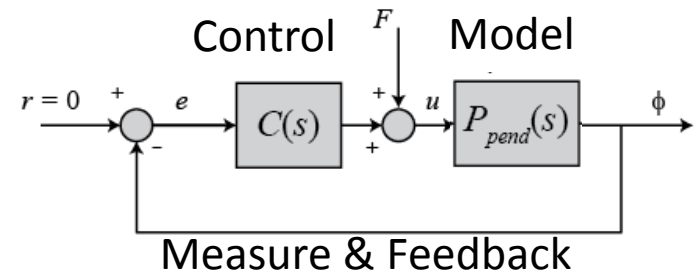
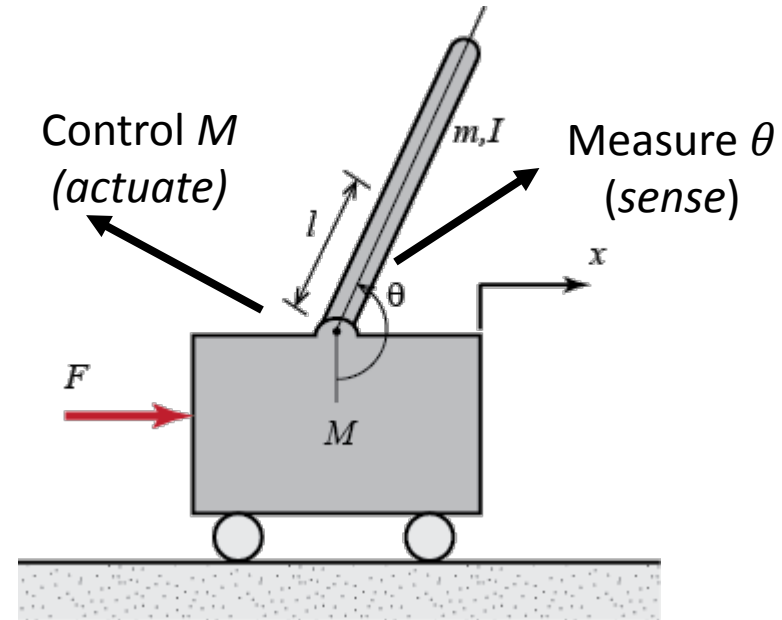


Inverted Pendulum

- How to control θ ?
Sense & actuate
– Measure θ and control M

1. **Model** dynamic behavior
2. **Design/model control**
3. **Simulate**

Virtual Design Environment



Inverted Pendulum: Control

1. Model dynamic behavior

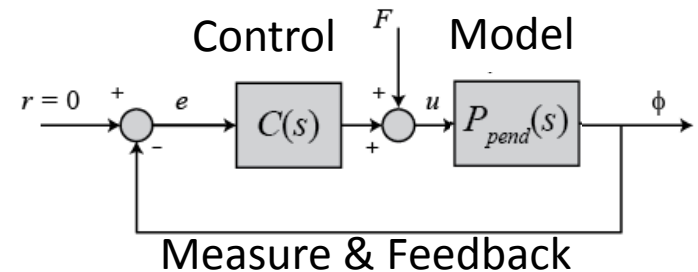
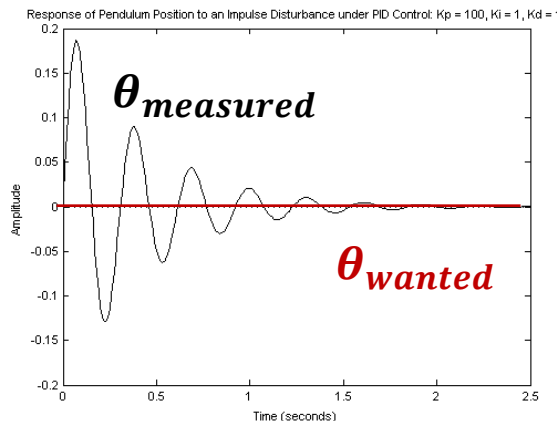
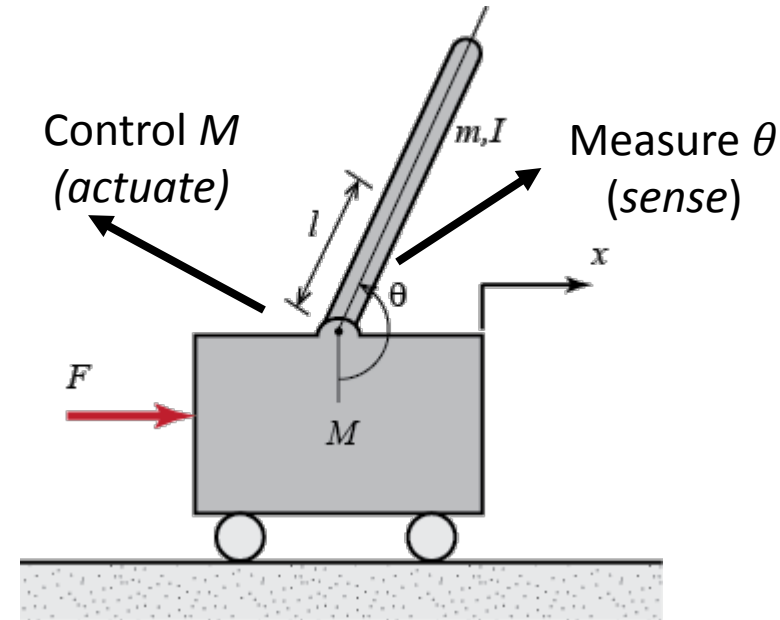
e.g. Equations of motion

$$(M + m) \ddot{x} - ml\ddot{\theta} \cos\theta + ml\dot{\theta}^2 \sin\theta = F$$
$$l\ddot{\theta} - g \sin\theta = \ddot{x} \cos\theta$$

2. Design control

$$\theta_{measured} \approx \theta_{wanted} = 0^\circ$$

3. Simulate (and iterate)



How would Google do this?

■ Driverless car

Autonomous Driving

Google's modified Toyota Prius uses an array of sensors to navigate public roads without a human driver. Other components, not shown, include a GPS receiver and an inertial motion sensor.

LIDAR

A rotating sensor on the roof scans more than 200 feet in all directions to generate a precise three-dimensional map of the car's surroundings.

VIDEO CAMERA

A camera mounted near the rear-view mirror detects traffic lights and helps the car's onboard computers recognize moving obstacles like pedestrians and bicyclists.



RADAR

Four standard automotive radar sensors, three in front and one in the rear, help determine the positions of distant objects.

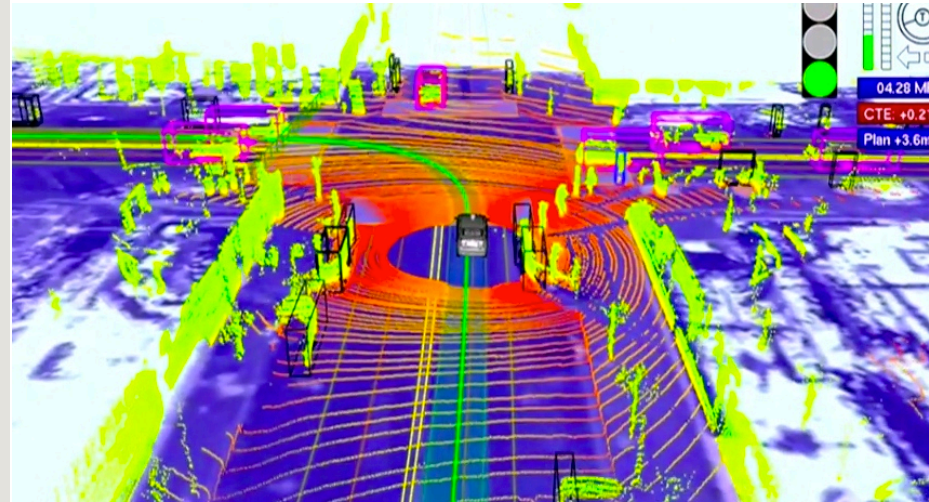
Sense

POSITION ESTIMATOR

A sensor mounted on the left rear wheel measures small movements made by the car and helps to accurately locate its position on the map.

Actuate

Real-time virtual model



Source: Google

THE NEW YORK TIMES, PHOTOS: GREGG DEGUZMAN FOR THE NEW YORK TIMES

How would do this?

ON Semiconductor enables energy efficient automotive solutions that reduce emissions, improve fuel economy, and enhance lighting, safety, connectivity, and infotainment power delivery systems. The company provides a broad array of power management, protection, processing, signal conditioning and control products that deliver solutions focused on powertrain, dynamic braking, lighting, climate control, door zone, collision warning, IVN, and infotainment applications.

Body & Convenience

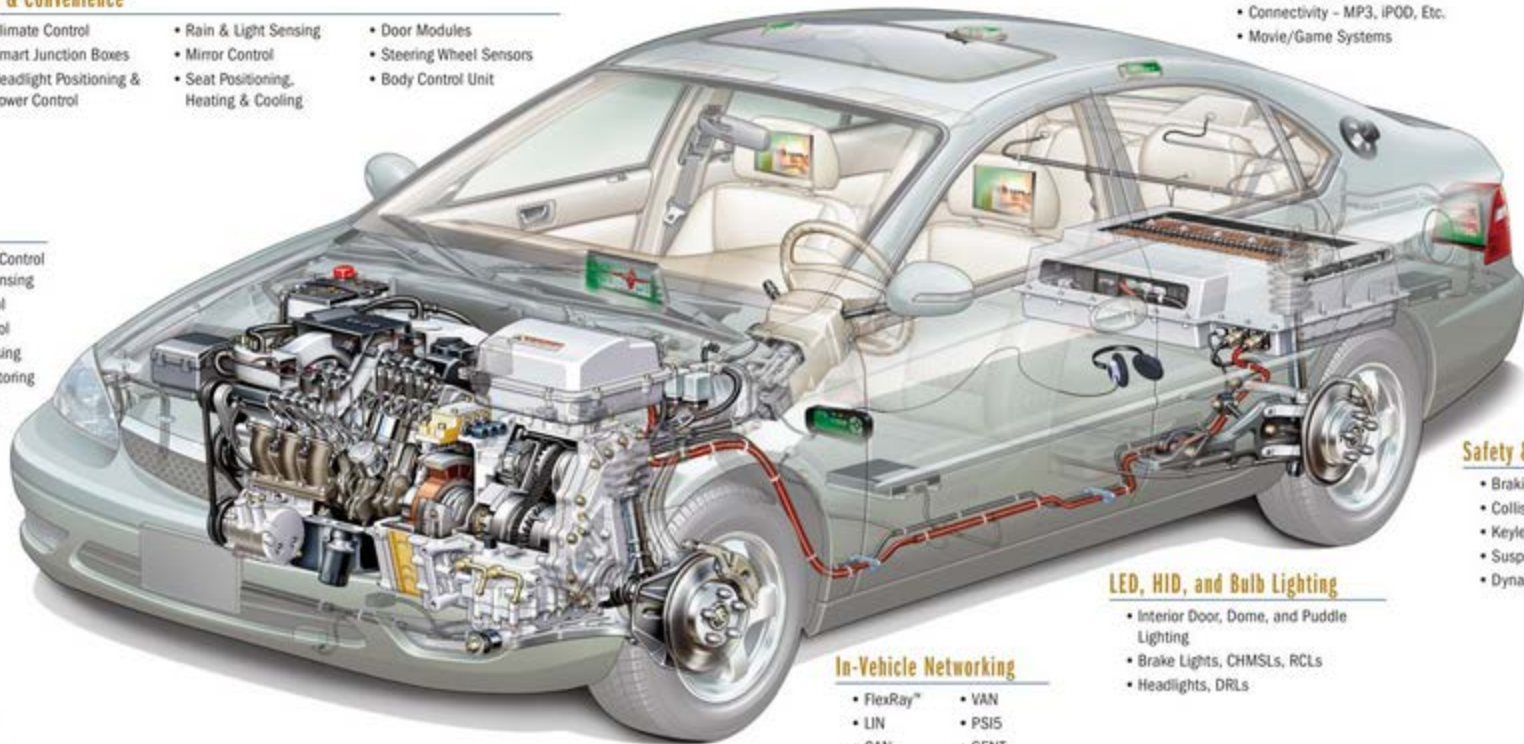
- Climate Control
- Smart Junction Boxes
- Headlight Positioning & Power Control
- Rain & Light Sensing
- Mirror Control
- Seat Positioning, Heating & Cooling
- Door Modules
- Steering Wheel Sensors
- Body Control Unit

Powertrain

- Transmission Control & Position Sensing
- Engine Control
- Throttle Control
- Oil Level Sensing
- Air Flow Monitoring
- Valve Control
- Fuel Injection Control

Audio & Infotainment

- Instrument Clusters
- GPS/Navigation Systems
- Satellite/Digital Radio
- Connectivity – MP3, IPOD, Etc.
- Movie/Game Systems



Safety & Chassis

- Braking/Traction/Stability
- Collision Avoidance
- Keyless Entry
- Suspension & Steering
- Dynamic Braking

LED, HID, and Bulb Lighting

- Interior Door, Dome, and Puddle Lighting
- Brake Lights, CHMSLs, RCLs
- Headlights, DRLs

In-Vehicle Networking

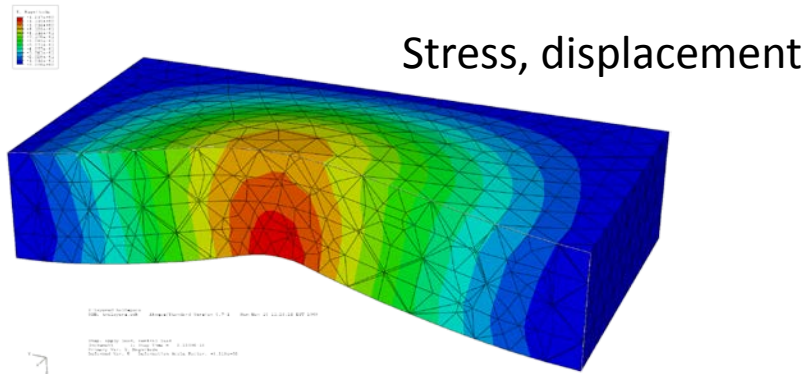
- FlexRay™
- LIN
- CAN
- VAN
- PSIS
- SENT



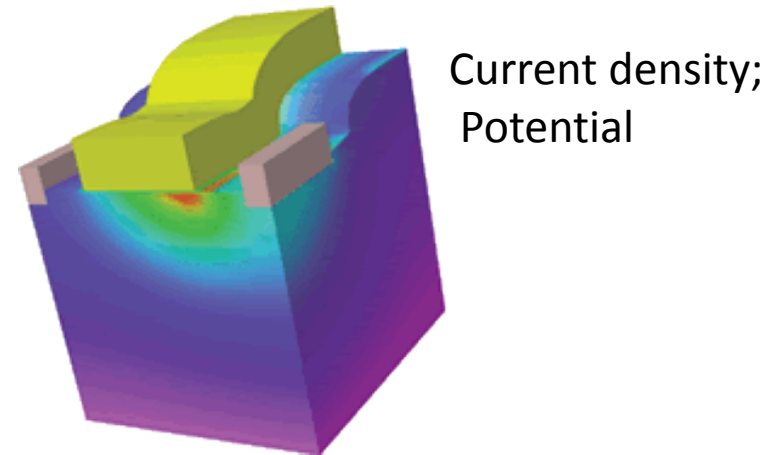
Plug in physical equations

- Finite Element Modeling (FEM)

Mechanical



Electrical



Mass, velocity (position), damping, force

$$[M]d\vec{v} + [K]\vec{v} = \vec{F}$$

Capacitance, voltage, conductance, current

$$[C]d\vec{v} + [G]\vec{v} = \vec{i}$$

Large, sparse matrices!

