Virtual Design of Integrated Circuits

Dynamical systems, model-order reduction and design optimization

Dimitri De Jonghe

Georges Gielen

KATHOLIEKE UNIVERSITEIT LEUVEN
ESAT

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

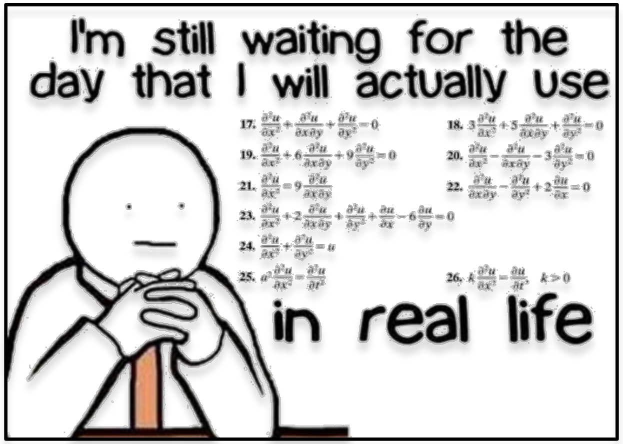
VIRTUAL DESIGN ENVIRONMENTS

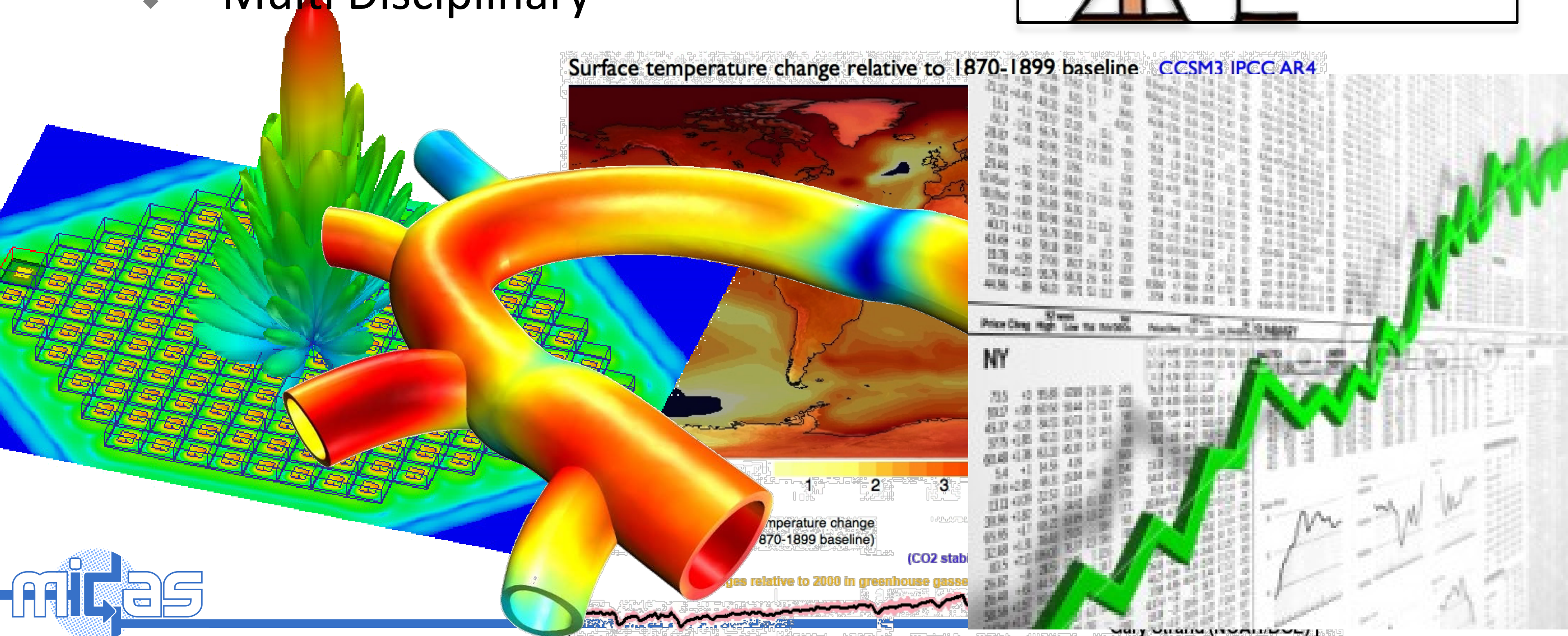
Computer Aided Design

- Modeling & Simulation
  - Applied mathematics

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

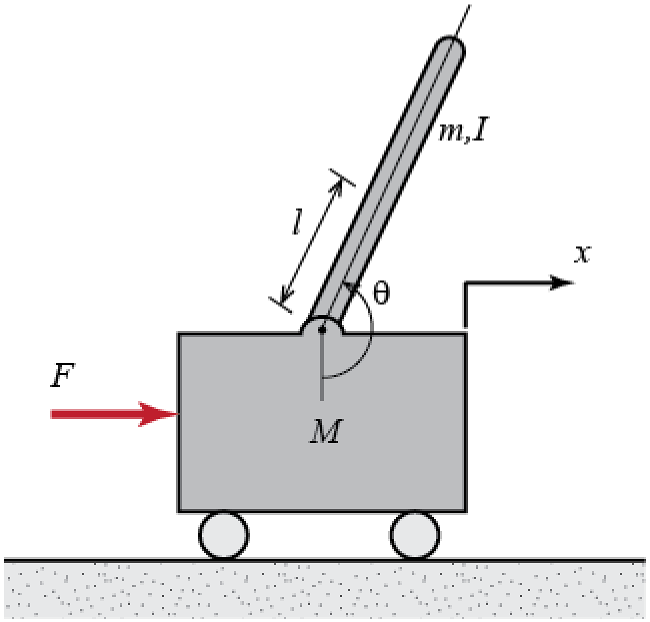
    > '90: + Machine Learning, + AI
  - Multi Disciplinary

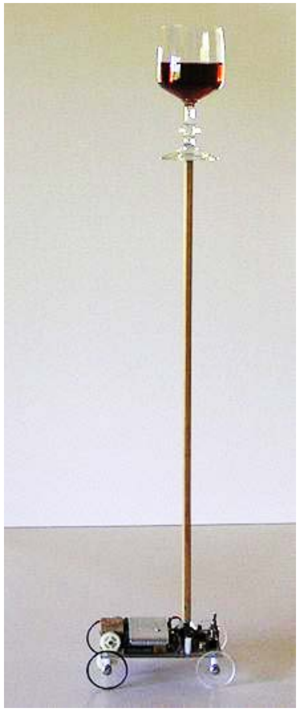




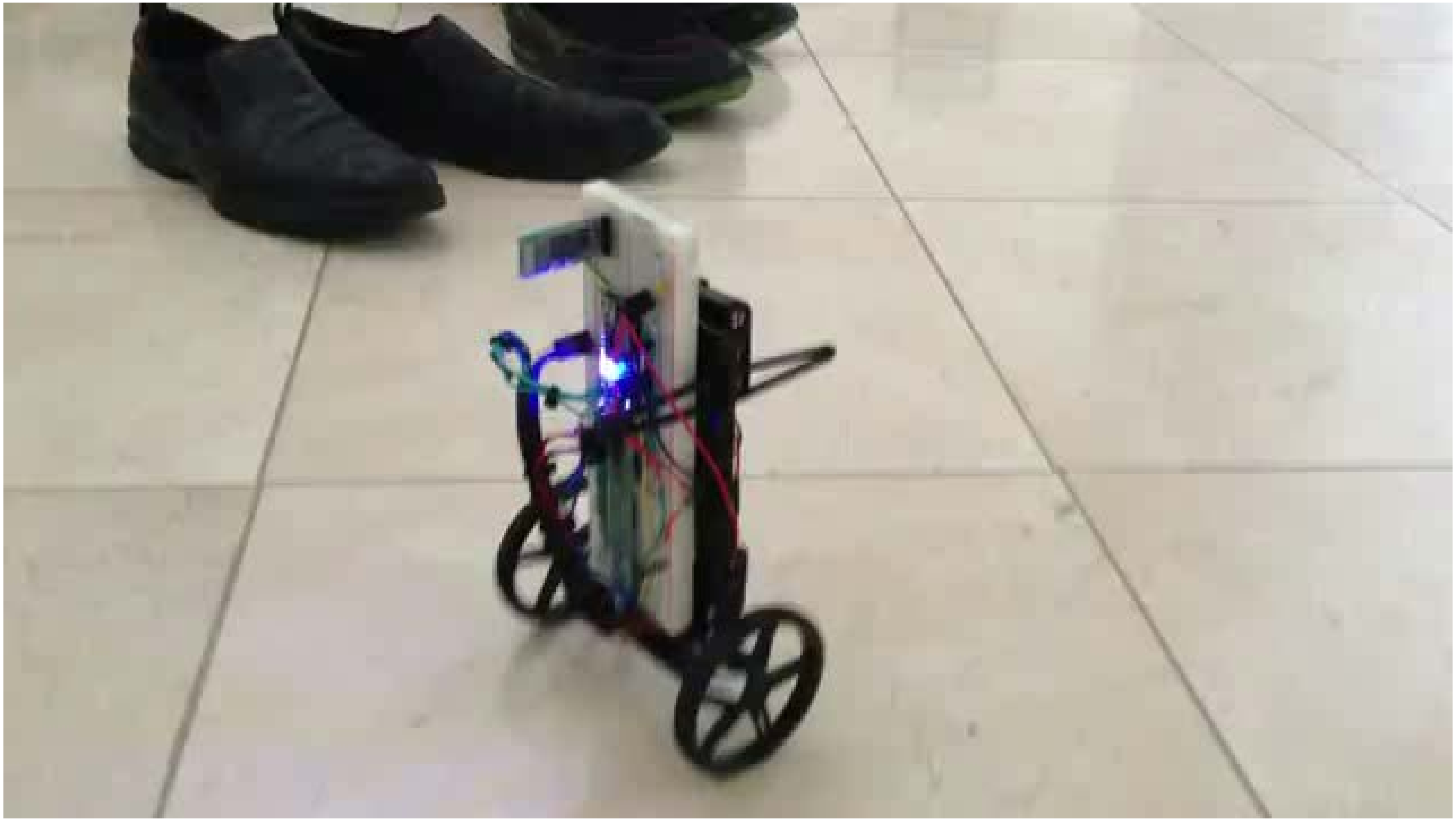
Some Examples

- Inverted pendulum





Inverted Pendulum

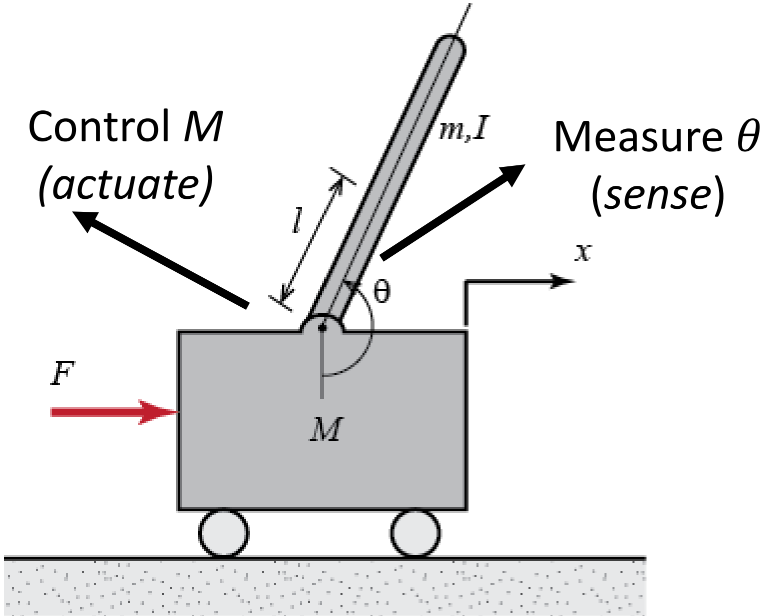


Inverted Pendulum

- How to control θ?

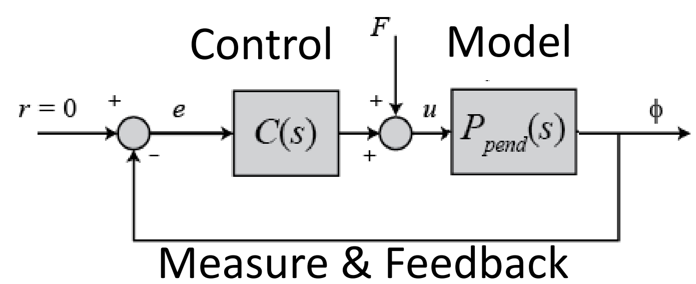
  Sense & actuate

  – 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

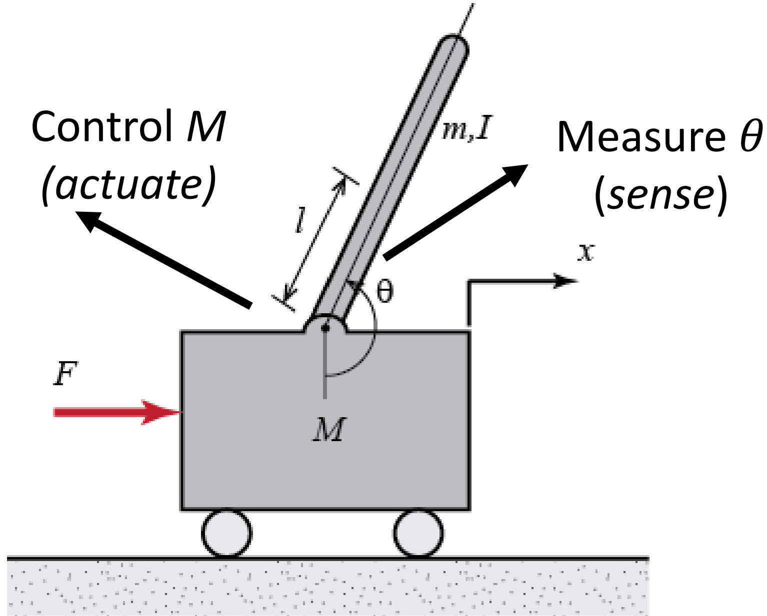
   e.g. Equations of motion

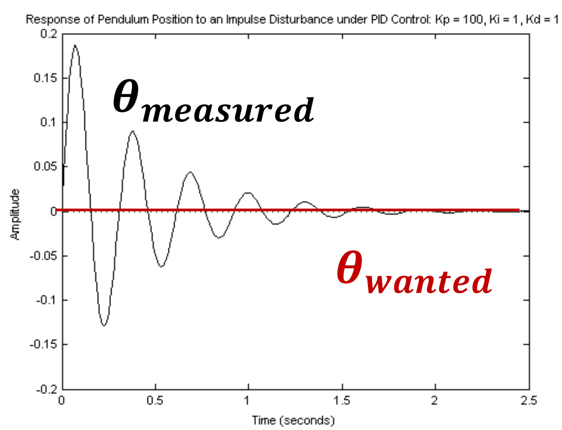
   $$(M+m)\,\ddot{x} - m\ell\ddot{\theta}\cos\theta + m\ell\dot{\theta}^2\sin\theta = F$$
   $$\ell\ddot{\theta} - g\sin\theta = \ddot{x}\cos\theta$$

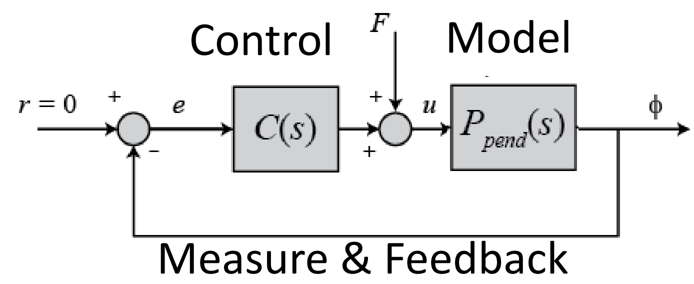
2. **Design** control

   $\boldsymbol{\theta_{measured}} \approx \boldsymbol{\theta_{wanted}} = 0°$

3. **Simulate** (and iterate)



