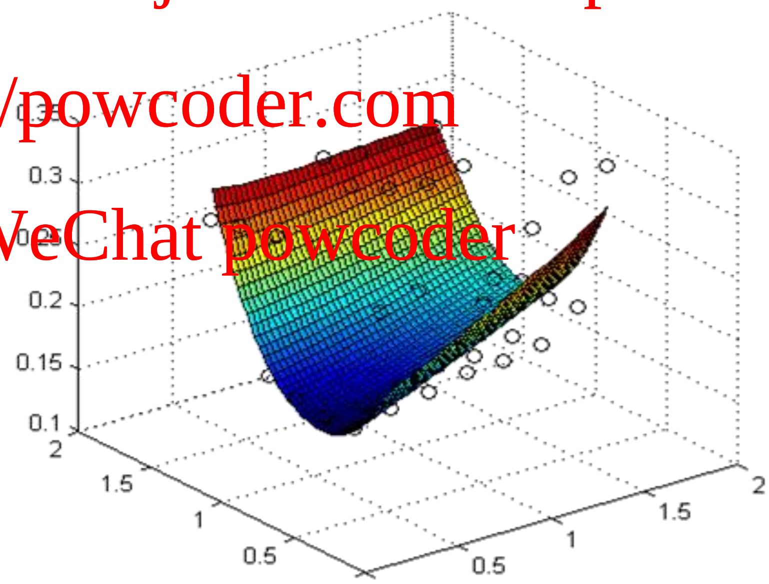
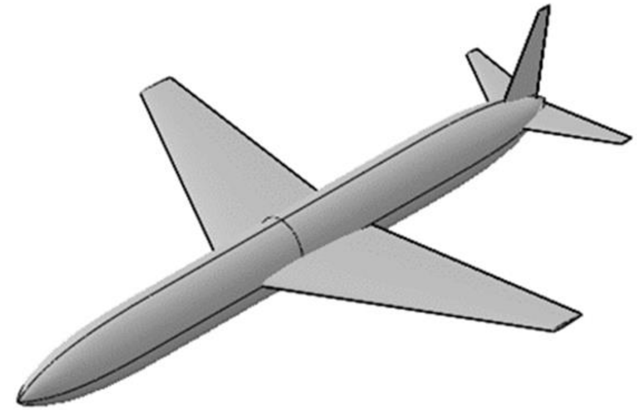
Sampling & Surrogate Modeling

ME 564/SYS 564
Wed. Sep. 19th 2018
Brian Chell

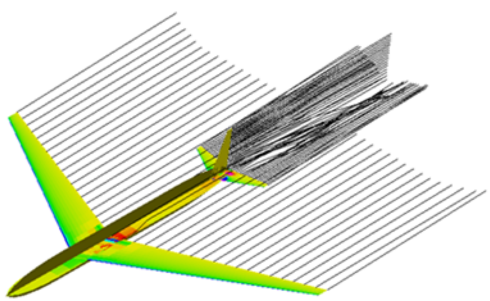


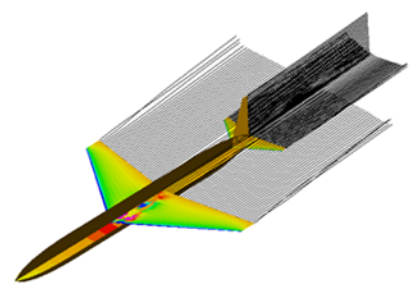
Goal of Week 4: To learn different strategies to sample your design space and build metamodels, and to explore some MATLAB tools that can help

Introduction



- Brian Chell
  - bchell@stevens.edu
- Ph.D. Student of Prof. Hoffenson
- Research based on Multidisciplinary Analysis and Optimization (MDAO)
- Sampling & surrogate modeling an important part of MDAO





Recap: How to optimize

1. **Formulate** the problem

(Weeks 1-2, 4, 9-12)

   a) Define system boundaries

   TODAY b) Develop analytical models

   c) Explore/reduce the problem space

   d) Formalize optimization problem

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}, \mathbf{p}) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{x}, \mathbf{p}) \leq 0 \\ & \mathbf{h}(\mathbf{x}, \mathbf{p}) = 0 \end{aligned}$$

2. **Solve** the problem

(Weeks 3, 5-8, 13)

   a) Choose the right approach/algorithm

   b) Solve (by hand, code, or software)

   c) Interpret the results

   d) Iterate if needed

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{H}(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_0)$$

Recap: Optimization formulation

Objective function

Parameters

Variables

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x},\mathbf{p}) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{x},\mathbf{p}) \le 0 \\ & \mathbf{h}(\mathbf{x},\mathbf{p}) = 0 \end{aligned}$$

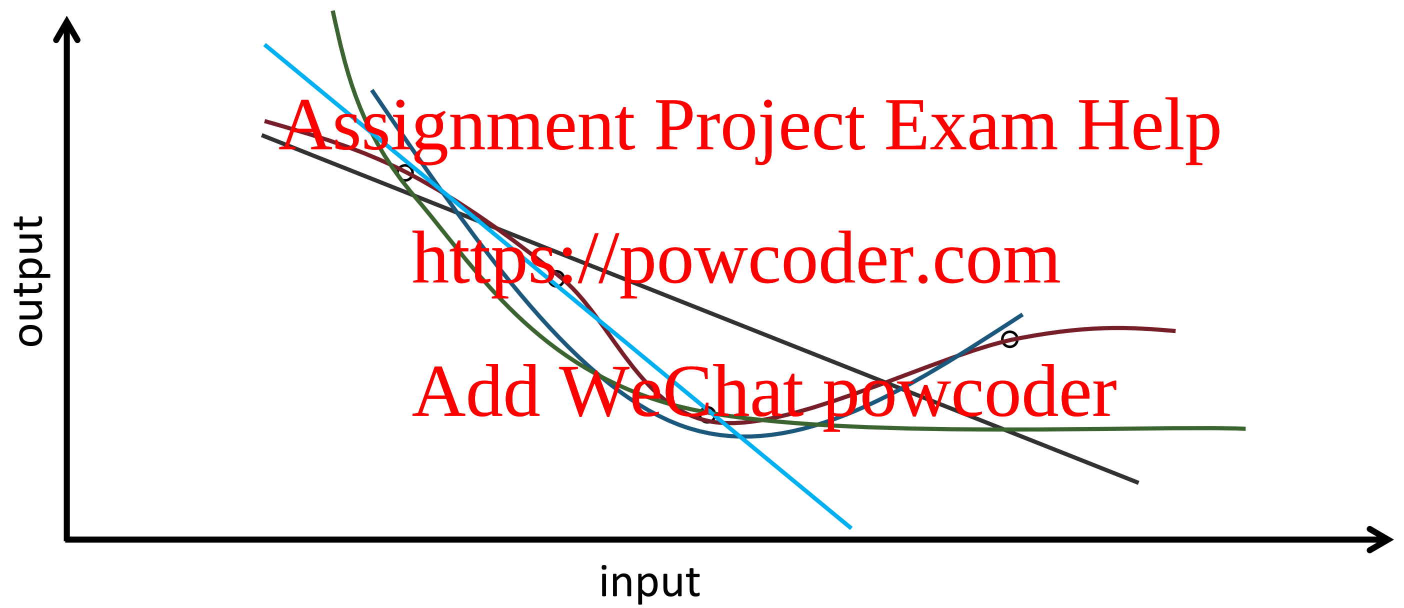
Constraints

Today we will discuss strategies for dealing with difficult functions *f*, *g*, and *h*

"negative null" form

What is surrogate modeling?

Fitting a function to some data



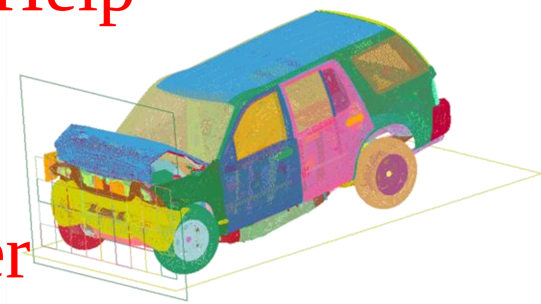
The *data points* represent the model

The *function* we fit is a meta- or "surrogate" model—a model of a model!

Why do we do it?