$10^3 - 10^6$ rows/cols

Dynamical Systems

Linear systems

$$[C]d\vec{v}(t) + [G]\vec{v}(t) = \vec{i}(t)$$

→ $[C]$, $[G]$ are constant matrices

→ No Memory: $[C] = 0$; $[G]\vec{v}(t) = \vec{i}(t)$

→ Passives: **capacitors**



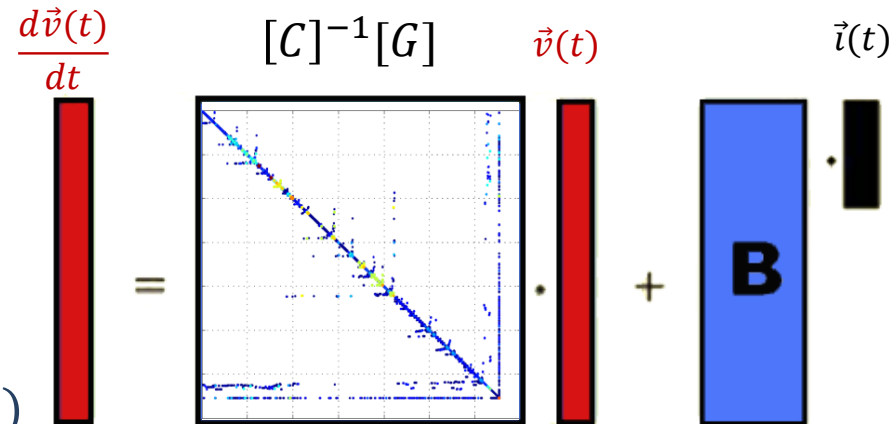
resistors



inductors



$$[L] = [C], [G]$$

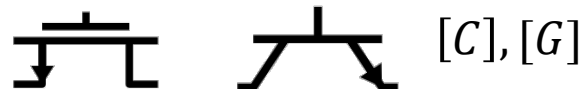


Nonlinear systems

$$[C(\vec{v}(t))]d\vec{v}(t) + [G(\vec{v}(t))] = \vec{i}(t)$$

→ $[C]$, $[G]$ are matrix functions of $\vec{v}(t)$ → *linearise!*

→ Actives: **transistors** (switches, logic AND, OR, amplifiers, ...)



Solve Dynamical Systems

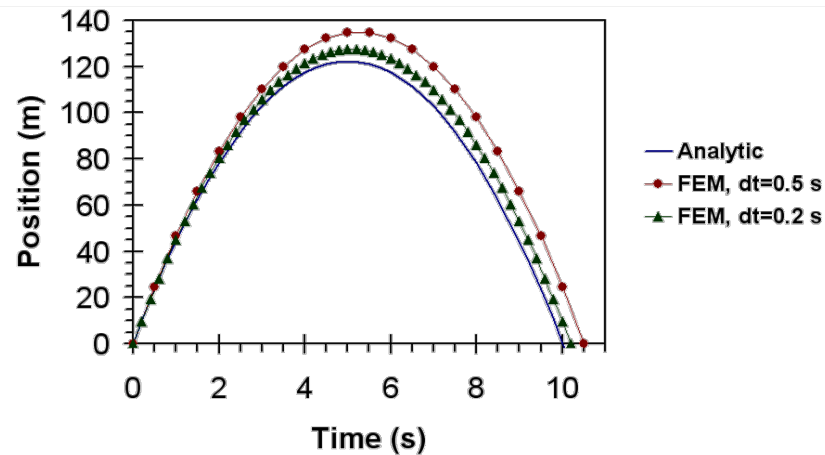
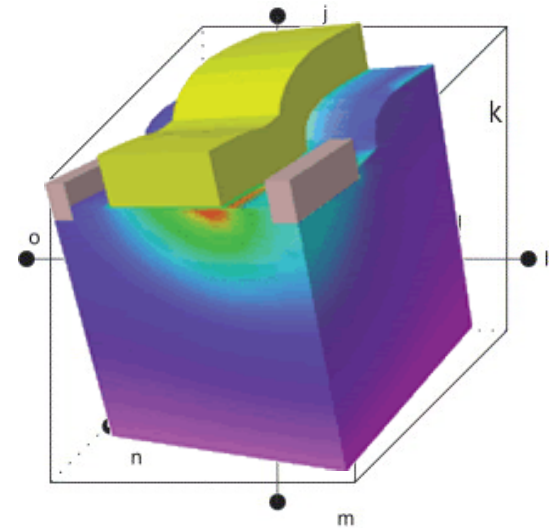
- Next-state equations

$$[C] \frac{d\vec{v}(t)}{dt} + [G]\vec{v}(t) = B\vec{i}(t)$$

- Solve for discrete time

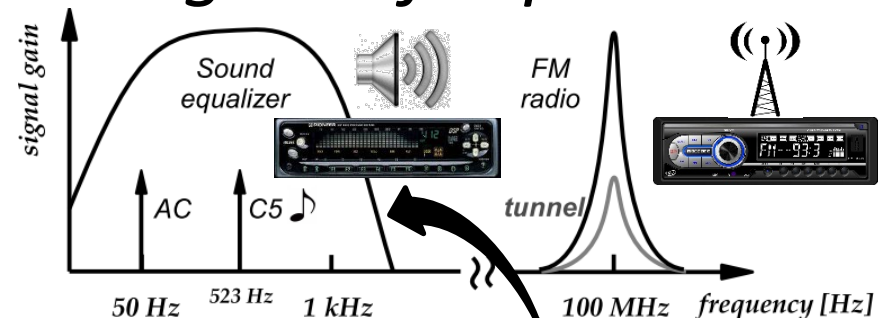
$$\frac{d\vec{v}(t)}{dt} \rightarrow \frac{\vec{v}(t_2) - \vec{v}(t_1)}{t_2 - t_1} \rightarrow \frac{\Delta\vec{v}}{\Delta t}$$

- Adapt time step if variations ($\Delta\vec{v}$) are large/small



Frequency Domain Analysis

- How does a system respond to *vibrations*?
 - Mechanical: Pressure, acoustic signals
 - Electromagnetic: AM, FM, WIFI, GSM, digital, ...
- A dynamical system *filters signal's frequencies*
 - Audio equalizer
 - Radio tuner, ...



- *Frequency (s) domain*

$$t \rightarrow s \quad \frac{d\vec{v}(t)}{dt} \rightarrow sV(s)$$

$$\frac{I(s)}{V(s)} = H(s)$$

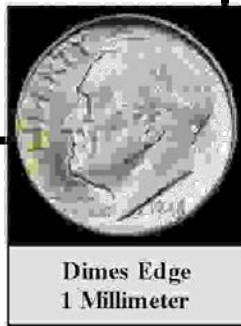
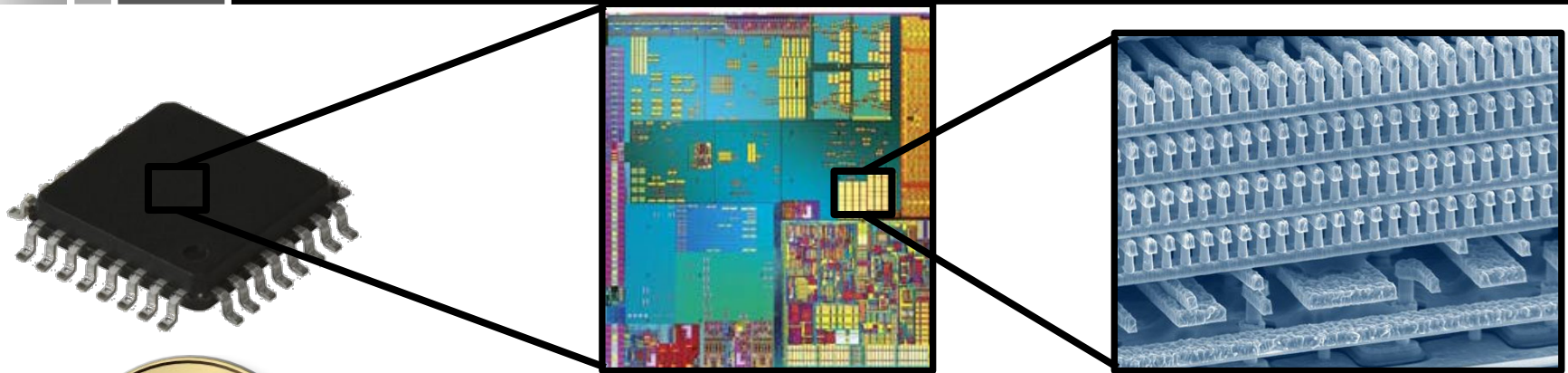
$$[C] \frac{d\vec{v}(t)}{dt} + [G]\vec{v}(t) = \vec{i}(t) \longrightarrow [C]sV(s) + [G]V(s) = I(s)$$

tough differential equation \longrightarrow easy linear equation (+, -, x, /)

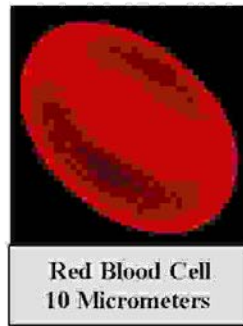
Our Focus

- Generation of **models for ICs**
 - To **verify** complex systems without producing them everytime
 - Complex models need **reduction** techniques
- **Use of existing models** in design cycle
 - Design **optimization & synthesis**

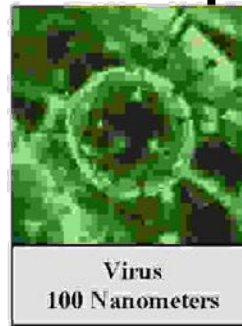
Chips are small but complex systems



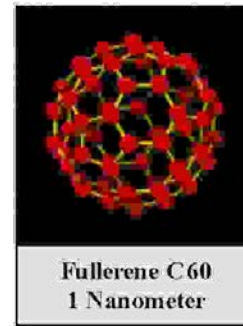
Dimes Edge
1 Millimeter



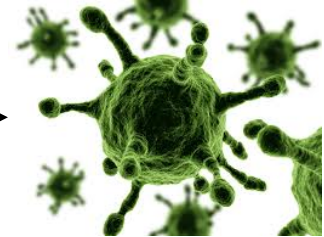
Red Blood Cell
10 Micrometers



Virus
100 Nanometers



Fullerene C60
1 Nanometer



SIZE



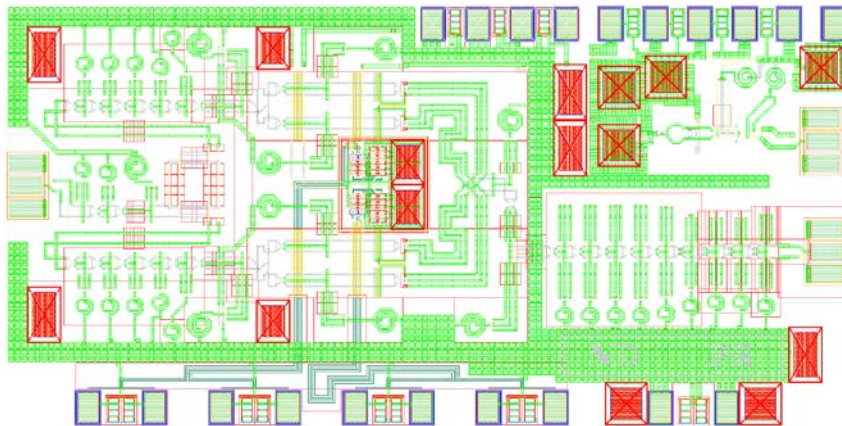
Fabrication of Integrated Circuits



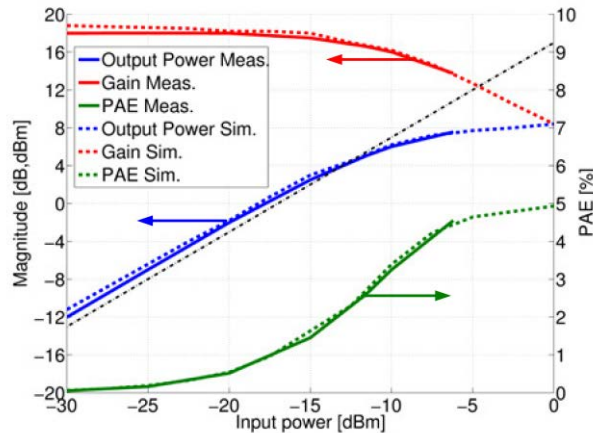
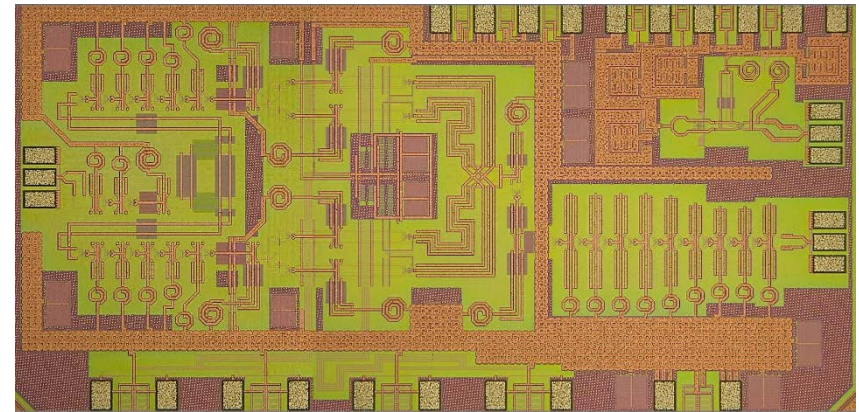
Computer Aided Design of Chips