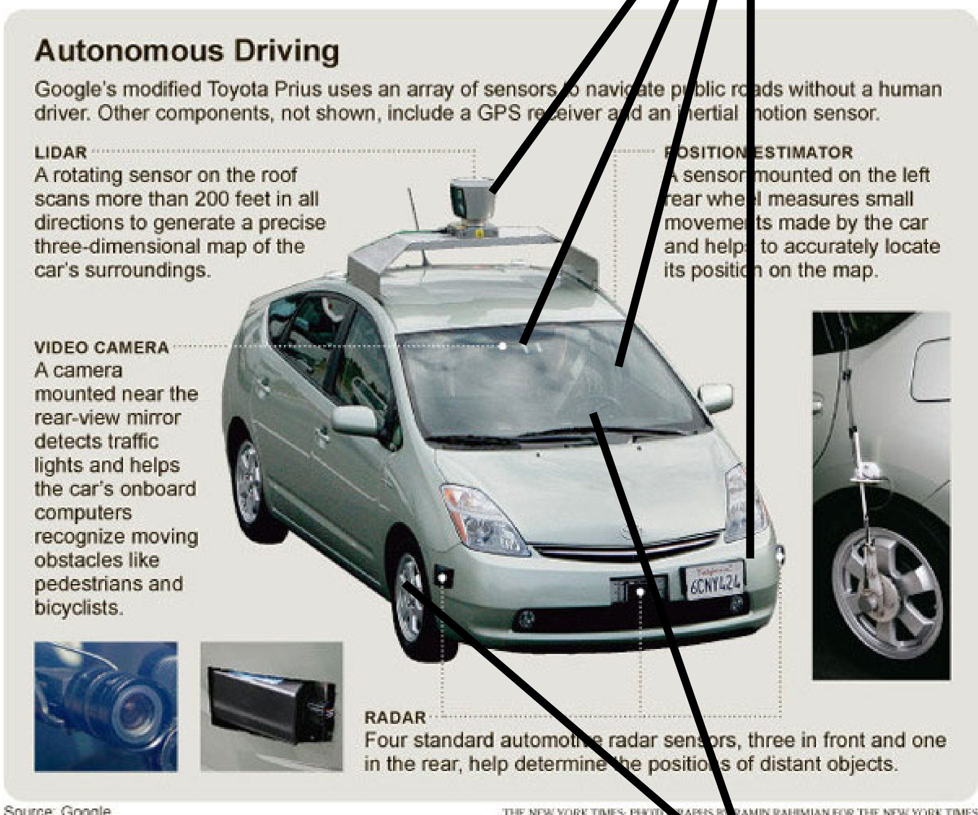




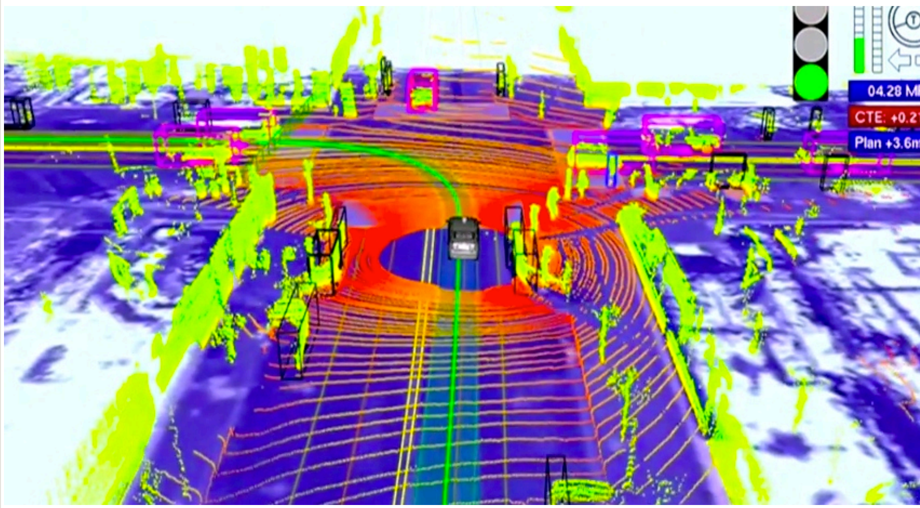
How would Google do this?

- Driverless car

Sense



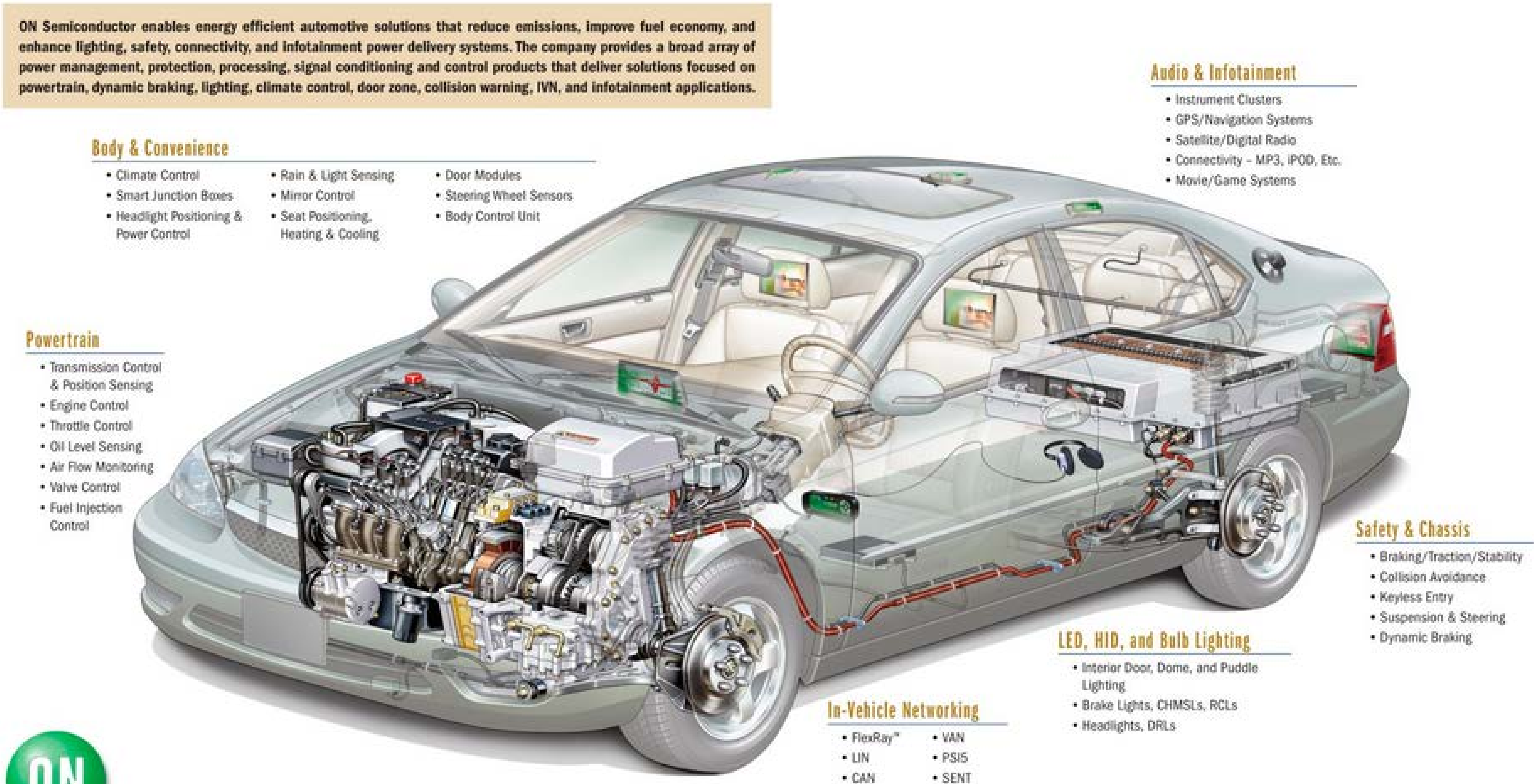
Real-time **virtual model**



Actuate

KATHOLIEKE UNIVERSITEIT LEUVEN

How would ON do this?

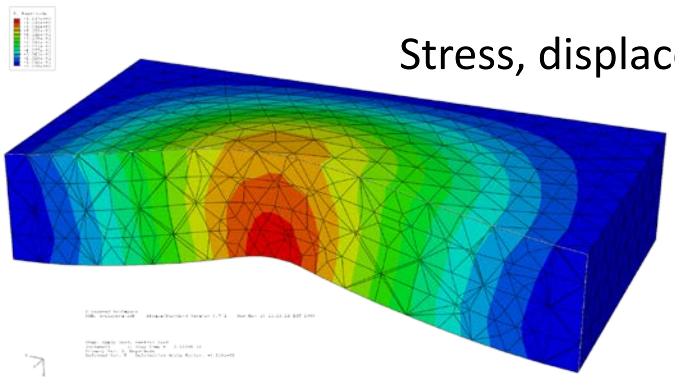


KATHOLIEKE UNIVERSITEIT LEUVEN

Plug in physical equations

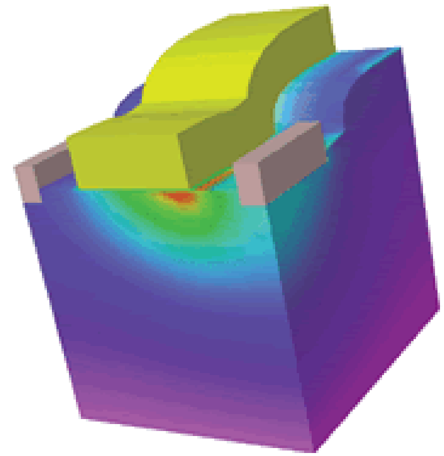
- Finite Element Modeling (FEM)

Mechanical



Stress, displacement

Electrical



Current density;
Potential

Mass, velocity (position), damping, force

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

Capacitance, voltage, conductance, current

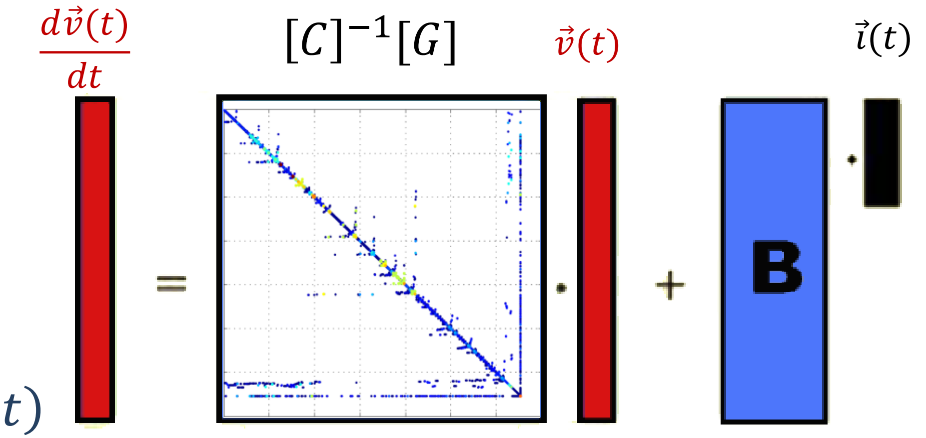
$$[C]d\vec{v} + [G]\vec{v} = \vec{\imath}$$

Large, sparse matrices!

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

KATHOLIEKE UNIVERSITEIT LEUVEN

Dynamical Systems



- Linear systems

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

  ➔ $[C]$, $[G]$ are constant matrices

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

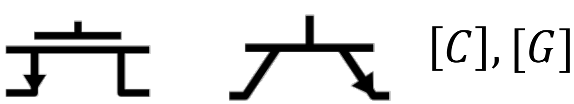
  ➔ Passives: **capacitors** **resistors** **inductors**

- Nonlinear systems

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

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

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



KATHOLIEKE UNIVERSITEIT LEUVEN

Solve Dynamical Systems

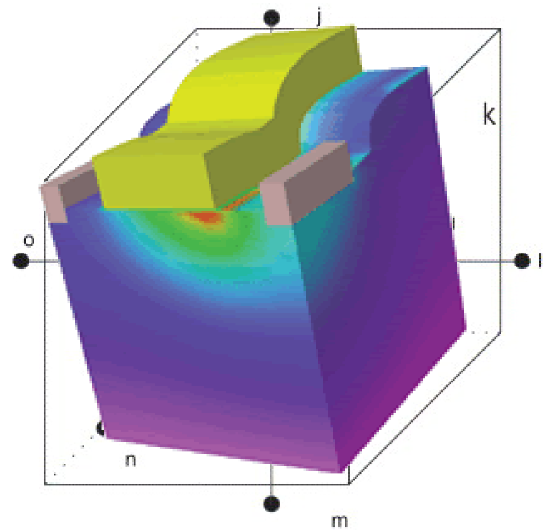
- Next-state equations

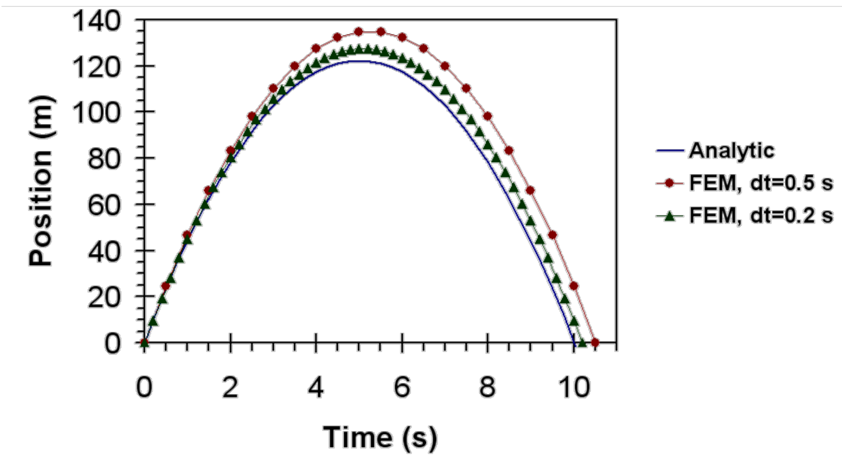
$$[C]\frac{d\vec{v}(t)}{dt} + [G]\vec{v}(t) = B\vec{i}(t)$$

  - Solve for discrete time

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

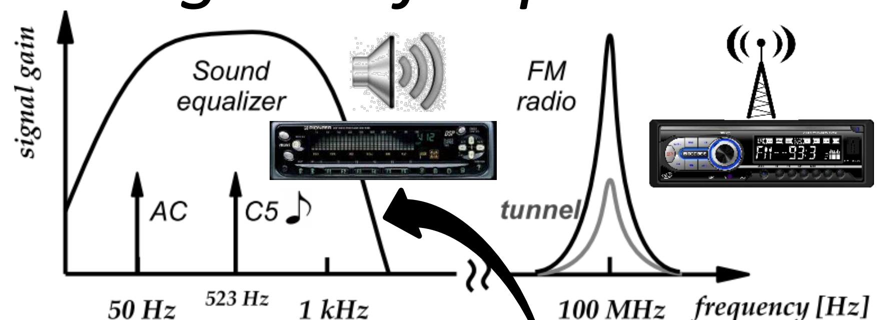
  - Adapt time step if variations ($\Delta\vec{v}$) are large/small