- When we have only physical experiments or observations to collect data (no computational models exist)

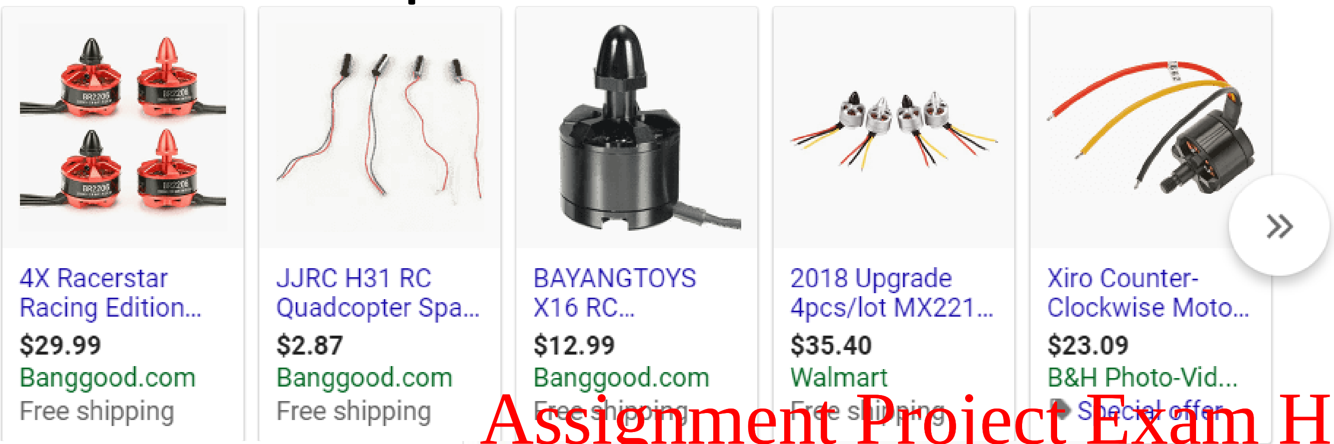
- When computational models are expensive to evaluate (usually time-intensive)

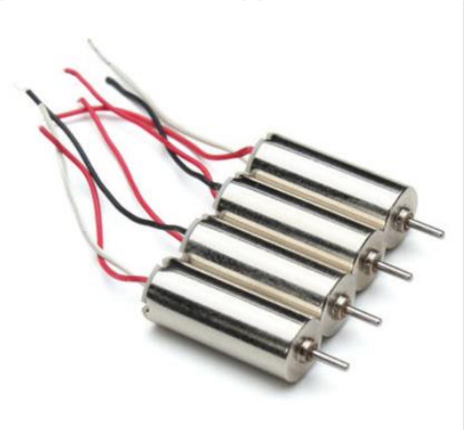


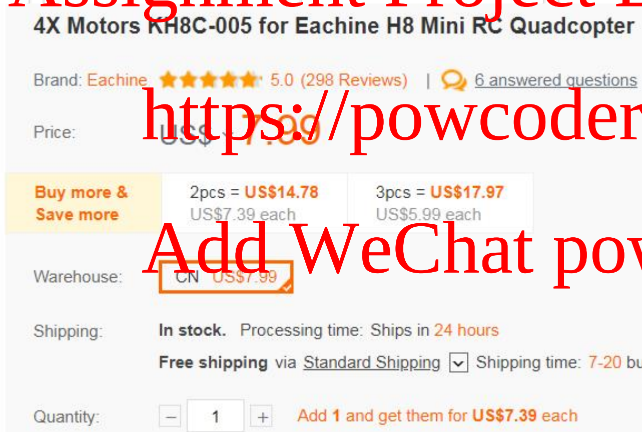
Surrogate models are generally much faster than simulations, and many optimization algorithms require really large numbers of simulations
(However, we often trade accuracy for speed)

Example: Quadcopter Motor Mass

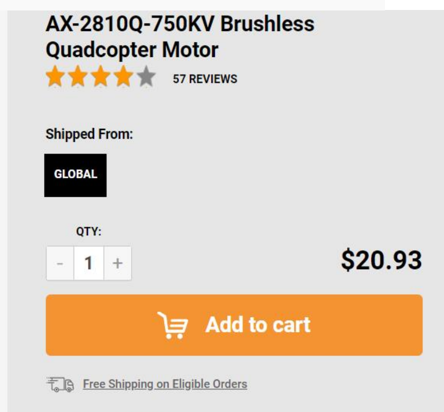
Example: Gather data







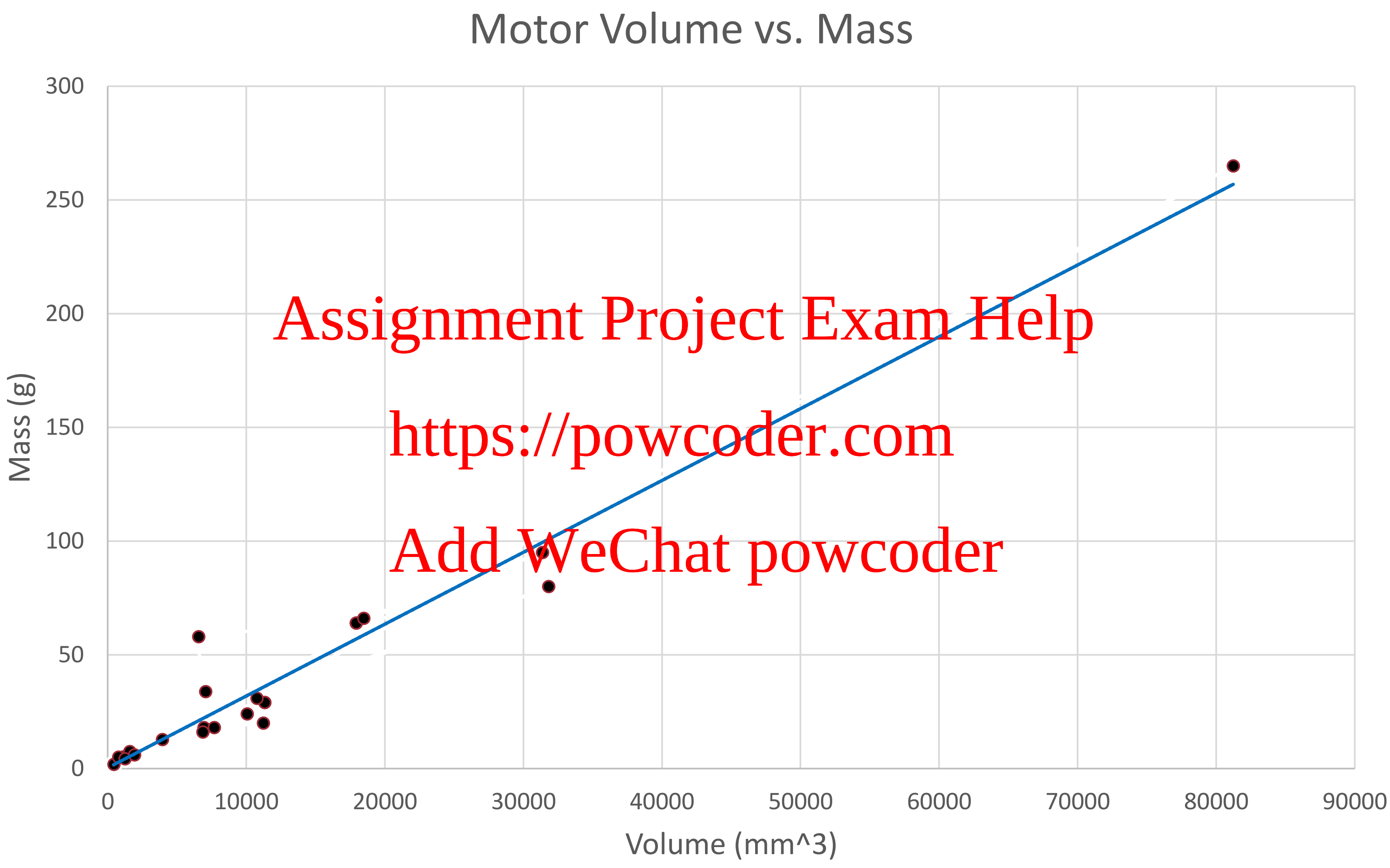
Specification :
Input: 2-4s Lipo
Constant: 15 Amps
Burst: 25 Amps
Weight: 4g bare board
Dimensions: 23x12.5x3mm
Motor wire: 70mm
Signal wire: 120mm
PCB - 4 Layer Blind hole PCB - 3oz Pour
Mosfet: N-Channel
OneShot and BLHeli 16.2 Pre Installed



