High-Dimensional Statistical Modeling and Analysis Of Custom Integrated Circuits

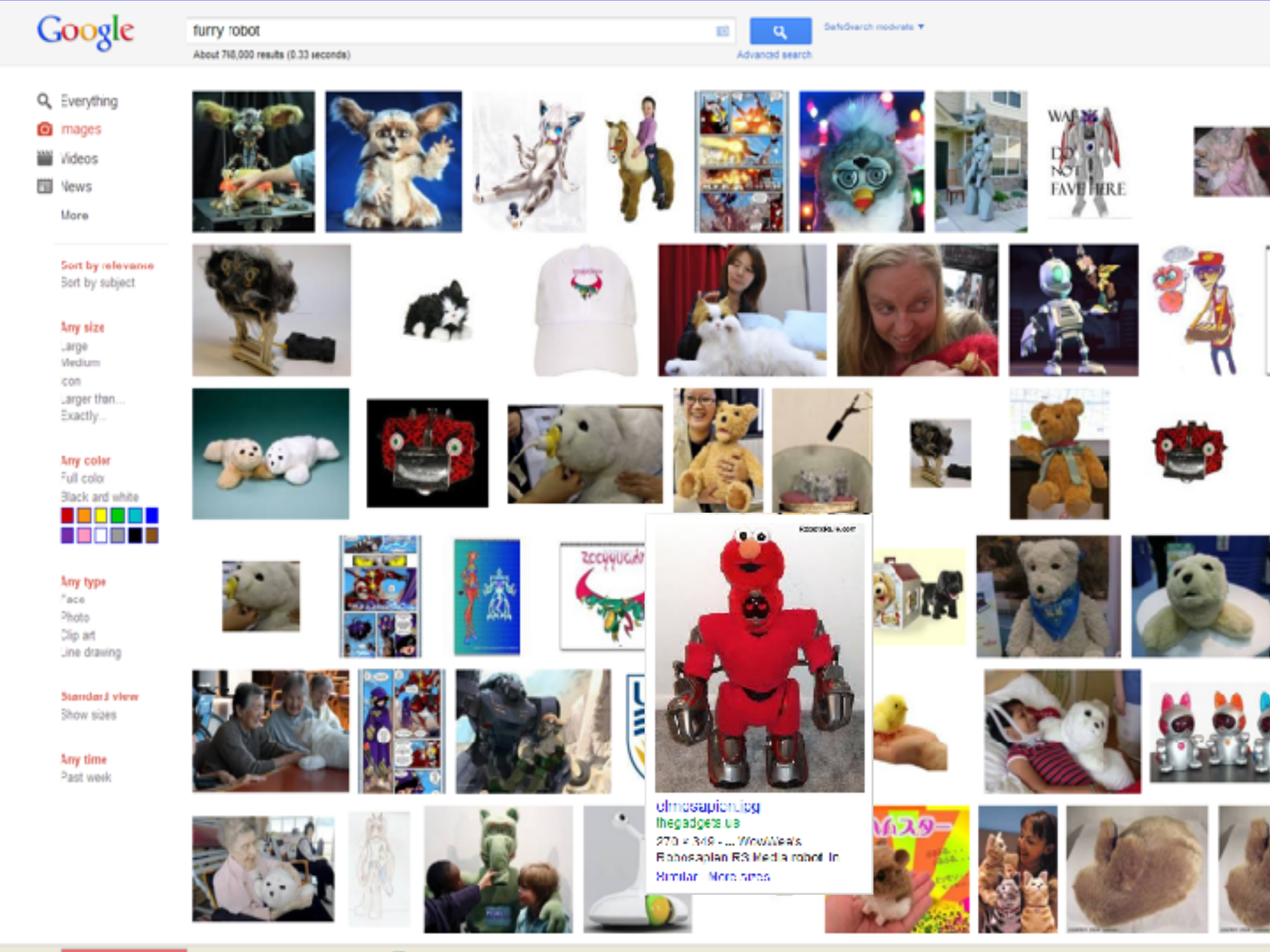
Trent McConaghy

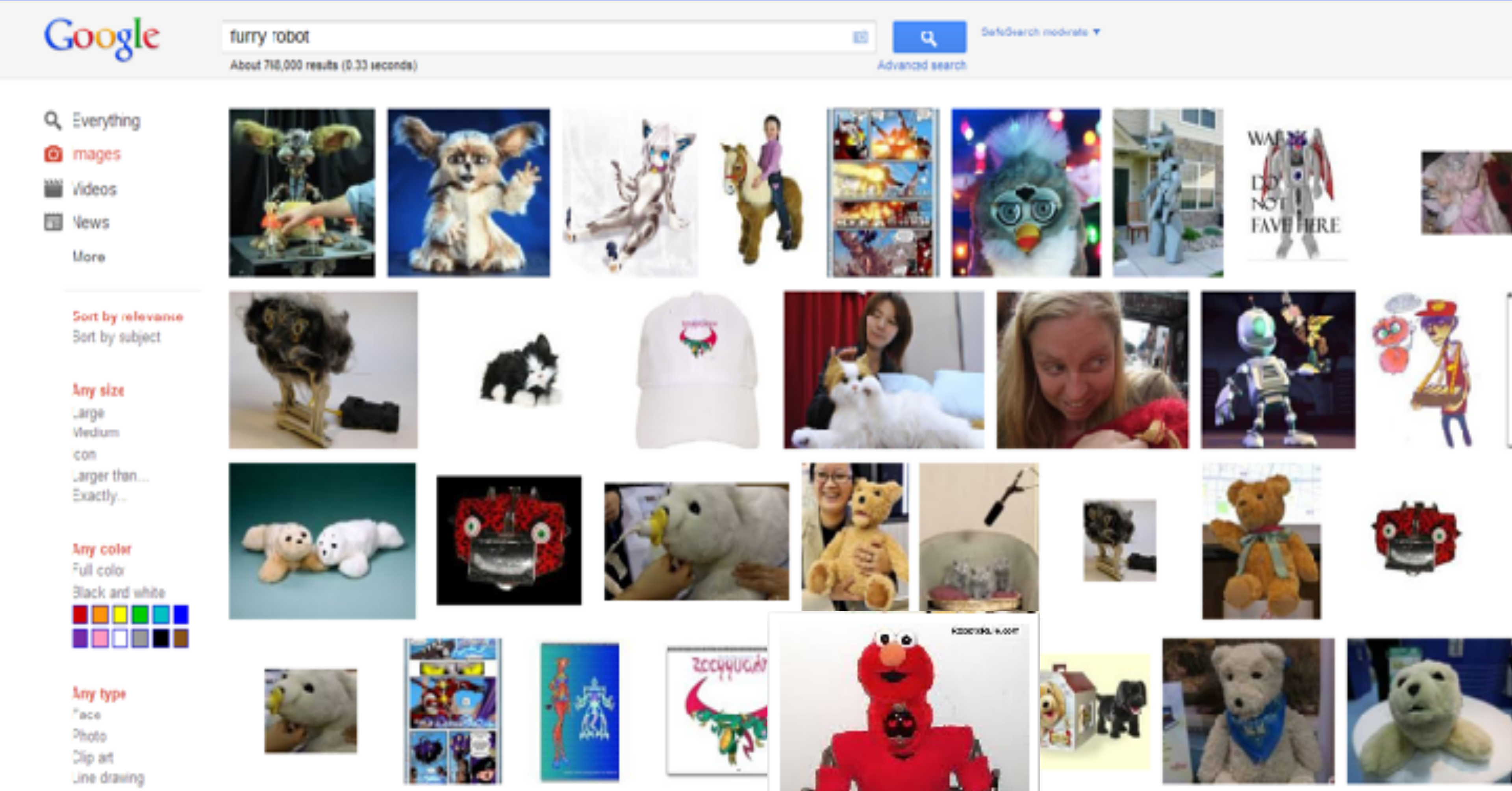
Co-founder & CSO, Solido Design Automation

Custom Integrated Circuits Conference (CICC)

San Jose, CA, Sept 2011


Google
images
furry robot
Search Images


Google
furry robot
About 718,000 results (0.33 seconds)
Advanced search
SafeSearch moderate
Everything
Images
Videos
News
More
Sort by relevance
Sort by subject
Any size
Large
Medium
Icon
Larger than...
Exactly...
Any color
Full color
Black and white
Any type
Face
Photo
Clip art
Line drawing
Standard view
Show sizes
Any time
Past week
elmosapien.jpg
thegadgets.us
Robosapien RS Media robot In
Similar
More sizes



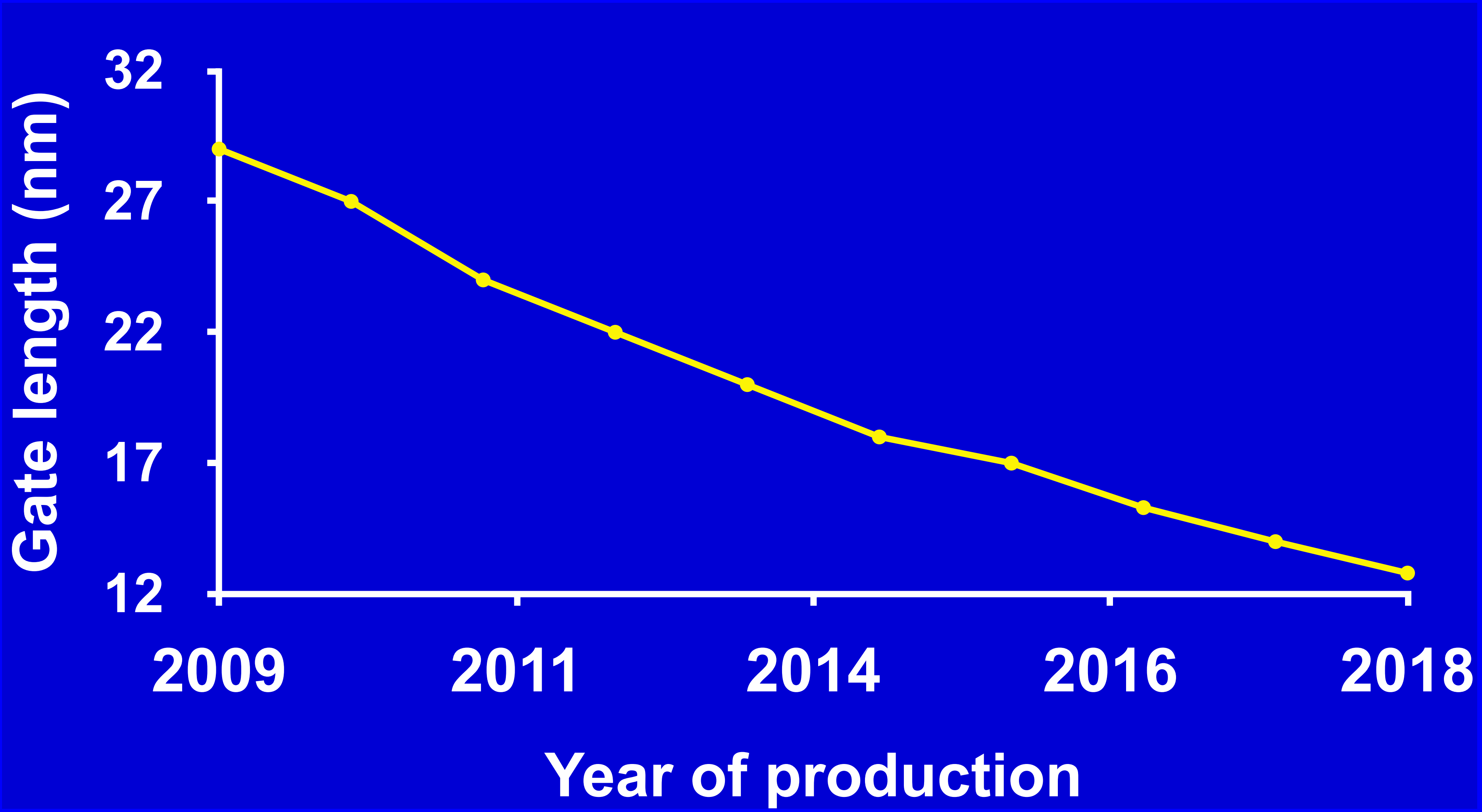
How *does* Google find furry robots?

(Not the aim of this talk, but we'll find out…)

Outline

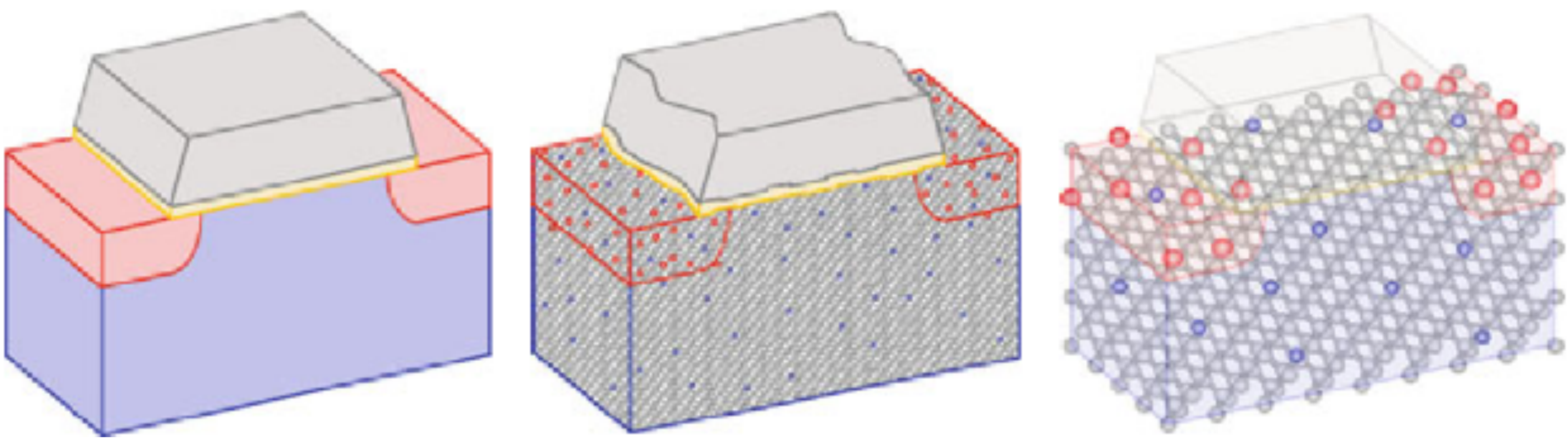
- <u>Motivation</u>
- Proposed flow
- Backgrounder
- Fast Function Extraction (FFX)
- Results
- Conclusion

With Moore's Law, Transistors Are Shrinking…



[ITRS, 2011]

Transistors are shrinking
But atoms aren't!