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



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

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

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

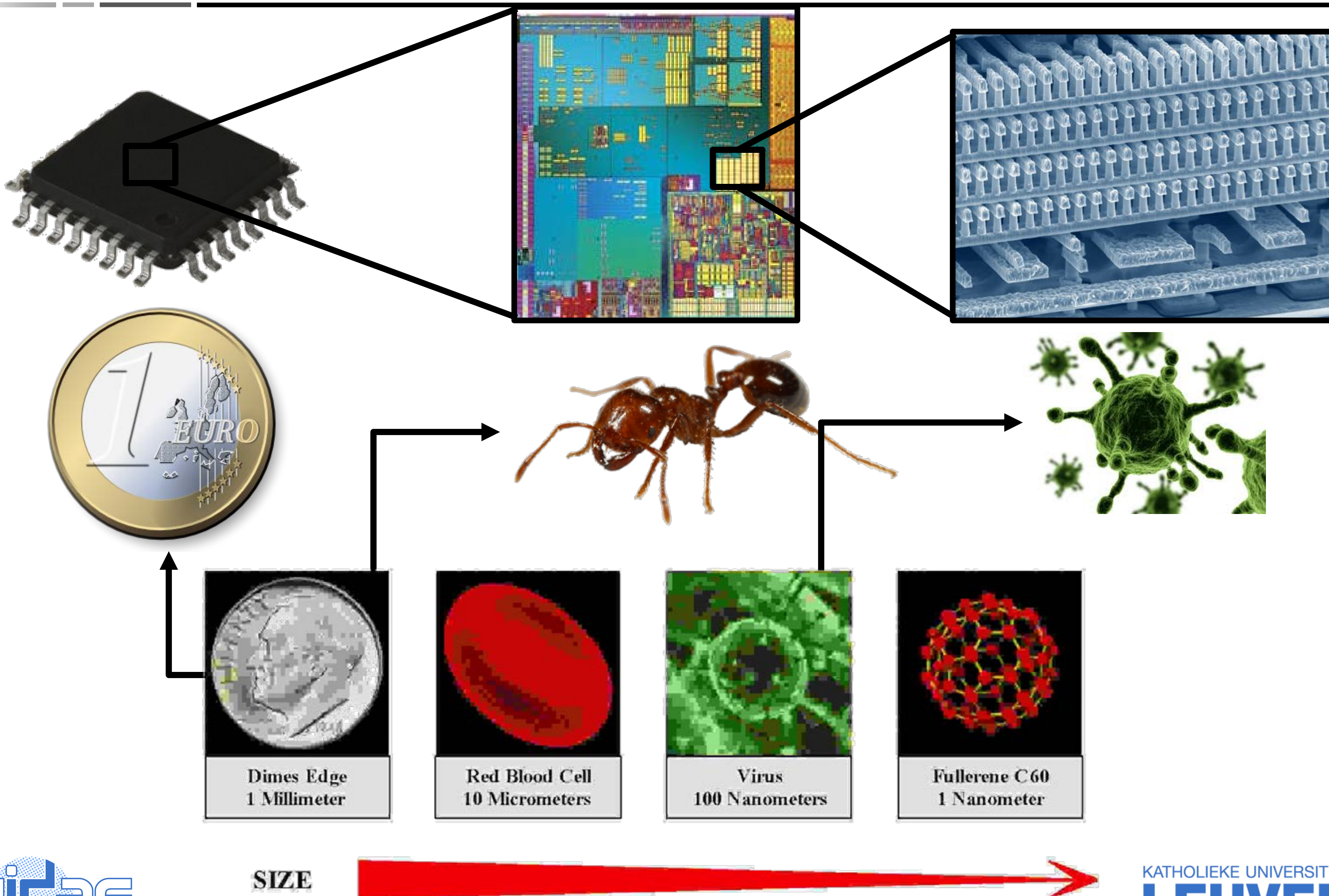
tough differential equation ⟶ *easy linear equation (+,-, x, /)*

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

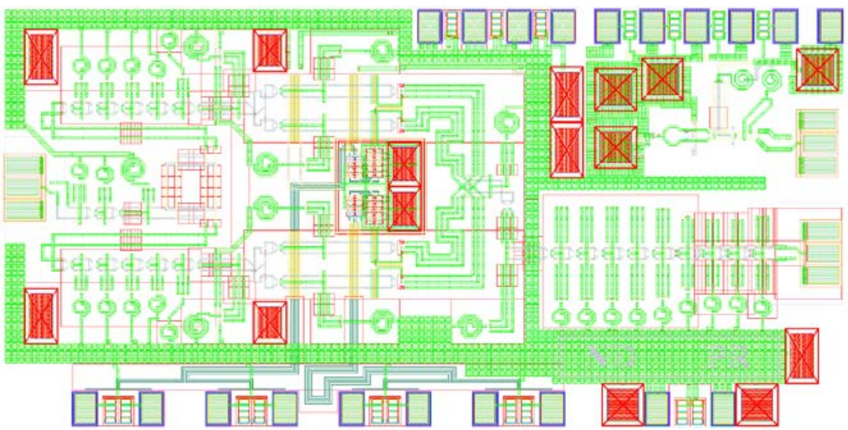


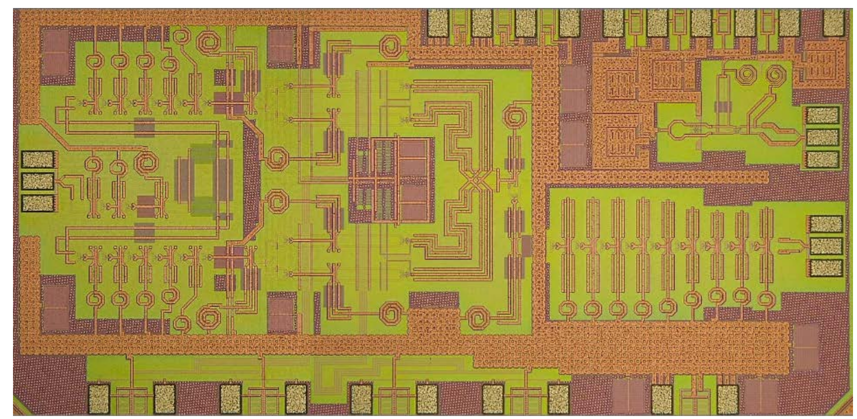
Fabrication of Integrated Circuits

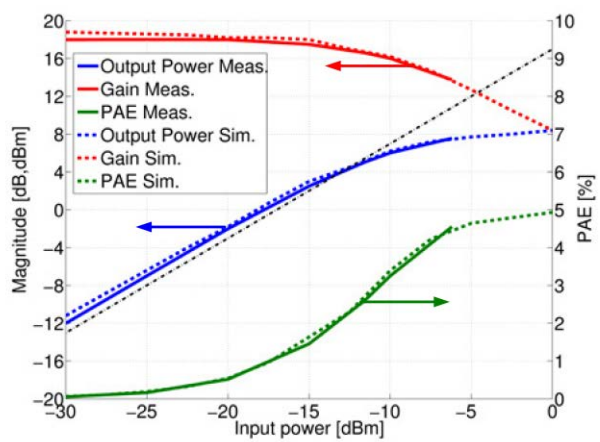
Computer Aided Design of Chips

e.g. 120 GHz Power Amplifier

- CAD model
- Actual chip

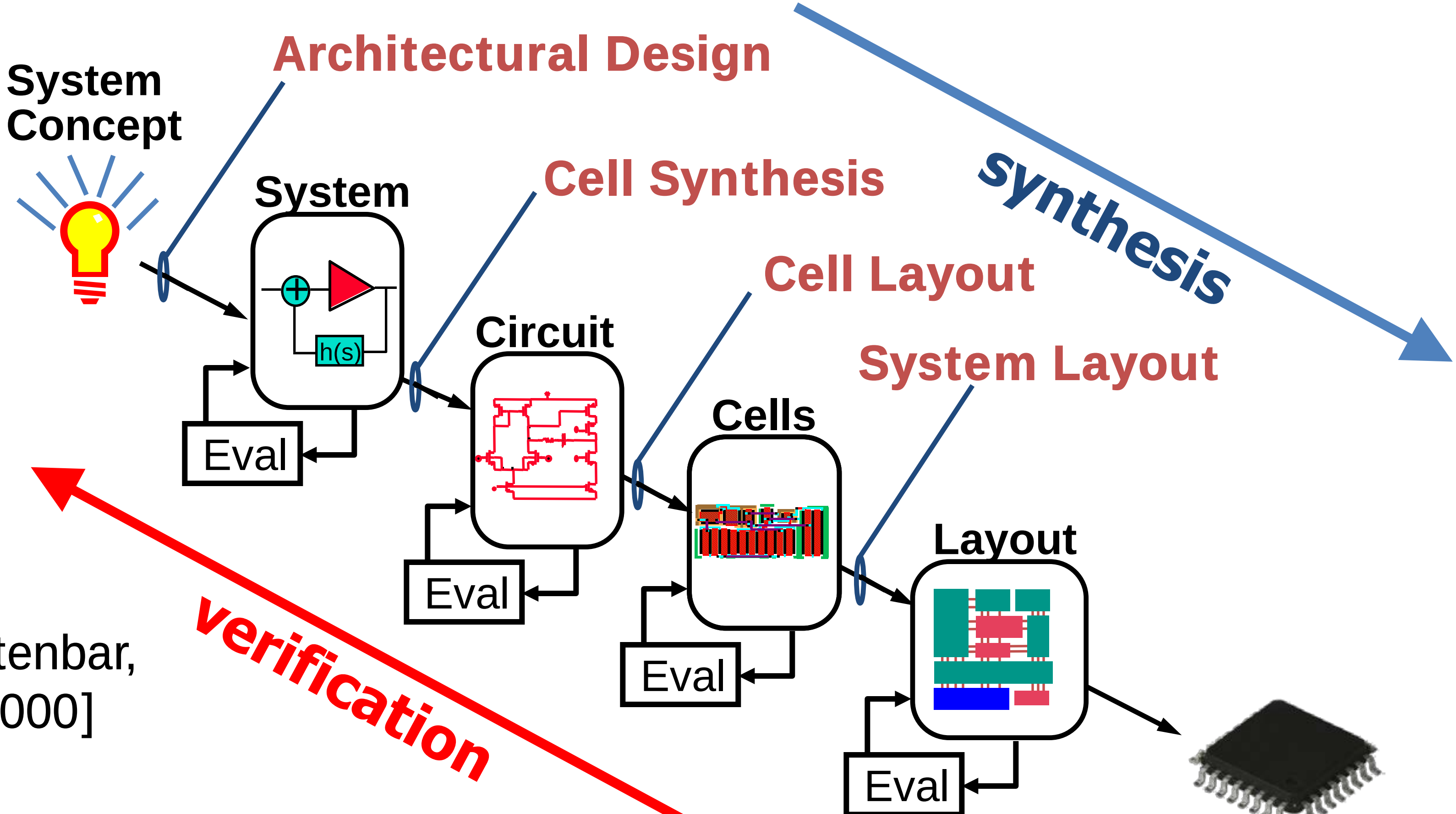






CAD layers of abstraction

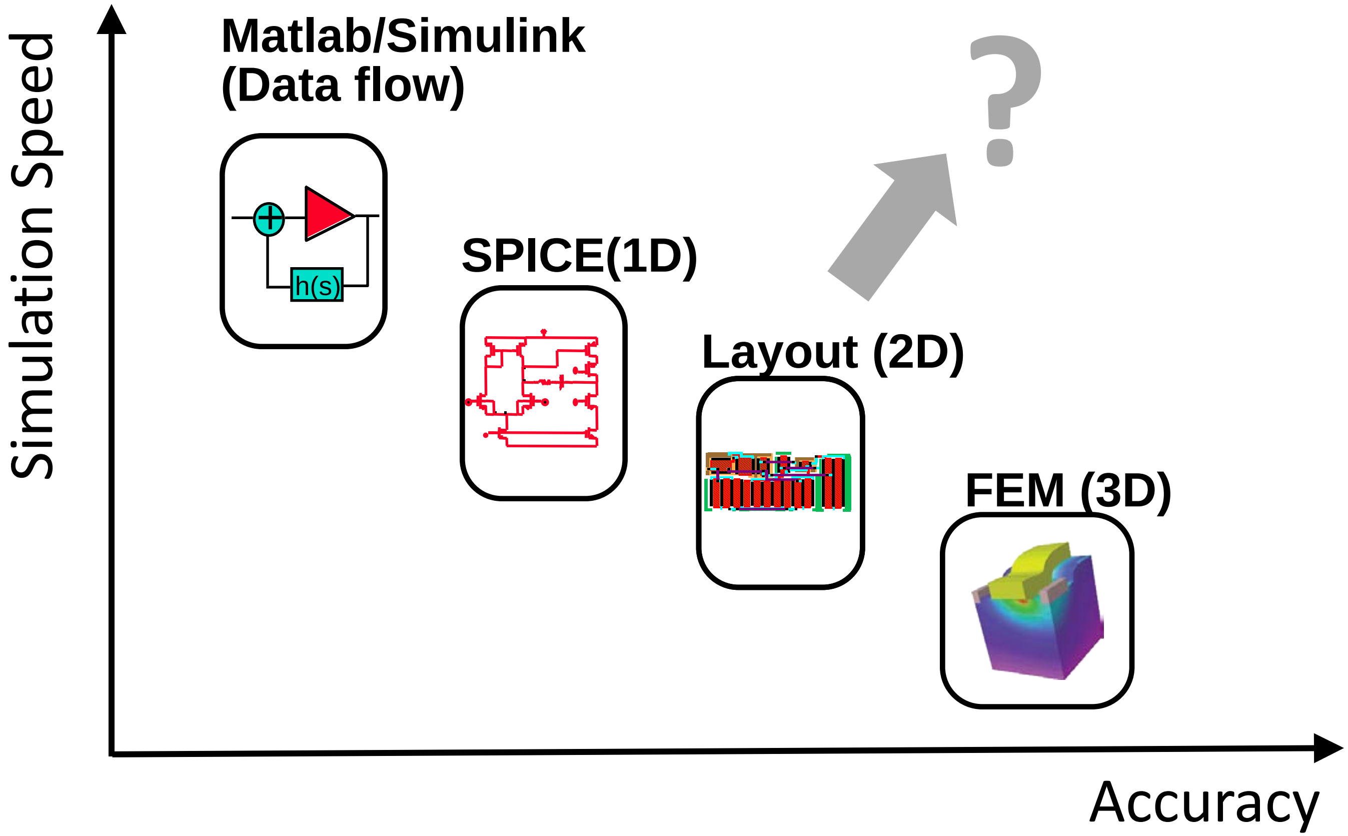
- Top-down design process
- Design iterations + bottom-up verification



[Gielen & Rutenbar, Proc. IEEE 2000]