Example: Gather data

Motor Name	Diameter (mm)	Length (mm)	Volume (mm^3)	Volume (m^3)	Mass (g) No wires
KDE1806XF-2350	23	16.7	6938.44299	6.9384E-06	18
KDE2304XF-2350	28.3	16	10064.2806	1.0064E-05	24
KDE2306XF-2550	28.3	18	11322.3156	1.1322E-05	29
KDE2306XF-2050	28.3	18	11322.3156	1.1322E-05	29
KDE2315XF-2050	28.3	28.5	17926.9997	1.7927E-05	64
KDE2315XF-965	28.3	28.5	17926.9997	1.7927E-05	64
KDE2814XF-775	35.5	31.7	31376.5977	3.1377E-05	95
BE2212	28	30	18472.5648	1.8473E-05	66
2400Kv 10A	22.5	16.5	6560.52903	6.5605E-06	58
MN1804 KV2400	23	16.5	6855.34787	6.8553E-06	16
MN1806 KV1400	23	18.5	7686.29913	7.6863E-06	18
F30 KV2800	23	27	11217.842	1.1218E-05	20
F40Ⅲ KV2400	28.4	17	10769.0026	1.0769E-05	30.8
F1000-Short Shaft KV545	53.6	36	81231.0303	8.1231E-05	265
Racerstar 8520	8.5	20	1134.90035	1.1349E-06	5.1
Racerstar 615	6	15	424.115008	4.2412E-07	1.8
Chaoli CL-1020	10	20	1570.79633	1.5708E-06	7.5
Chaoli CL 720	7	20	769.6902	7.6969E-07	5
EMAX RS2306	23	17	7063.08568	7.0631E-06	33.84
EMAX RS1306	18	15.5	3944.26958	3.9443E-06	12.7
MN3110-17	37.7	28.5	31813.9388	3.1814E-05	80
BE0905	12	11	1244.07069	1.2441E-06	4.2
BE1104	14	12.5	1924.2255	1.9242E-06	6

Example: Choose and fit function



mass = 0.0032*Vol + 0.2874

$g_1(r, h)$: $n*(0.0032*(\pi r^2 h) + 0.2874) \leq 150$

Surrogate modeling steps

1. Gather data

   Sample the design space by efficiently choosing which "experiments" to run

2. Choose a function structure

   E.g., linear regression, terms in a regression model, kriging, neural networks, etc.

3. Fit a function to data

   Find best-fit parameters for function structure

4. Assess fitness

   Measure how well the surrogate model fits the data

Surrogate modeling steps

1. **Gather data**

   Sample the design space by efficiently choosing which "experiments" to run

2. Choose a function structure

   E.g., linear regression, terms in a regression model, kriging, neural networks, etc.

3. Fit a function to data

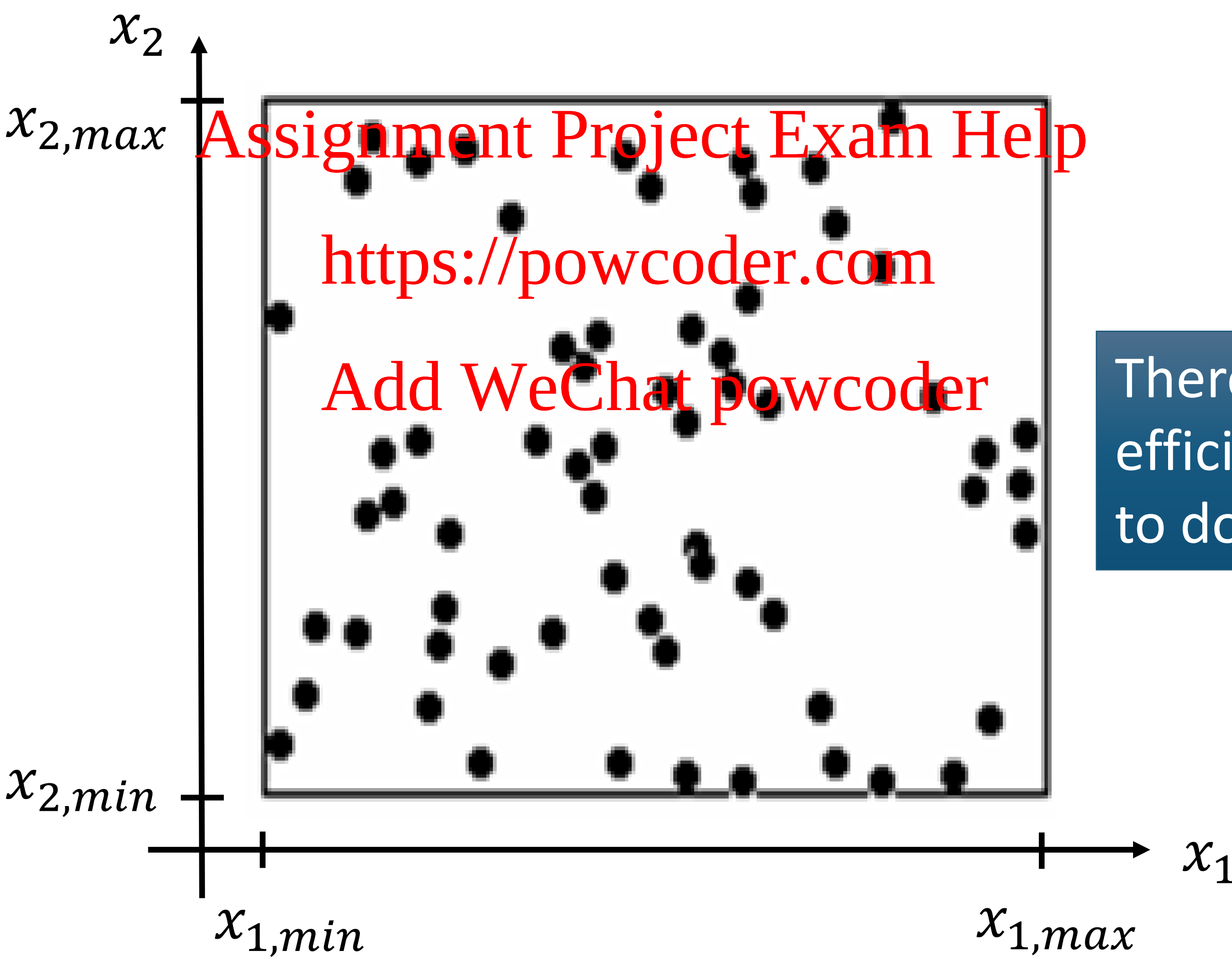
   Find best-fit parameters for function structure

4. Assess fitness

   Measure how well the surrogate model fits the data

Monte Carlo approach

Choose random experiments/points to sample



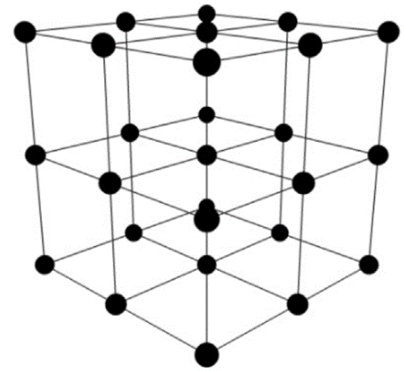
There are more efficient ways to do this!