e.g. 120 GHz Power Amplifier

- CAD model

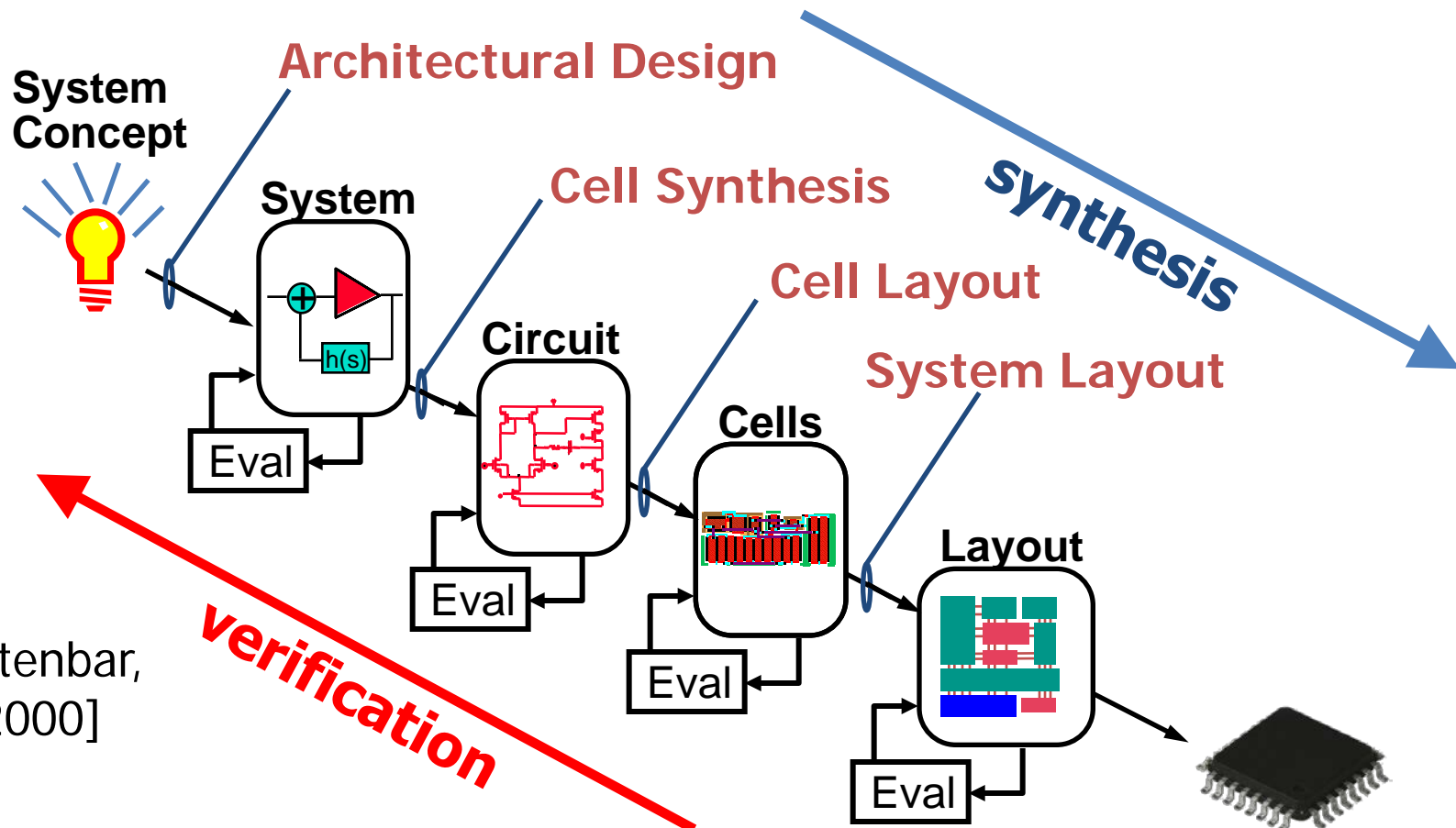


- Actual chip



CAD layers of abstraction

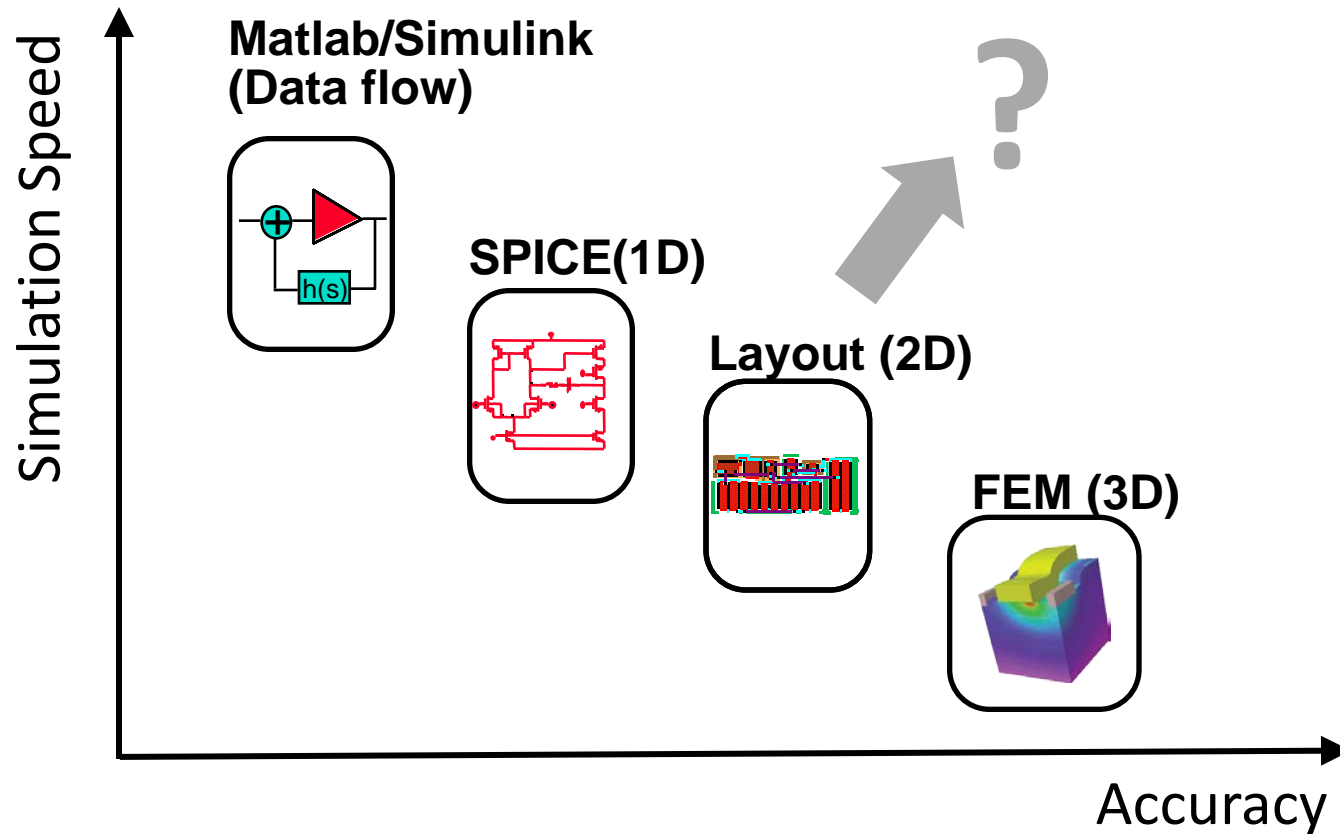
- Top-down design process
- Design iterations + bottom-up verification



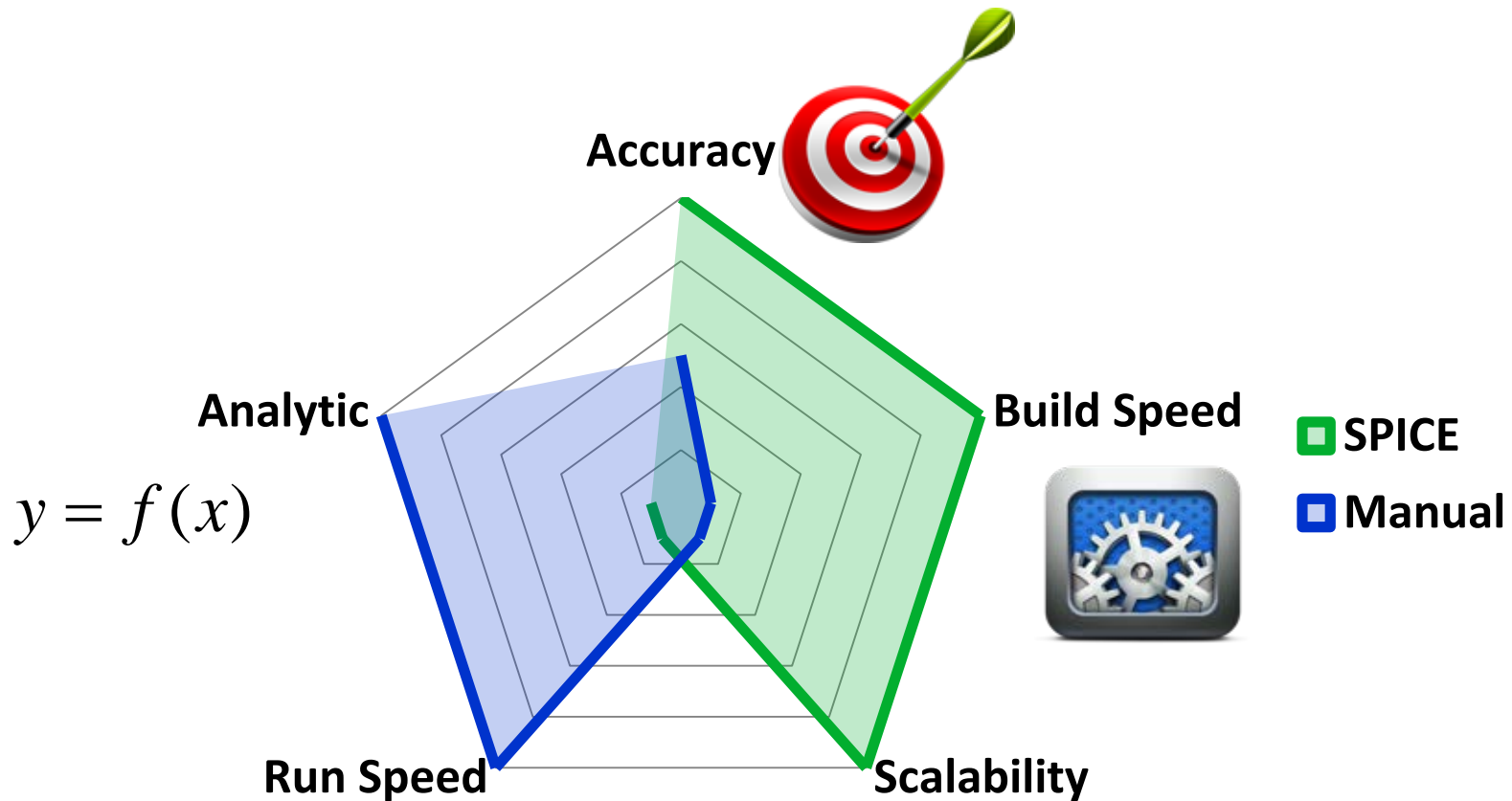
[Gielen & Rutenbar,
Proc. IEEE 2000]

Simulation Trade-off

- Course (fast, rough) vs fine (slow, accurate)



Modeling Trade-off



Computer Aided Design: Remarks

- Full 3D finite element simulation:
 - Very accurate simulations
 - First-time right design (low cost!)

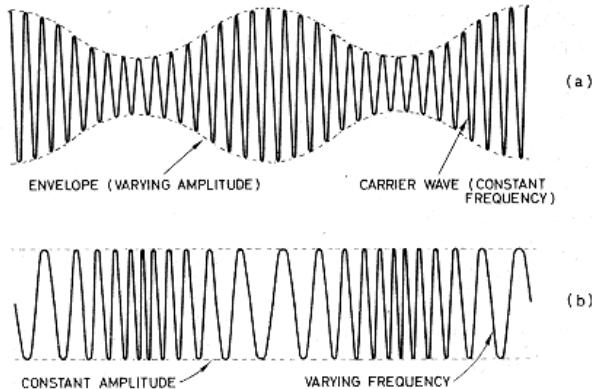
- But...
 - Computationally expensive!
 - ➔ Full system simulations may take weeks!

Virtual design environments for real-world applications

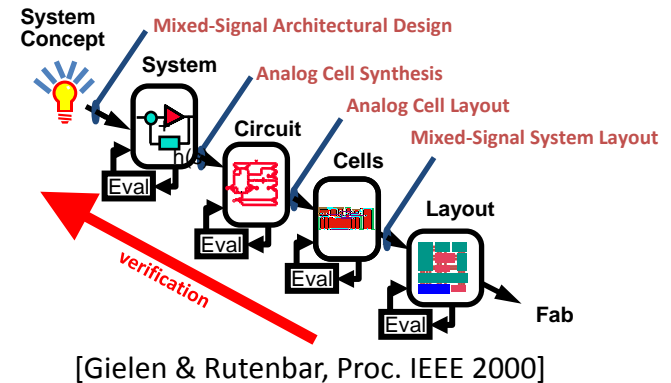
SIMULATION OF LARGE SYSTEMS

Motivation

- **Verification** of systems often **too slow** (> days)
 - depends on size, nonlinearity, time constants
- IC vendors want to **protect and reuse** their IP
- IC designers want **accuracy at all design levels**



$$y = f(x)$$



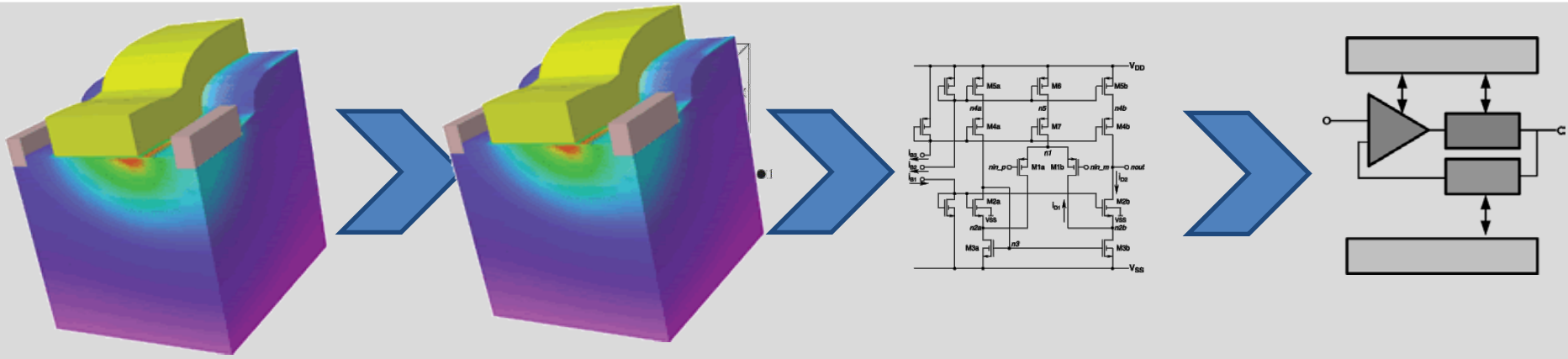
System-level simulation

FEM

macromodel

component

system

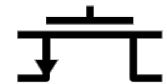
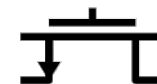


1 X

1 X

100-1000 X

10^3-10^6 X



10^3-10^4 eqns

10 eqns

10^3-10^4 eqns

10^4-10^7 eqns

[C], [G]