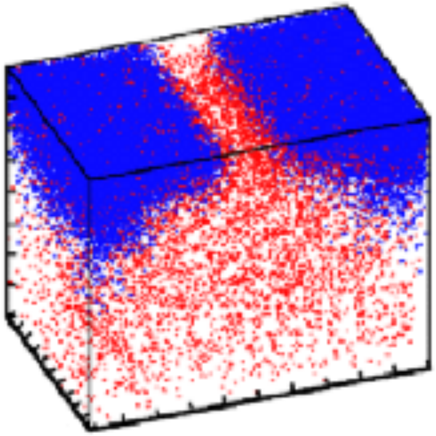


traditional 22nm sub 10 nm

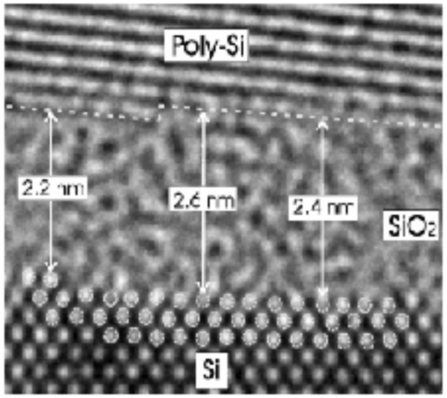
A. Asenov, "Statistical Nano CMOS Variability and Its Impact on SRAM", Chapter 3, A. Singhee and R. Rutenbar, Eds., Extreme Statistics in Nanoscale Memory Design, Springer, 2010

Variation = atoms out of place

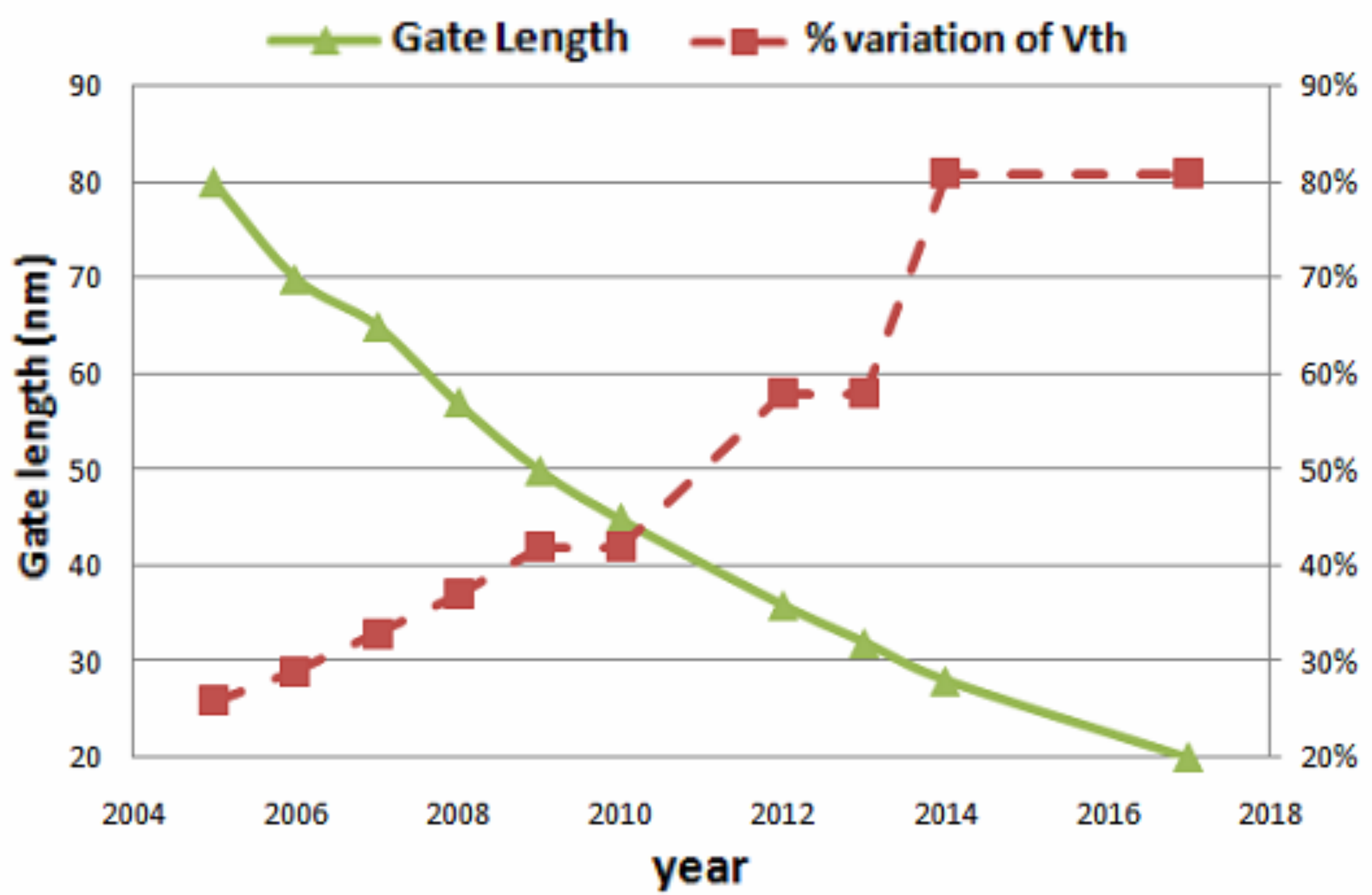
Relative variation worsens as transistors shrink



Random dopant effects



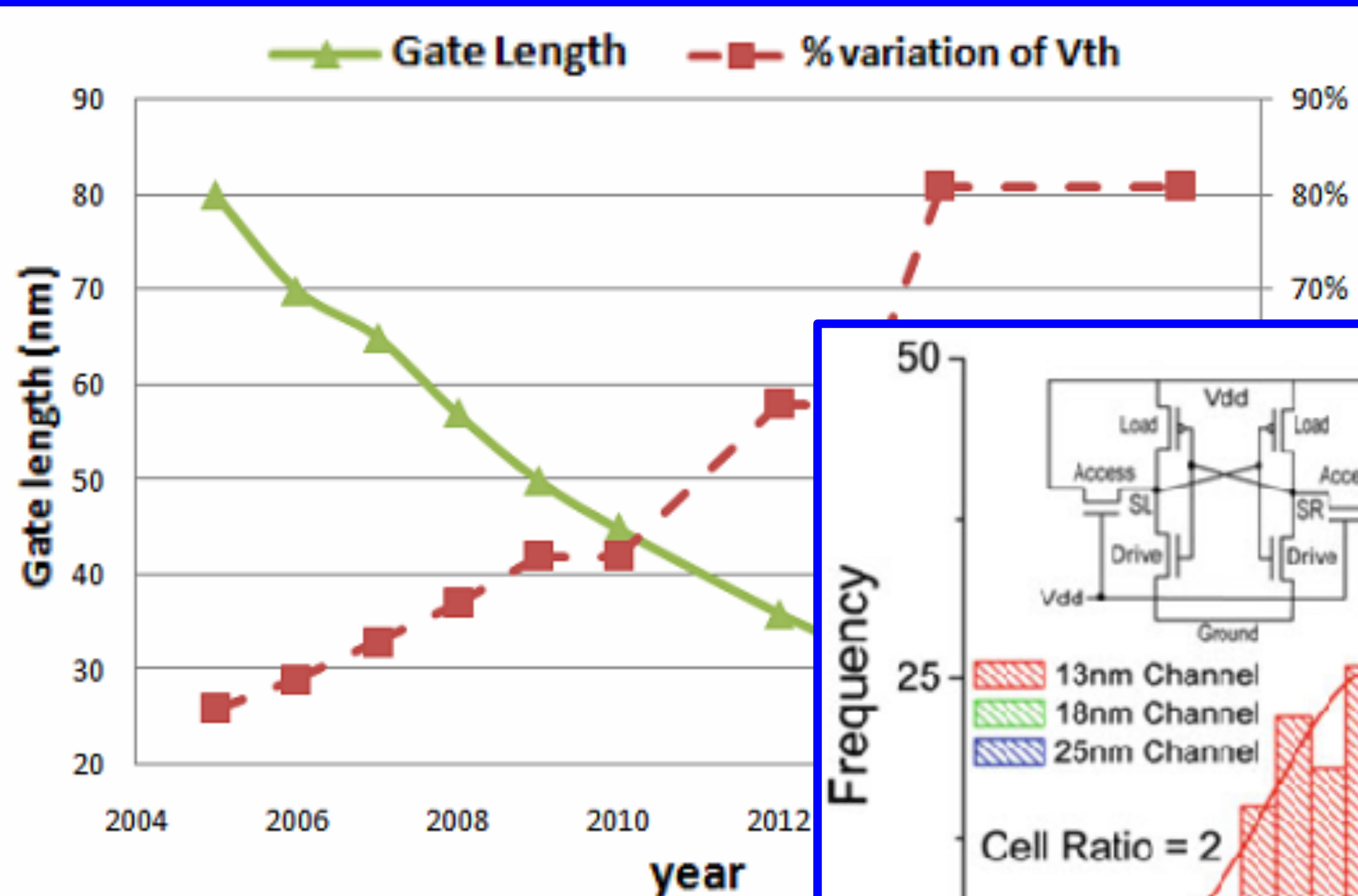
Oxide thickness

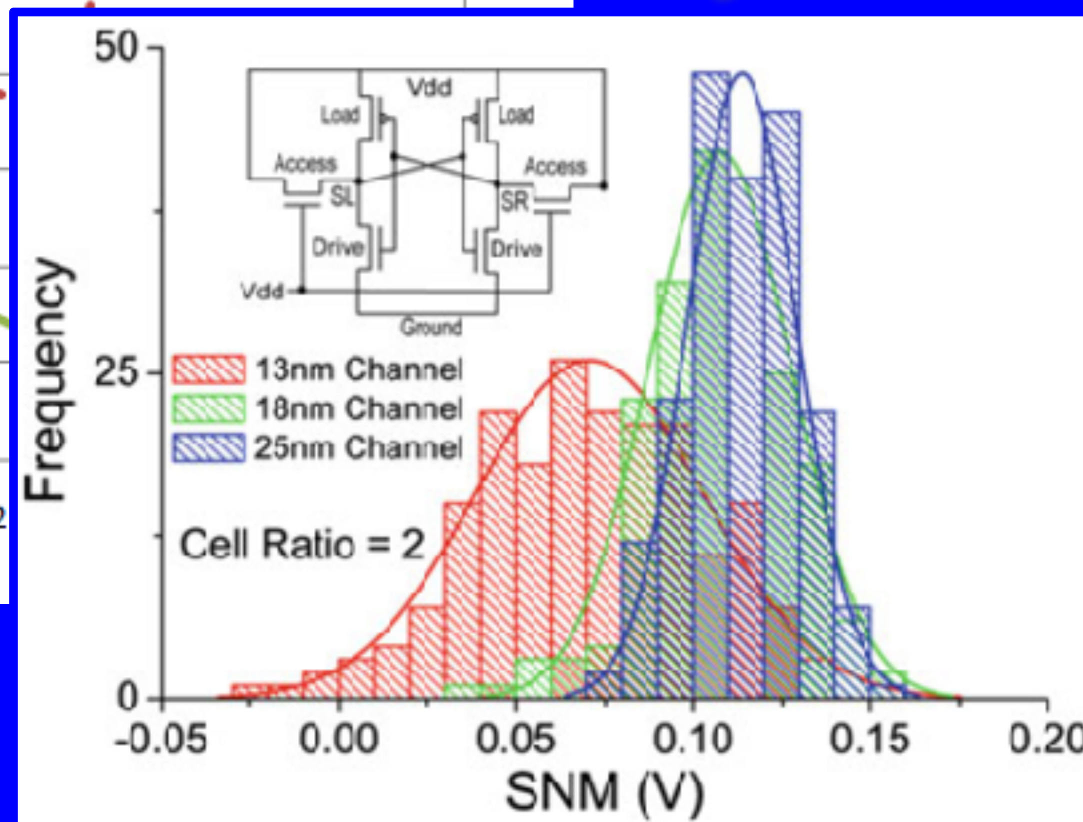


⋮

-A. Asenov, "Statistical Nano CMOS Variability and Its Impact on SRAM", Extreme Statistics, Springer, 2010
-C. Visweswariah, “Mathematics and Engineering: A Clash of Cultures?”, IBM, 2005
-ITRS, 2006

Higher device variation
→ Higher performance variation
→ Lower yield





Typical Custom Design Flow

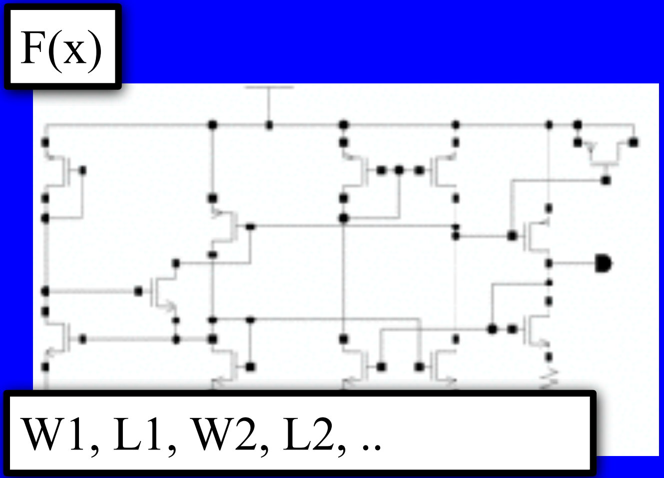
(Manual) develop eqns. of design → performance

Early-stage design (topology, initial sizing)

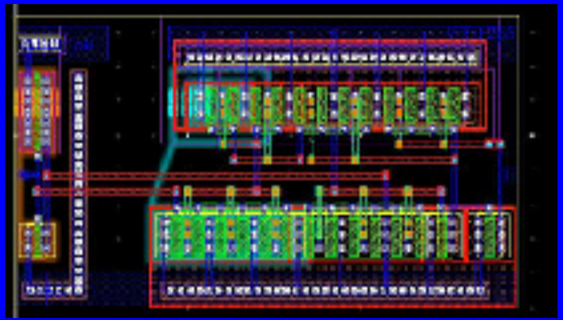
Tune sizings for performance

Layout

⋮



W1, L1, W2, L2, ..