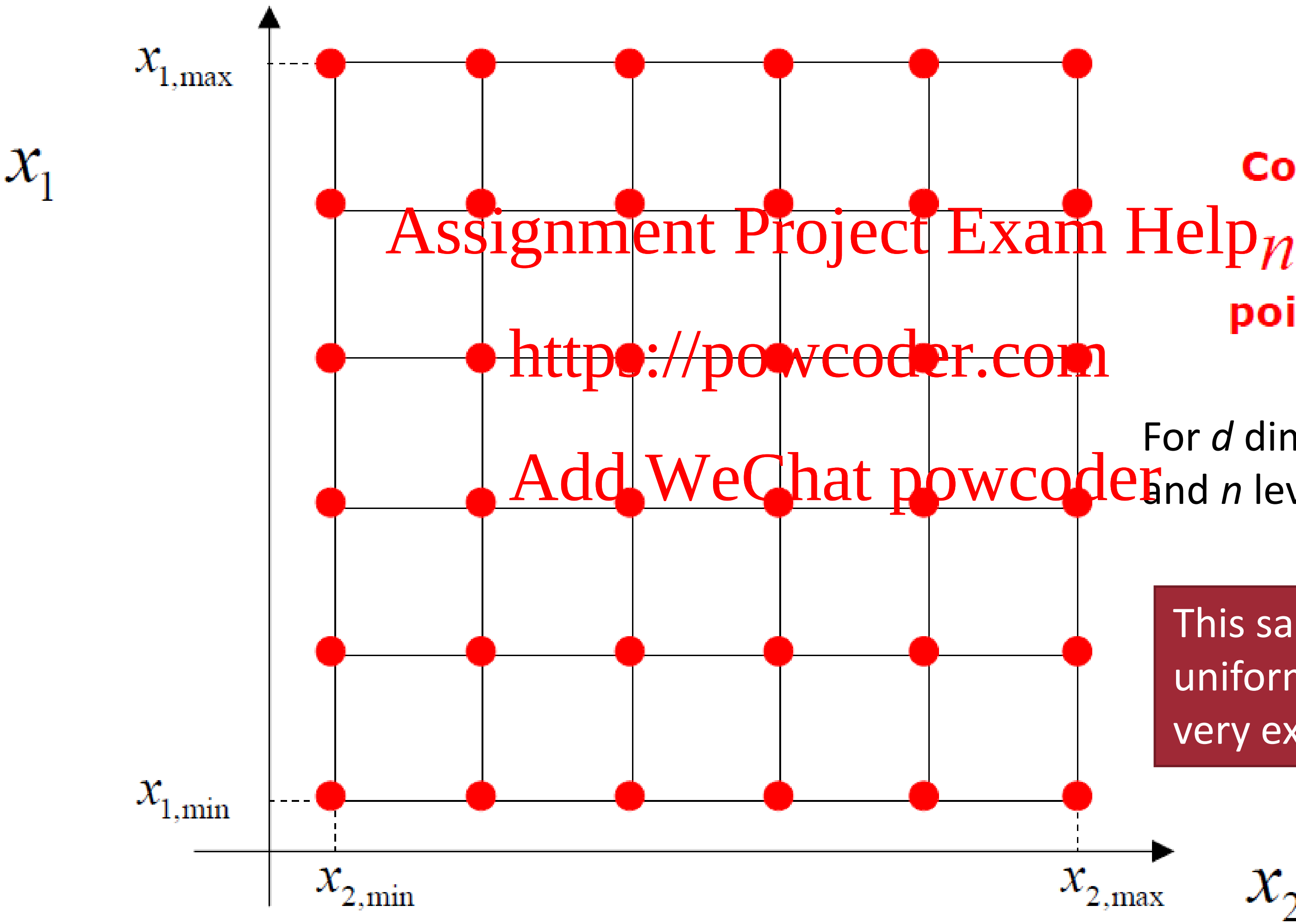
Gathering data

Design of Experiments (DOE): Collect a ***useful*** set of data that ***efficiently*** spans the design/input space

RunOrder	Pressure	Speed	Temp	Output
1	10	50	45	
2	20	50	45	
3	10	100	45	
4	20	100	45	
5	10	50	65	
6	20	50	65	
7	10	100	65	
8	20	100	65	

DOE: Full factorial





Cost:

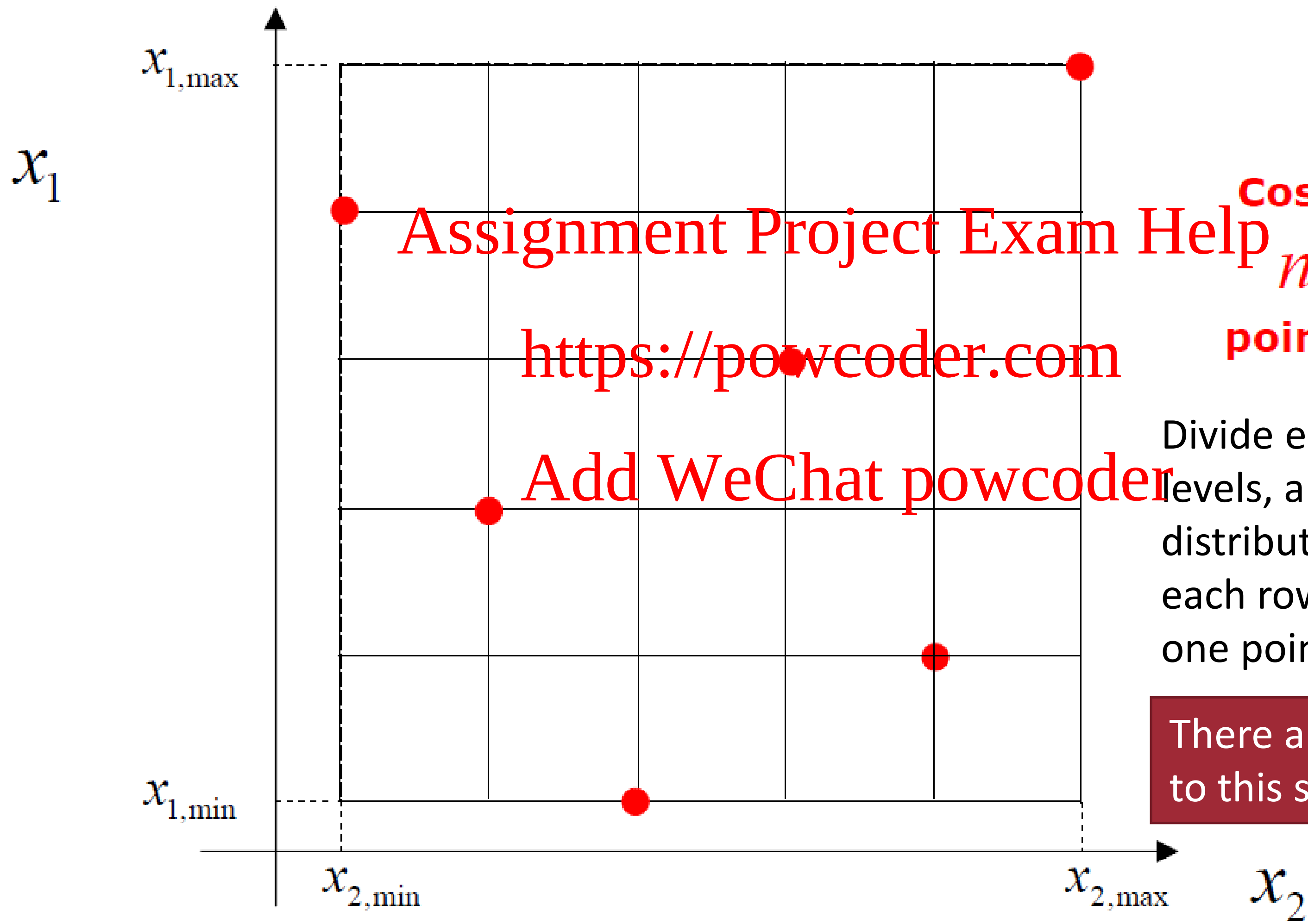
n^d

points

For *d* dimensions/variables and *n* levels for each variable

This samples the space uniformly, but can get very expensive

DOE: Latin hypercube

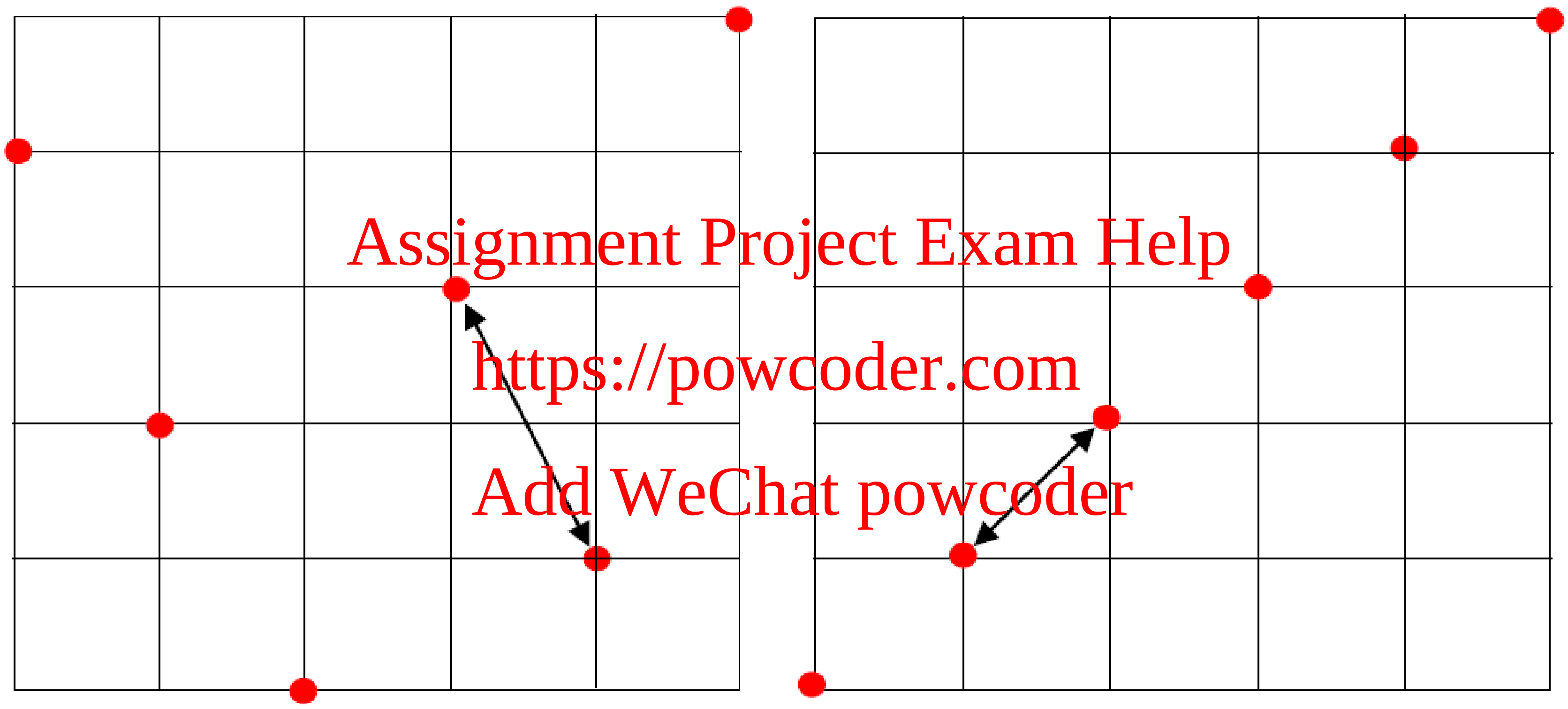


Cost:
n
points

Divide each variable into n levels, and then randomly distribute points such that each row and column has one point