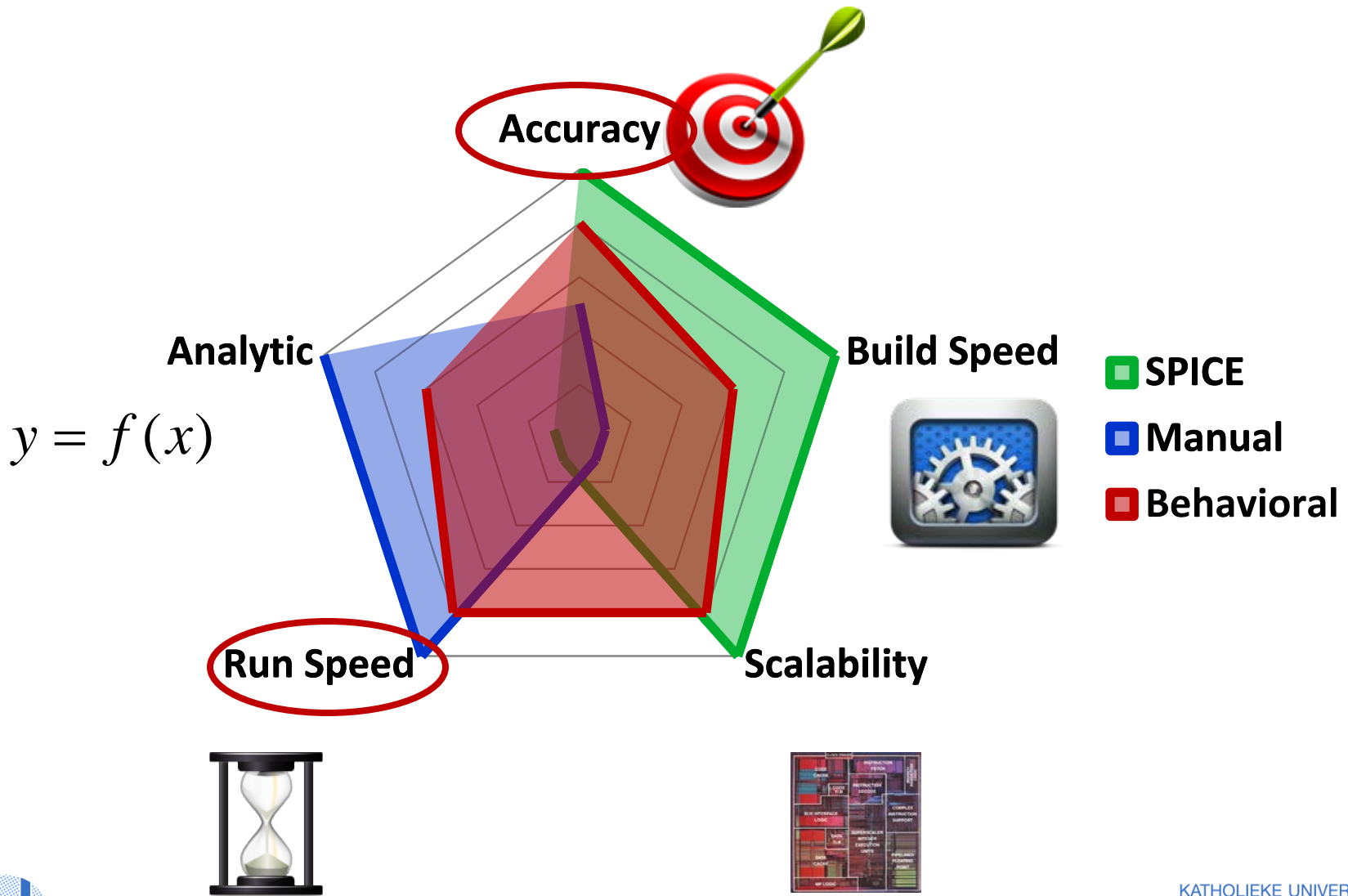
[C], [G]

[C], [G]

[C], [G]

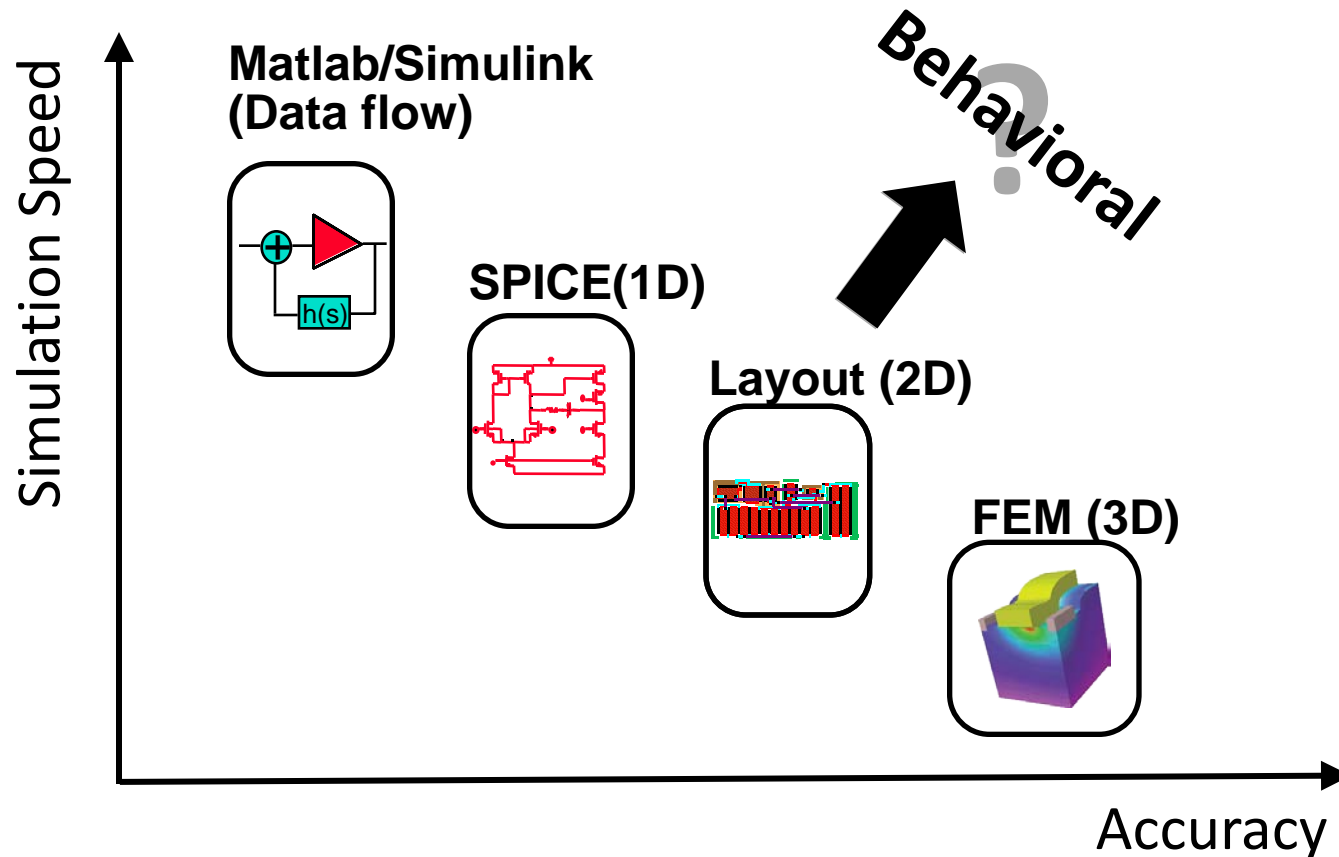
How to deal with system level complexity??

Modeling Trade-off



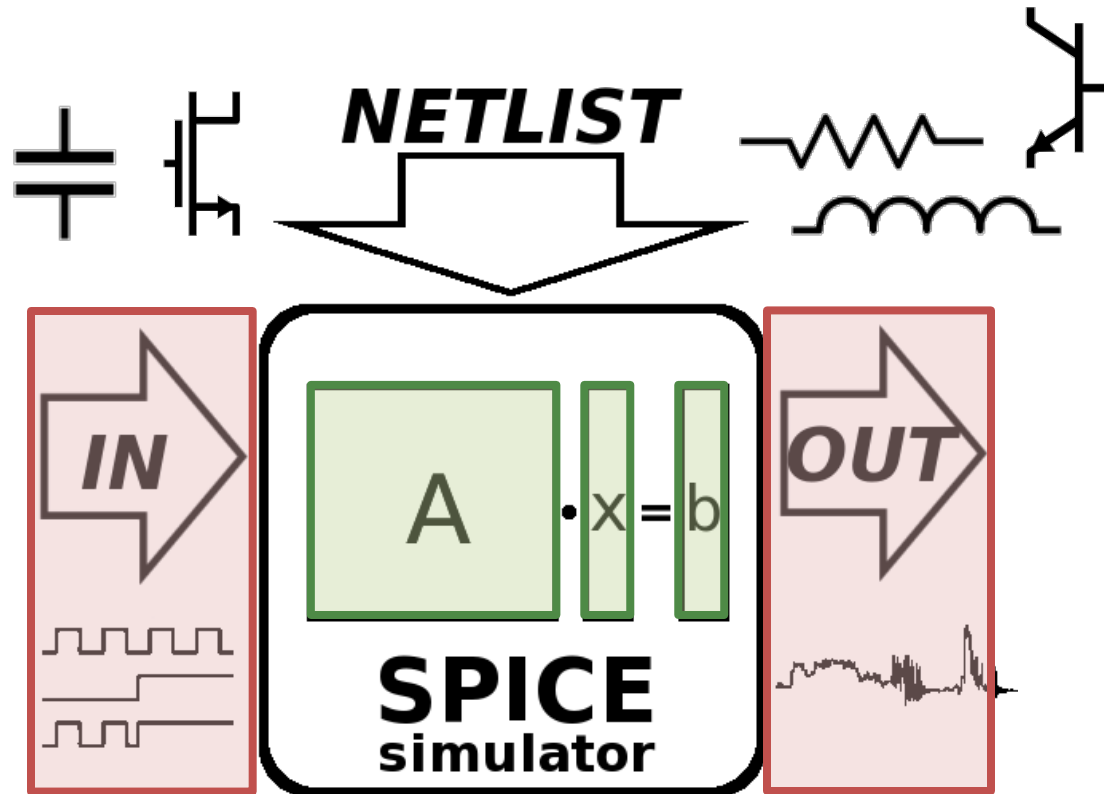
Simulation Trade-off

- Course (fast, rough) vs fine (slow, accurate)



Which behavior?

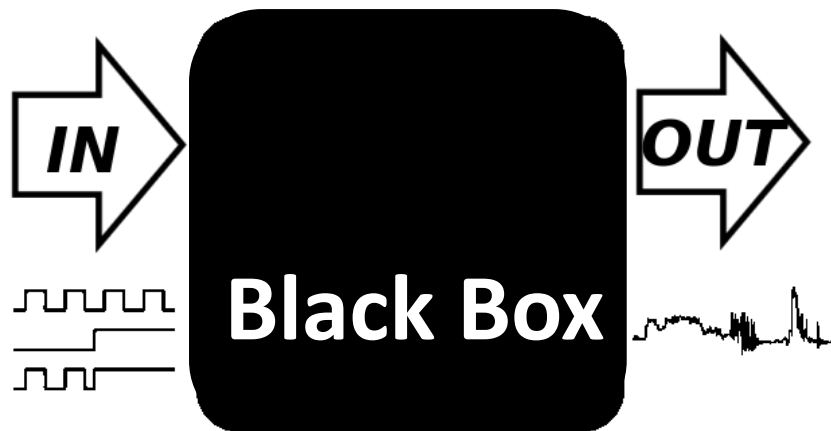
- Input-Output behavior
- Dynamical equations



Which Model?

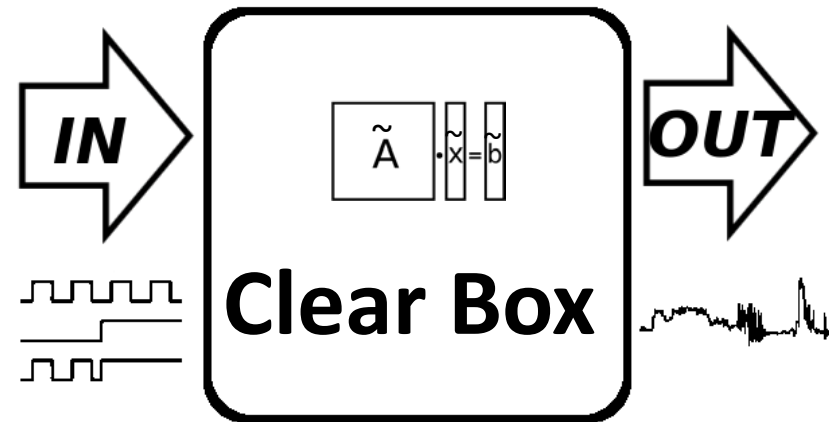
- Black Box

- Only look at terminals
- System **Identification**
- “*Generic*”



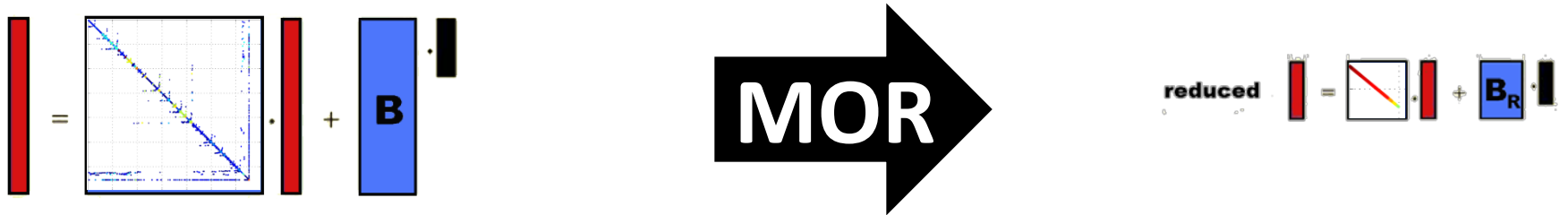
- Clear Box

- Internal representation
- System **Reduction**
- “*Natural*”



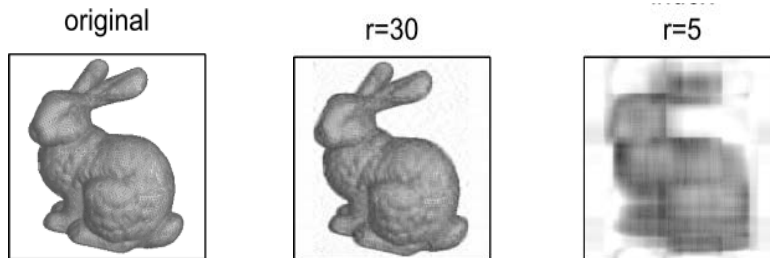
Reduction of *Linear Systems*

- Model Order Reduction (MOR)

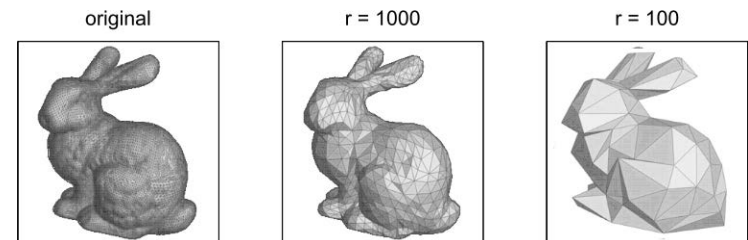


⇒ **Principal Component Analysis**
Eigenvalues, SVD, ... + Truncation

[A] = pixel array → JPEG

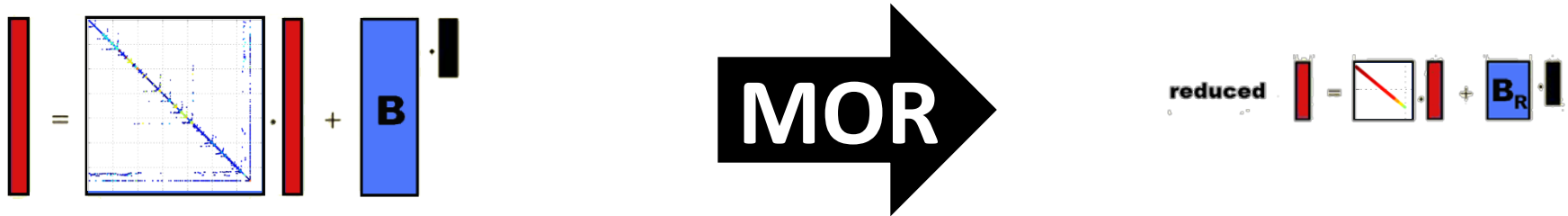


[A] = vertices → reduce mesh



Reduction of *Linear* Systems

■ Model Order Reduction (MOR)