Variation-Aware Design Flow

(Manual) develop eqns. of design → performance

Early-stage design (topology, initial sizing)

Tune sizings for perf. & *yield*

Layout

⋮

Variation-Aware Approaches:

- Direct Monte Carlo
- Design-specific 3σ corners
- …

Direct Monte Carlo (MC) For Variation-Aware Design

(Manual) develop eqns. of design → performance

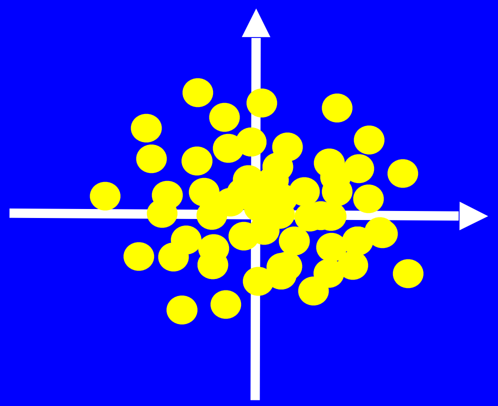
Early-stage design (topology, initial sizing)

Tune sizings for perf. & *yield* *via direct MC*

Layout

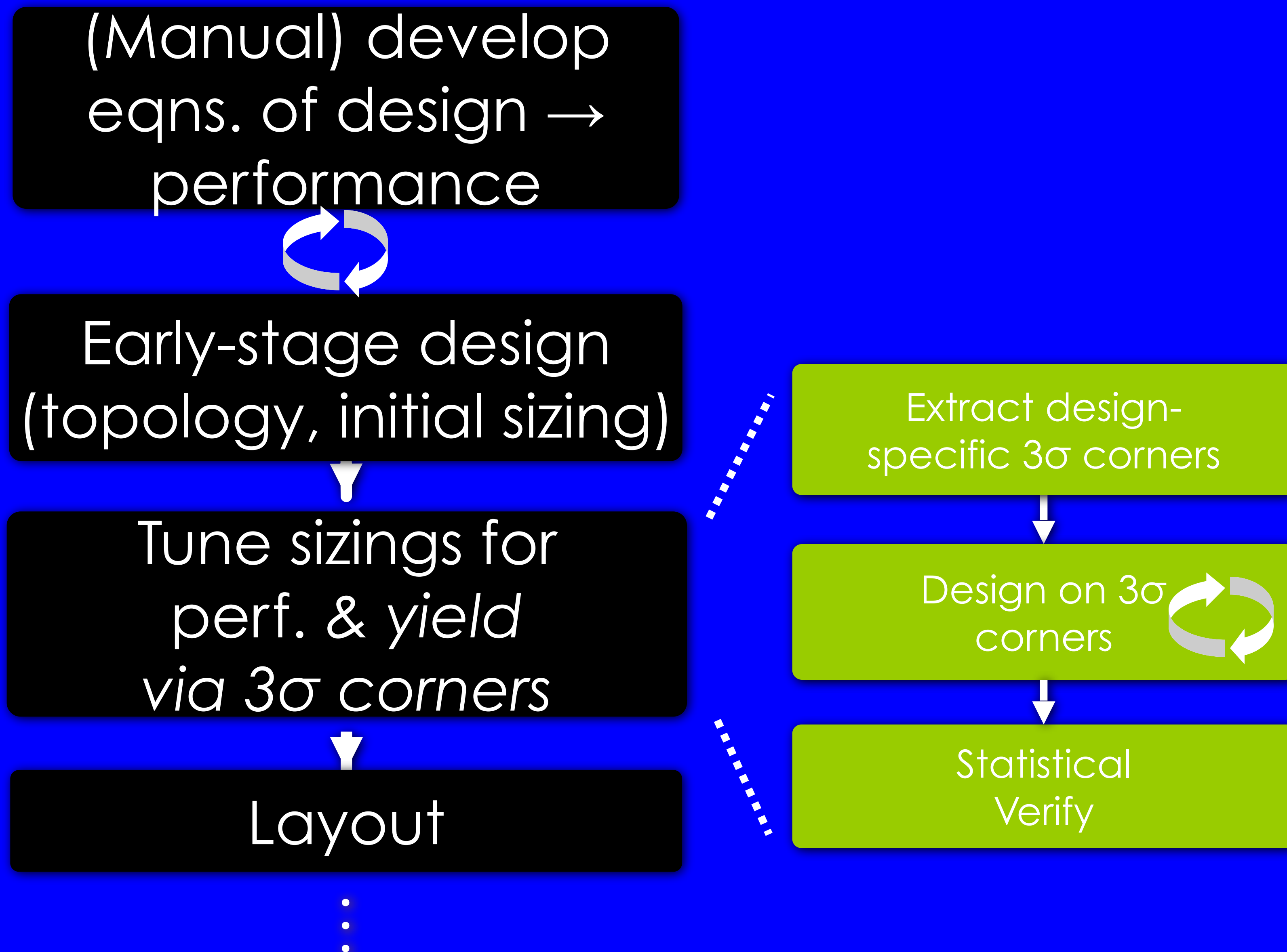
⋮

SPICE on all 50 (or 1K!) (or 10K!) MC samples



Q: How to get MC out of the design loop, yet retain accuracy of MC?

Design-Specific 3σ Corners For Variation-Aware Design



Design-Specific 3σ Corners For Variation-Aware Design

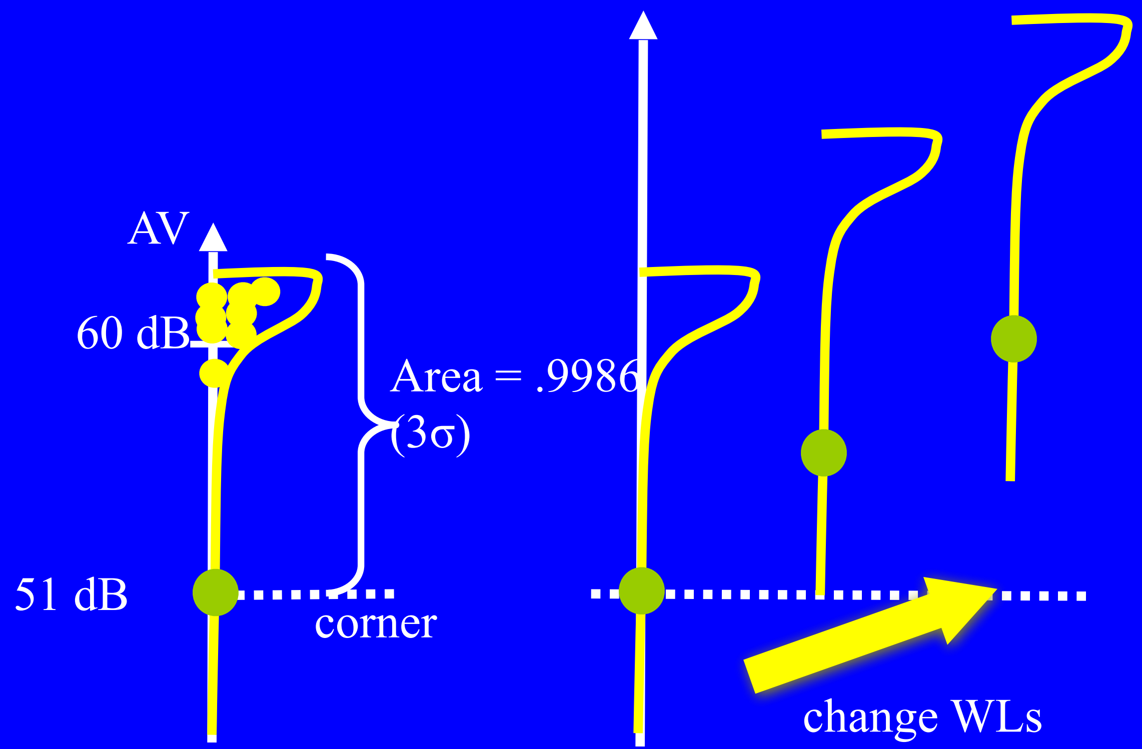
Extract design-specific 3σ corners

↓

Design on 3σ corners

↓

Statistical Verify



Variation-Aware Design Flow

(Manual) develop eqns. of design → performance

Early-stage design (topology, initial sizing)

Tune sizings for perf. & yield

Layout

⋮

There are effective industrial-scale methods *here*…

…but not *here*

This paper aims to help!

Outline

- Motivation
- <u>Proposed flow</u>
- Backgrounder
- Fast Function Extraction (FFX)
- Results
- Conclusion

An Ideal Flow
… Is Infeasible

(Manual) develop eqns. of design & *process vars.* → circuit performance

Early-stage design (topology, initial sizing)

Tune sizings for perf. & yield

Layout

⋮

"Dear designer,
To handle mismatch in early design stages,
please add 1000 extra variables
Δtox_1, $\Delta Nsub_1$, …
Δtox_{100}, $\Delta Nsub_{100}$, …"

Are *you* an expert at process modeling?

Who is an expert at *both* process modeling & topology development?

Proposed Flow

(Manual) develop eqns. of design vars. → circuit perf.

(Auto) extract eqns. of process vars. → circuit perf.

Early-stage design (topology, initial sizing)

↓

Tune sizings for perf. & yield

↓

Layout

⋮

Equation-Extraction Step: Details, by Example (1D)

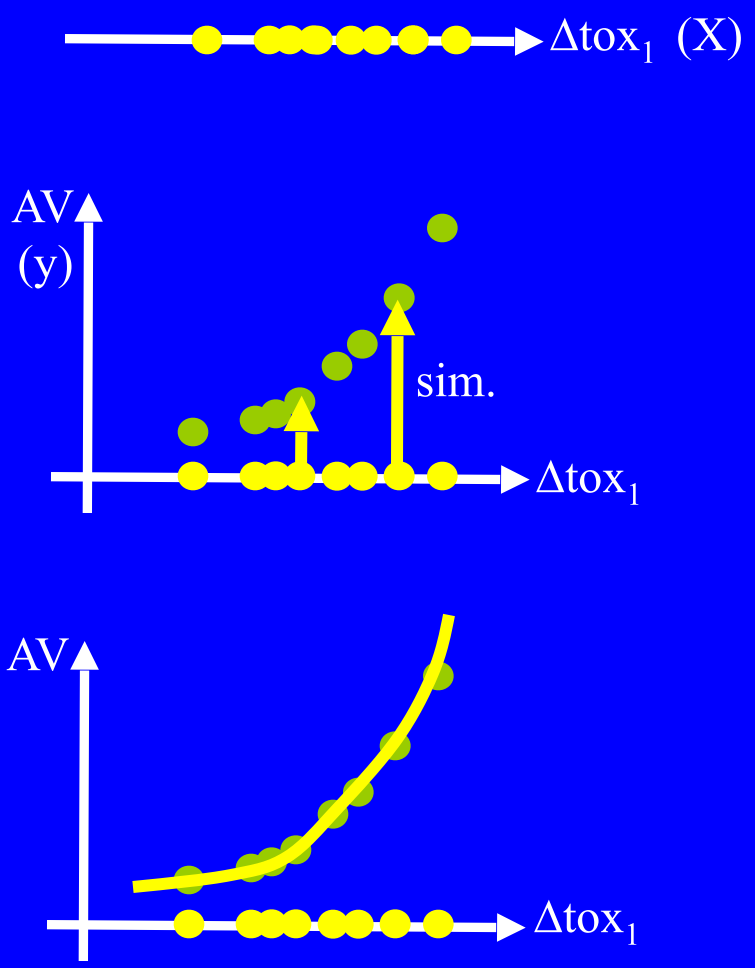
Draw 100-5000 MC samples (X)

↓

SPICE-Simulate on samples (y)

↓

Build whitebox Model of X → y



$$AV = 50.2 + 9.1 \cdot \Delta tox_1 + 3.2 \cdot \max(0, \Delta tox_1{}^2)$$

Equation-Extraction Step: Problem Scope

- 5-10+ process variables per device (e.g. BPV model)
- 10-100+ devices
- Therefore 50-1000+ input variables
- 100-5000 simulations (runtime cost)
- Need whitebox model, for insight!
- Ideally get a tradeoff of complexity vs. accuracy

Equation Extraction Step: Off-the-Shelf Modeling Options

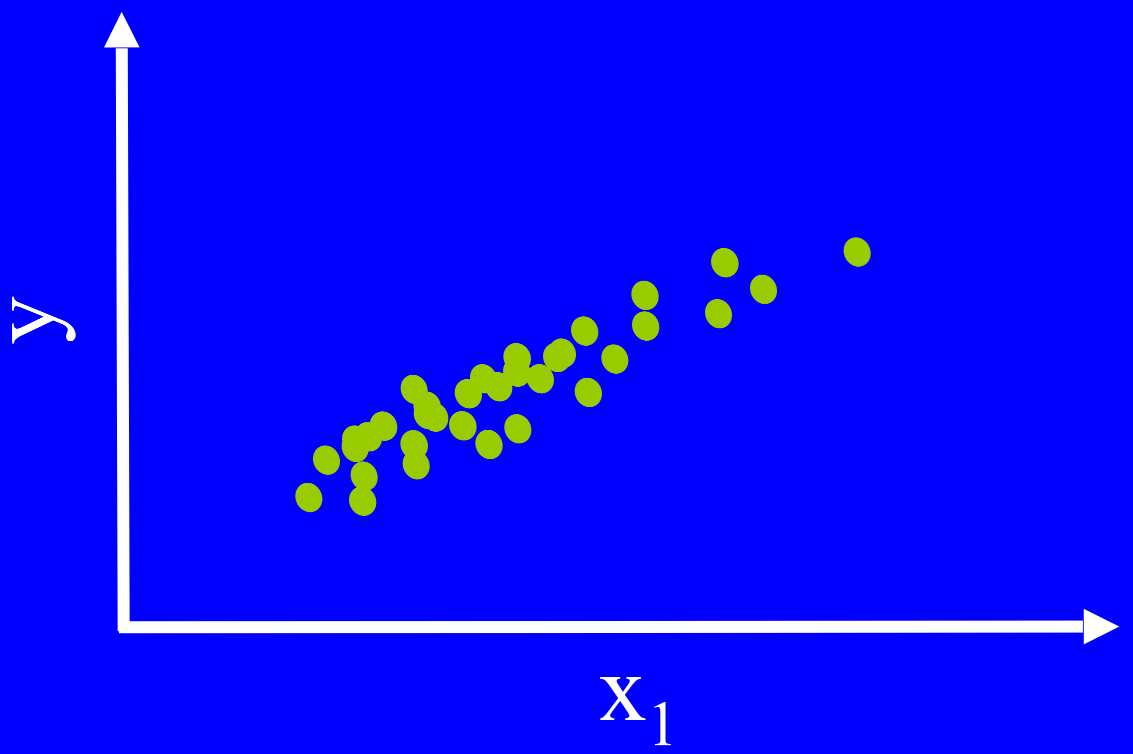
- Linear model – can’t handle nonlinear
- Quadratic – not nonlinear enough, scaling?
- MARS, SVM, neural net, others – not whitebox

Nothing meets problem scope!