There are many solutions to this sampling problem

DOE: Optimal Latin hypercube



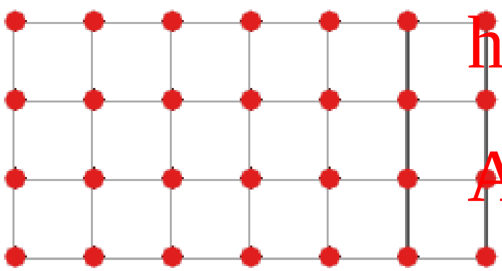
Maximize the minimum distance between any two points

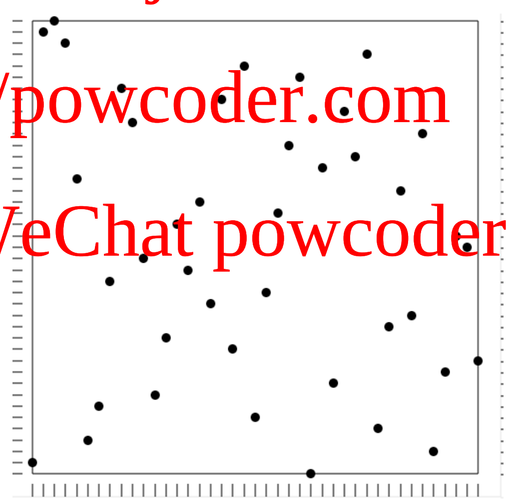
DOE: MATLAB codes

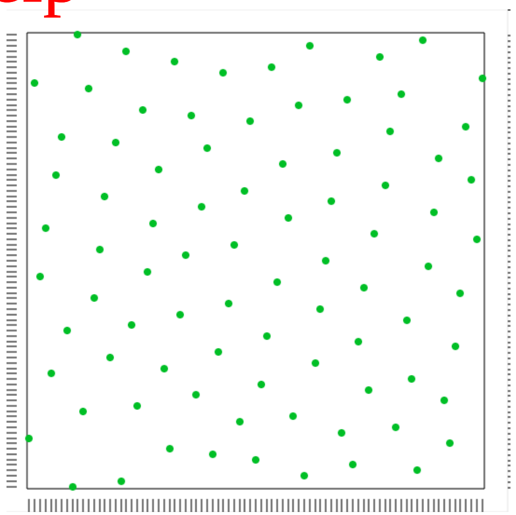
Design of Experiments (DOE): Collect a useful set of data that efficiently spans the design/input space

Full Factorial

Latin Hypercube/Optimal Latin Hypercube







design = fullfact([4,7])

for 2 variables: 4 levels on the first and 7 on the second

design = lhsdesign(n,p,‘iterations’,20,‘criterion’,‘maximin’)

for n samples and p variables

Adaptive Sampling

Instead of deciding the sampling points at the beginning, you can choose them on the way.

Efficient Global Optimization (EGO)

- Assume that you want to find minimum of $y = f(\mathbf{x})$

- Start with an initial set of points $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots \mathbf{x}^{(k)}\}$

- Fit a a kriging model to those points
- Find the point that maximizes the **expected improvement** using a merit function.
- Sample at this point and continue …

We will discuss this again in Week 12

Surrogate modeling steps

1. Gather data

   Sample the design space by efficiently choosing which "experiments" to run

2. **Choose a function structure**

   E.g., linear regression, terms in a regression model, kriging, neural networks, etc.

3. **Fit a function to data**

   Find best-fit parameters for function structure

4. Assess fitness

   Measure how well the surrogate model fits the data

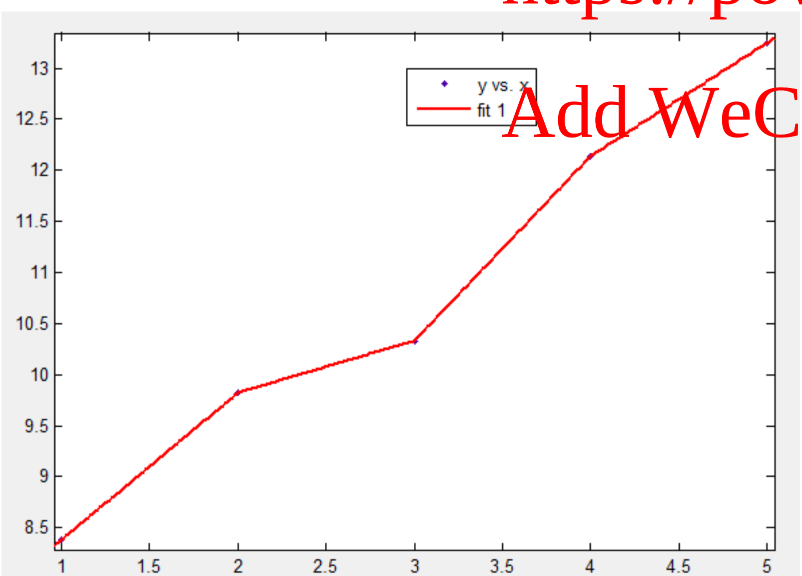
Some common surrogate model structures

- Interpolation
- Regression
- Artificial neural networks (ANNs)
- Kriging

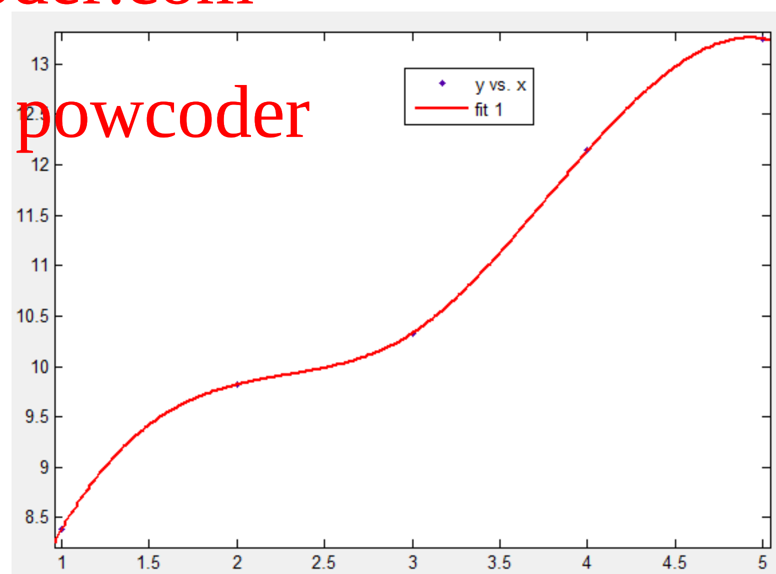
Interpolation

Use data points, and estimate in between them

yi = interp1(x,y,xi,'spline') – supply input and output data (x,y), choose among methods like linear or spline, supply new inputs to be estimated (xi)