➔ **Principal Component Analysis**
Eigenvalues, SVD, ... + Truncation

[A] = dynamical equations

Flavours

Modal Approximation (*Eigenvalue decomposition*)

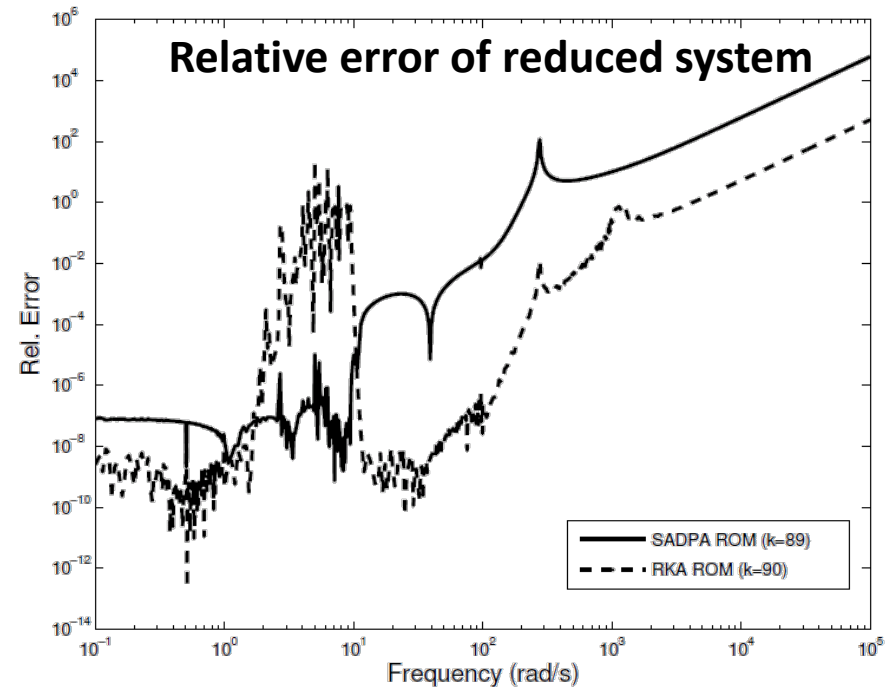
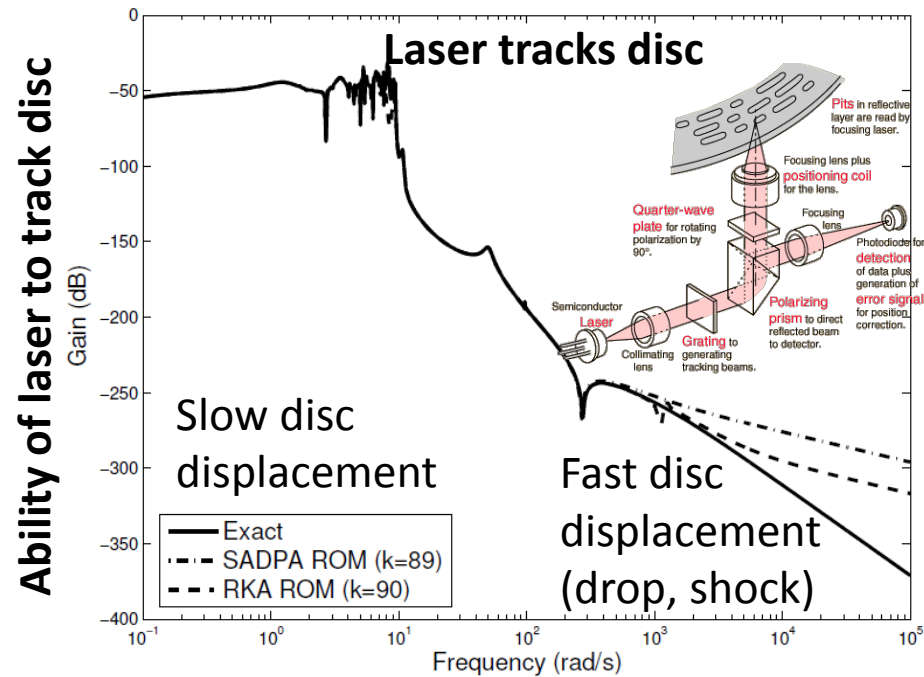
Truncated Moment matching (*Krylov projection*)

Proper Orthogonal Decomposition (*Identification*)

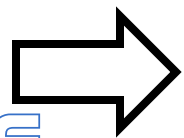
Vector Fitting (*Function approximation*)

Reduction of *Linear* Systems

- CD player (optical control) : 480 \rightarrow 90 eqns






- Used in large (linear) system solvers (FEM, ...)

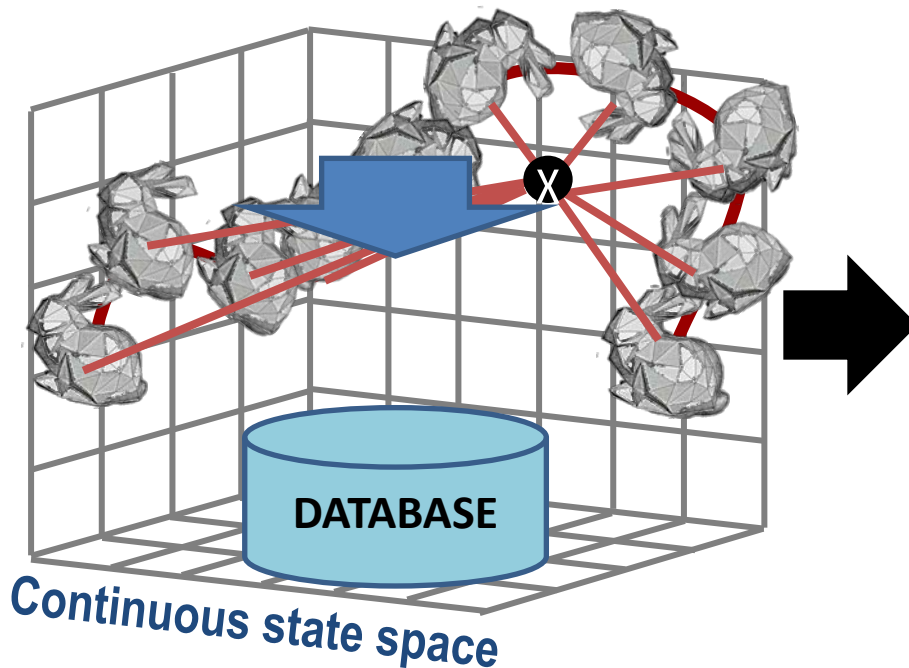


What about *nonlinear* systems?


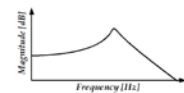
Reduction of *Nonlinear* Systems

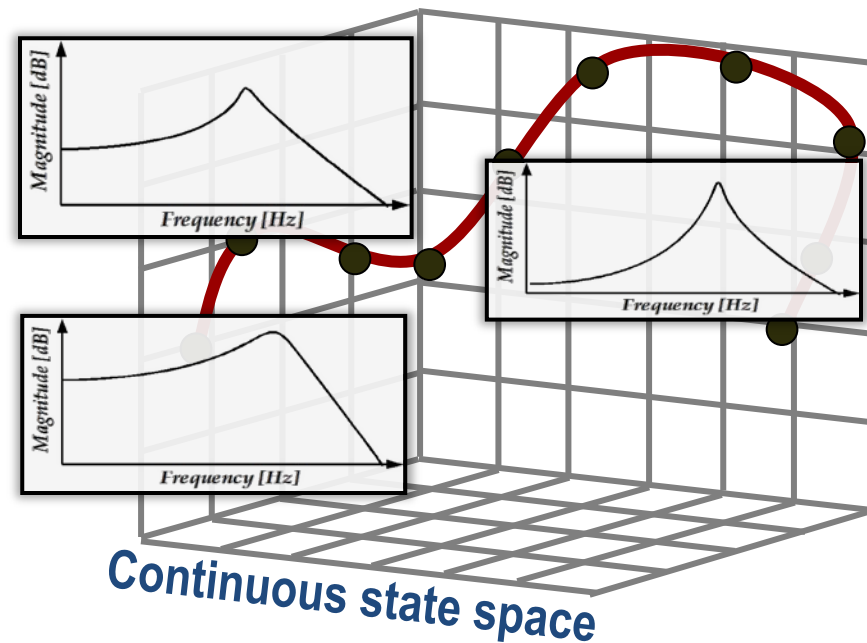
■ Trajectory PieceWise

1. Sample & linearize states 
2. Reduce linearized systems 
3. Interpolate reduced states 






■ Transfer Function Trajectories

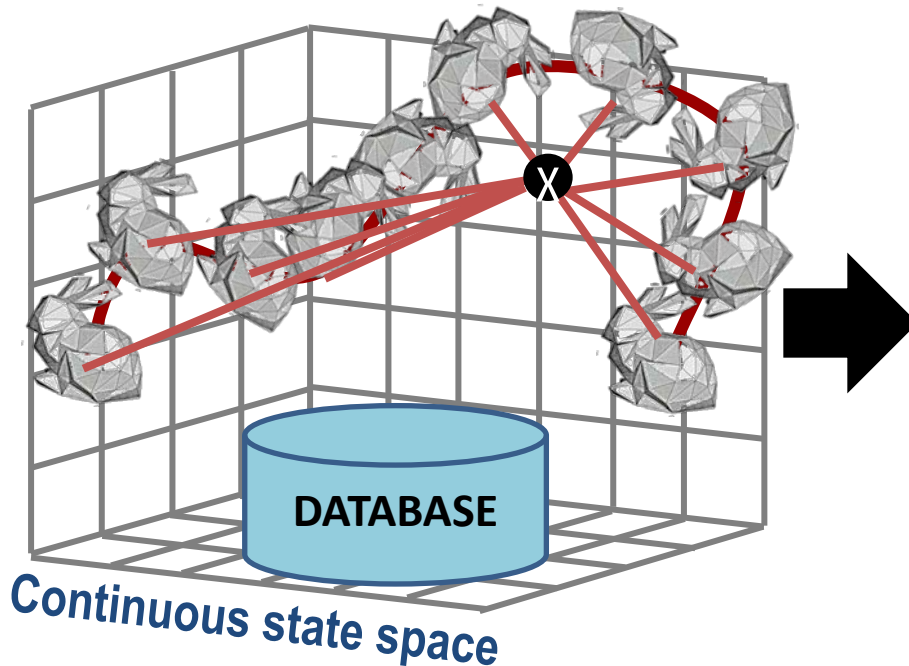
1. Sample & linearize states 
2. Frequency transform states 




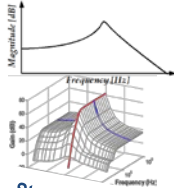

Reduction of *Nonlinear* Systems

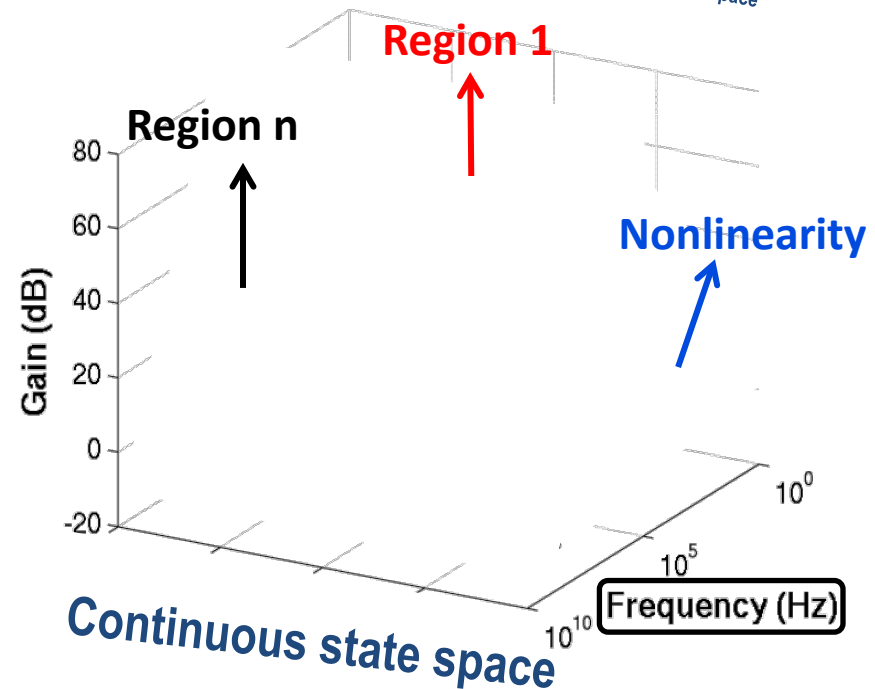
■ Trajectory PieceWise

1. Sample & linearize states 
2. Reduce linearized systems 
3. Interpolate reduced states 

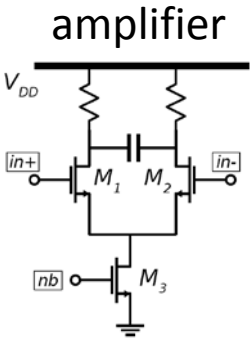


■ Transfer Function Trajectories

1. Sample & linearize states 
2. Frequency transform states 
3. *Fit surface with math* 

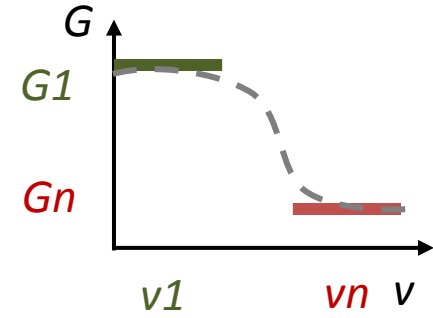


Reduction of *Nonlinear* Systems



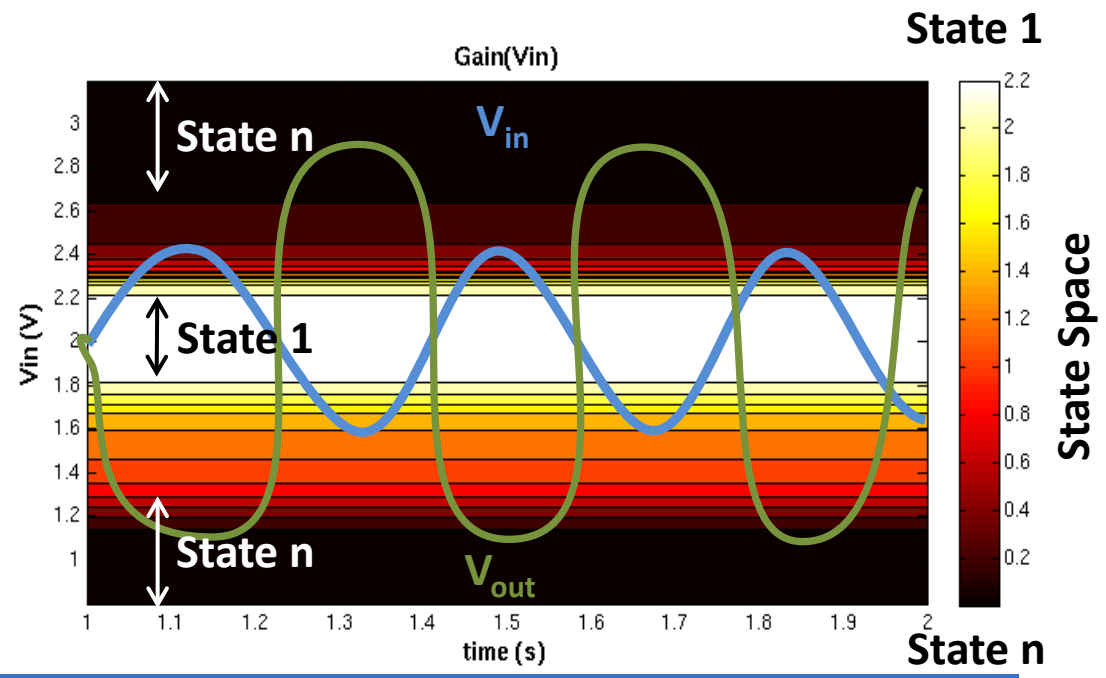
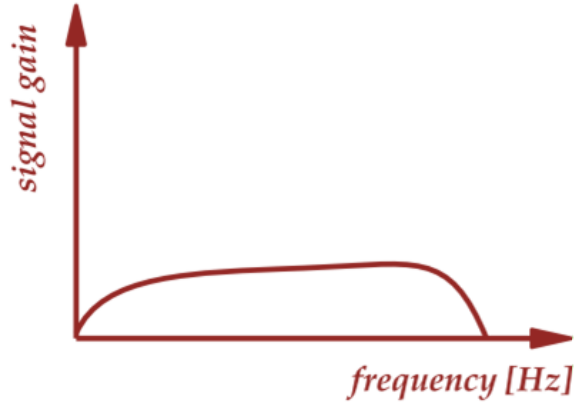
■ $[C(v)]$ and $[G(v)]$ vary with *state* of $v(t)$