Outline

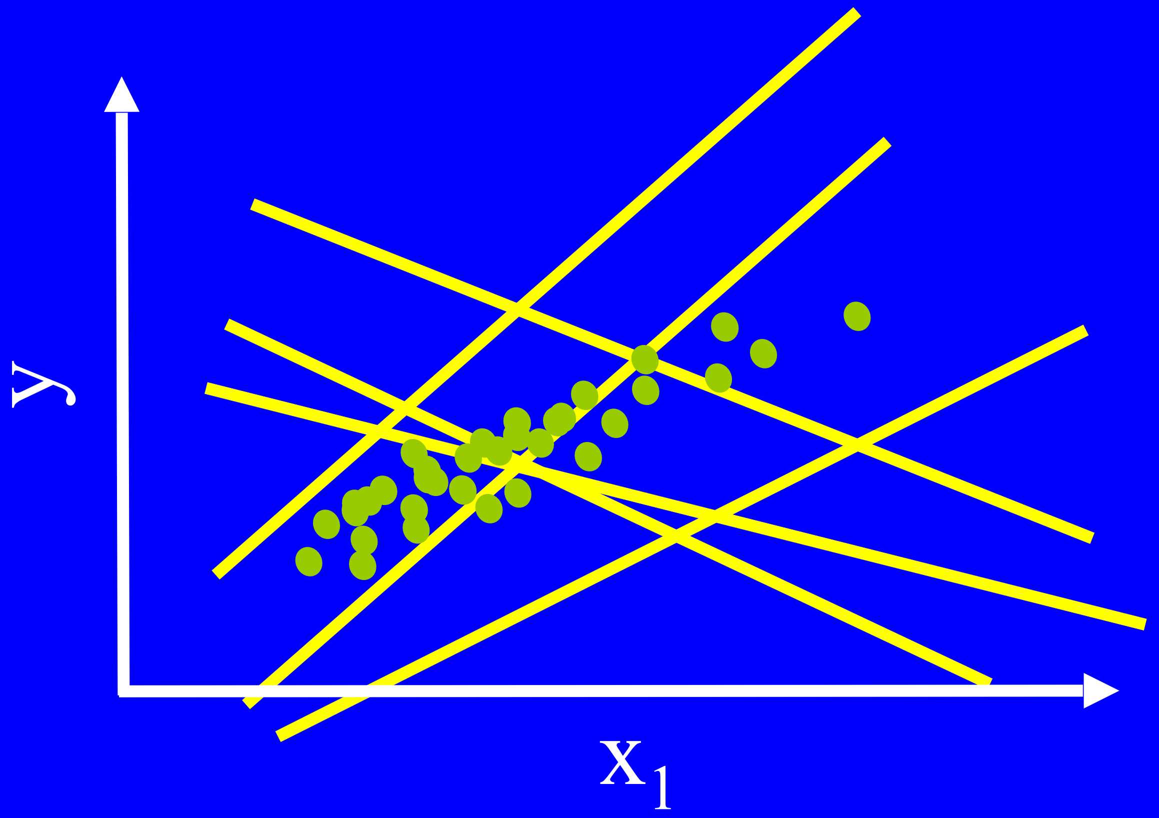
- Motivation
- Proposed flow
- <u>Backgrounder</u>
  - <u>Least-squares regression</u>
  - Regularized learning
- Fast Function Extraction (FFX)
- Results
- Conclusion

Given some x,y data…



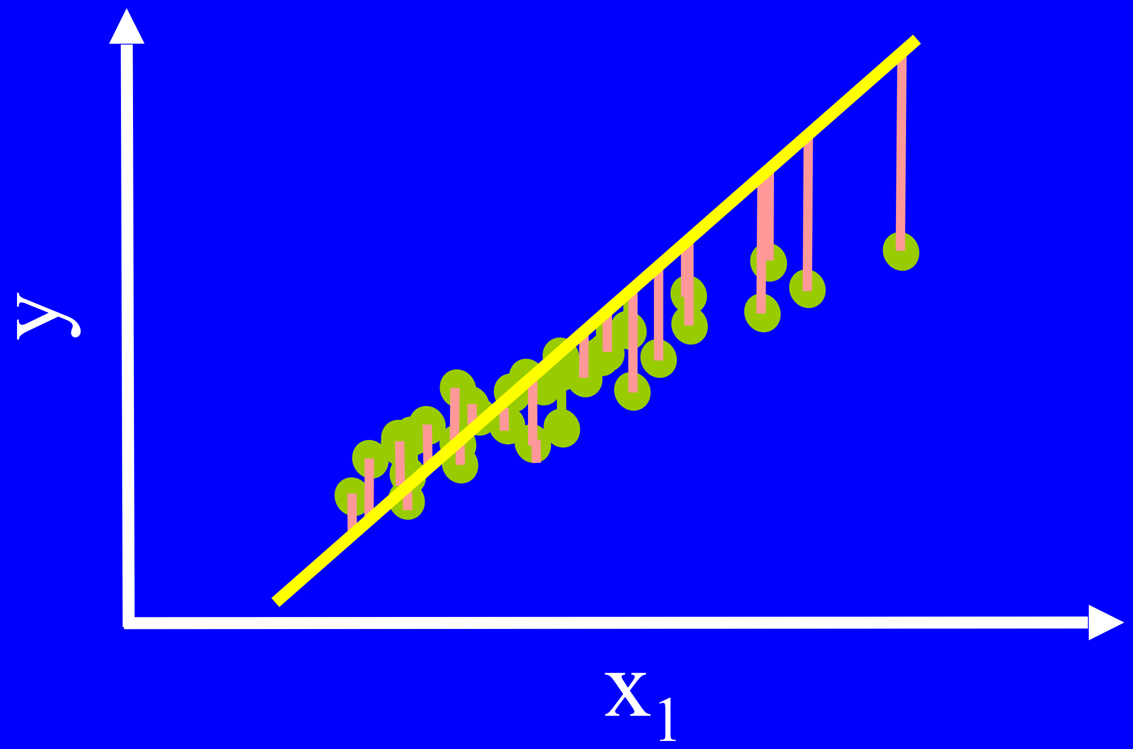
…we can build 1D linear models

Many possible linear models!
(∞ to be precise)



1D Linear Least-Squares (LS) Regression

Find linear model that minimizes $\sum(\hat{y}_i - y_i)^2$ *(across all i in training data)*

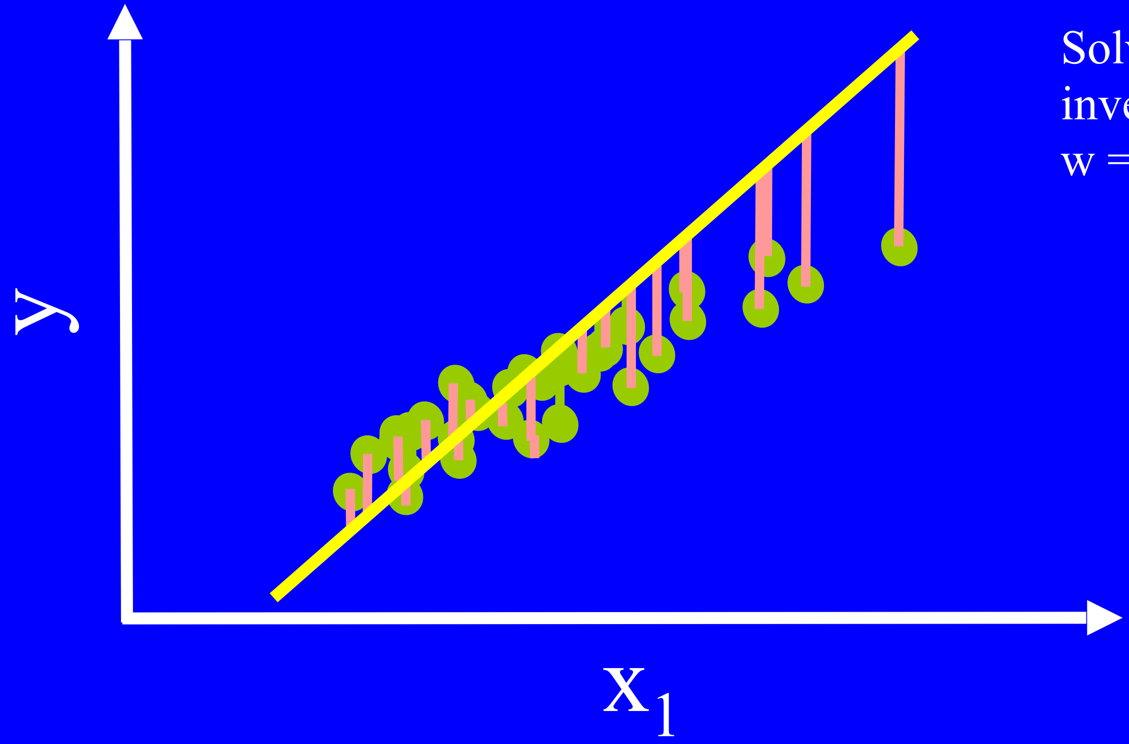


1D Linear LS Regression

Find linear model that minimizes $\sum(\hat{y}_i - y_i)^2$

That is: $[w_0, w_1]^* = \text{minimize} \sum(\hat{y}_i - y_i)^2$

where $\hat{y}(x_1) = w_0 + w_1 \cdot x_1$



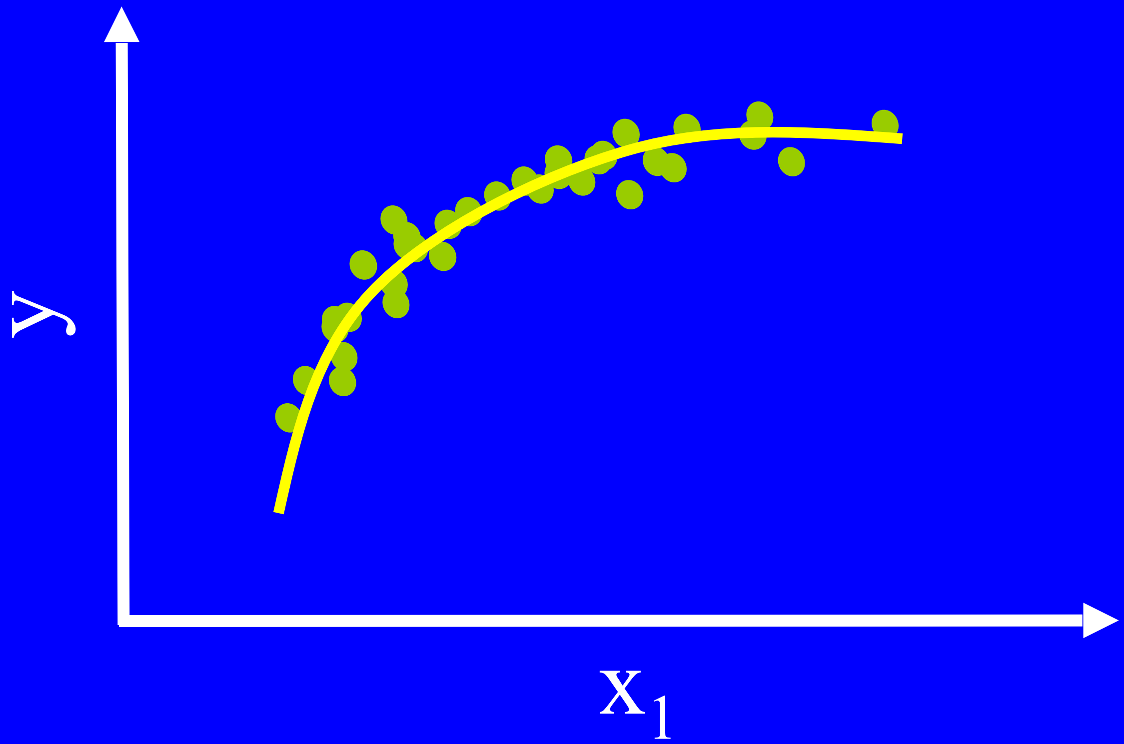
Solving this amounts to matrix inversion:
w = [1 X] / y

1D *Quadratic* LS Regression

$[w_0, w_1, w_{11}]^* = \text{minimize} \sum(\hat{y}_i - y_i)^2$

where $\hat{y}(x_1) = w_0 + w_1 \cdot x_1 + w_{11} \cdot x_1^2$

We are applying linear (LS) learning on linear & nonlinear basis functions. OK!

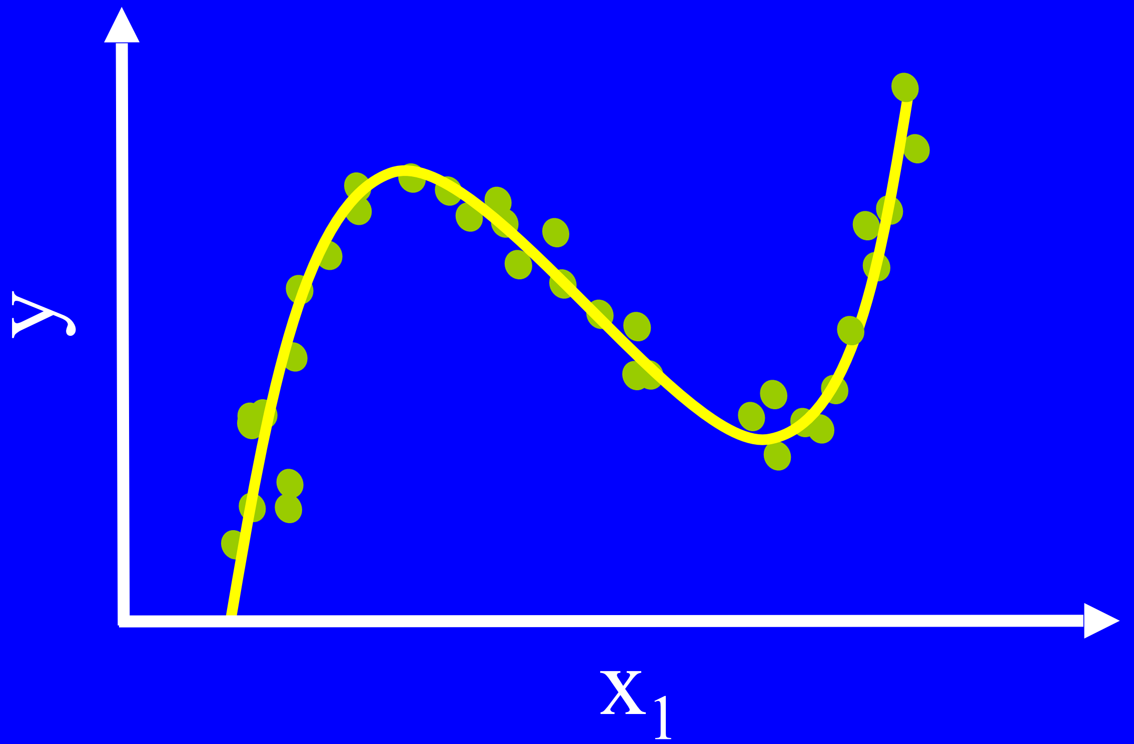


1D *Nonlinear* LS Regression

$[w_0, w_1, w_{sin}]^* = \text{minimize} \sum(\hat{y}_i - y_i)^2$

where $\hat{y}(x_1) = w_0 + w_1 \cdot x_1 + w_{sin} \cdot \sin(x_1)$

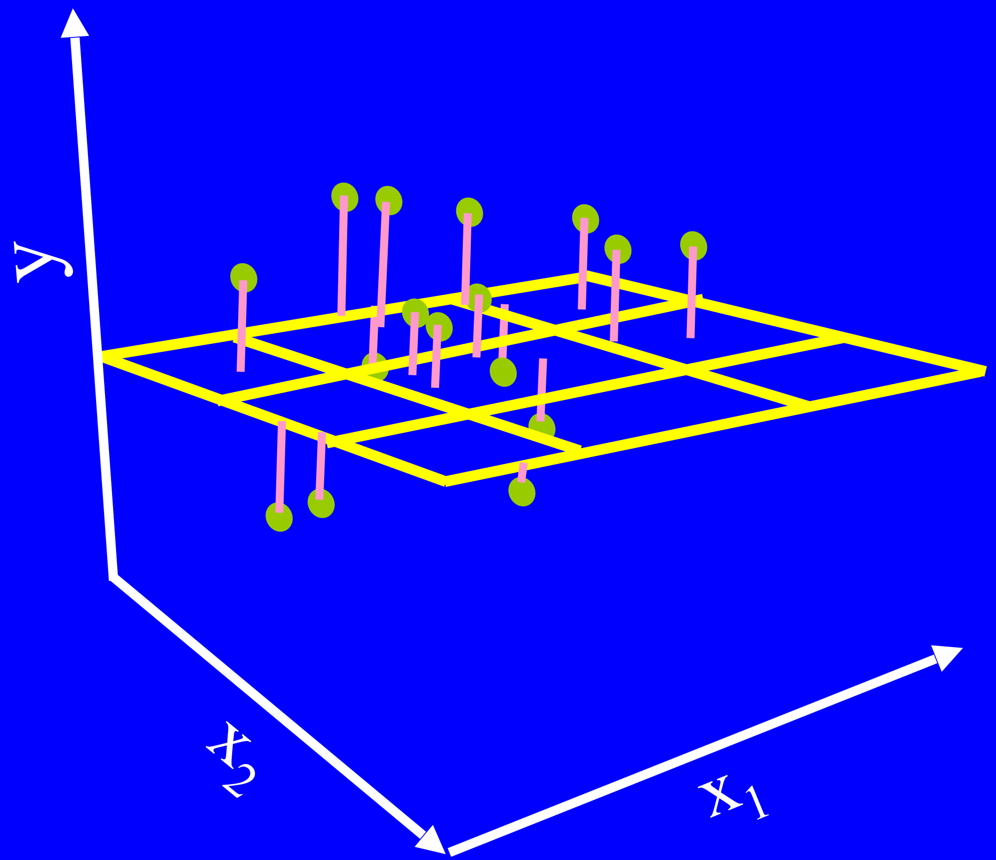
We are applying linear (LS) learning on linear & nonlinear basis functions. OK!