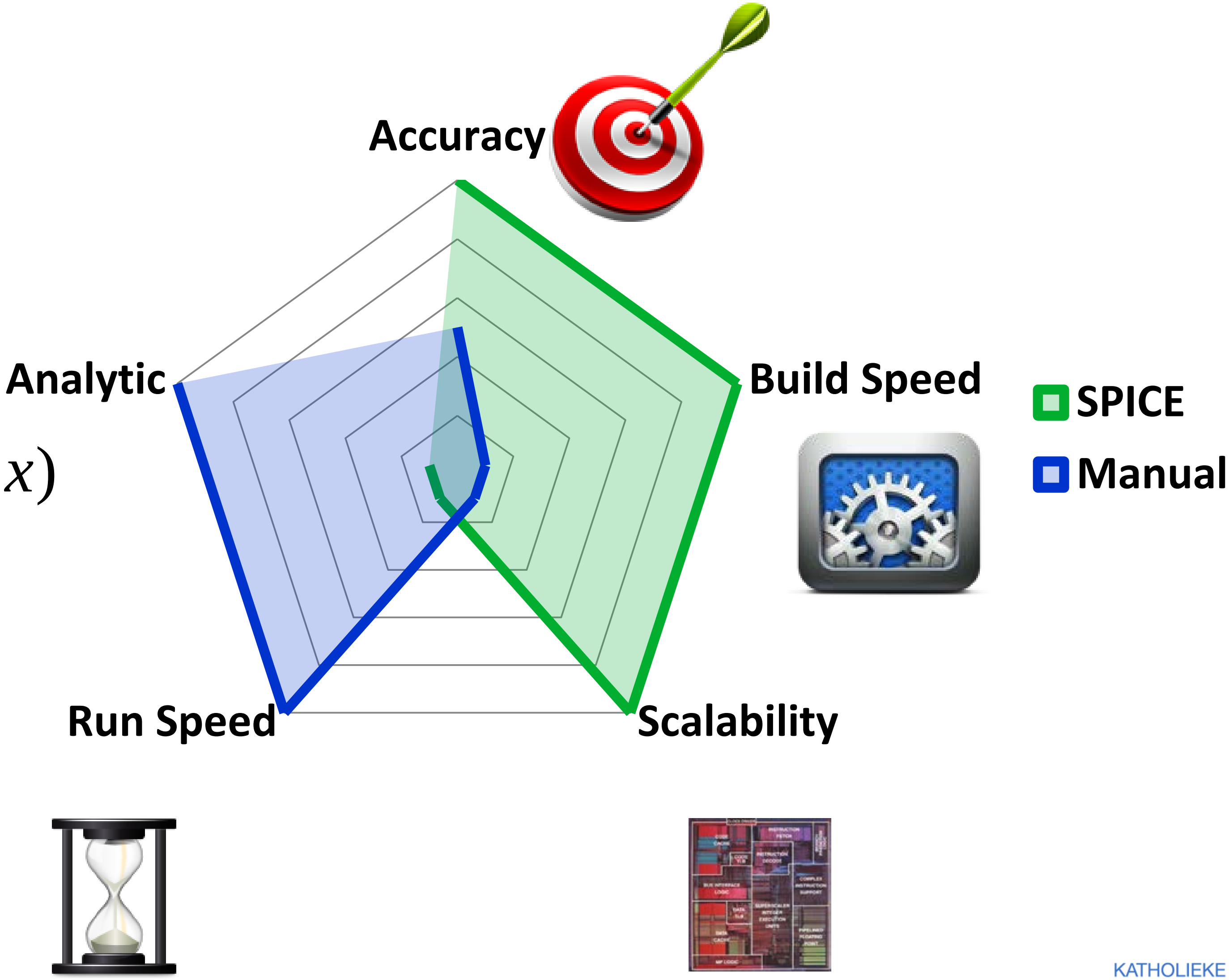
Simulation Trade-off

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



Modeling Trade-off

$$y = f(x)$$



Computer Aided Design: Remarks

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

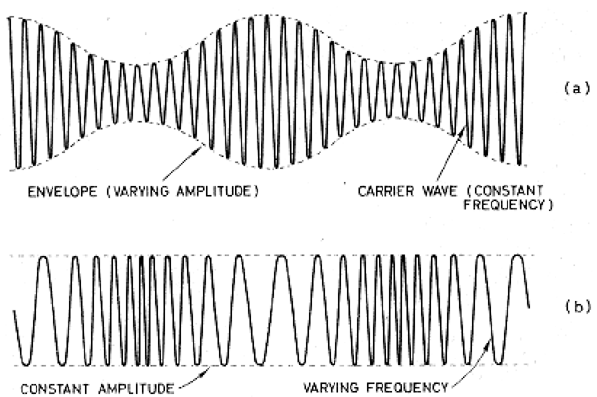
- But...
  - Computationally expensive!
  - ➔ Full system simulations may take weeks!

Virtual design environments for real-world applications

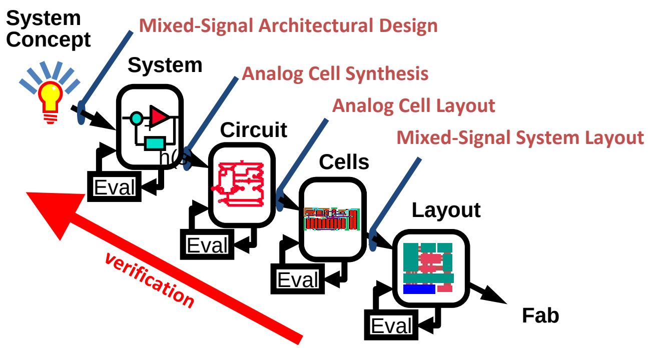
SIMULATION OF LARGE SYSTEMS

Motivation

- **Verfication** 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**

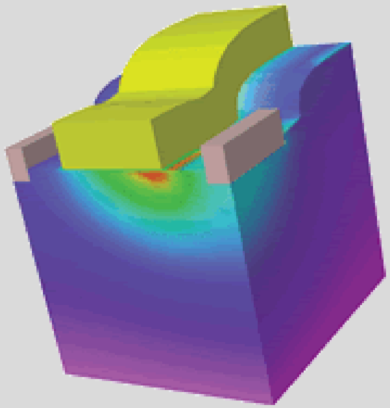
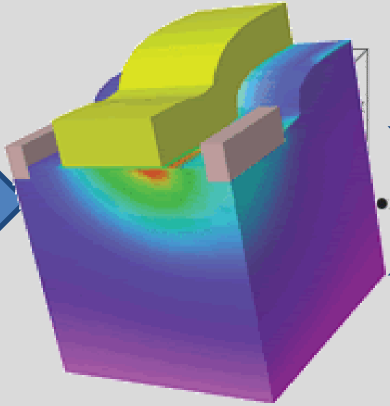
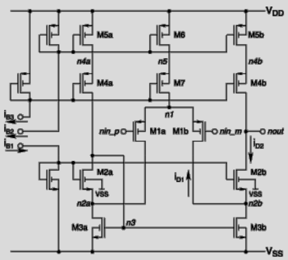
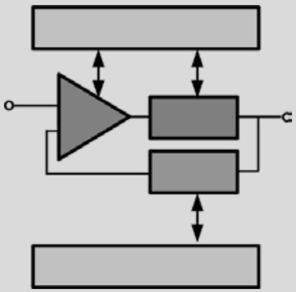
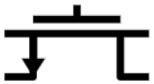
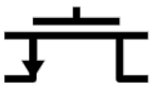
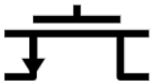


$$y = f(x)$$



[Gielen & Rutenbar, Proc. IEEE 2000]

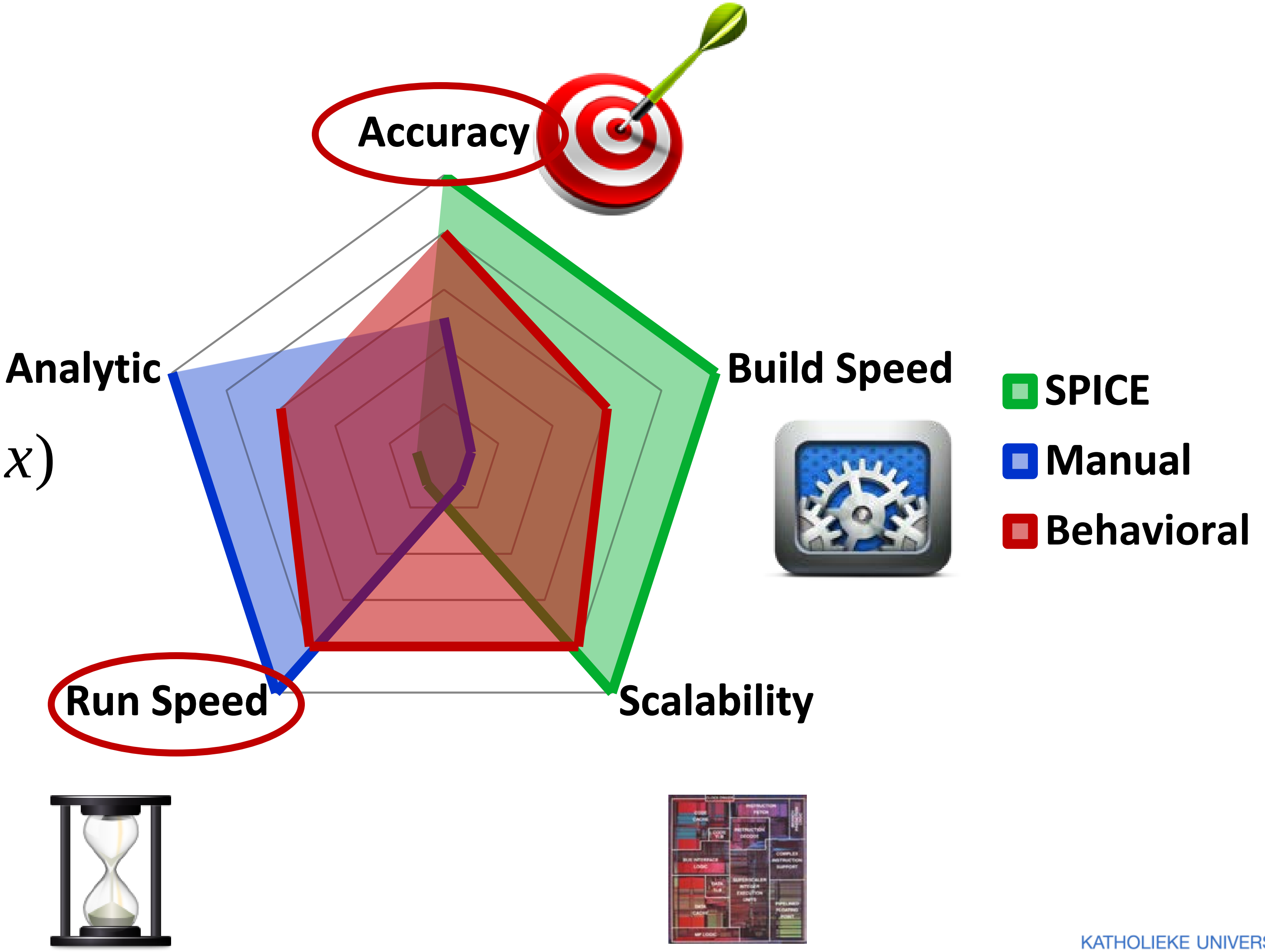
System-level simulation

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??

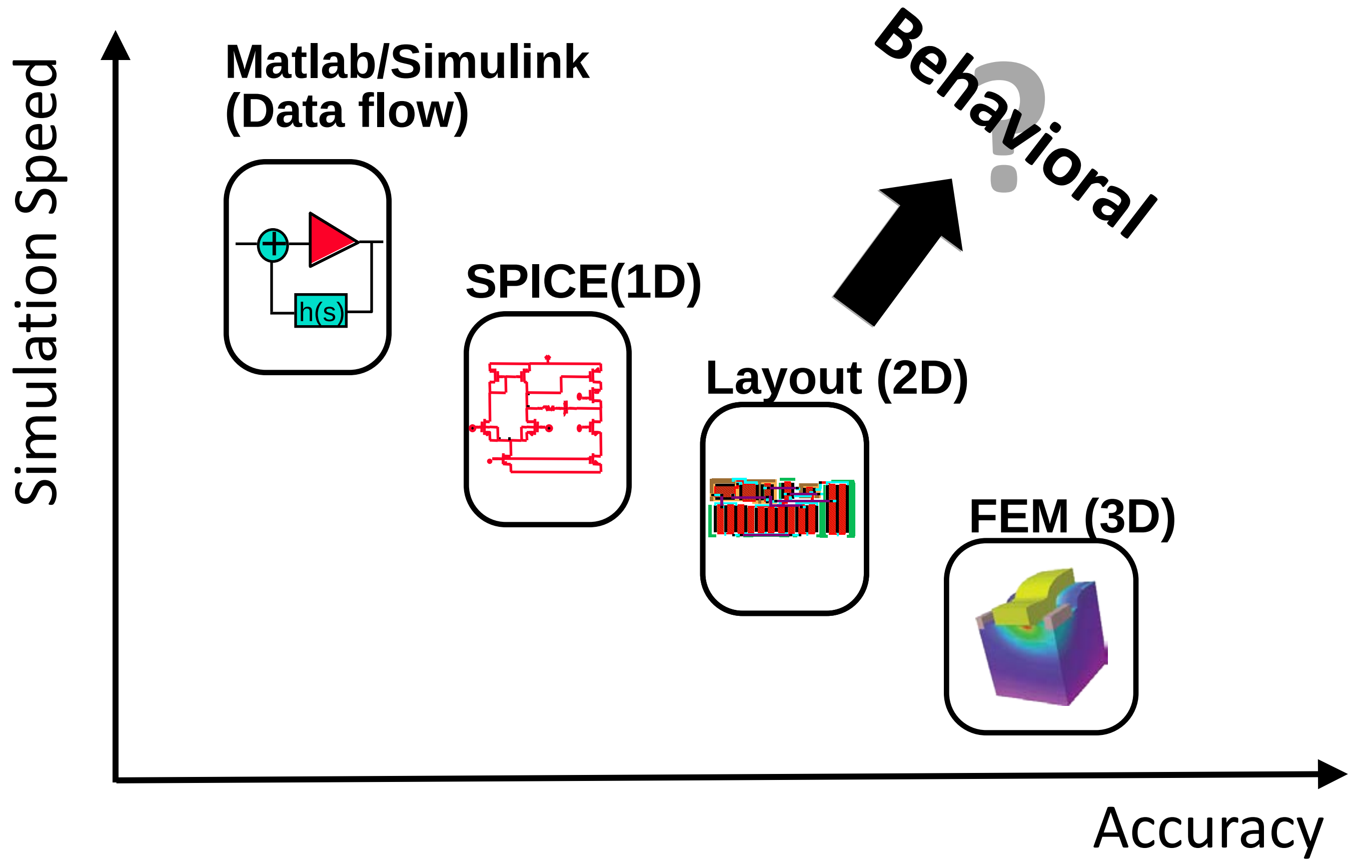
Modeling Trade-off

$$y = f(x)$$



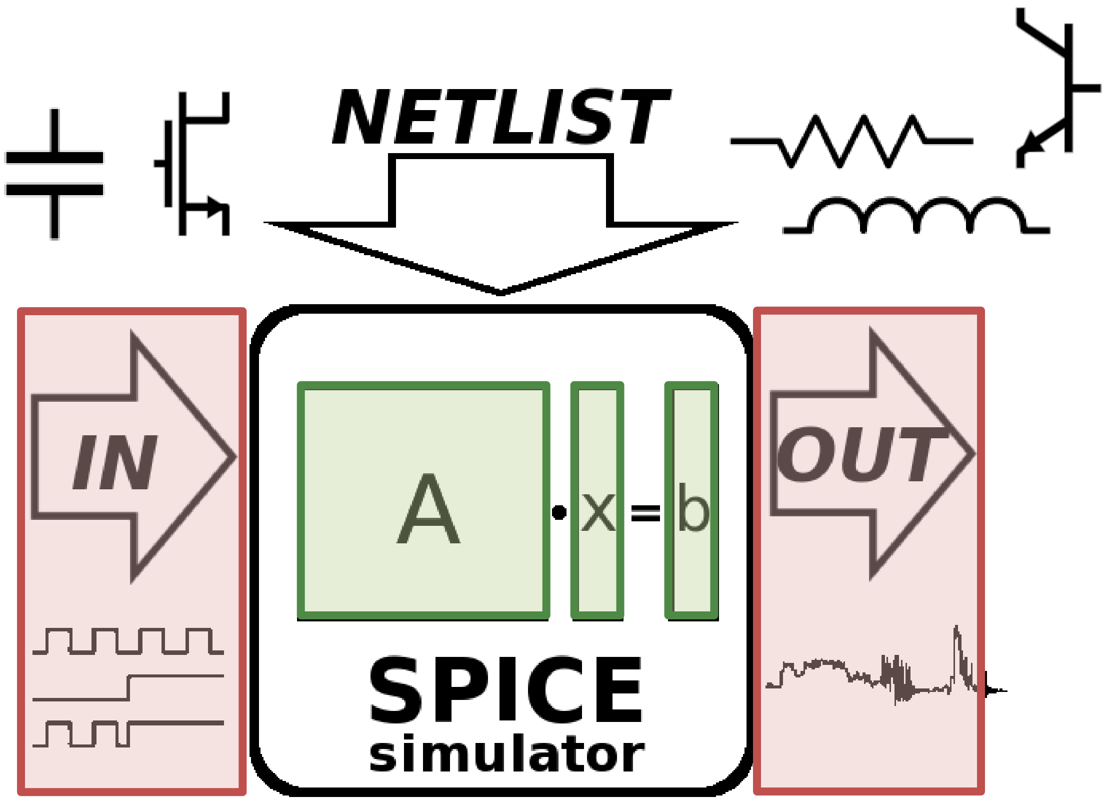
Simulation Trade-off

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



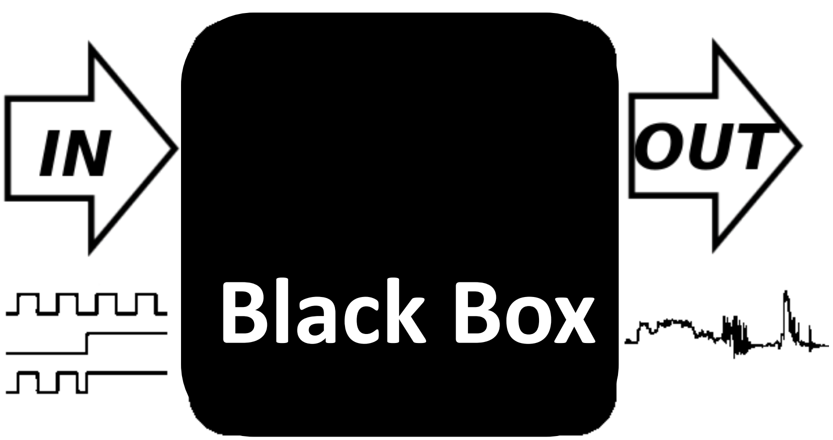
Which behavior?