- If $(v == v1) \rightarrow$ state 1
- If $(v == vn) \rightarrow$ state n



$$[C(\vec{v}(t))]d\vec{v}(t) + [G(\vec{v}(t))] = \vec{i}(t)$$

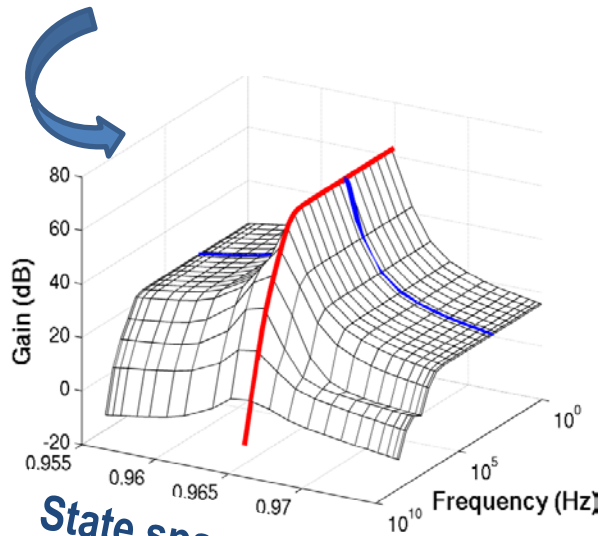
$$\begin{aligned} \rightarrow [C1]d\vec{v}(t) + [G1]\vec{v}(t) &= \vec{i}(t) \\ &\vdots \\ \rightarrow [Cn]d\vec{v}(t) + [Gn]\vec{v}(t) &= \vec{i}(t) \end{aligned}$$



Model nonlinearity and memory effects

Data

$$[C(\vec{v}(t))]d\vec{v}(t) + [G(\vec{v}(t))] = \vec{i}(t)$$



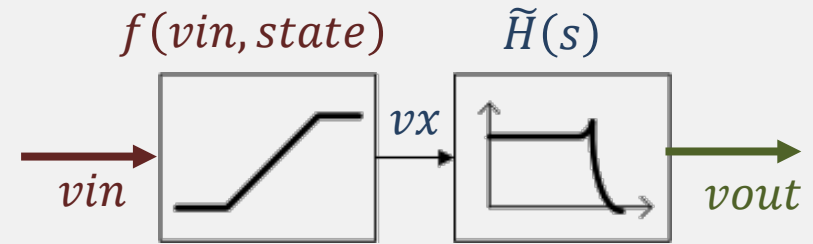
State space



fitting

$$\{H(s)@state\} \Rightarrow H(s, state) \approx \sum_{p=1}^{P \ll N} \left(\frac{r_p(state)}{s + a_p} \right)$$

Model



1. Nonlinear function

$$vx = f(vin, state) = \int r(state) dvin$$

2. Transfer function (memory)

$$\frac{vout}{vx} = \tilde{H}(s) = \sum_{p=1}^{P \ll N} \left(\frac{1}{s + a_p} \right)$$

internal delay

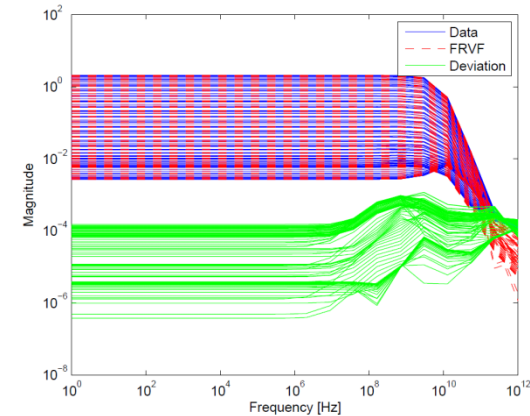
Fitting for nonlinear systems

1. Fit the frequency function

$$\frac{v_{out}}{v_x} = \tilde{H}(s) = \sum_{p=1}^{P \ll N} \left(\frac{1}{s + a_p} \right)$$

→ Vector Fitting Algorithm:

Optimizes the internal delays a_p of the model



2. Fit the nonlinear function(s)

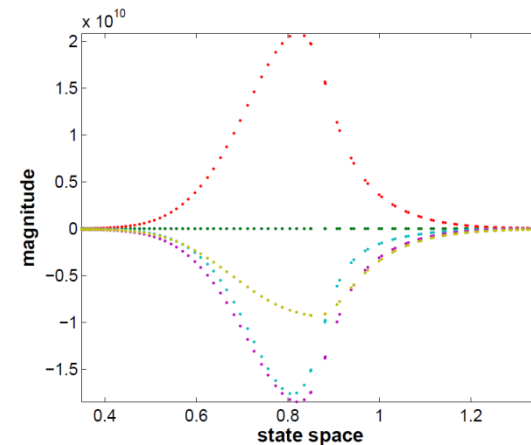
$$v_x = f(v_{in}, state)$$

→ Machine Learning (Classifiers, Regressors)

Optimally fit a given set of data with a mathematical function

Neural Networks, Support Vector Machines,

Nearest Neighbours, Evolutionary algorithms, ...



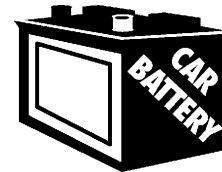
The proof is in the pudding...

PRACTICAL APPLICATIONS

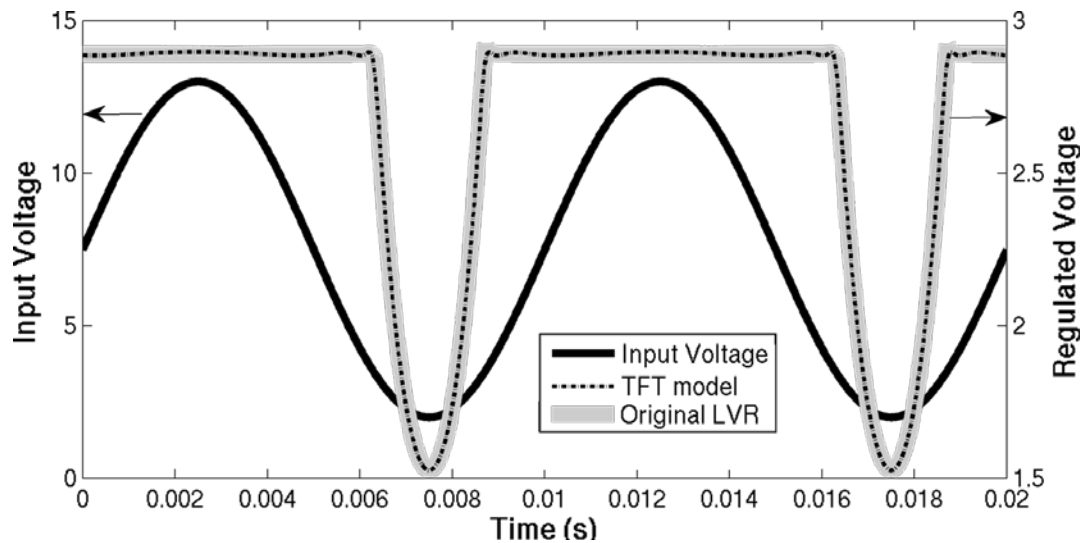
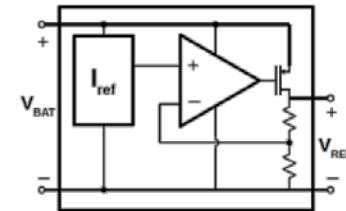
Application Example (1)

■ Linear Voltage Regulator

- Original size: 1250 eqns
- ❖ Model size: **12** eqns
- ❖ Simulation speedup: **110X**
- ❖ Error < **2%**



Linear Voltage Regulator

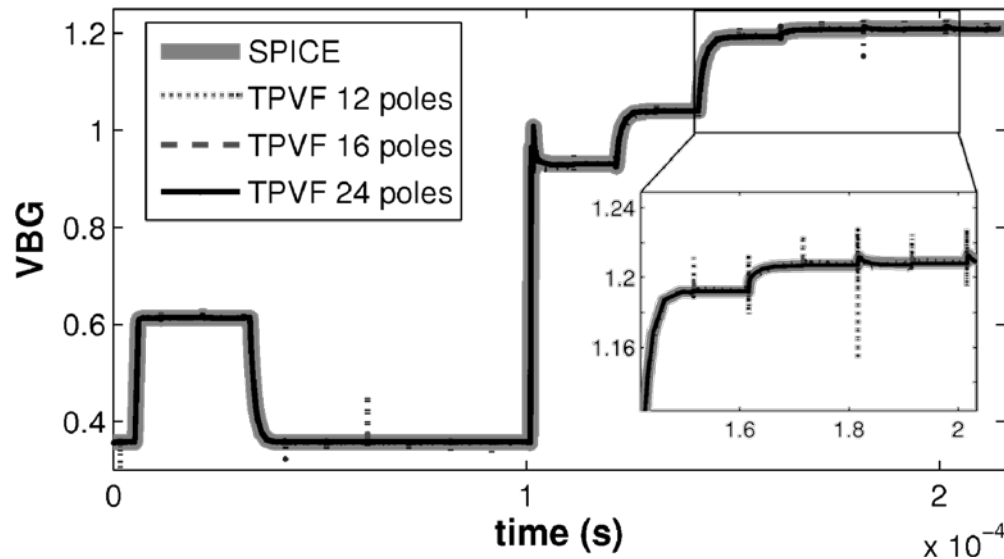
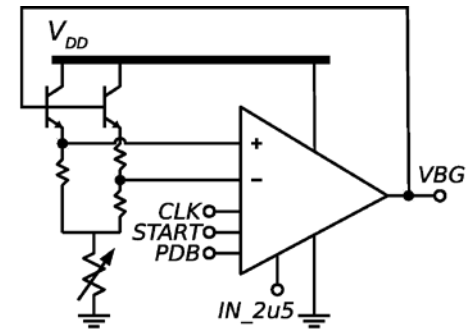


Application Example (2)

■ Auto-Zero Bandgap

- Original size: 650 eqns
- ❖ Model size: **20** eqns
- ❖ Simulation speedup: **50X**
- ❖ Error < **2%**

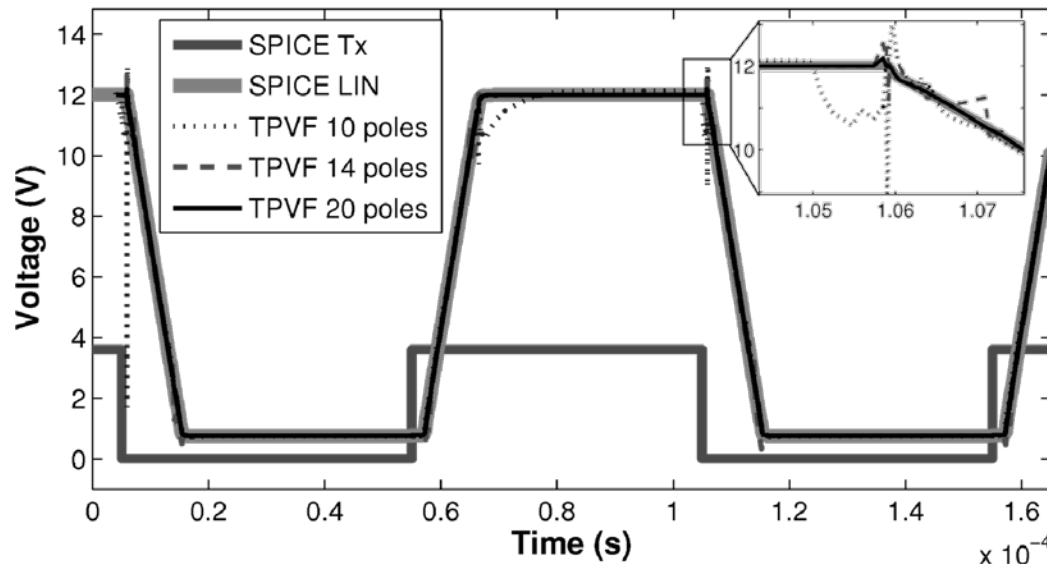
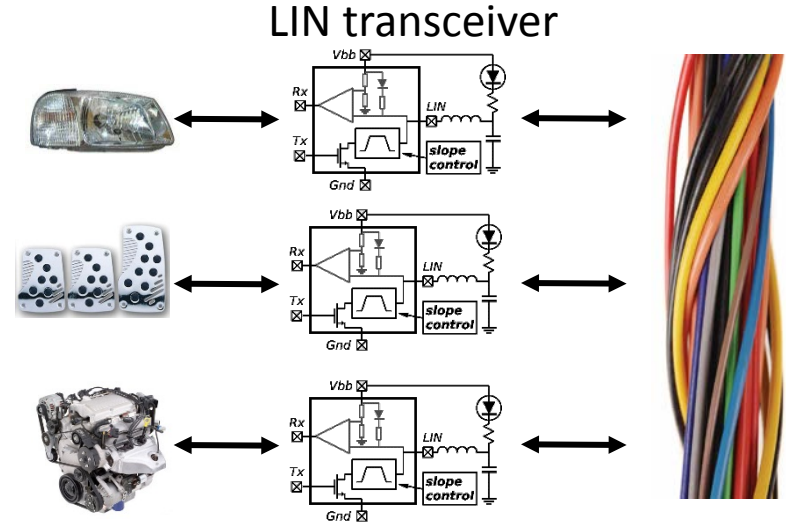
Absolute voltage reference (1.2V)



Application Example (3)