linear



cubic spline

Linear regression

Design Space

$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(k)}$

data

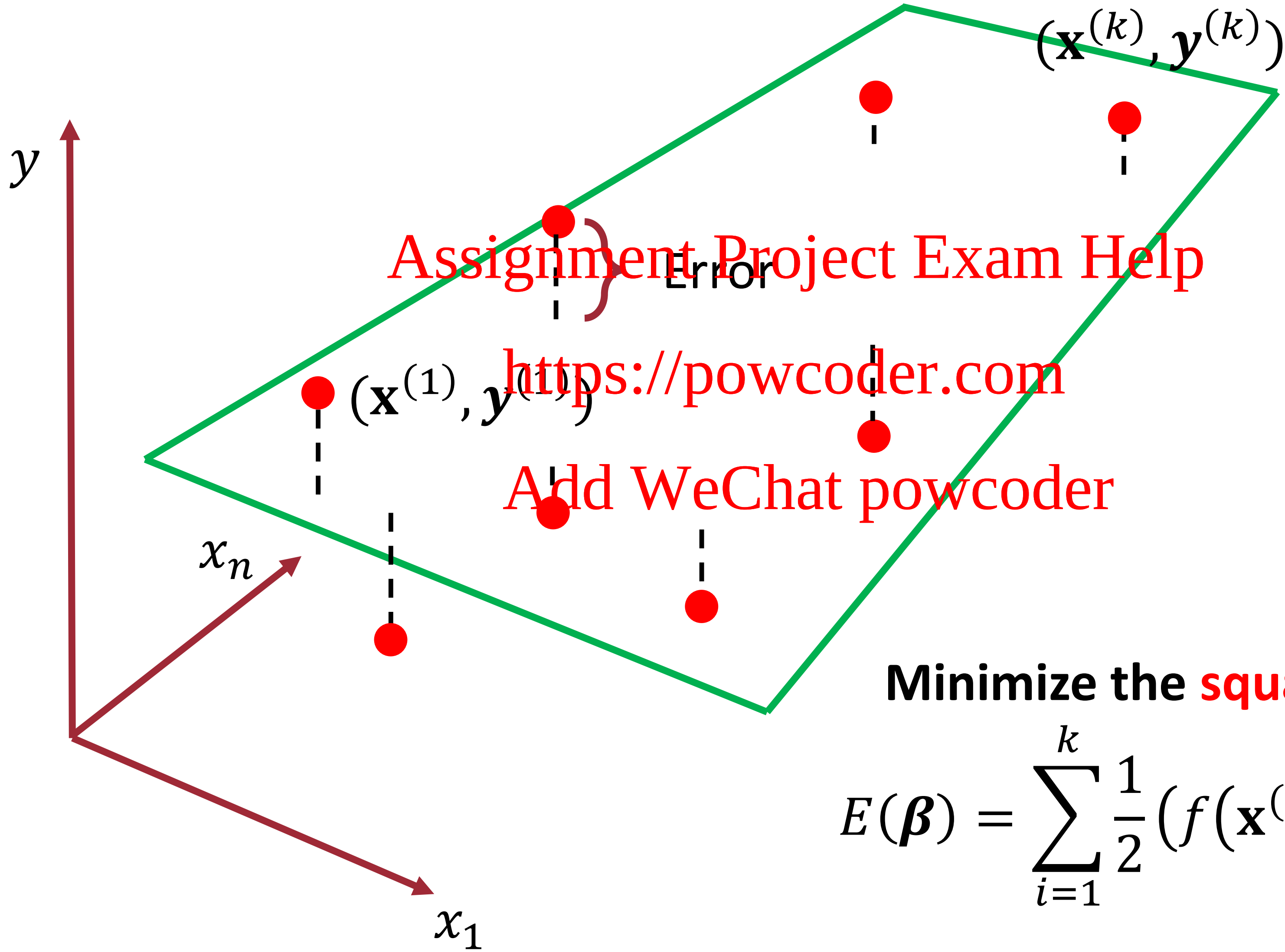
Output Space

$y^{(1)}, y^{(2)}, \ldots, y^{(k)}$

- Finding parameters ($\boldsymbol{\beta}$) to make a function of $\mathbf{x}$ best match the data points of $\mathbf{y}$
- Linear model: $f(\mathbf{x}) = \boldsymbol{\beta}^{\boldsymbol{T}}\mathbf{x}$
- Nonlinear model: $f(\mathbf{x}) = \boldsymbol{\beta}^{\boldsymbol{T}}\varphi(\mathbf{x})$

A good model should closely match $f(\mathbf{x})$ to $\mathbf{y}$

Linear Regression



Minimize the square error:

$$E(\boldsymbol{\beta}) = \sum_{i=1}^{k} \frac{1}{2}\left(f\left(\mathbf{x}^{(i)}, \boldsymbol{\beta}\right) - y^{(i)}\right)^2$$

Linear Regression

Modeling becomes an optimization problem

$$E(\boldsymbol{\beta}) = \sum_{i=1}^{k} \frac{1}{2}\left(f\left(\mathbf{x}^{(i)}, \boldsymbol{\beta}\right) - y^{(i)}\right)^2$$

$$E(\boldsymbol{\beta}) = \sum_{i=1}^{k} \frac{1}{2}\left(\boldsymbol{\beta}^T \mathbf{x}^{(i)} - y^{(i)}\right)^2$$

We will discuss this in week 5. This is the first order necessary condition: the derivative equals zero

Design variable

$$\frac{\partial E(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \begin{bmatrix} \mathbf{x}^{(1)}\boldsymbol{\beta} - y^{(1)} \\ \vdots \\ \mathbf{x}^{(k)}\boldsymbol{\beta} - y^{(k)} \end{bmatrix}^T = \mathbf{0}^T$$

Linear Regression

Modeling becomes an optimization problem with a closed-form solution

$$\frac{\partial E(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \begin{bmatrix} \mathbf{x}^{(1)}\boldsymbol{\beta} - y^{(1)} \\ \vdots \\ \mathbf{x}^{(k)}\boldsymbol{\beta} - y^{(k)} \end{bmatrix}^T = \mathbf{0}^T$$

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_n^{(1)} \\ \vdots & \ddots & & \vdots \\ 1 & x_1^{(k)} & \dots & x_n^{(k)} \end{bmatrix} \qquad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_n \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(k)} \end{bmatrix}$$

$$\boldsymbol{\beta}_* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \longrightarrow$$

Linear coefficients for least square error

Example

Find a linear model to predict the exam grade given HW grade.

Student	HW Grade	Exam Grade
1	95	85
2	85	95
3	80	70
4	70	65
5	60	70
	$x^{(i)}$	$y^{(i)}$

A linear model of 1 variable has 2 coefficients: β_0, β_1

$$f(\boldsymbol{x}) = \beta_0 + \beta_1 x$$

Recall: $\boldsymbol{\beta}_* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$

Example

HW Grade	Exam Grade
95	85
85	95
80	70
70	65
60	70

$x^{(i)}$ $\quad$ $y^{(i)}$

$$\boldsymbol{X} = \begin{bmatrix} 1 & 95 \\ 1 & 85 \\ 1 & 80 \\ 1 & 70 \\ 1 & 60 \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} 85 \\ 95 \\ 70 \\ 65 \\ 70 \end{bmatrix}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

$\beta_0, \beta_1 = ?$

$$\boldsymbol{\beta}_* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \begin{bmatrix} 26.78 \\ 0.64 \end{bmatrix}$$

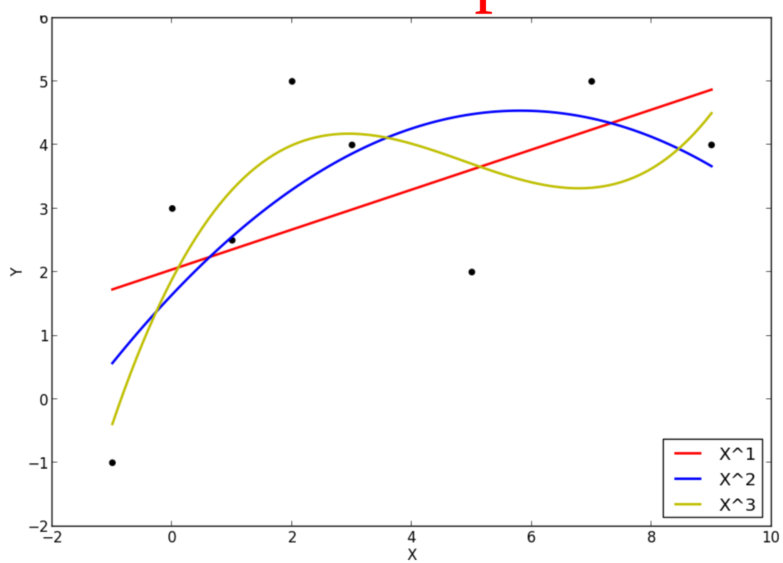
$$Exam\ Grade = 26.78 + 0.64(HW\ Grade)$$

Nonlinear models

You can do linear regression with more complex models: just reform your x matrix to have the appropriate columns:

$$\mathbf{x} = [x_1^2 \;\; x_1 \;\; x_1 x_2 \; x_2^2 \; x_2 \;\; \sin(x_1)]$$

$$\hat{y} = \beta_1 x_1^2 + \beta_2 x_1 + \beta_3 x_1 x_2 + \beta_4 x_2^2 + \beta_5 x_2 + \beta_6 \sin(x_1)$$