- Input-Output behavior
- Dynamical equations

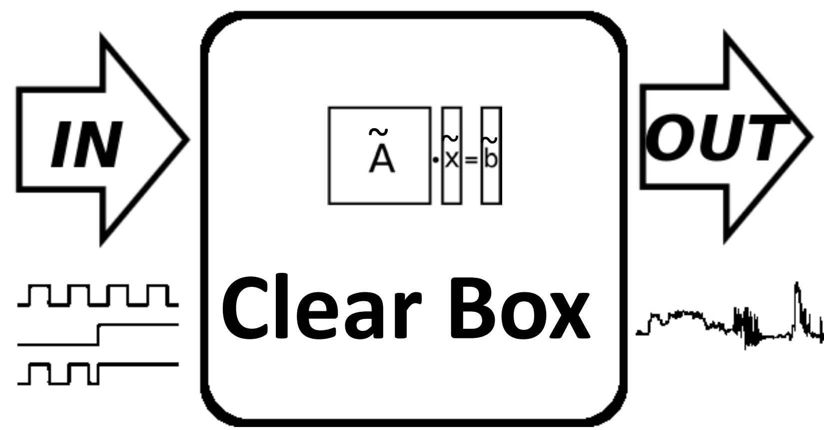


Which Model?

- Black Box
  - Only look at terminals
  - System **Identification**
  - *"Generic"*



- Clear Box
  - Internal representation
  - System **Reduction**
  - *"Natural"*

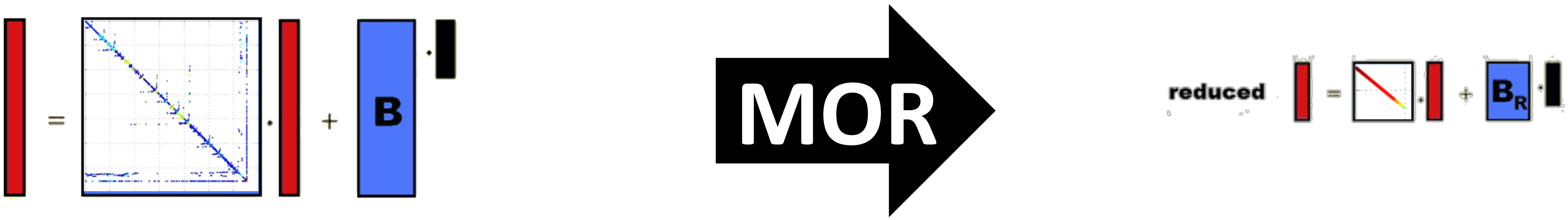


[Rutenbar-Gielen-Roychowdhury, Proc.IEEE 2007]

KATHOLIEKE UNIVERSITEIT LEUVEN

Reduction of *Linear* Systems

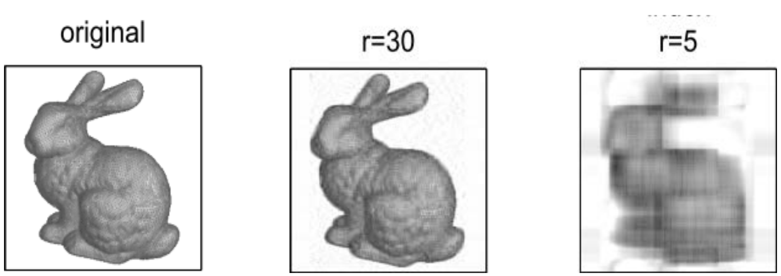
- **Model Order Reduction (MOR)**



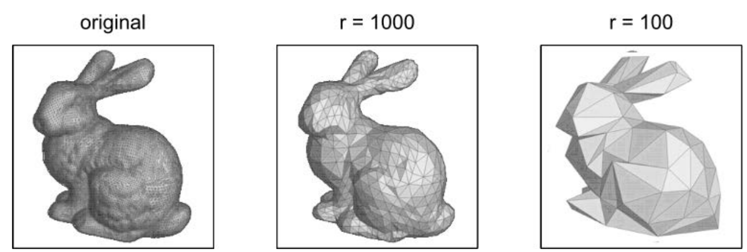
Principal Component Analysis

Eigenvalues, SVD, ... + Truncation

[A] = pixel array ➔ **JPEG**



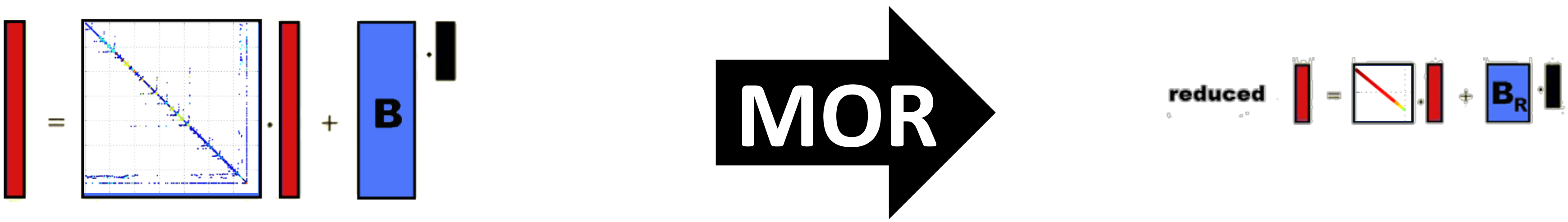
[A] = vertices ➔ **reduce mesh**



KATHOLIEKE UNIVERSITEIT LEUVEN

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)*

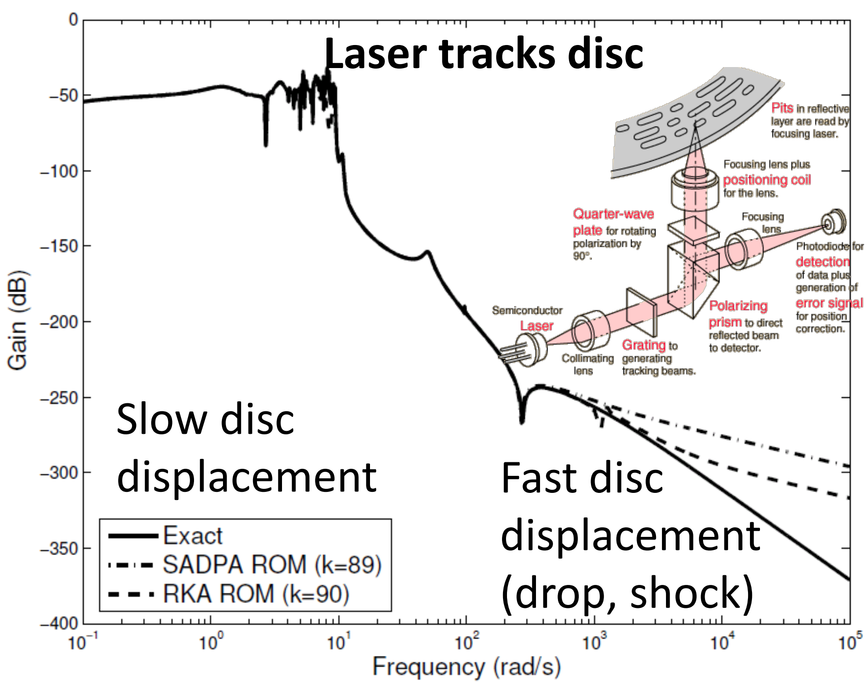
Vector Fitting *(Function approximation)*

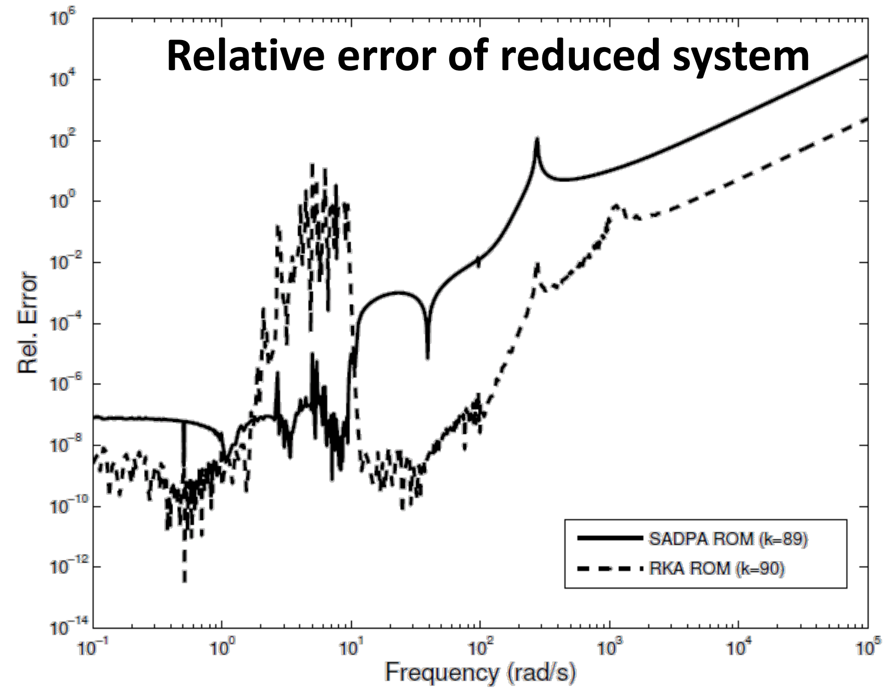
[Grimme '97, Odabasioglu '98, Phillips '05, Astrid '08, Deschrijver '08]

Reduction of *Linear* Systems

- CD player (optical control) : 480 → 90 eqns

Ability of laser to track disc



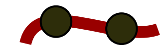
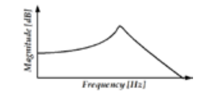


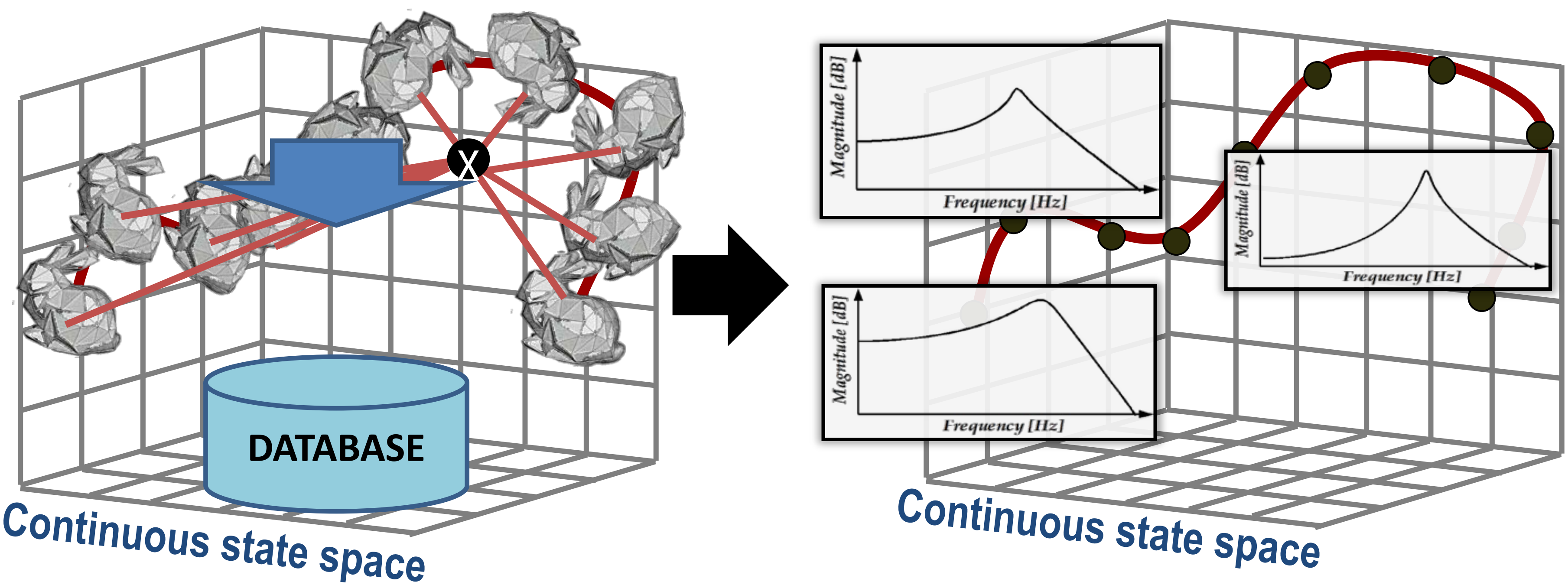
- Used in large (linear) system solvers (FEM, …)

⇨ **What about *nonlinear* systems?**

KATHOLIEKE UNIVERSITEIT LEUVEN

Reduction of *Nonlinear* Systems

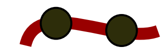
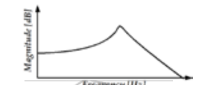
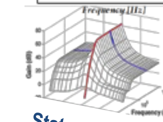
- **Trajectory PieceWise** 
  1. Sample & linearize states
  2. Reduce linearized systems 
  3. Interpolate reduced states 
- **Transfer Function Trajectories**
  1. Sample & linearize states 
  2. Frequency transform states 

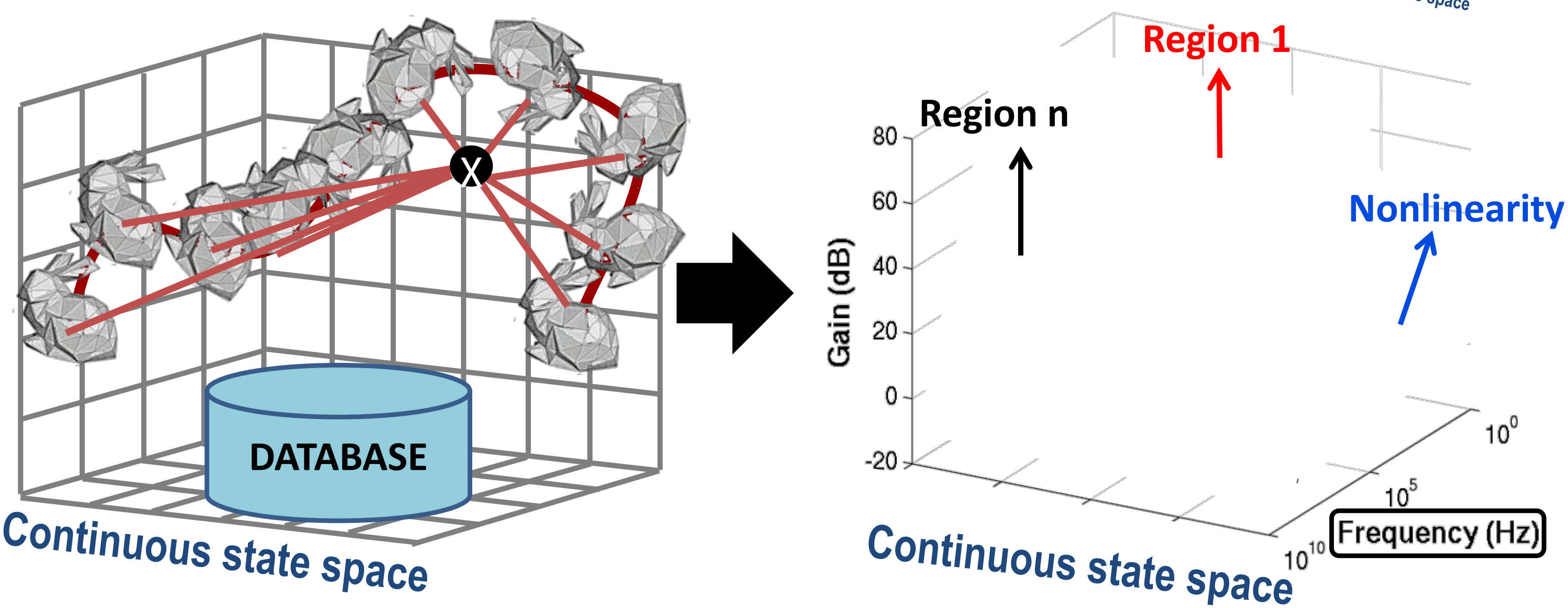


[Rewienski'03, Tiwary'06, Dong'08, Bond'11, DeJonghe'11/'12]