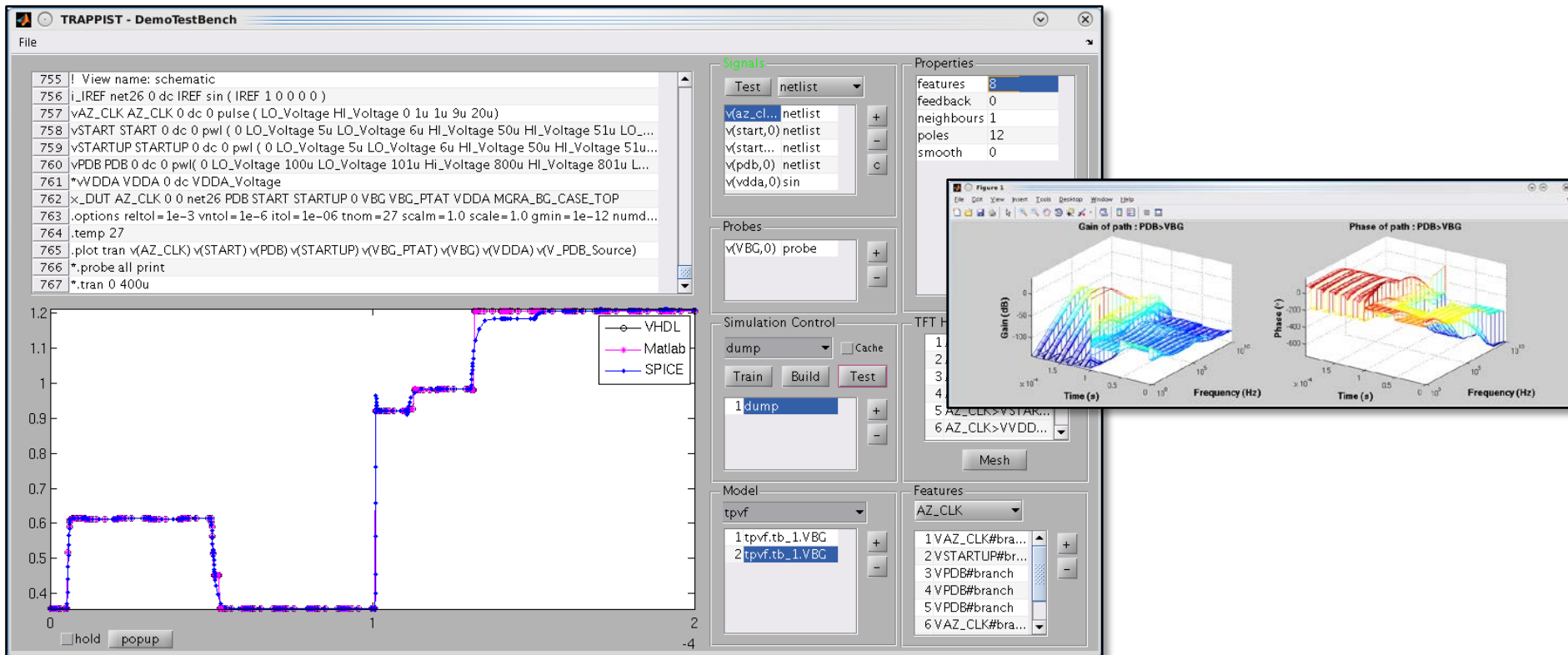
■ LIN-transceiver

- Original size: 1509 eqns
- ❖ Model size: **20 eqns**
- ❖ Simulation speedup: **60X**
- ❖ Error < **2%**



GUI: TRAPPIST

- **T**Rajjectory **A**Pproximation by **P**iecewise Interpolation of **S**tate-dependent **T**ransfer Functions



Conclusion & Challenges

- **Virtual design environments**

- Design of **complex** structures and systems
- **Accurate** physical models are **expensive**
- Model order reduction

- **Challenges remain**

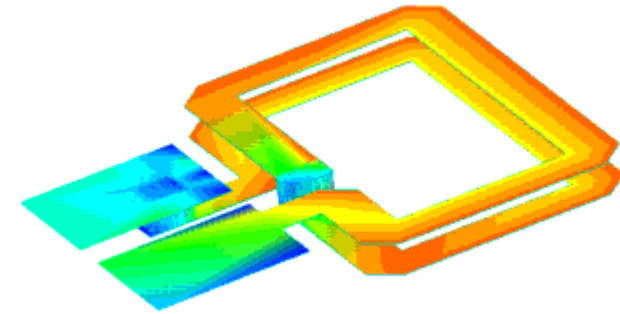
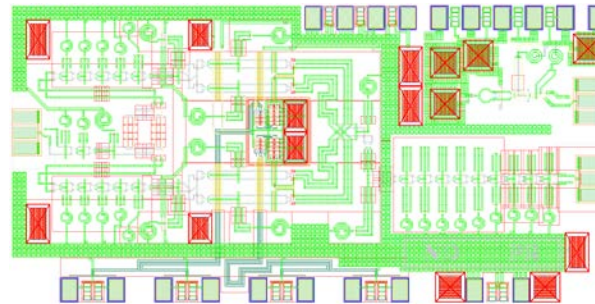
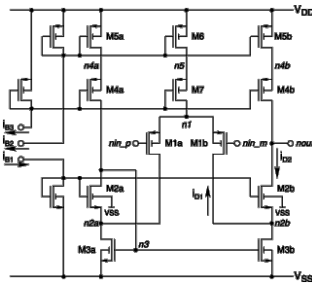
- **Multiphysics** (MEMS): EM + stress + heat + flow + ...
- **Stochastic** systems (nano): $[C, G] \rightarrow \mu[C, G], \sigma[C, G]$
- **Extremely high frequencies**: EMC, mmWave, ...

– ...

OPTIMIZATION OF COMPLEX STRUCTURES

Design issues

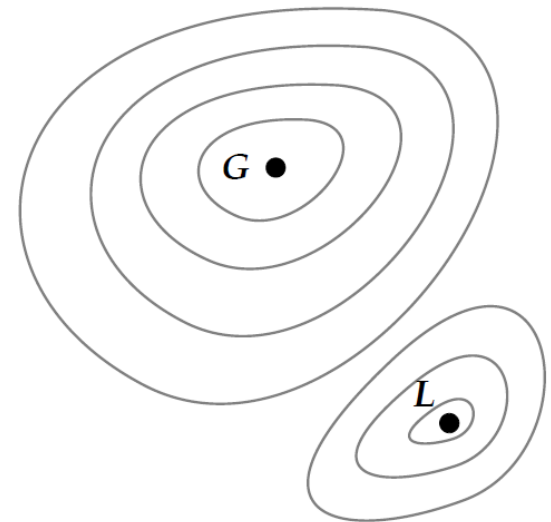
- High-dimensional design space
 - Lots of *free variables*



- Complex trade-offs: *accurate (expensive) models*
- **Automate design (partially) by**
 - *Efficient optimization*
 - *Automated synthesis* of structures

Optimization algorithms

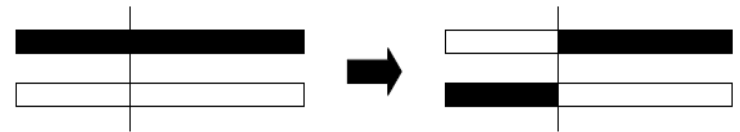
- Local unconstrained optimizers
 - e.g., gradient- or Newton-based approaches
- Constrained optimization
 - solve linear or nonlinear program
 - special case: Geometric Program
- Greedy stochastic algorithms
 - only improvements are allowed
- Annealing approaches
 - also up-hill moves can occur
- **Evolutionary techniques**
 - e.g., genetic algorithms, evolutionary strategies



Evolutionary Algorithms (EA)

- Mimic *evolutionary biology*

- *Inheritance, mutation, selection, crossover*
- Variables → genetic material
- Objective → fitness



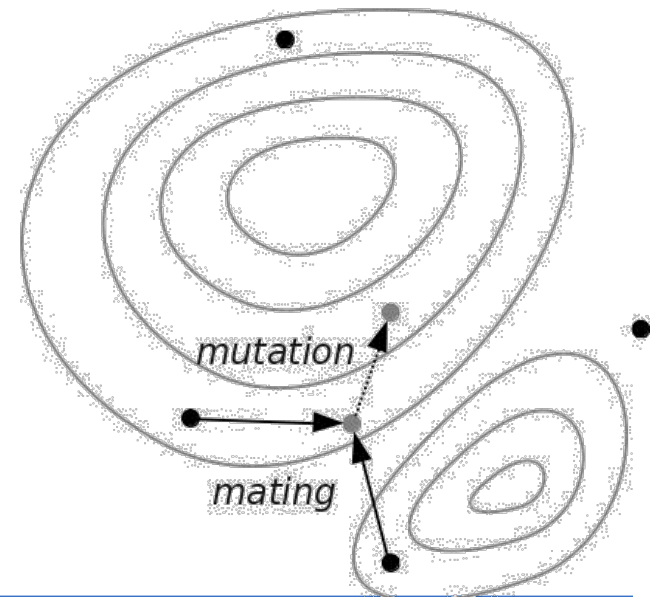
crossover



mutation

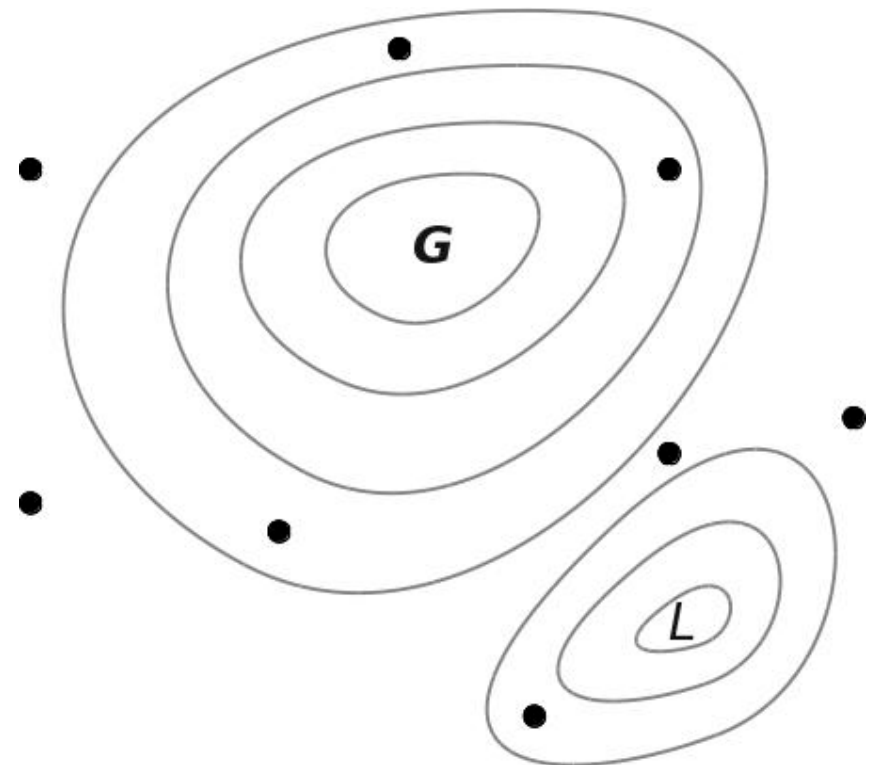
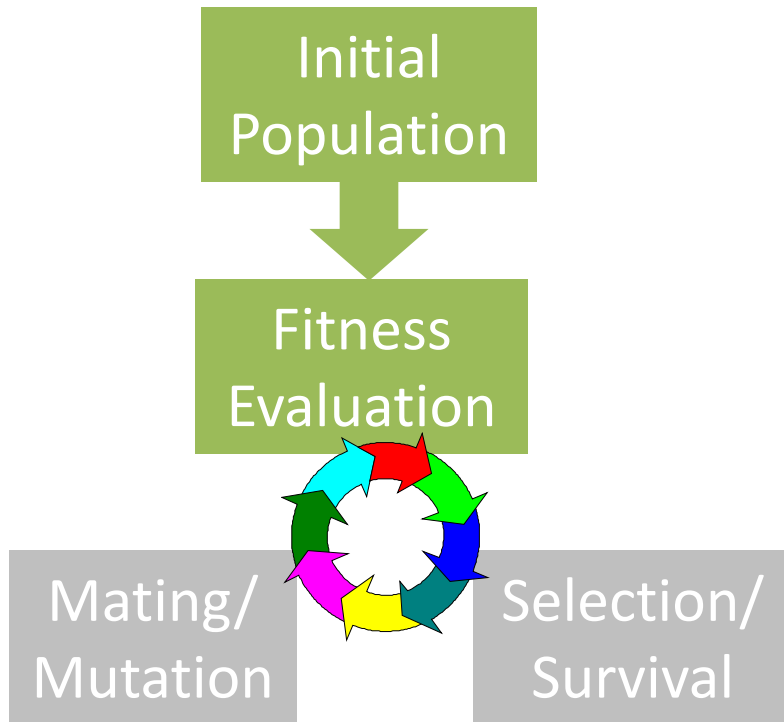
- Global optimizer

- If population large enough
- *Randomness* in each generation
- *Convergence* in each generation



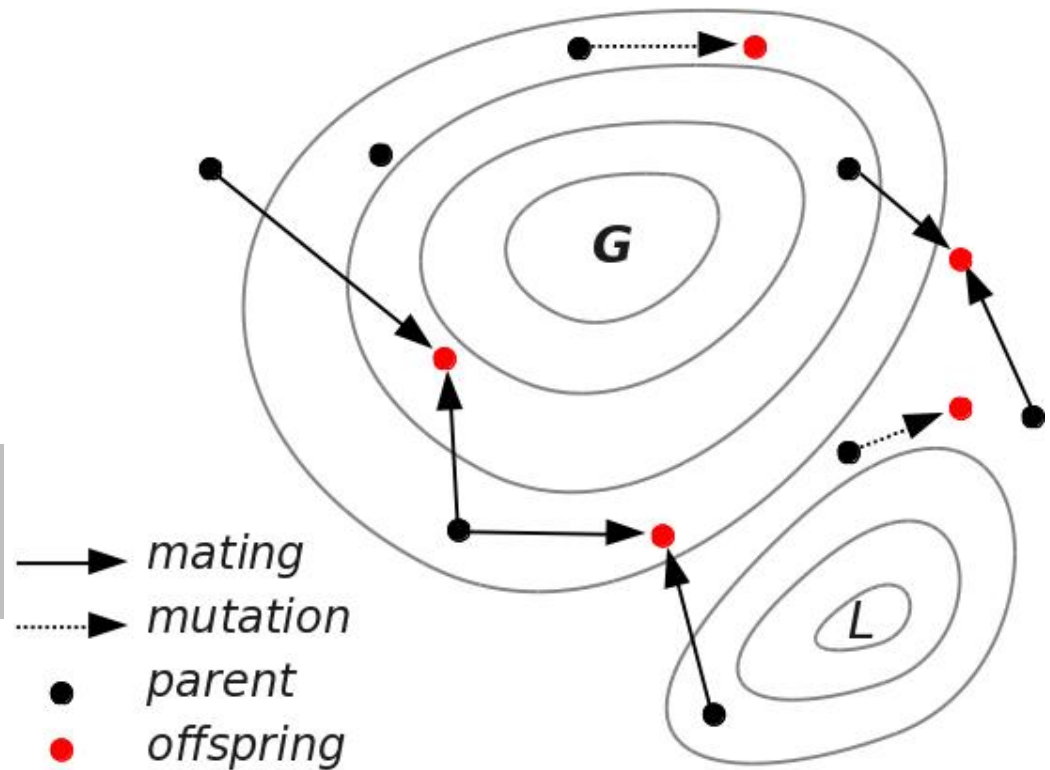
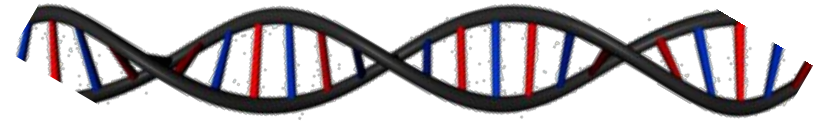
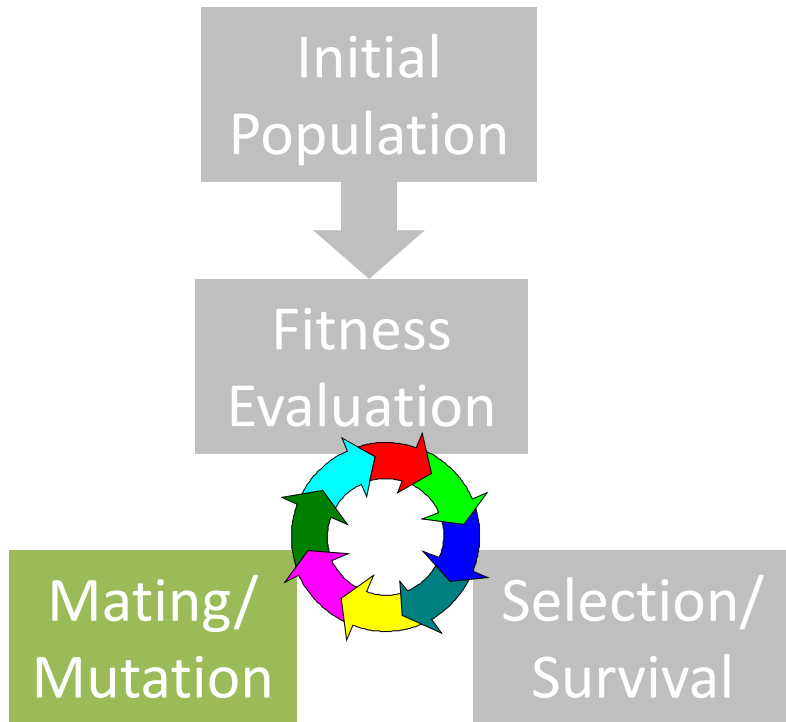
EA: Random Initialization