2D Linear LS Regression

$$[w_0, w_1, w_2]^* = \text{minimize} \sum(\hat{y}_i - y_i)^2$$

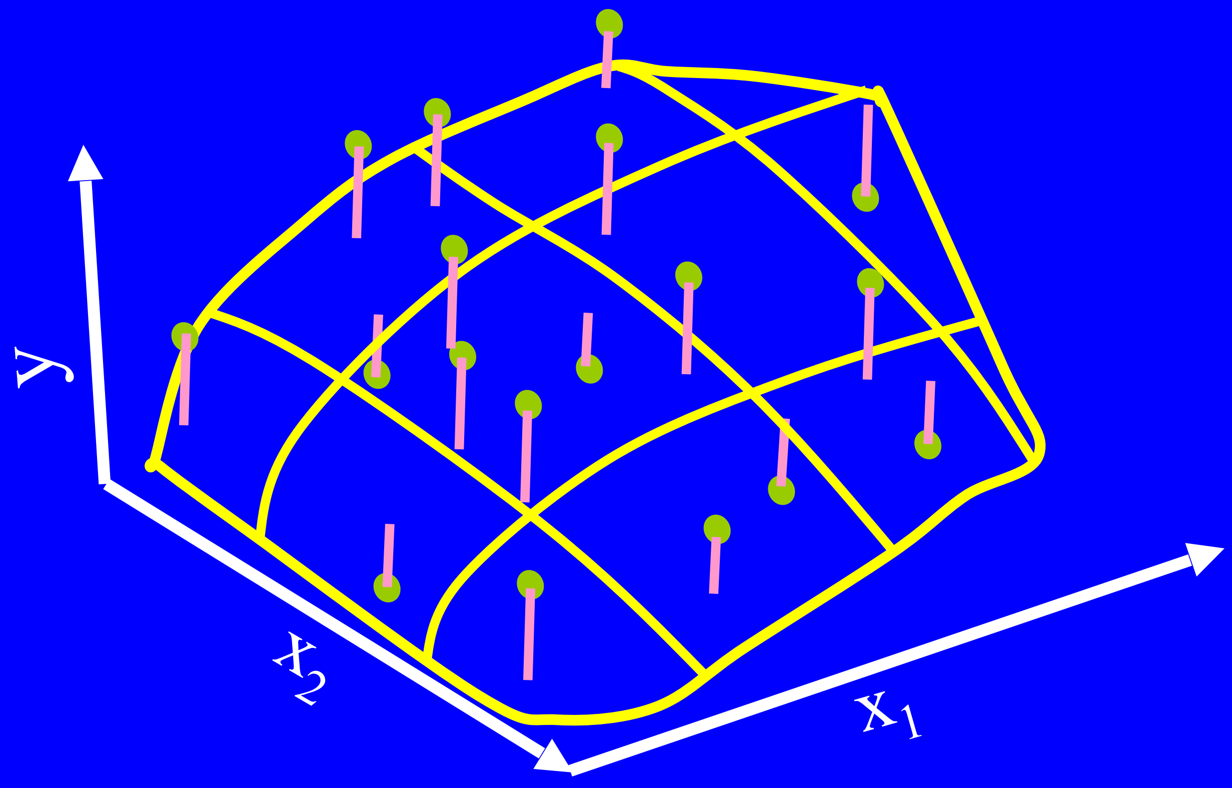
$$\text{where } \hat{y}(x) = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2$$



2D Quadratic LS Regression

$[w_0, w_1, w_2, w_{11}, w_{22}, w_{12}]^* = \text{minimize} \sum(\hat{y}_i - y_i)^2$

where $\hat{y}(x) = w_0 + w_1 \bullet x_1 + w_{11} \bullet x_1^2 + w_{22} \bullet x_2^2 + w_{12} \bullet x_1 \bullet x_2$



Outline

- Motivation
- Proposed flow
- Backgrounder
  - Least-squares regression
  - Regularized learning
- Fast Function Extraction (FFX)
- Results
- Conclusion

Regularized Linear Regression

- Minimizes a combination of training error *and model sensitivity*

- Formulation:

$$w^* = \text{minimize}\ \Big(\underbrace{\sum(\hat{y}_i(w) - y_i)^2}_{\substack{\text{minimize training error (fit}\\ \text{training data better)}}} + \lambda \cdot \underbrace{\sum|w_j|}_{\substack{\text{model sensitivity}\\ \text{(reduce risk}\\ \text{in future predictions)}}}\Big)$$

Regularized Linear Regression

When solving $w^* = \text{minimize}\ (\sum(\hat{y}_i(w) - y_i)^2 + \lambda \bullet \sum|w_j|)$,
consider different values for λ…

λ

$\lambda = \infty$

$\lambda = 0$

$\cancelto{0}{\sum(\hat{y}_i(w) - y_i)^2 + \lambda \bullet \sum|w_j|}$

$\sum(\hat{y}_i(w) - y_i)^2 + \lambda \bullet \sum|w_j|$

0

$w^* = [0, 0, \ldots, 0]$

(constant response)

w^* reduces to LS

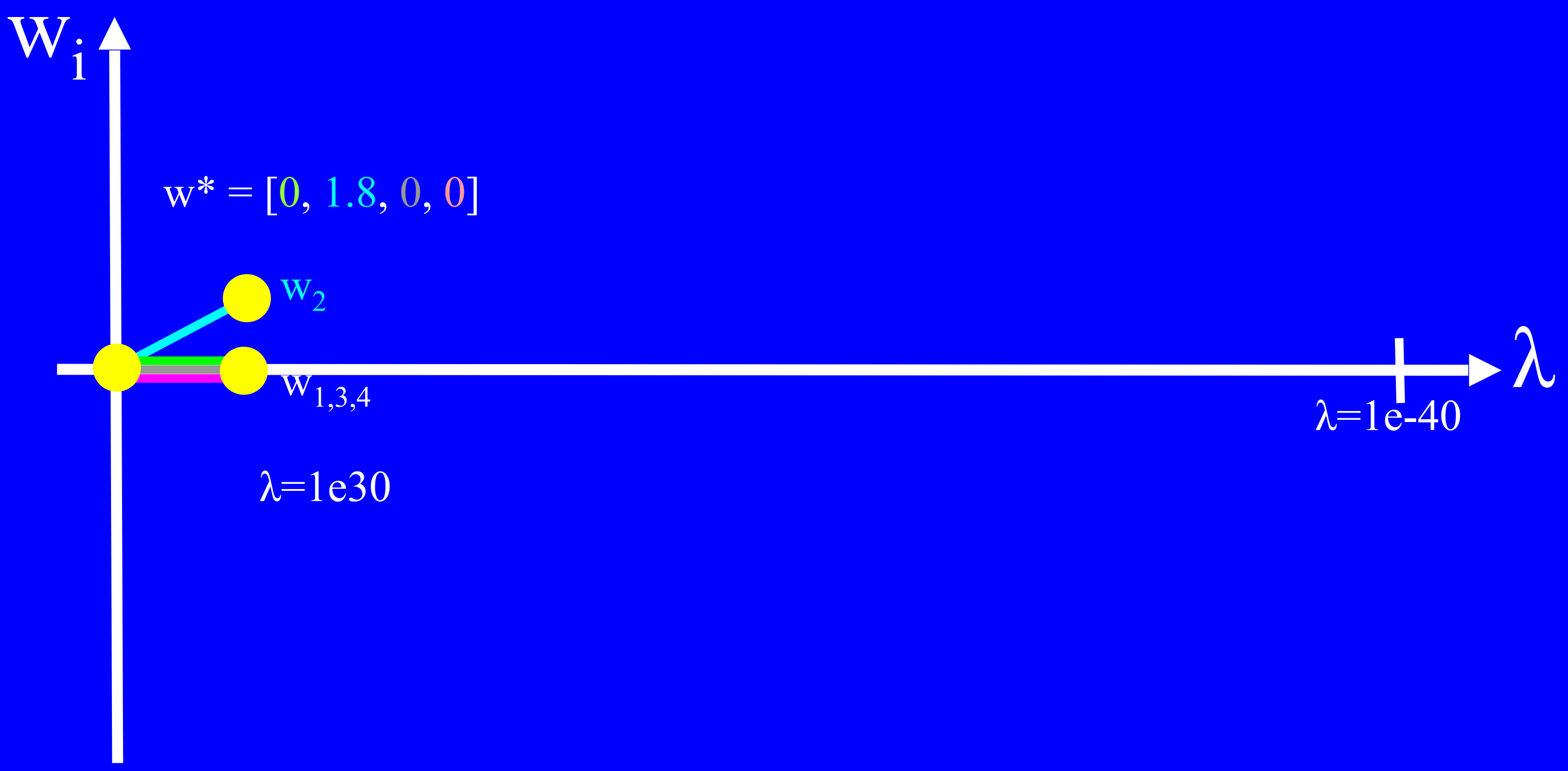
solution

Pathwise Regularized Linear Regression

solve $w^* = \text{minimize} \left(\sum(\hat{y}_i(w) - y_i)^2 + \lambda \bullet \sum|w_j| \right)$

at λ=1e40 ($\lambda \longrightarrow \infty$)

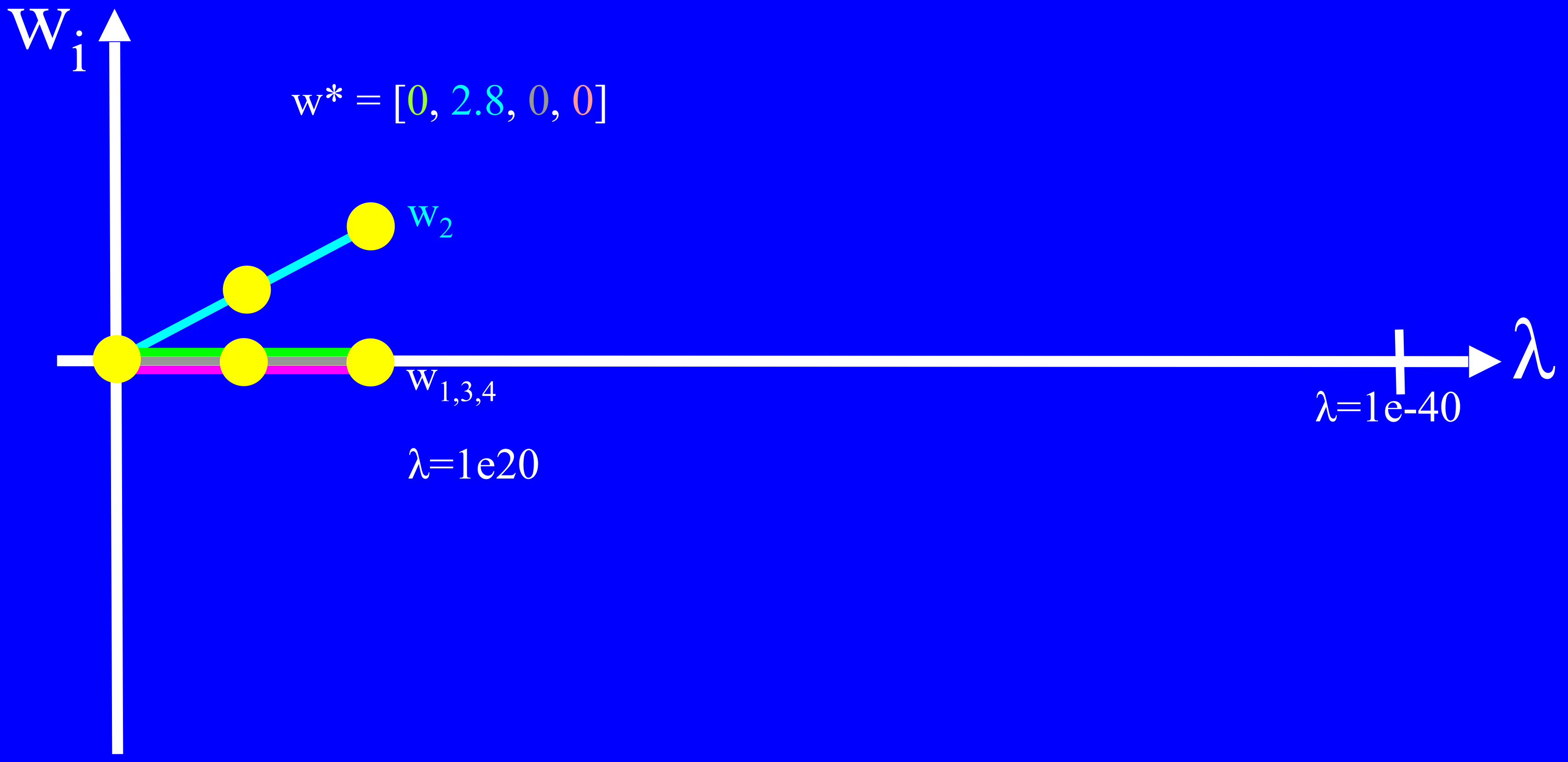
Pathwise Regularized Linear Regression



solve $w^* = \text{minimize}\ (\sum(\hat{y}_i(w) - y_i)^2 + \lambda \bullet \sum|w_j|)$