Reduction of *Nonlinear* Systems

- **Trajectory PieceWise**
  1. Sample & linearize states 
  2. Reduce linearized systems 
  3. Interpolate reduced states 

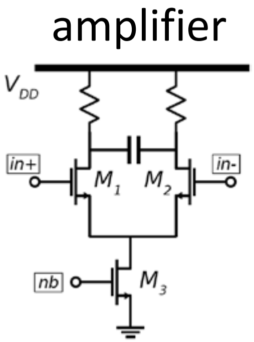
- **Transfer Function Trajectories**
  1. Sample & linearize states 
  2. Frequency transform states 
  3. *Fit surface with math* 



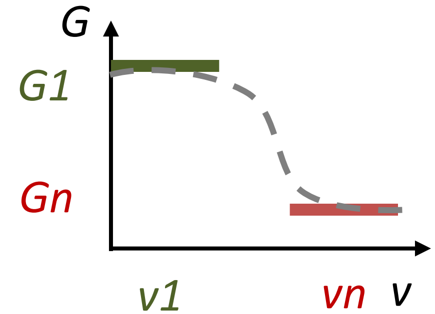
[Rewienski'03, Tiwary'06, Dong'08, Bond'11, DeJonghe'11/'12]

KATHOLIEKE UNIVERSITEIT LEUVEN

Reduction of *Nonlinear* Systems



- *[C(v)]* and *[G(v)]* vary with *state of v(t)*
  - If *(v == v1)* ➔ state *1*
  - If *(v == vn)* ➔ state *n*

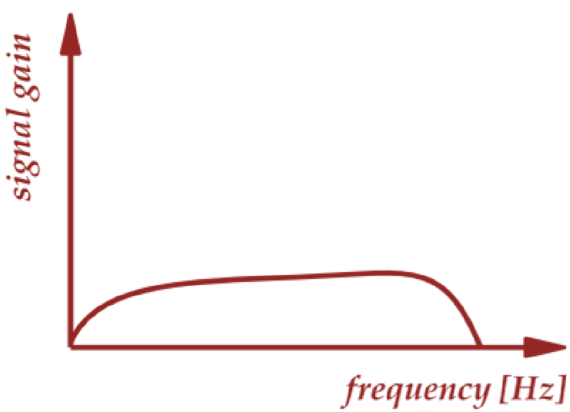


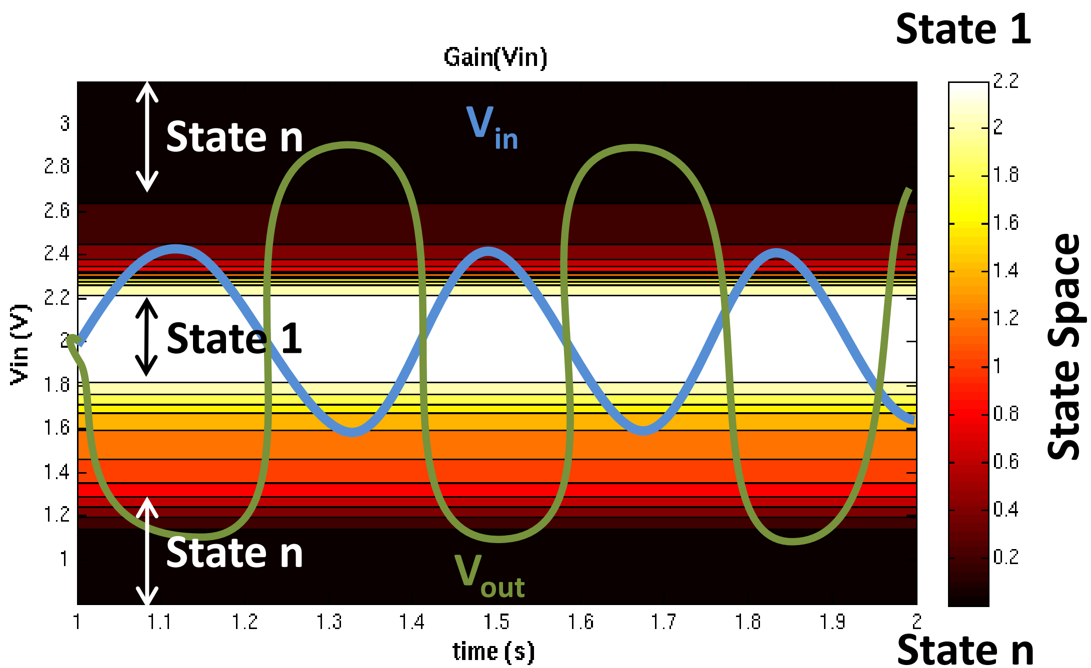
$$[C(\vec{v}(t))]d\vec{v}(t) + [G(\vec{v}(t))] = \vec{\imath}(t)$$

$$[C1]d\vec{v}(t) + [G1]\vec{v}(t) = \vec{\imath}(t)$$

$$\vdots$$

$$[Cn]d\vec{v}(t) + [Gn]\vec{v}(t) = \vec{\imath}(t)$$

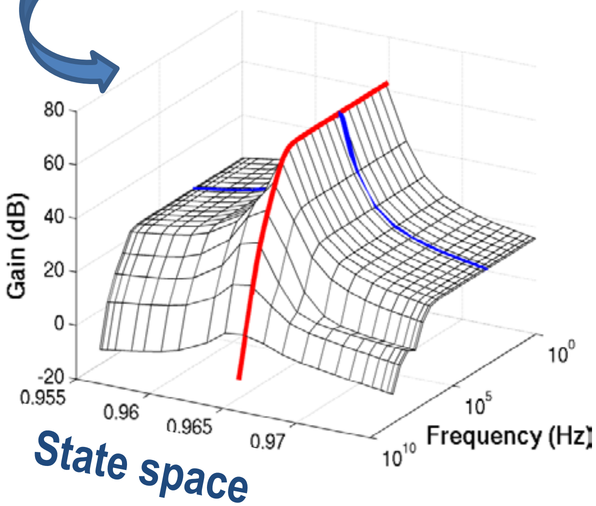




Model nonlinearity and memory effects

Data

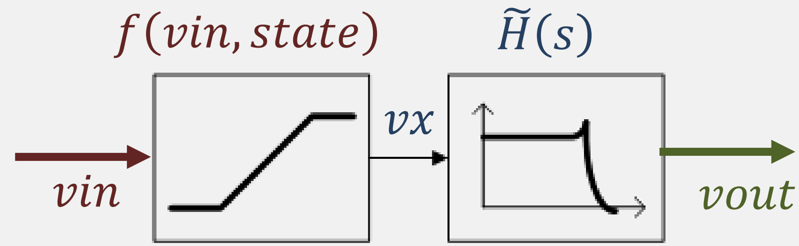
$$[C(\vec{v}(t))]d\vec{v}(t) + [G(\vec{v}(t))] = \vec{i}(t)$$



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

fitting

Model



1. <u>*Nonlinear function*</u>

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

2. <u>*Transfer function (memory)*</u>

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

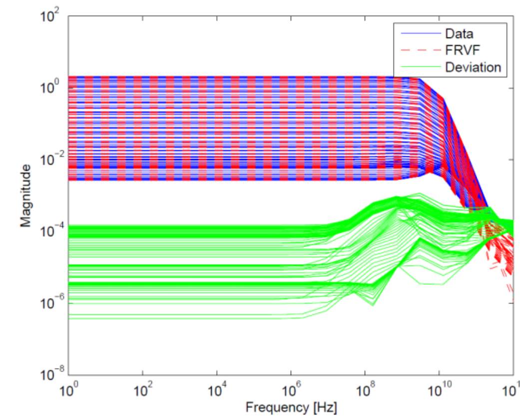
internal delay

[De Jonghe & Gielen, TCAS-I 2012]

Fitting for nonlinear systems

1. Fit the frequency function

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

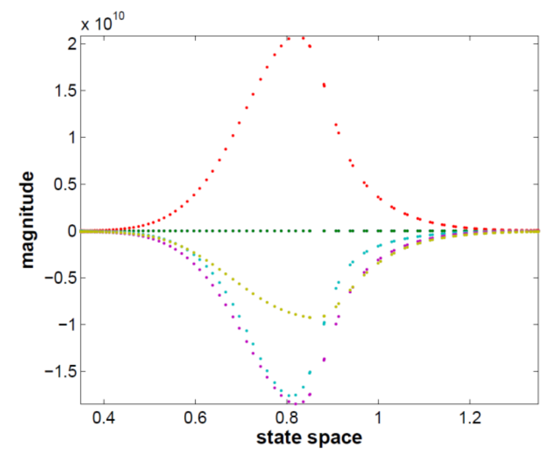


→ **Vector Fitting** Algorithm:

Optimizes the internal delays a_p of the model

2. Fit the nonlinear function(s)

$$vx = f(vin, state)$$



→ **Machine Learning** (Classifiers, Regressors)

Optimally fit a given set of data with a mathematical function

Neural Networks, Support Vector Machines,

Nearest Neighbours, Evolutionary algorithms, …

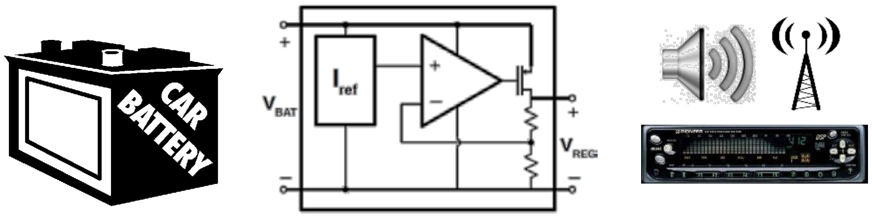
The proof is in the pudding…

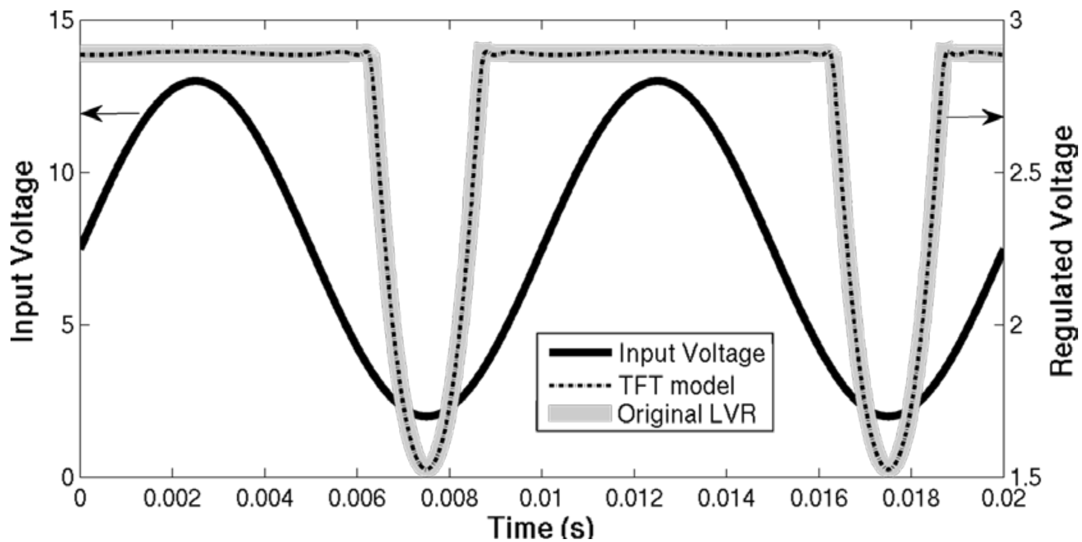
PRACTICAL APPLICATIONS

Application Example (1)

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

Linear Voltage Regulator

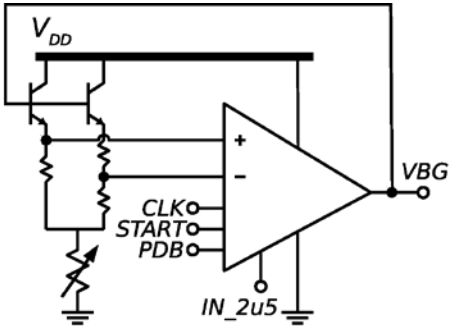


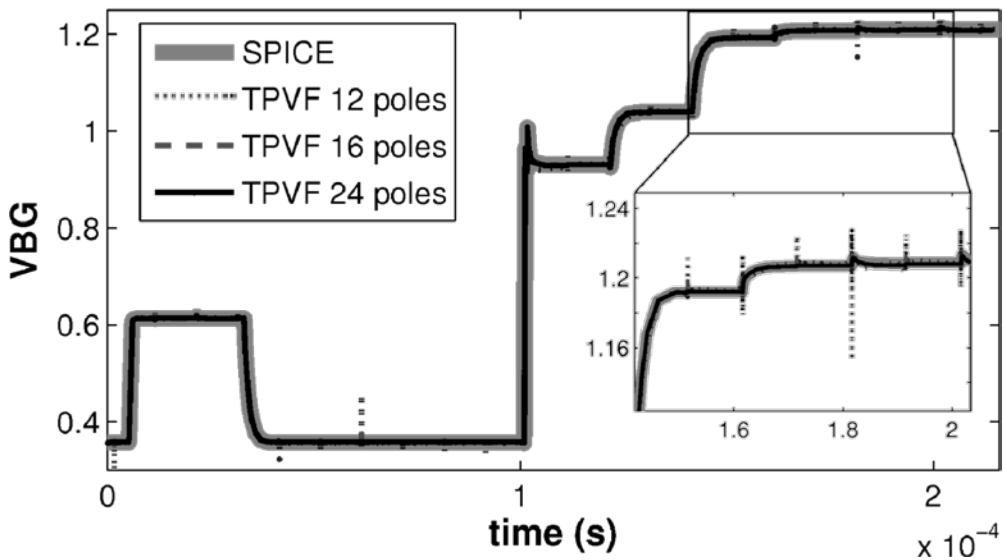


Application Example (2)

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

Absolute voltage reference (1.2V)