Regression in MATLAB

Use the backslash operator \ for linear regression:

With input data matrix x and output vector y, command:

beta = x\y

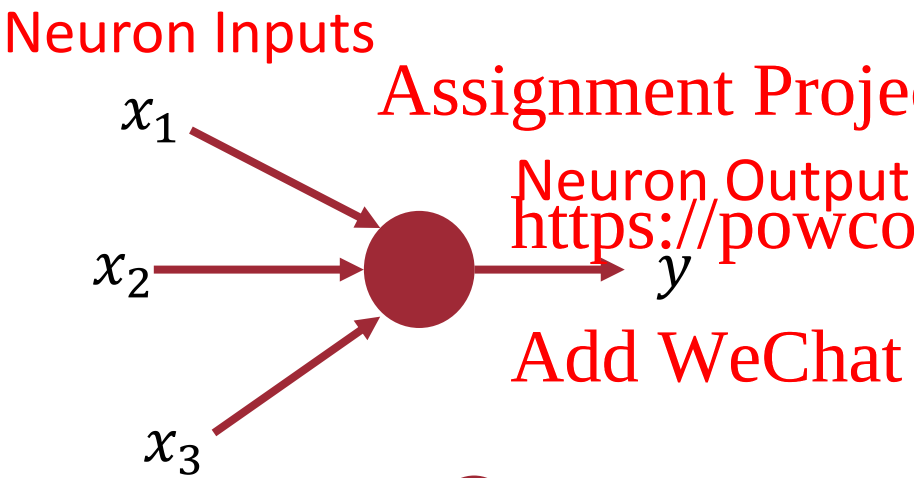
will output a list of beta values such that:

$$\hat{y} = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots$$

You can also use **cftool** to open a graphical user interface for the curve-fitting toolbox

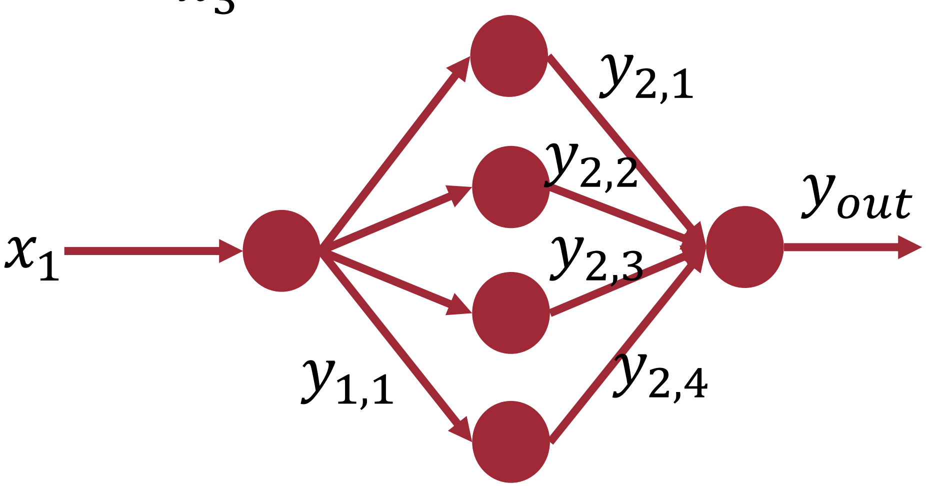
Artificial Neural Networks (ANNs)

When you don't know the structure of your function (e.g., which terms to include), ANNs offer an automated way



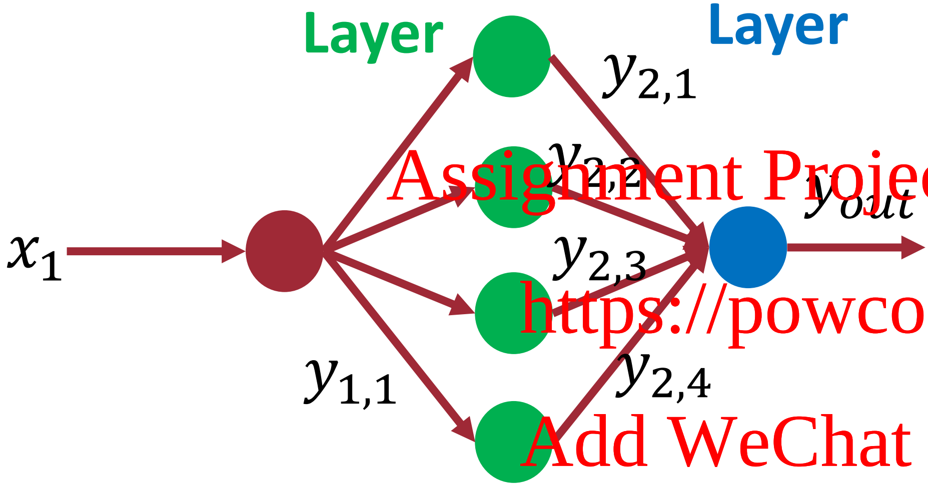
Each neuron has a transfer function

$$f_{i,j}(\mathbf{x}) = y_{i,j}$$

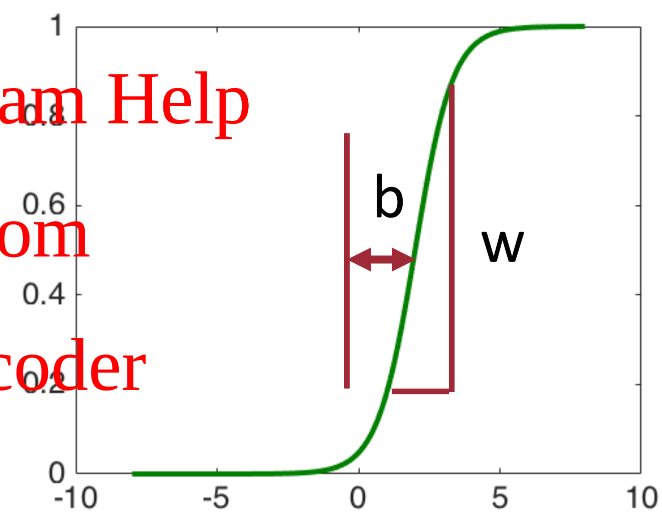


A chain of these functions gives the final model

Neural Network Model



Transfer Function (logistic)