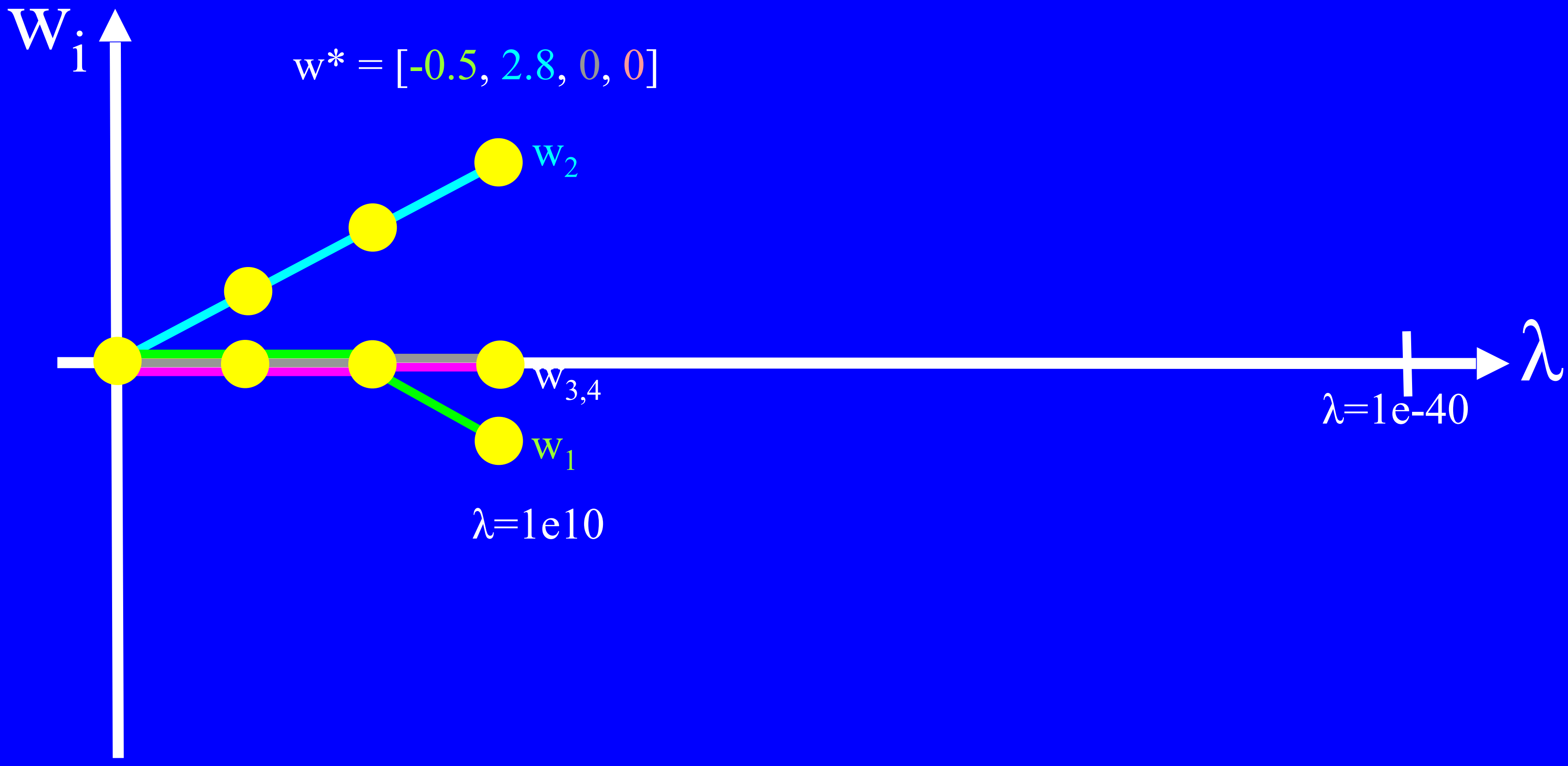
at λ=1e30

Pathwise Regularized Linear Regression



solve $w^* = \text{minimize} \left(\sum(\hat{y}_i(w) - y_i)^2 + \lambda \bullet \sum|w_j| \right)$
at λ=1e20

Pathwise Regularized Linear Regression

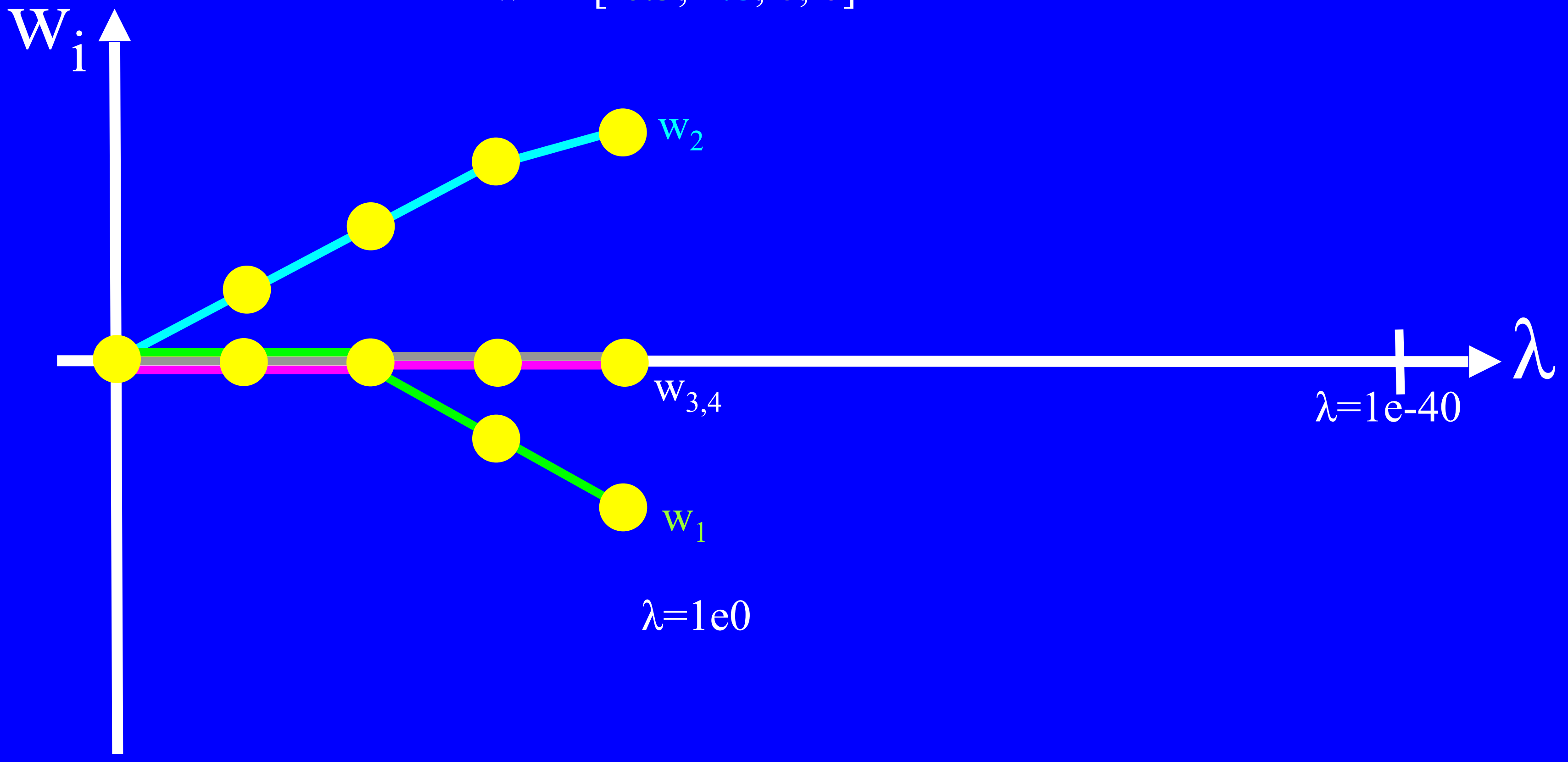


solve $w^* = \text{minimize} \left(\sum(\hat{y}_i(w) - y_i)^2 + \lambda \bullet \sum|w_j| \right)$

at λ=1e10

Pathwise Regularized Linear Regression

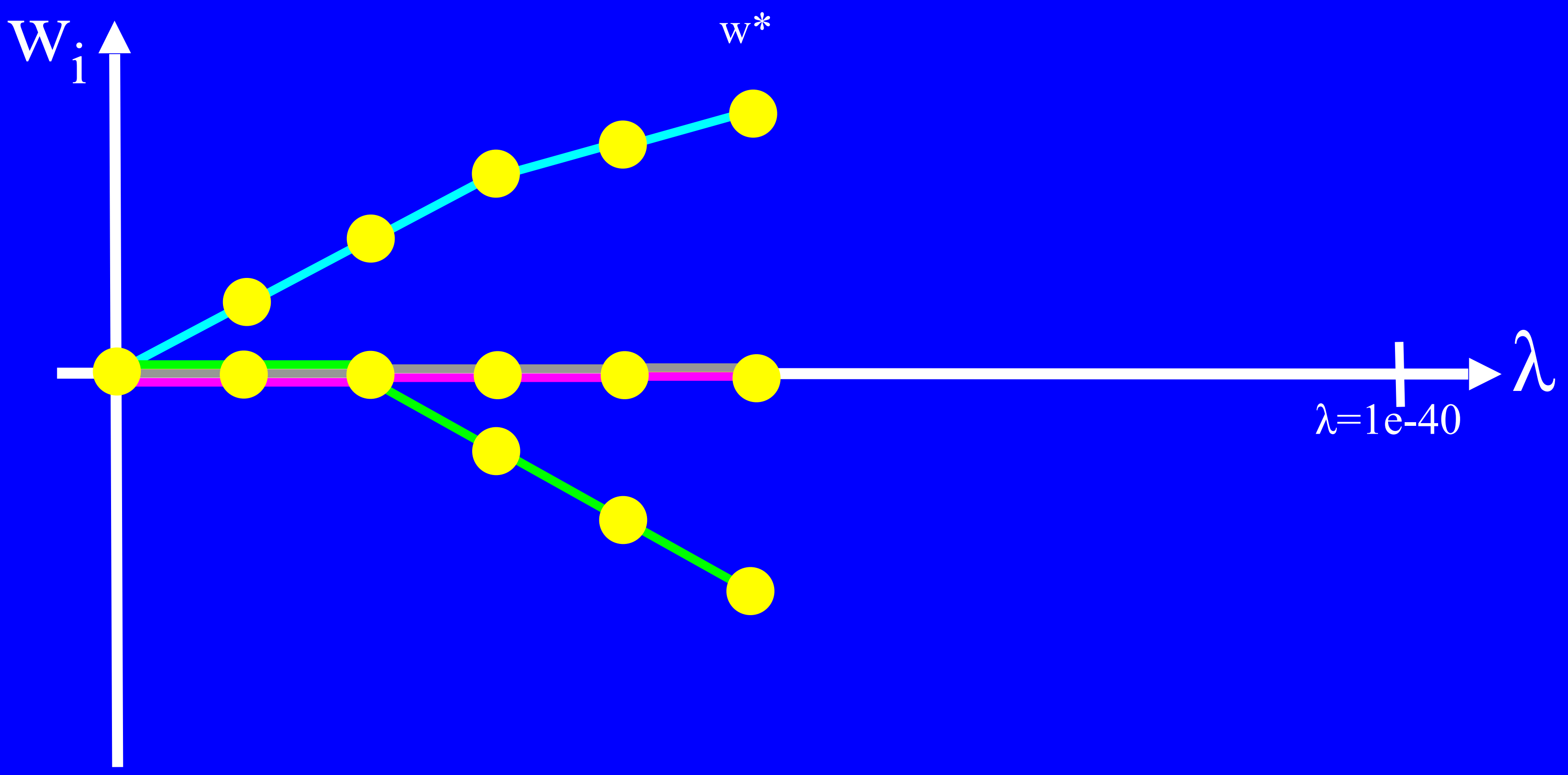
w* = [-0.5, 2.8, 0, 0]



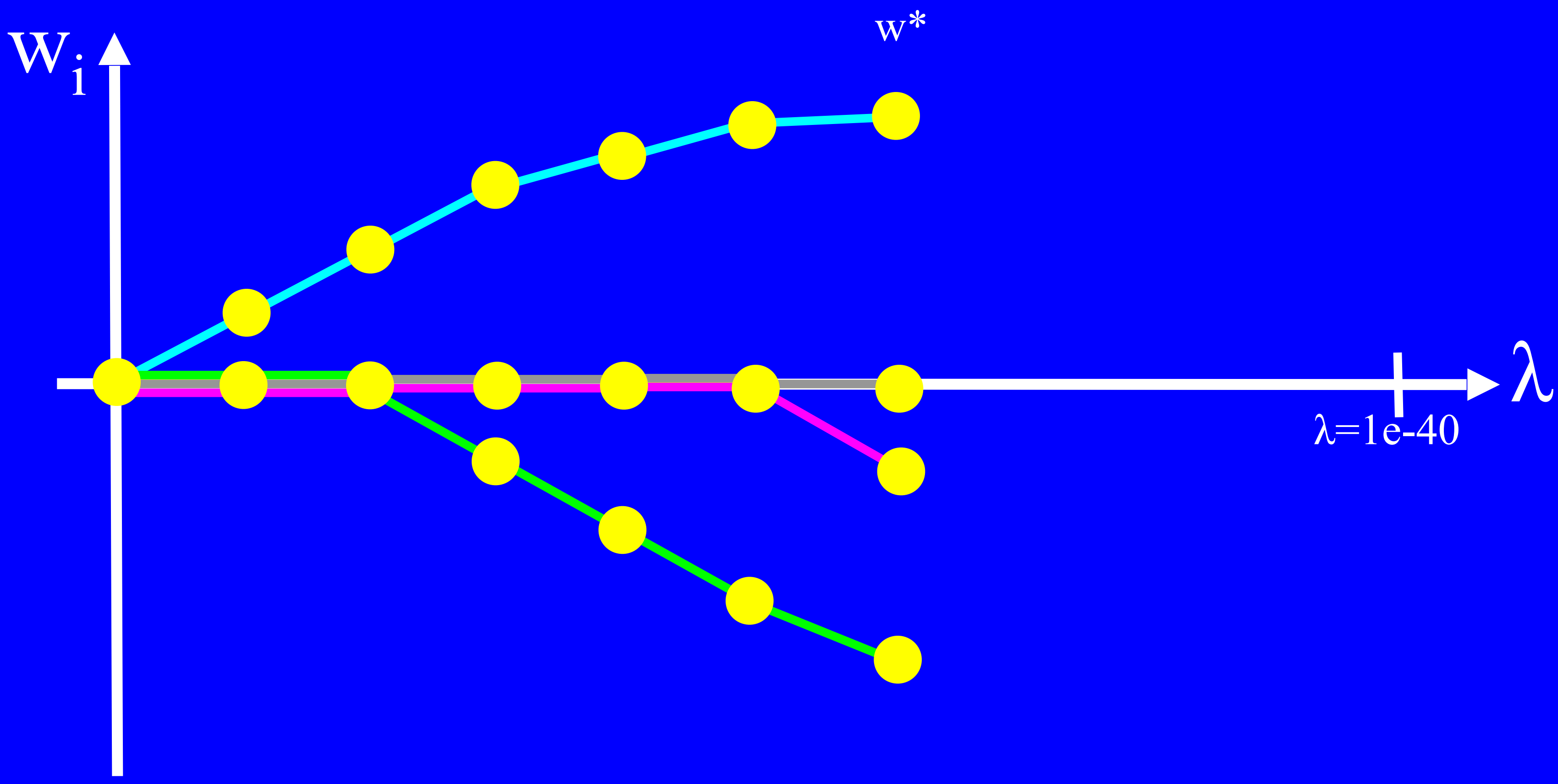
solve $w^* = \text{minimize}\ (\sum(\hat{y}_i(w) - y_i)^2 + \lambda \bullet \sum|w_j|)$