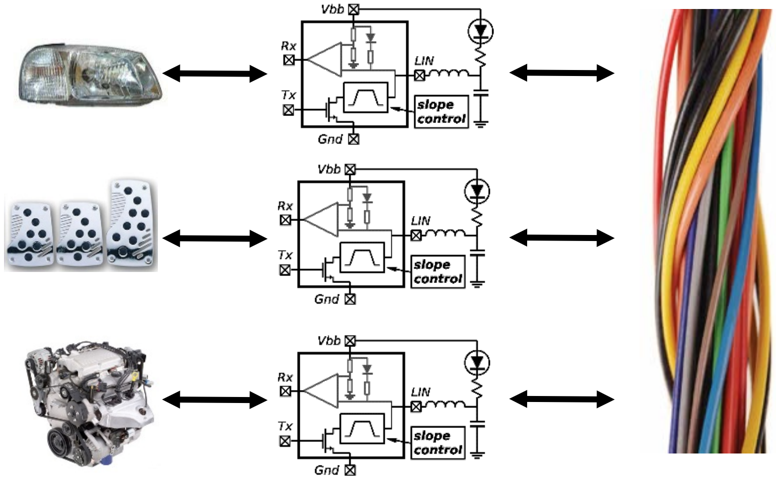


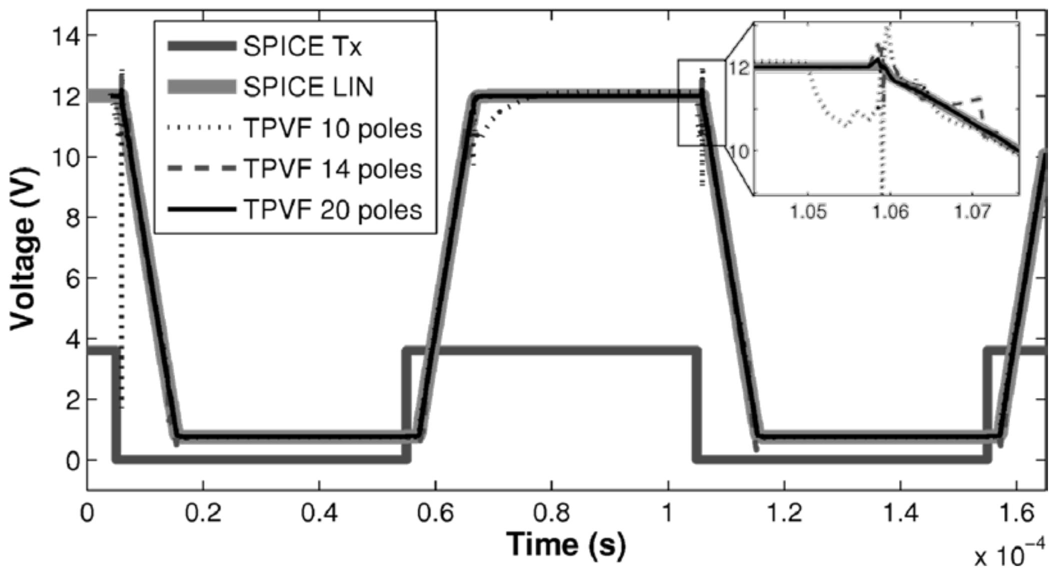


Application Example (3)

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

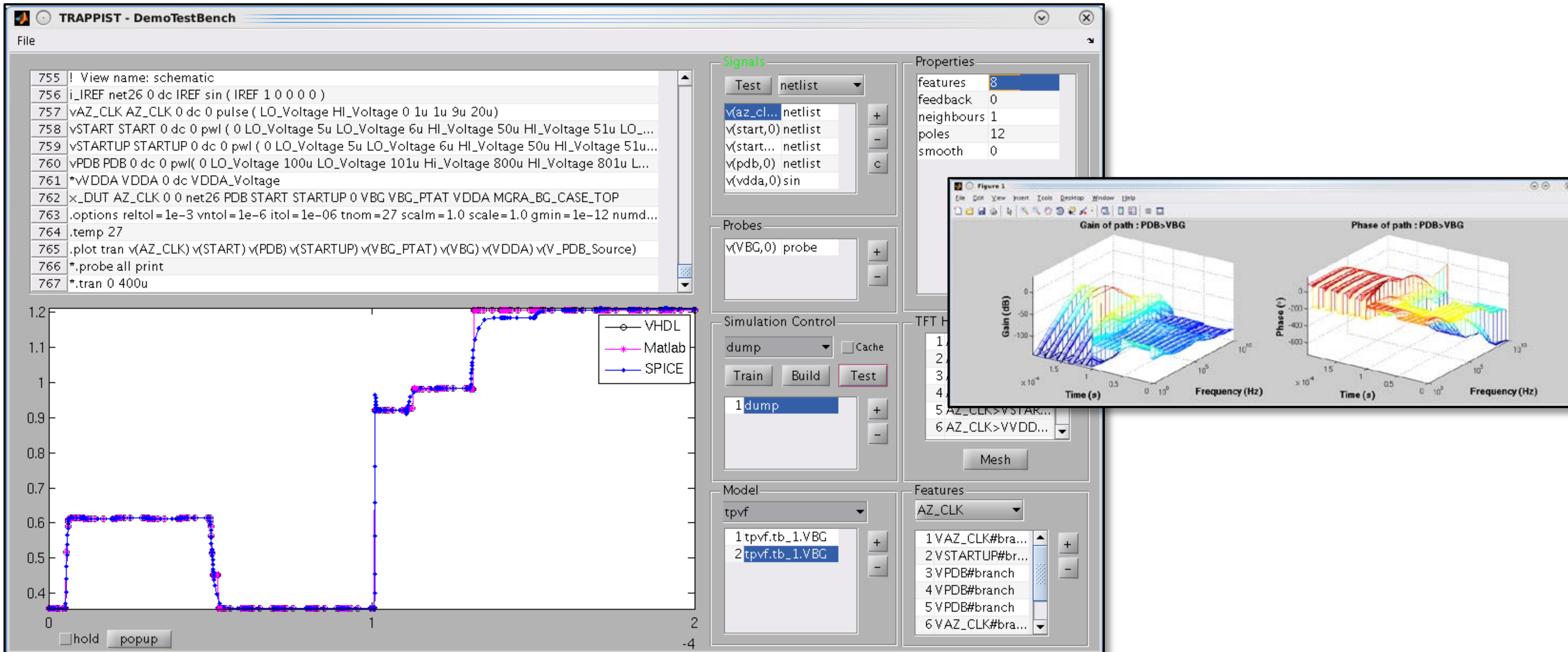
LIN transceiver





GUI: TRAPPIST

- **TR**ajectory **AP**proximation by **P**iecewise **I**nterpolation of **S**tate-dependent **T**ransfer Functions



Conclusion & Challenges

- **Virtual design environments**
  - Design of ***complex*** structures and systems
  - ***Accurate*** physical models are ***expensive***

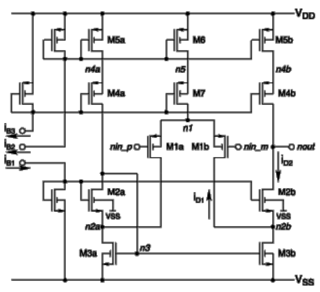
  → <u>Model order reduction</u>

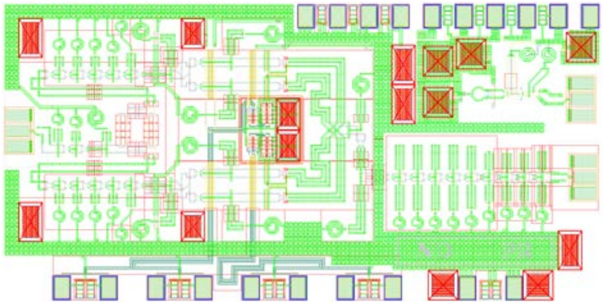
- **Challenges remain**
  - ***Multiphysics*** (MEMS): EM + stress + heat + flow +…
  - ***Stochastic*** systems (nano): [C, G] → μ[C,G], σ[C,G]
  - ***Extremely high frequencies***: EMC, mmWave, …
  - **…**

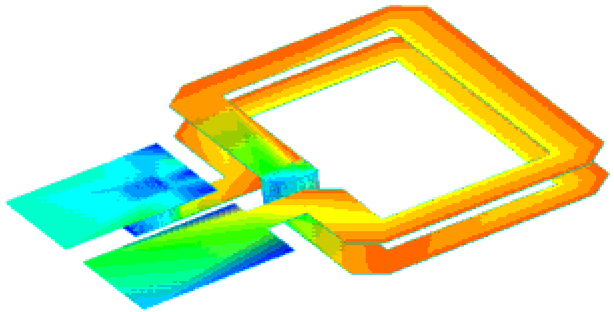
OPTIMIZATION OF COMPLEX STRUCTURES

Design issues

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



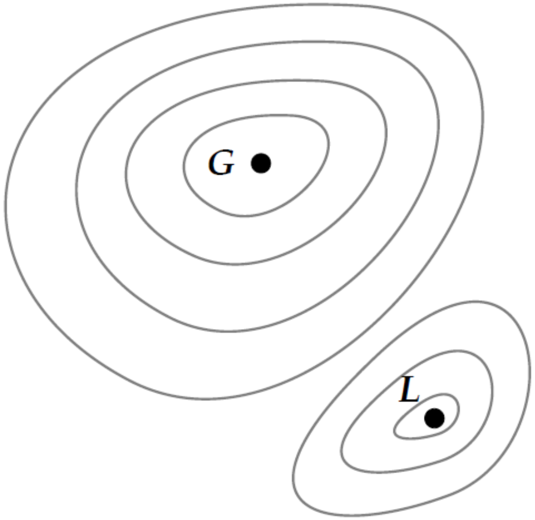




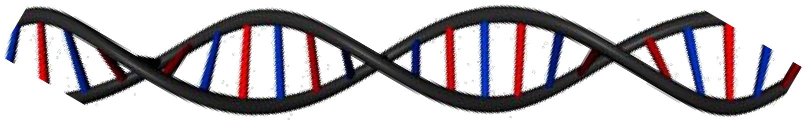
- Complex trade-offs: *accurate (expensive)* models
- ***Automate*** *design (partially) by*
  - *Efficient* ***optimization***
  - *Automated* ***synthesis*** of structures

Optimization algorithms

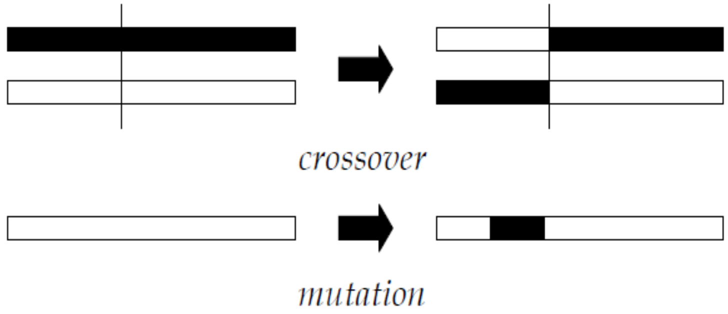
- Local unconstrained optimizers
  - e.g., gradient- or Newton-based approaches
- Constrained optimization
  - solve linear or nonlinear program
  - special case: Geometric Program
- Greedy stochastic algorithms
  - only improvements are allowed
- Annealing approaches
  - also up-hill moves can occur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



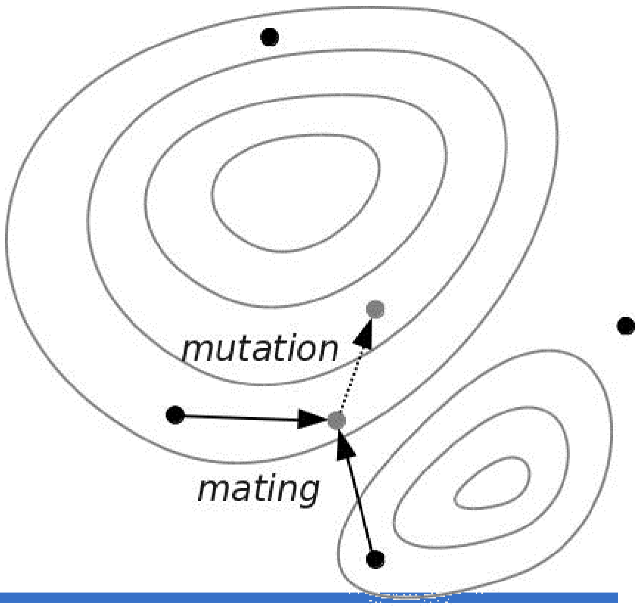
Evolutionary Algorithms (EA)



- Mimic *evolutionary biology*
  - *Inheritance, mutation, selection, crossover*
  - Variables ➔ genetic material
    Objective ➔ fitness

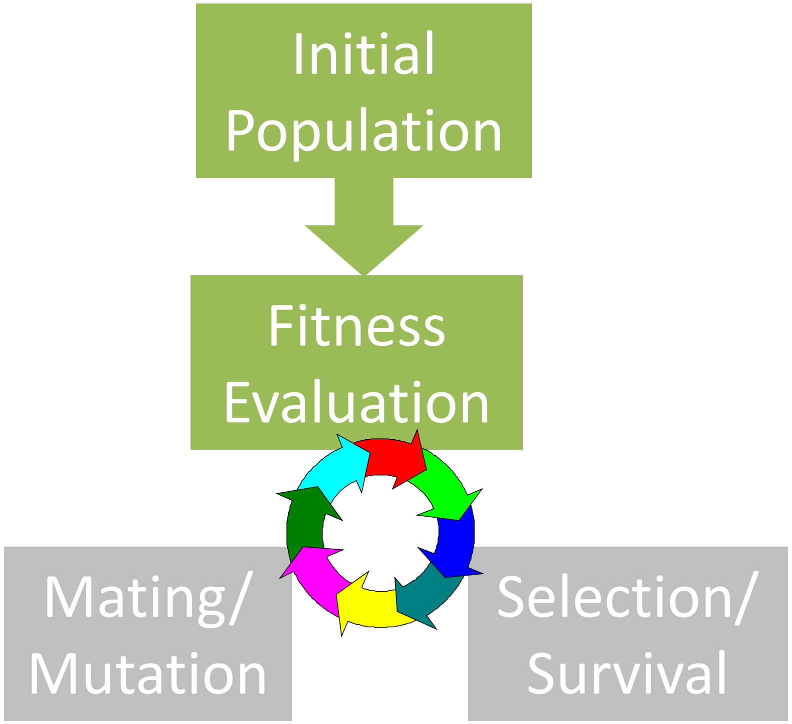


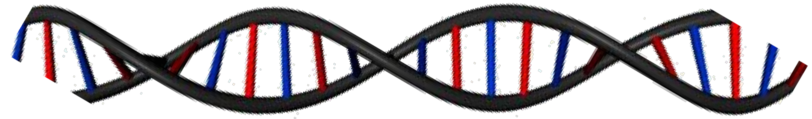
- Global optimizer
  - If population large enough
  - *Randomness* in each generation
  - *Convergence* in each generation