- Evolutionary loop



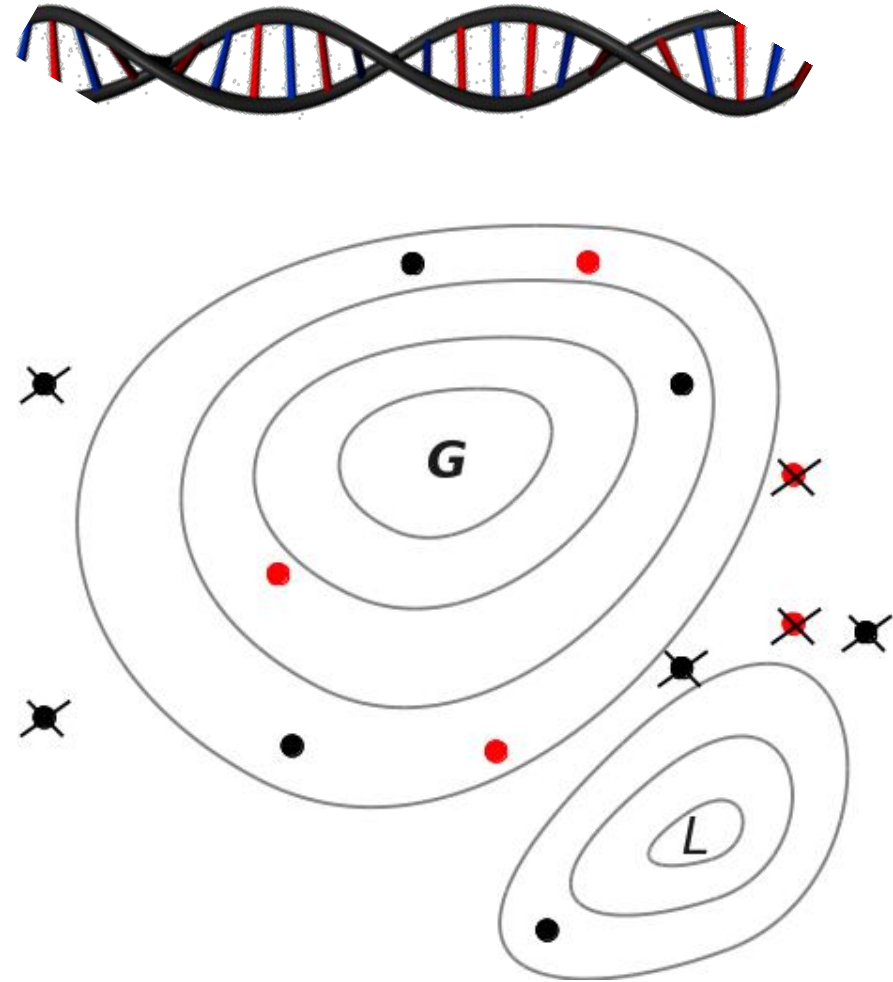
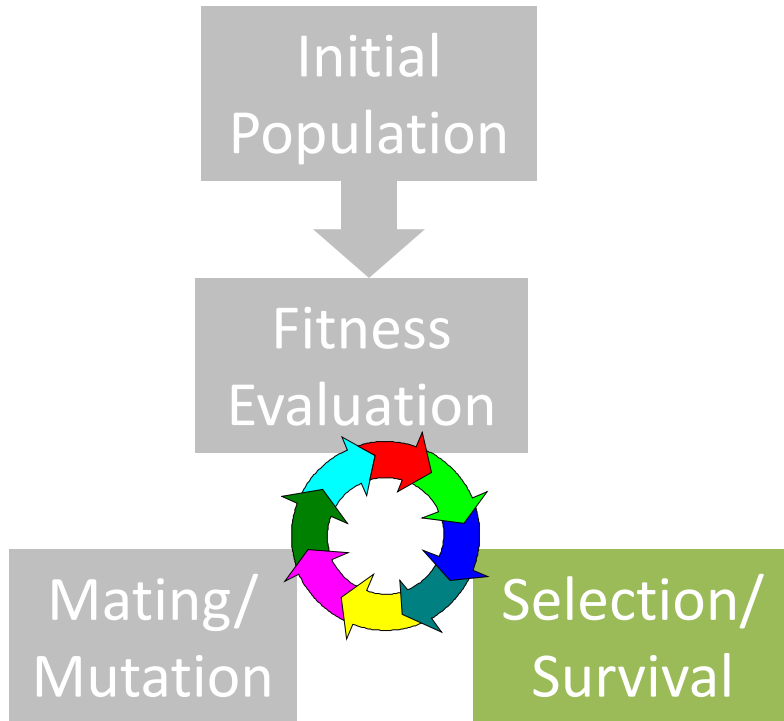
EA: Mating and Mutation

Evolutionary loop



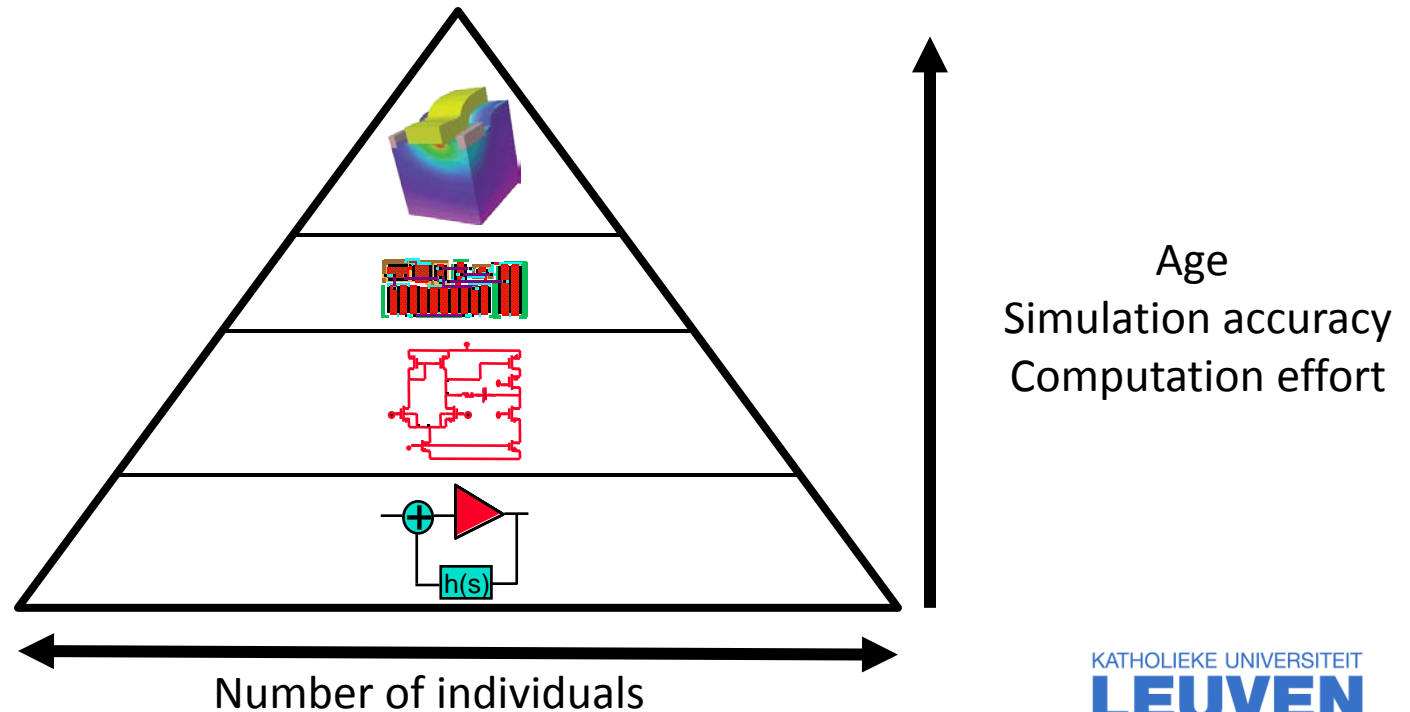
EA: Survival of the fittest

- Evolutionary loop



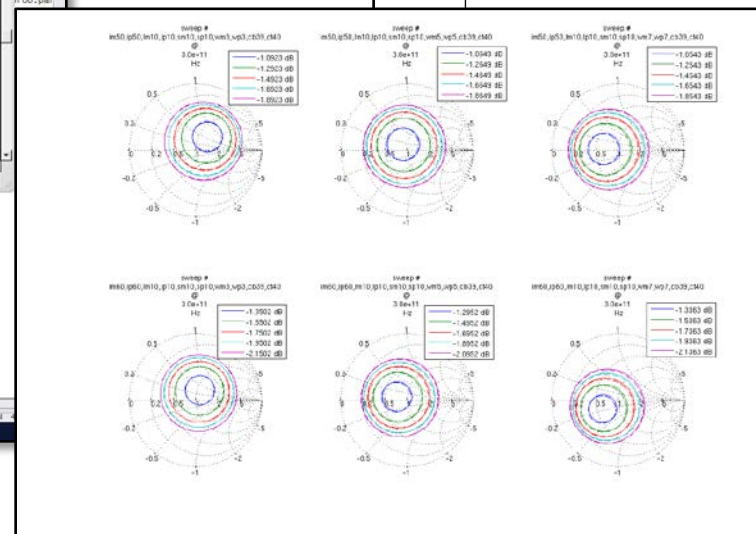
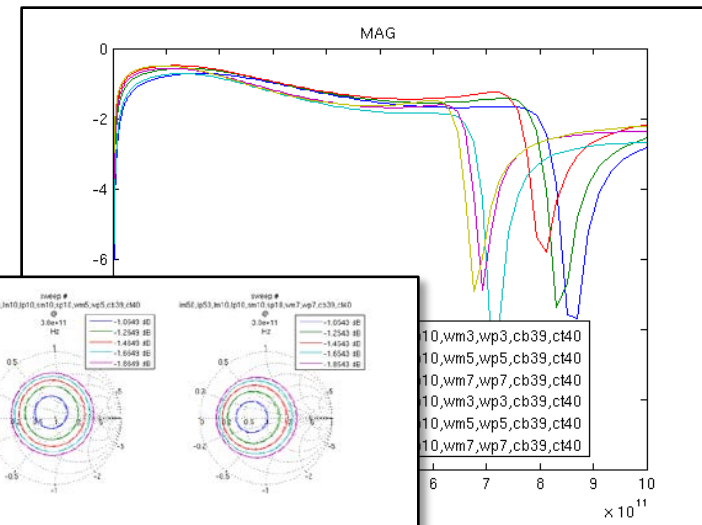
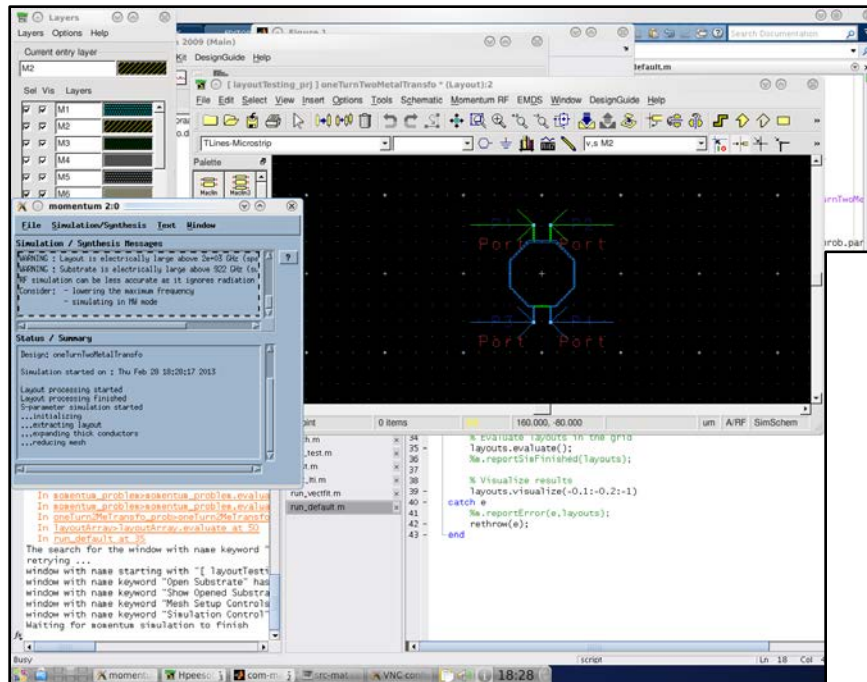
ALPS: Age layered population structure

- Only “zoom in” on promising survivors
 - **Accurate** models (FEM, 2D) only for **mature** survivors
 - **Cheap** models (Data-flow, 1D) for **young** individuals
- ➔ Different abstractions levels of the system



LAICO: Layout-Aware IC Optimizer

- Layout Generation tool
 - high frequencies and power converters
 - Synthesize, simulate and optimize



10_wm3_wp3_cb39_ct40
10_wm5_wp5_cb39_ct40
10_wm7_wp7_cb39_ct40
10_wm3_wp3_cb39_ct40
10_wm5_wp5_cb39_ct40
10_wm7_wp7_cb39_ct40

Conclusion & Challenges

- **Optimization of complex structures**
 - Various algorithms *mimic biology*
EA, swarm/particle optimization, annealing, ...
 - Efficient use of brute computation force

- **Challenges remain**
 - *Expensive* optimization
stochastic systems, 3D structures, nano stuff...
 - How to *filter out nonsense?*
constrain the design space

Questions?



Thanks for your attention...