at λ=1e0

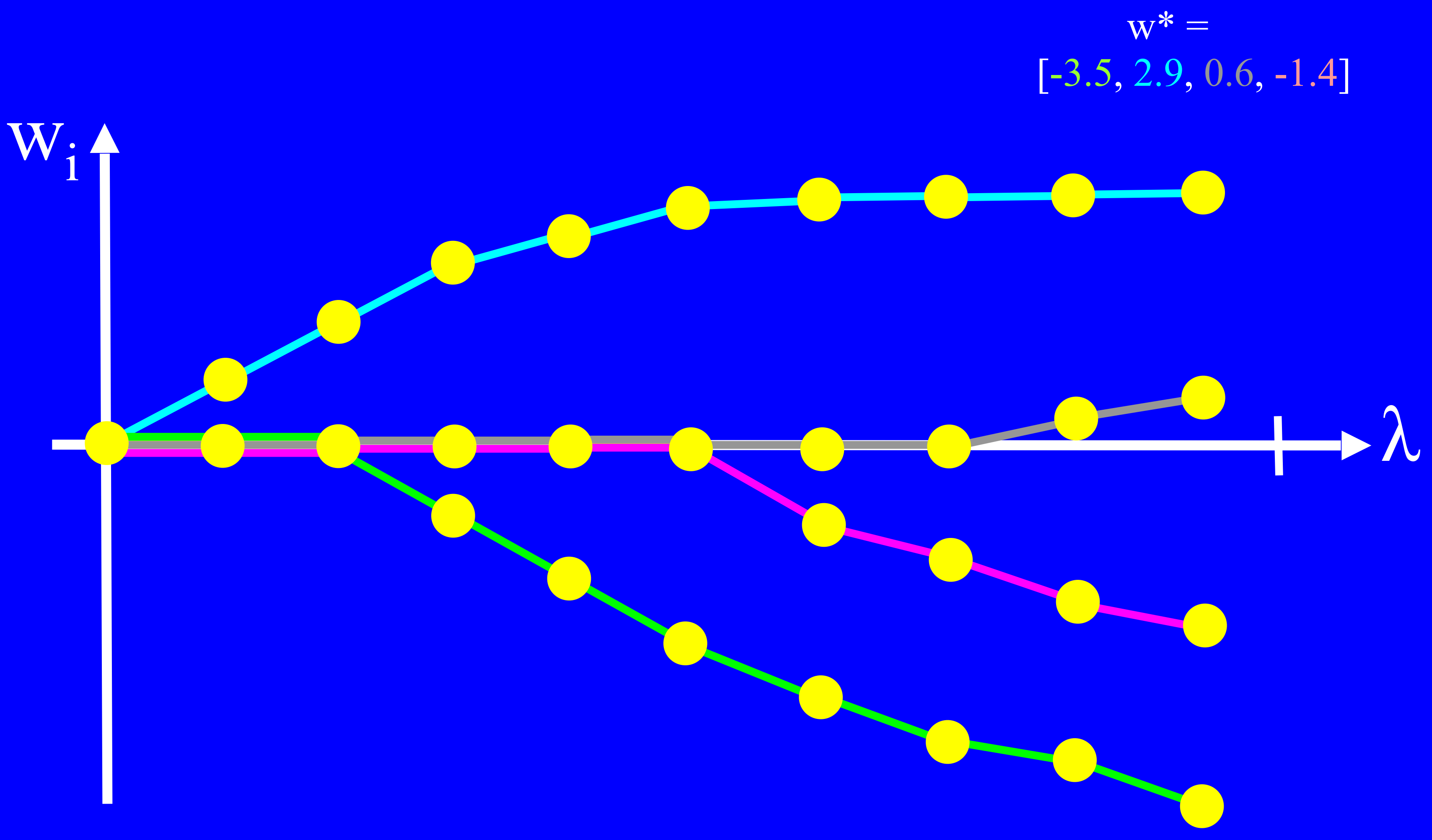
Pathwise Regularized Linear Regression



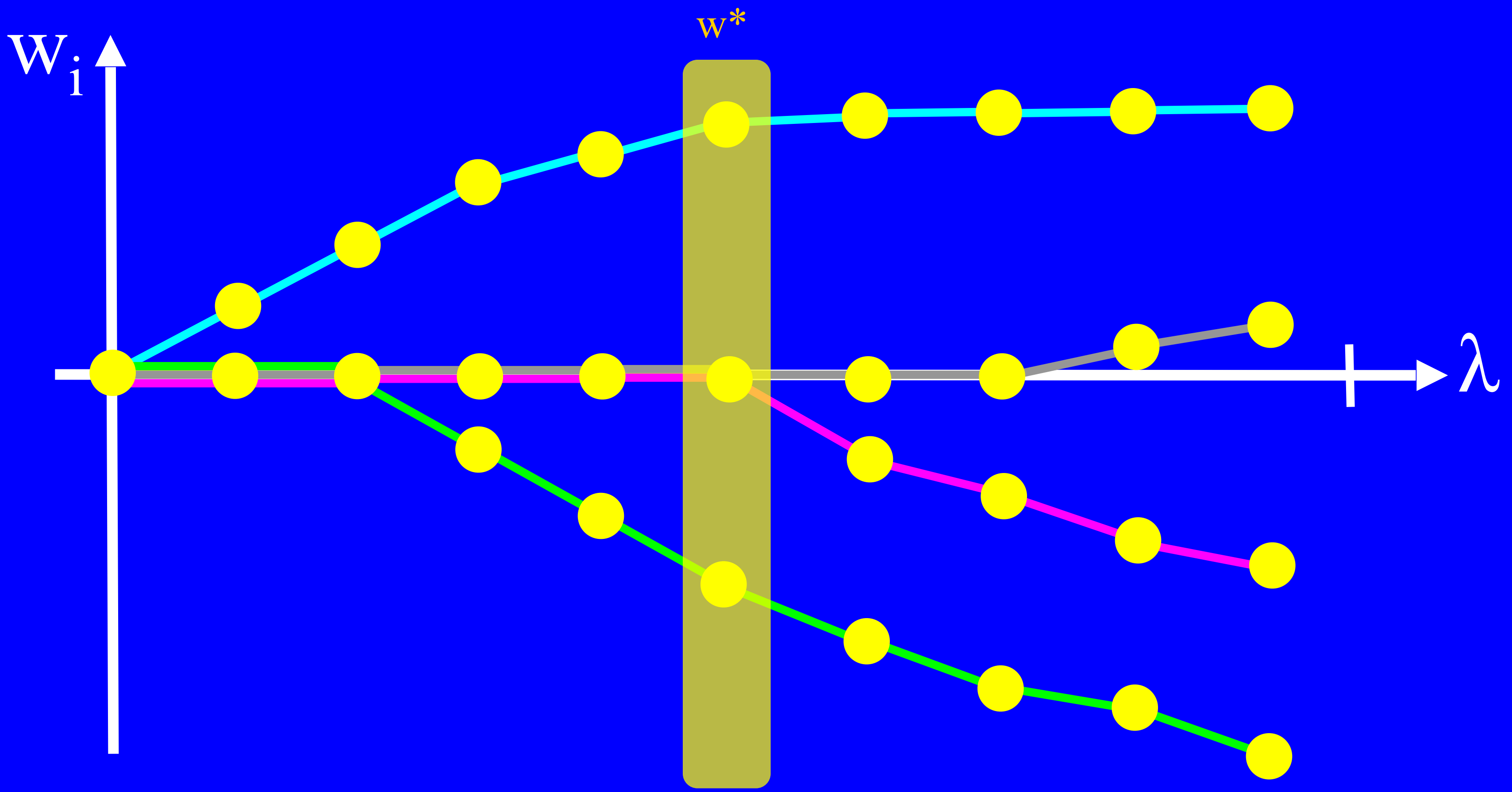
Pathwise Regularized Linear Regression



Pathwise Regularized Linear Regression

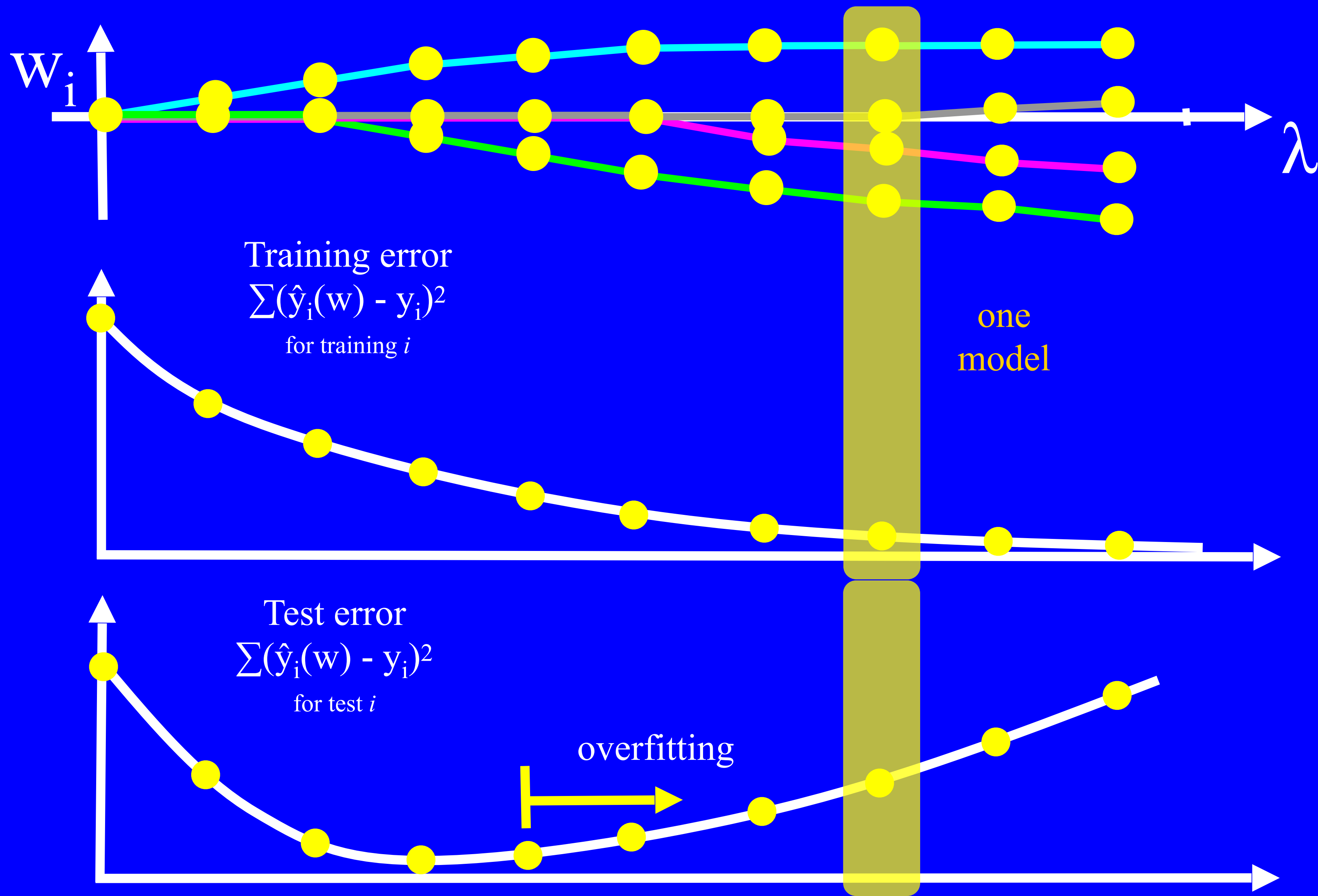


Pathwise Regularized Linear Regression

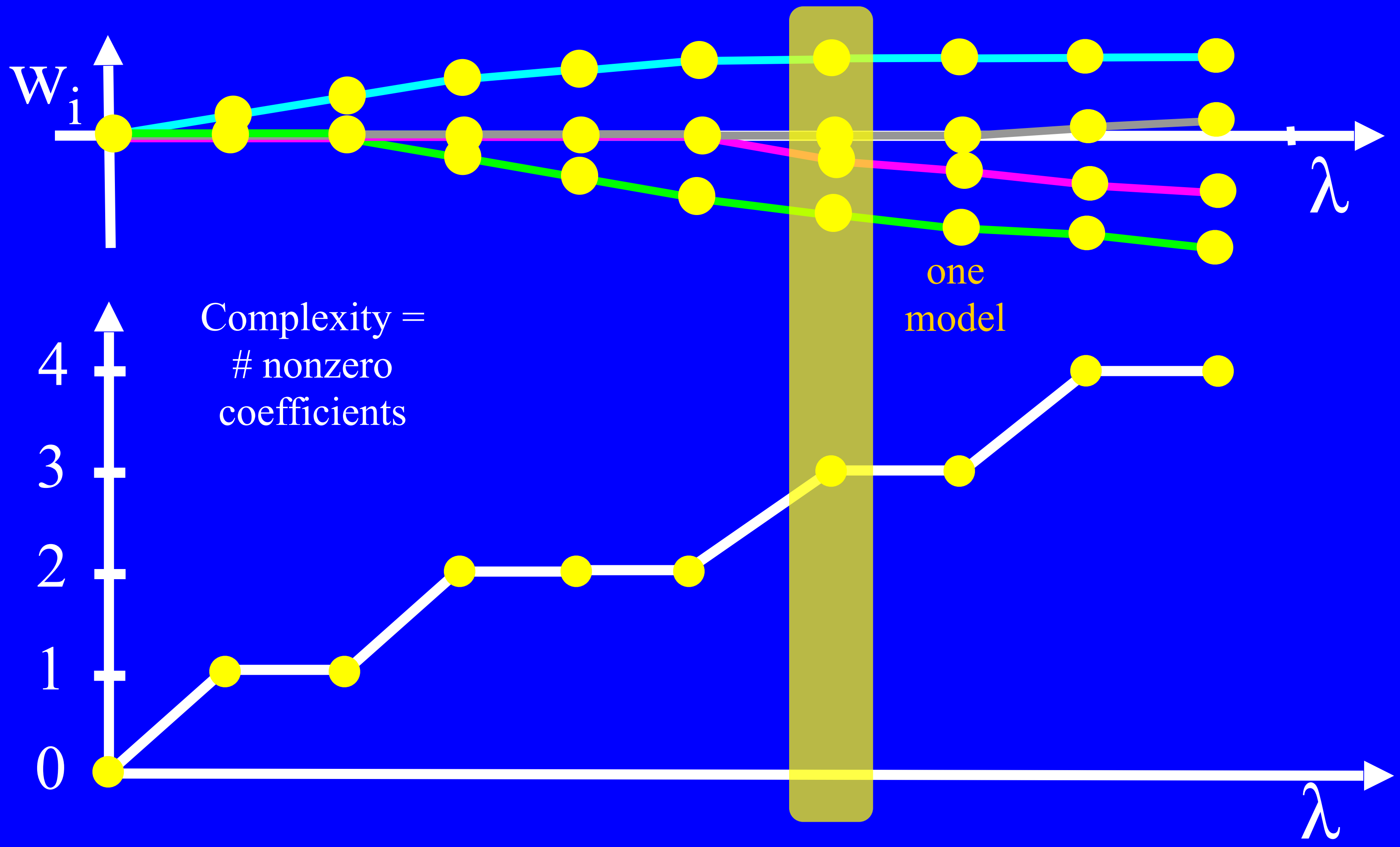


Each column vector of w* is a different linear model.
Which model is better / best? Accuracy vs. complexity…