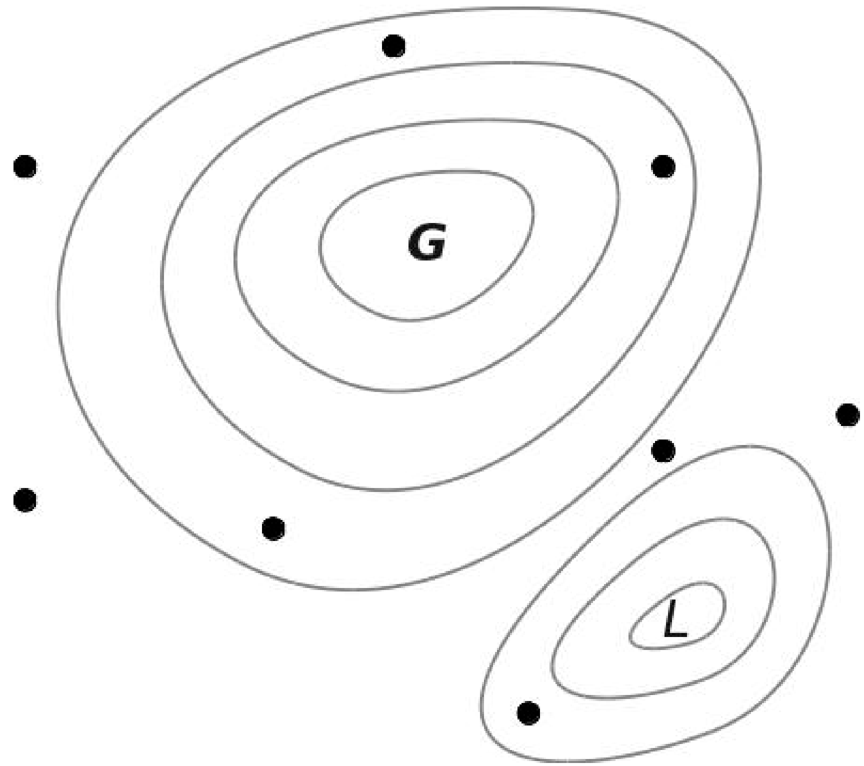


EA: Random Initialization

- Evolutionary loop

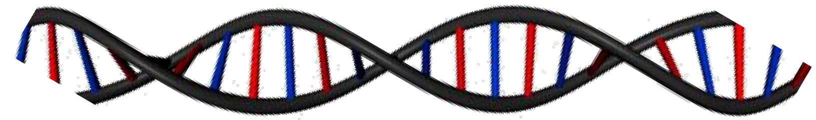


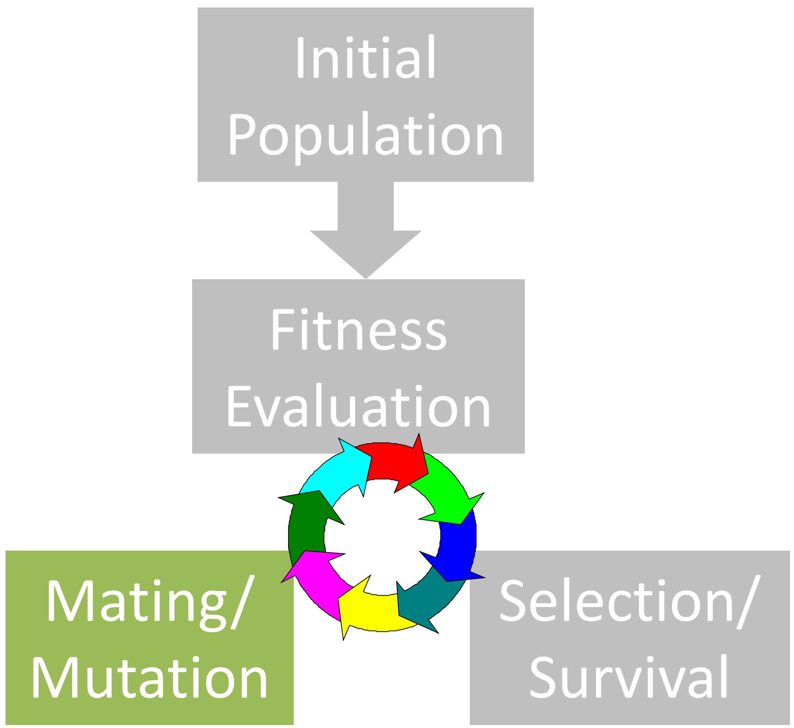


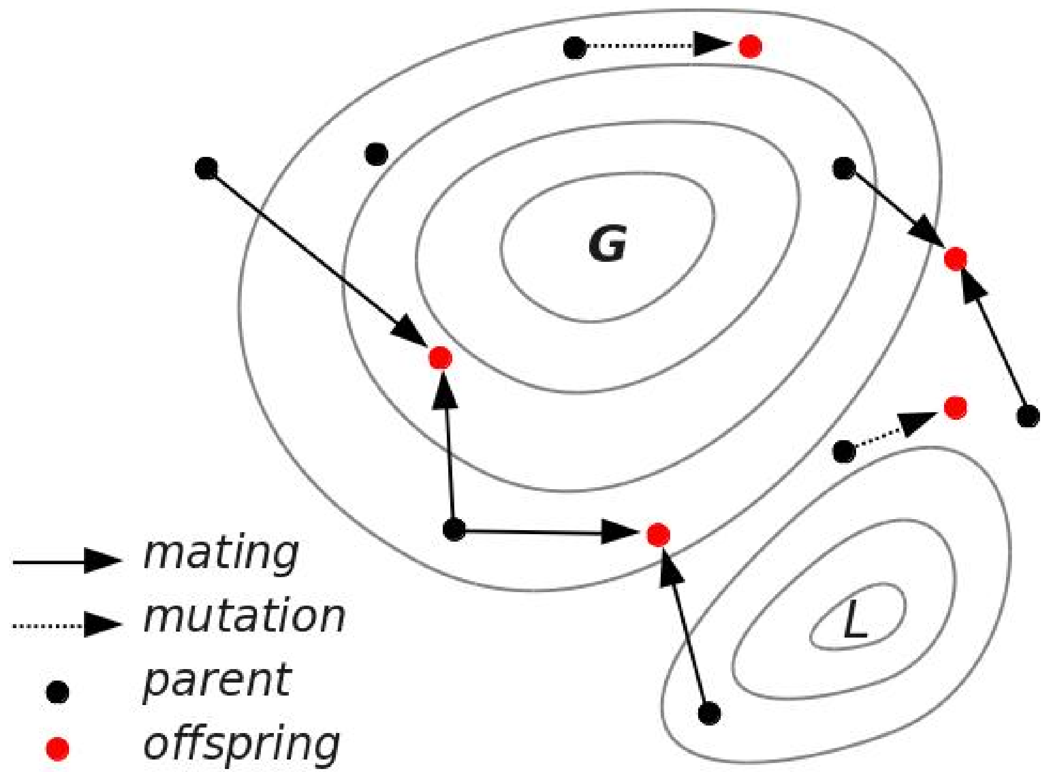


EA: Mating and Mutation

- Evolutionary loop

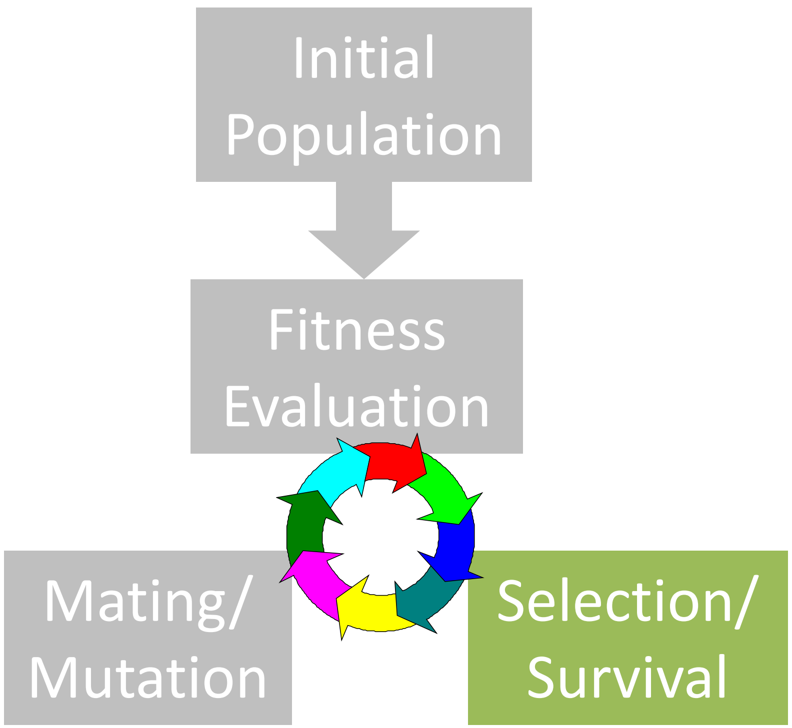


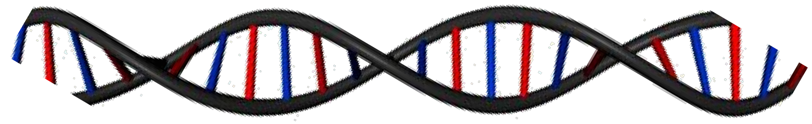


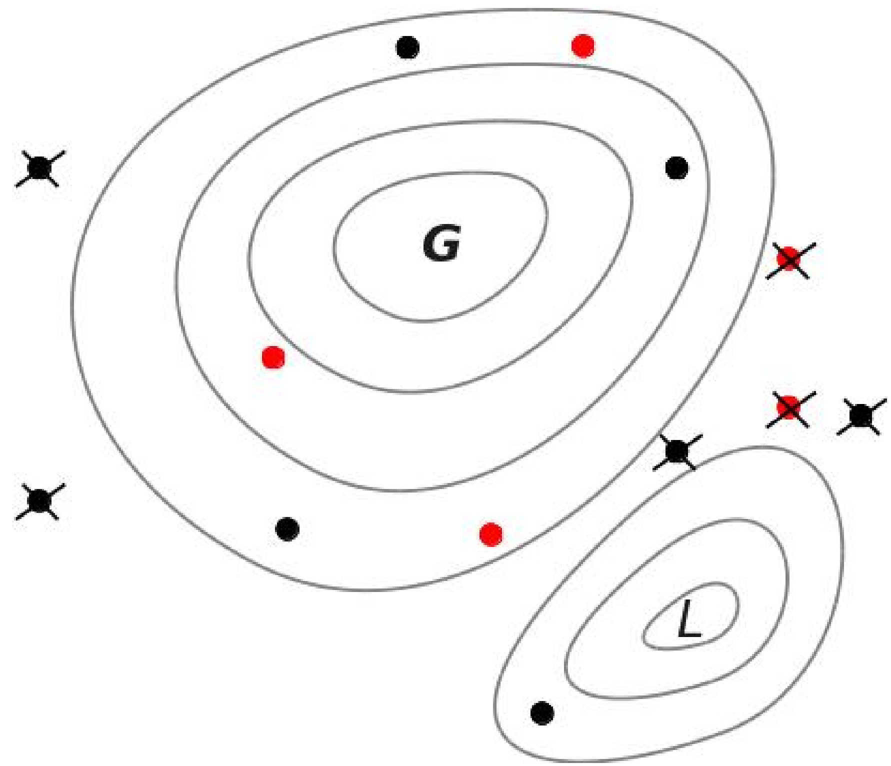


EA: Survival of the fittest

- Evolutionary loop

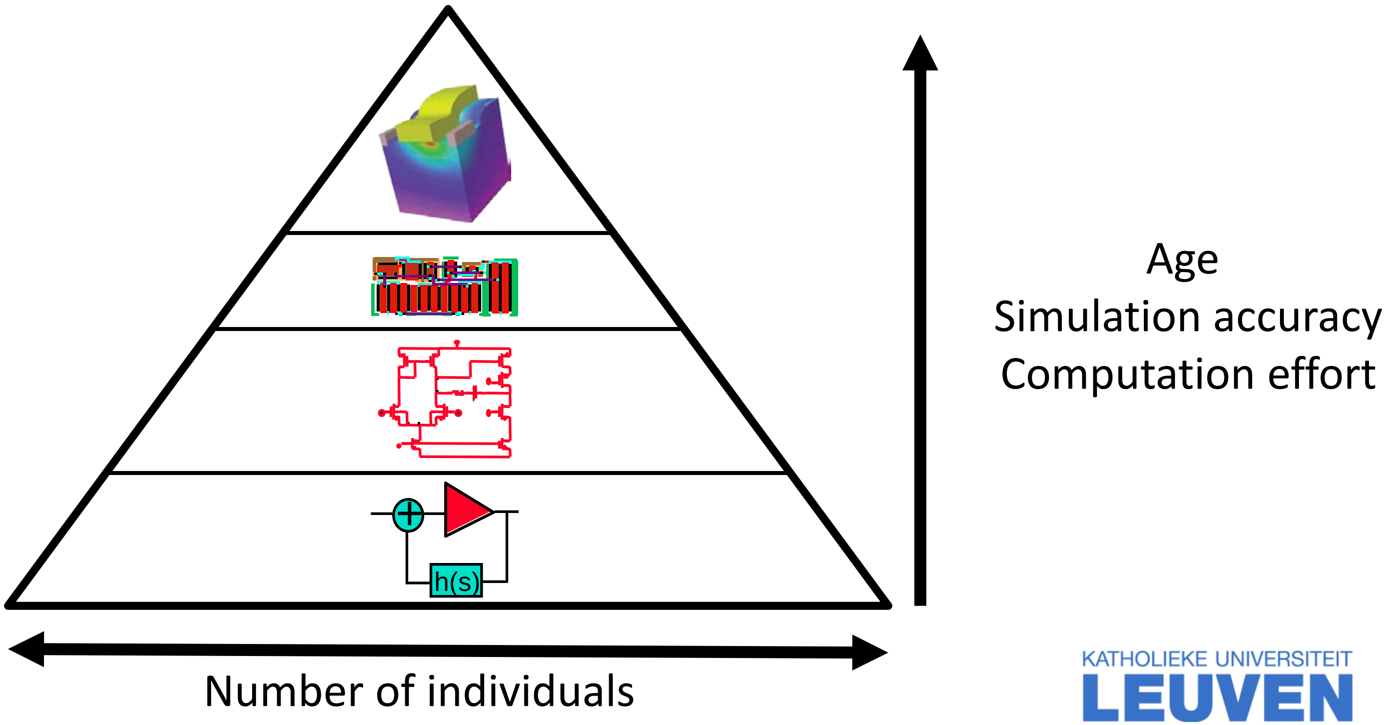






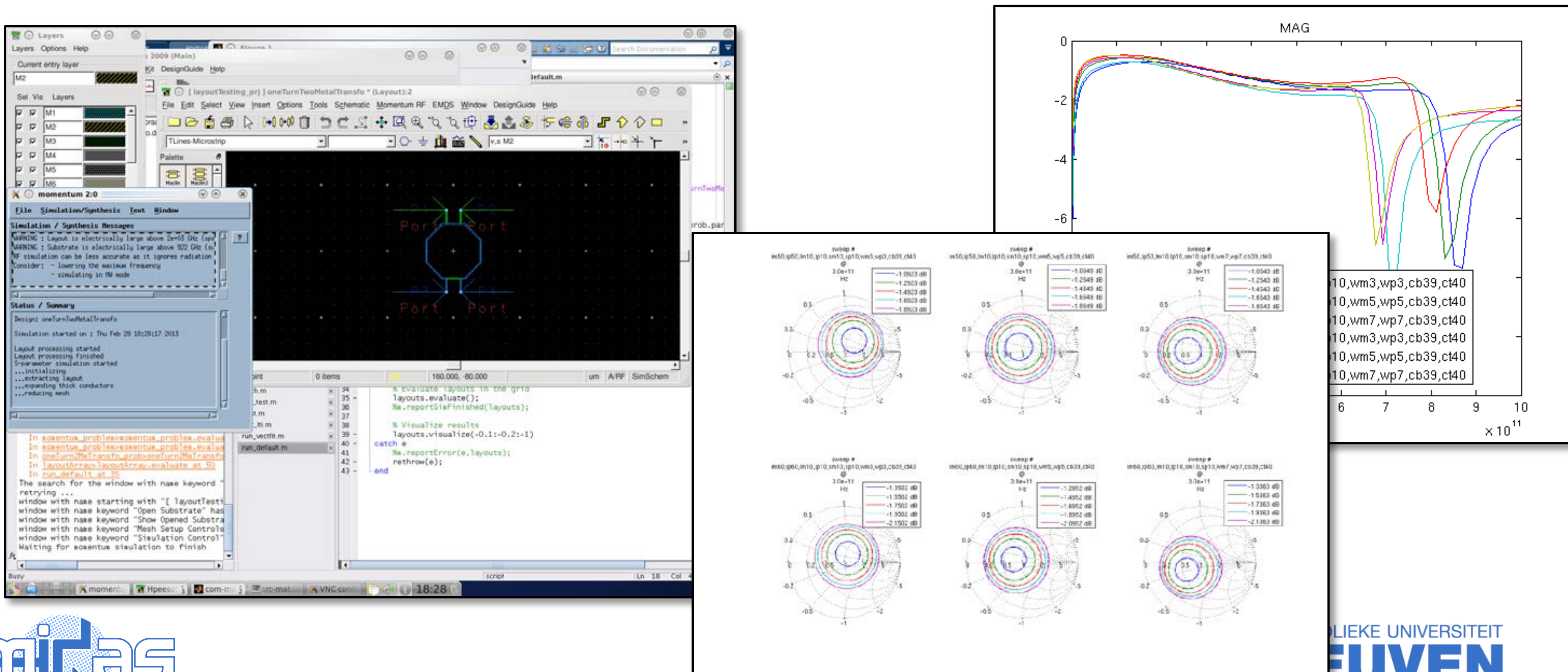
ALPS: Age layered population structure

- 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 convertors
  - Synthesize, simulate and optimize



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…