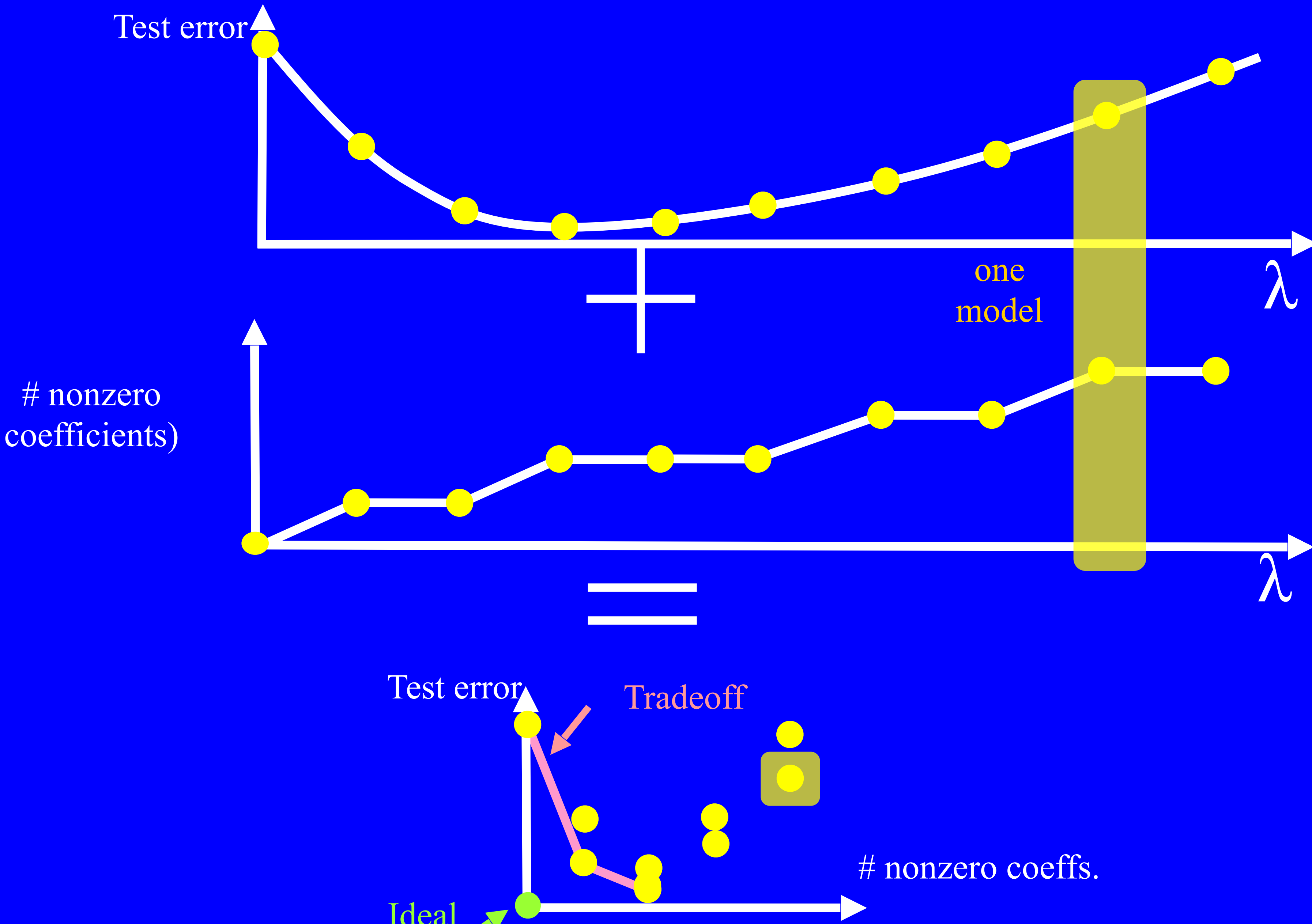
Pathwise Regression
Accuracy Per Model



Pathwise Regression
Complexity Per Model



Accuracy – Complexity Tradeoff



Cool Properties of Pathwise Regression (versus plain old LS)

- Cool property #1: solving a full regularized path gives us accuracy-vs-complexity tradeoffs!

- Cool property #2: can have more coefficients than samples!
  - Regularization term $\sum|w_i|$ means no underdetermined problems

- Cool property #3: solving a regularized learning problem is just as fast (or faster) than solving a least-squares learning problem!
  - Why: convex optimization problem – one big hill

Q: How *does* Google find furry robots?

Answer (NIPS 2010):

1. Treat images as 1000x1000 = 1M input vars. (x)
2. Crawl the web:
   - Find all images with "furry" & "robot" in filename. Assign y-value =
   - Find ≤5K images with "furry", "robot". y-value = 0.75
   - Randomly grab 5K more images. y-value = 0.0
3. Run regularized linear learning on X → y (learning on 1M input variables!!)
4. On all unseen images, run model. Return images sorted from highest $\hat{y}$ down.

- Cool property #4: regularized learning can handle a *ridiculous* number of input variables!

Outline

- Motivation
- Proposed flow
- Backgrounder
  - Least-squares regression
  - Regularized learning
- **Fast Function Extraction (FFX)**
- Results
- Conclusion

Recap: Proposed Flow

(Manual) develop eqns. of design vars. → circuit perf.

(Auto) extract eqns. of process vars. → circuit perf.

Early-stage design (topology, initial sizing)

Tune sizings for perf. & yield

Layout

⋮

Draw 100-1000 MC samples (X)

SPICE-Simulate on samples (y)

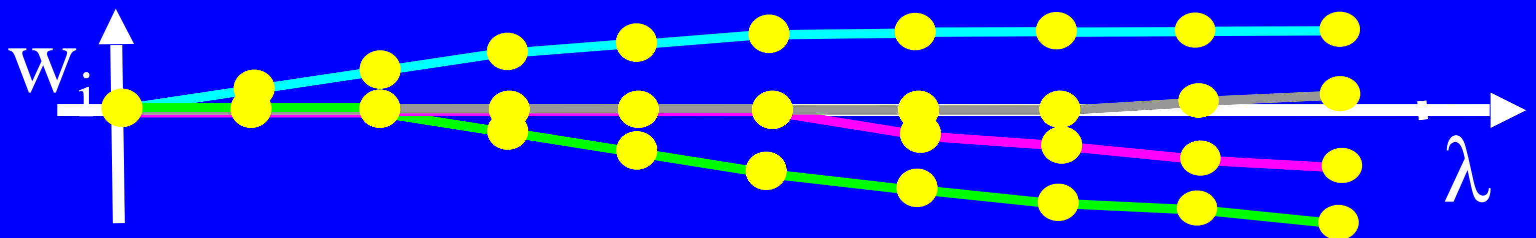
Build whitebox Model of X → y

A New Modeling Algorithm: Fast Function Extraction (FFX)

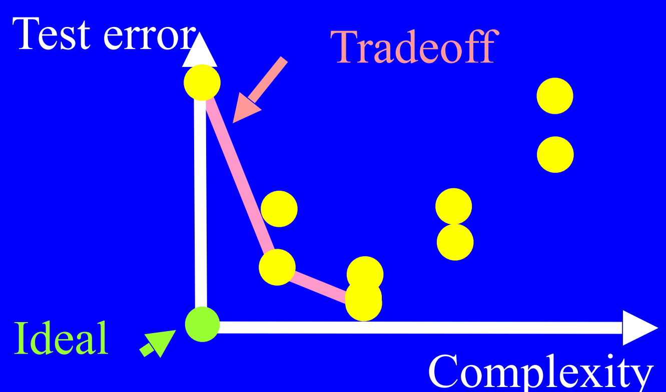
1. Replace linear bases with a *ridiculous* number *of* nonlinear ones

– x_i^2, $x_i \bullet x_j$, log, abs, max(0, x – thr), …

2. Pathwise regularized learn on bases



3. Filter for tradeoff of error vs. # coefficients



(That's it!) (+ a couple tweaks)

Outline

- Motivation
- Proposed flow
- Backgrounder
- Fast Function Extraction (FFX)
- Results
- Conclusion

Circuit Test Problems

Circuit	# Devices	# Process Vars.	Outputs Modeled
Opamp	30	215	AV (gain), BW (bandwidth), PM (phase margin), SR (slew rate)
Bitcell	6	30	$cell_i$ (read current)
Sense amp	12	125	delay, pwr (power)
Voltage ref.	11	105	DVREF (difference in voltage), PWR (power)
GMC filter	140	1468	ATTEN (attenuation), IL
Comp-arator	62	639	BW (bandwidth)

Hspice™, industrial MOS models < 65nm, 800-5K MC samples

Circuit Test Problems

Circuit	# Devices	# Process Vars.	Outputs Modeled
GMC filter	140	1468	ATTEN (attenuation), IL

Replace linear bases with

a *ridiculous* number *of* nonlinear ones

x_i^2, log(xi), abs(xi), max(0, x – thr),

$x_i \bullet x_j$, $\log(x_i) \bullet \mathrm{abs}(x_j)$, …

$\approx (10 * 1468)^2 / 2$

≈100M **bases to select from (!)**

Circuit Test Problems

Circuit	# Devices	# Process Vars.	Outputs Modeled
Opamp	30	215	AV (gain), BW (bandwidth), PM (phase margin), SR (slew rate)
Bitcell	6	30	$cell_i$ (read current)
Sense amp	12	125	delay, pwr (power)
Voltage ref.	11	105	DVREF (difference in voltage), PWR (power)
GMC filter	140	1468	ATTEN (attenuation), IL
Comp-arator	62	639	BW (bandwidth)

Results summary: <30 s build time. Error always < linear & quadratic, sometimes dramatically.

Opamp PM Equations

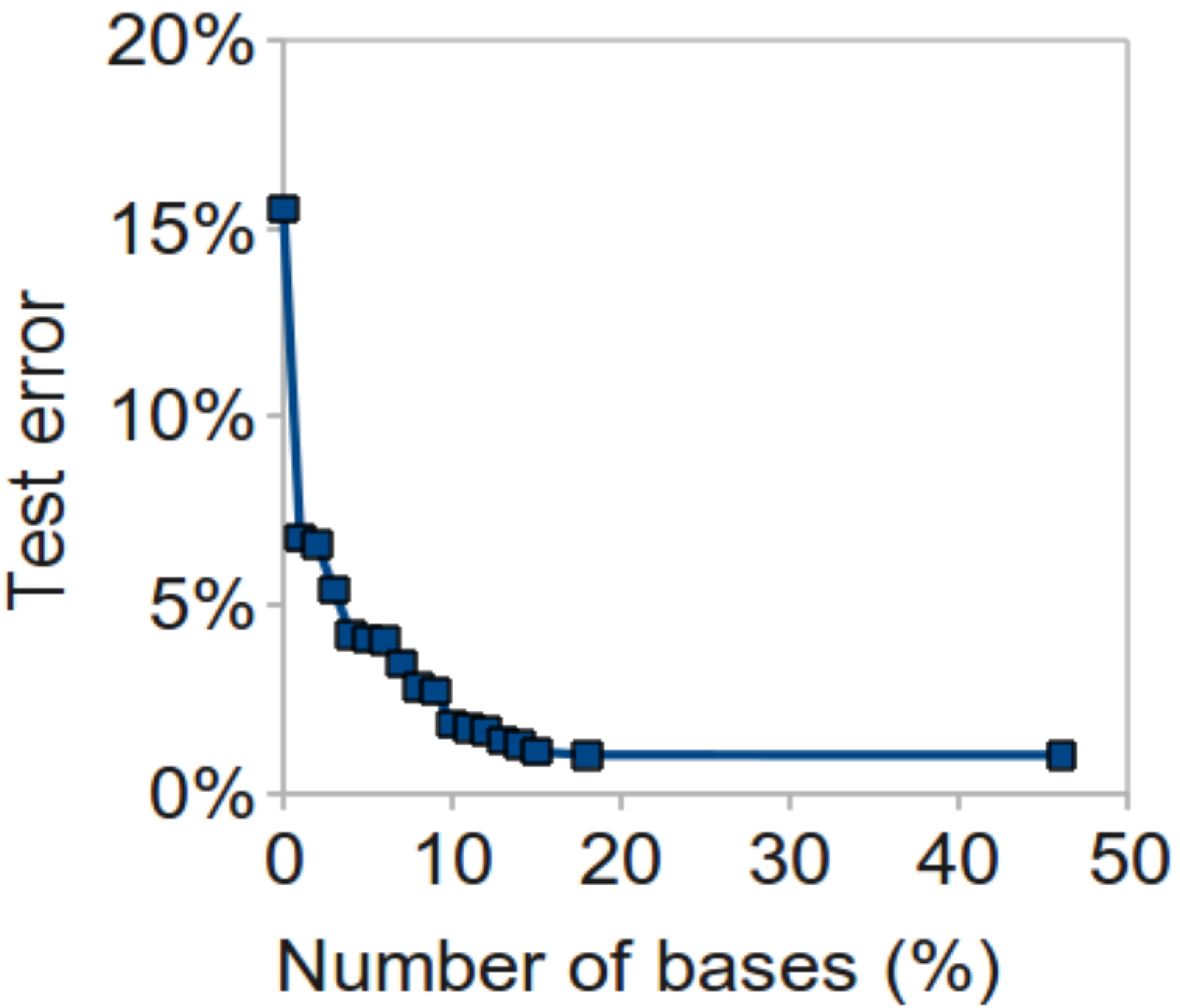
(215 global + local process variables. Modeling time < 30 s)

# bases	Test error	Extracted Equation
0	15.5 %	59.6
1	6.8	59.6 – 0.303 • dxl
2	6.6	59.6 – 0.308 • dxl – 0.00460 • cgop
3	5.4	59.6 – 0.332 • dxl – 0.0268 • cgop + 0.0215 • dvthn
4	4.2	59.6 – 0.353 • dxl – 0.0457 • cgop + 0.0211 • dvthn – 0.0211 • dvthp
5	4.1	59.6 – 0.354 • dxl – 0.0460 • cgop + 0.0198 • dvthn – 0.0217 • dvthp + 0.0135 • abs(dvthn) • dvthn
…	…	
46	1.0	58.9 – 0.136 • dxl + 0.0299 • dvthn – 0.0194…

Opamp PM Equations

Visualize Accuracy-Complexity Tradeoff

# bases	Test error
0	15.5 %
1	6.8
2	6.6
3	5.4
4	4.2
5	4.1
…	…
46	1.0



Voltage Ref. DVREF Equations

(105 global + local process variables. Modeling time < 30 s)

# bases	Test error	Extracted Equation
0	2.6 %	512.7
1	2.1	504 / (1.0 + 0.121 • max(0, dvthn + 0.875))
…	…	…
8	0.9	476 / (1.0 + 0.105 • max(0, dvthn + 1.61) – 0.0397 • max(0, -1.64 – dvthp) + …)

Shows: FFX is highly nonlinear if needed

Global vs. Local Variation?

FFX uses whatever variables help most, and sometimes patterns emerge

# bases	Test error	Opamp PM Extracted Equation
3	5.4	$59.6 - 0.332 \cdot dxl - 0.0268 \cdot cgop + 0.0215 \cdot dvthn$

global process variables

# bases	Test error	Comparator BW Extracted Equation
3	18.1	$171e6 - 4.57e5 \cdot x_{cm1,m1,lint} \cdot x_{cm1,m2,lint} + 5.23e4 \cdot x^2_{cm1,m1,lint} + 4.80e4 \cdot x^2_{cm1,m2,lint}$

local (mismatch) variables

Highest-Impact Variables for Opamp PM

% Impact	Variable Name
46.5	dxl
10.2	cgop
9.7	dvthn
7.4	dvthp
3.9	RCN_nsmm_DXL
3.8	RCP_nsmm_DXL
3.6	dxw
3.1	cgon
2.3	RCP_nsmm_DXW
...	...
0.3	CM1_M1_nsmm_LINT
0.3	CMB2_M1_nsmm_NSUB
...	...

Outline

- Motivation
- Proposed flow
- Backgrounder
- Fast Function Extraction (FFX)
- Results
- <u>Conclusion</u>

Conclusion

- Process variation is bad, and getting worse
- There are solutions for design tuning

… But not for early-stage manual topology design

- Idea: complement hand-derived equations with auto-extracted equations of variation → performance
- FFX builds the models (fast, scalable, nonlinear)

…with the help of pathwise regularized learning

- Easy to get started: code at trent.st/FFX
  - Just ≈3 pages of python!
  - Solido using it extensively. Others too, for analog test, behavioral modeling, and even web search (!)