$$y_{1,1} = \left(1 + e^{b_{1,1} - w_{1,1}x}\right)^{-1}$$

$$y_{2,1} = \left(1 + e^{b_{2,1} - w_{2,1}y_{1,1}}\right)^{-1}$$

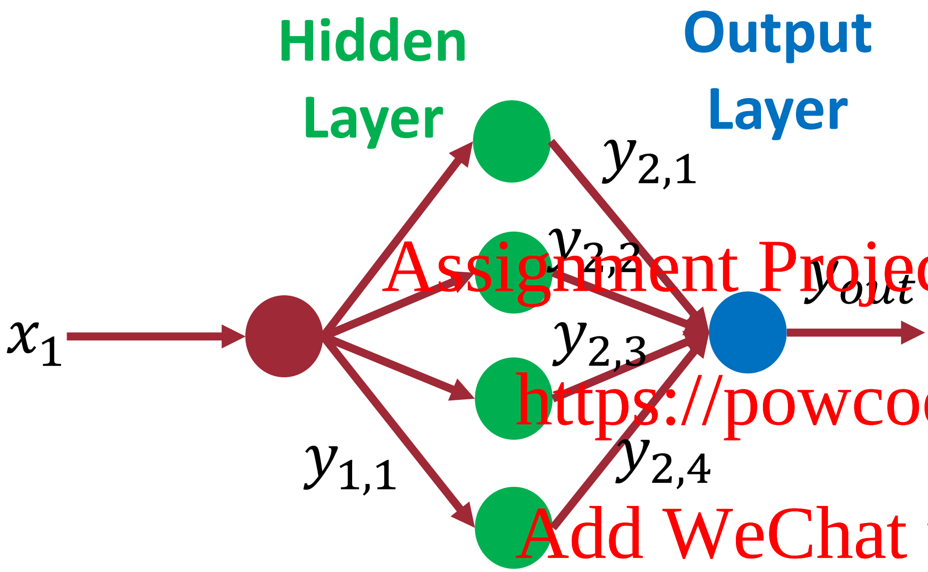
$$\ldots$$

$$y_{2,4} = \left(1 + e^{b_{2,4} - w_{2,4}y_{1,1}}\right)^{-1}$$

Hidden layer function

$$0 \leq y_{2,i} \leq 1$$

Neural Network Model



Final output

$$y_{out} = b_{3,1} - \sum_{i=1}^{4} w_{2,i} y_{2,i}$$

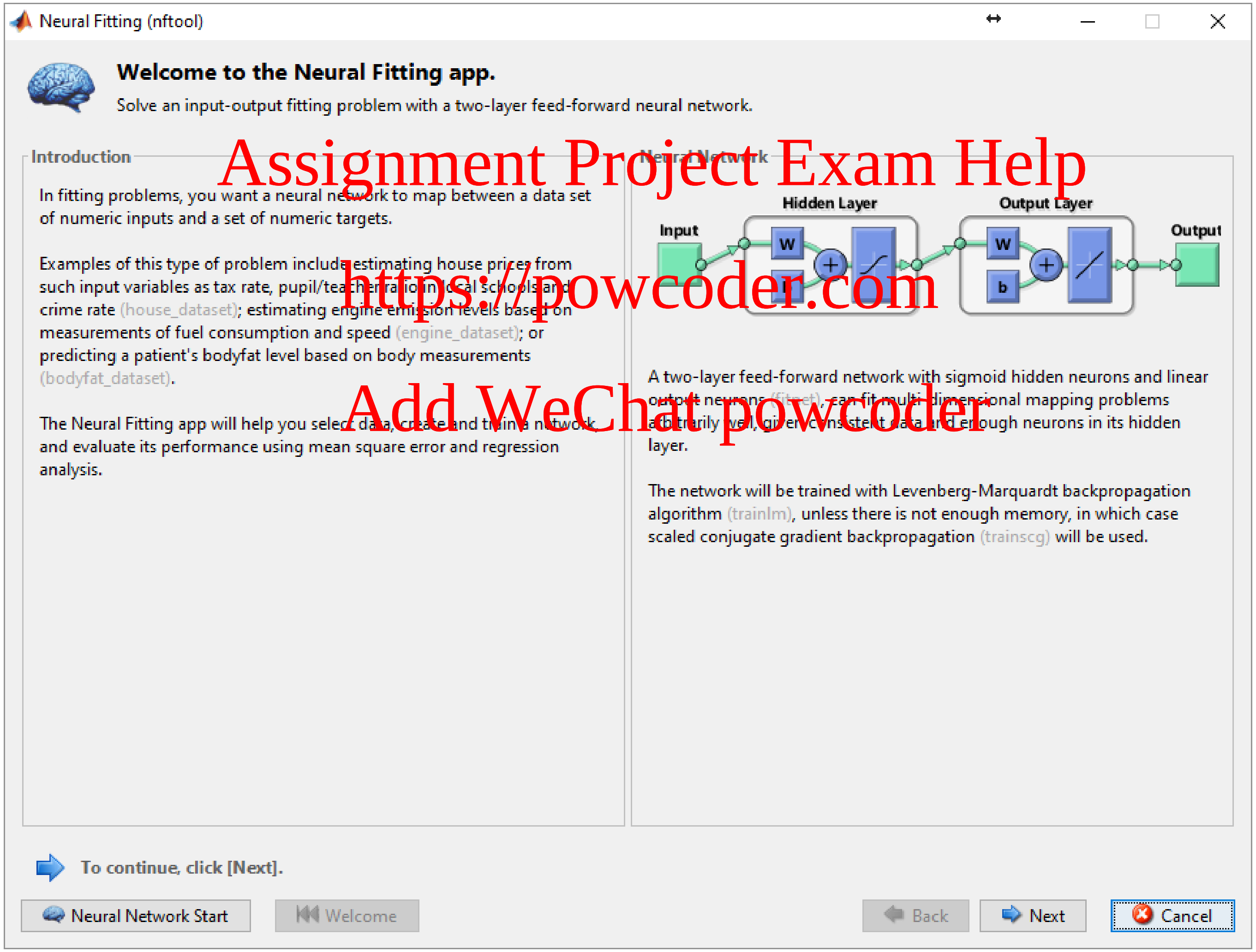
Can take any desired value based on **bias** and **weight** values

Find the set of $b_{i,j}$ and $w_{i,j}$ to minimize the square error

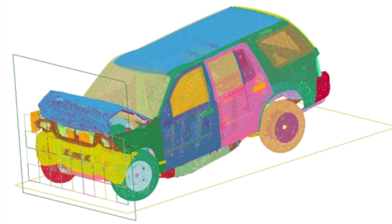
$$Error\left(b_{i,j}\,, w_{i,j}\right) = \sum_{i=1}^{k} \left(y^{(i)} - y_{out}\left(\mathbf{x}^{(i)}\right)\right)^2$$

Neural Networks with MATLAB

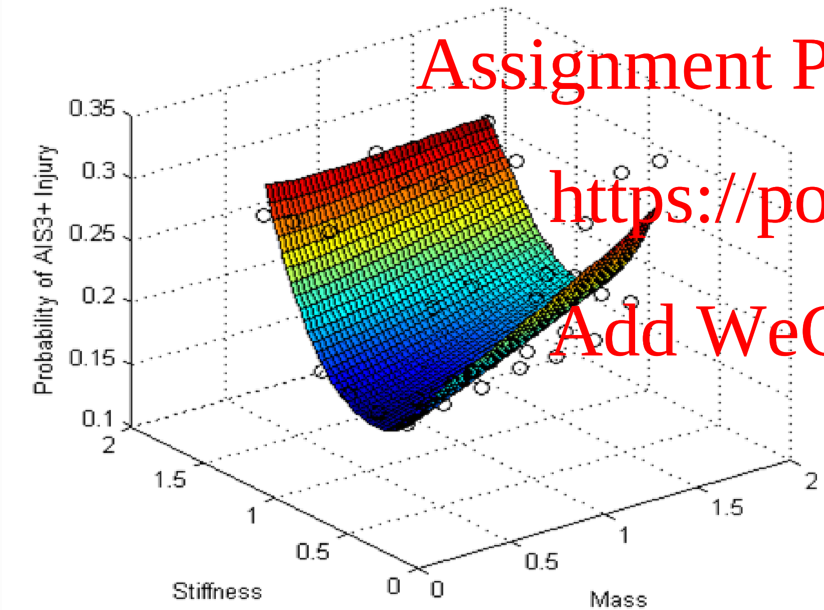
MATLAB has an easy-to-use neural network toolbox: **nftool**



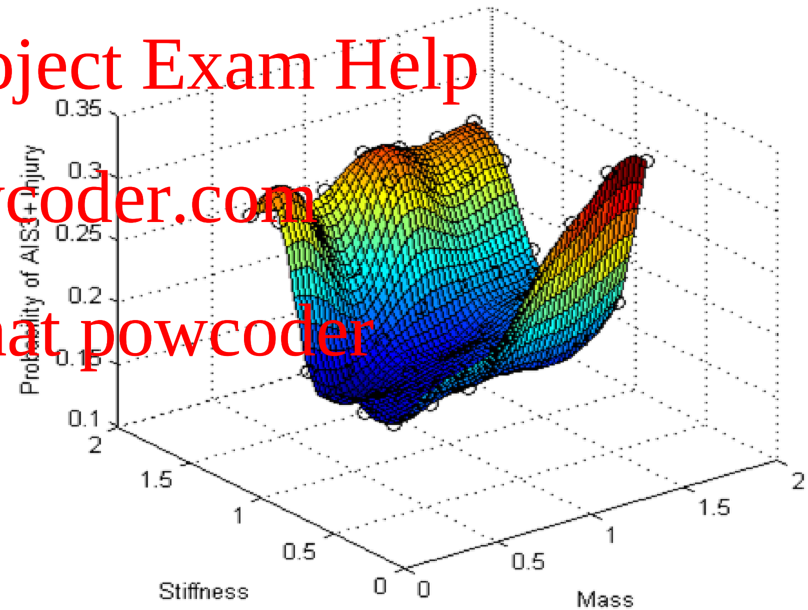
Example: Linear regression vs. ANN



64-point, 2-variable full factorial sample (?)



Least-squares linear regression w/ second-order terms



Artificial Neural Network w/ radial basis functions (exact fit)

Kriging

Typical surrogate models have a prediction error, ε, which is assumed to be independent:

$$y(x) = f(x) + \varepsilon$$

Non-exact-fitting function (e.g., linear, polynomial)

Independent, identically distributed (i.i.d.) random error

In kriging, ε is correlated with x:

$$y(x) = f(x) + Z(x)$$

Non-exact-fitting function (e.g., linear, polynomial)

Exact-fit error element that is a function of x

Kriging

- There are available software packages (e.g., ModelCenter) or public MATLAB codes for kriging implementation

https://www.mathworks.com/matlabcentral/fileexchange/?term=kriging

- Kriging goes through data points exactly
  - Good for data from deterministic simulation models
  - Bad for data with measurement error or noise

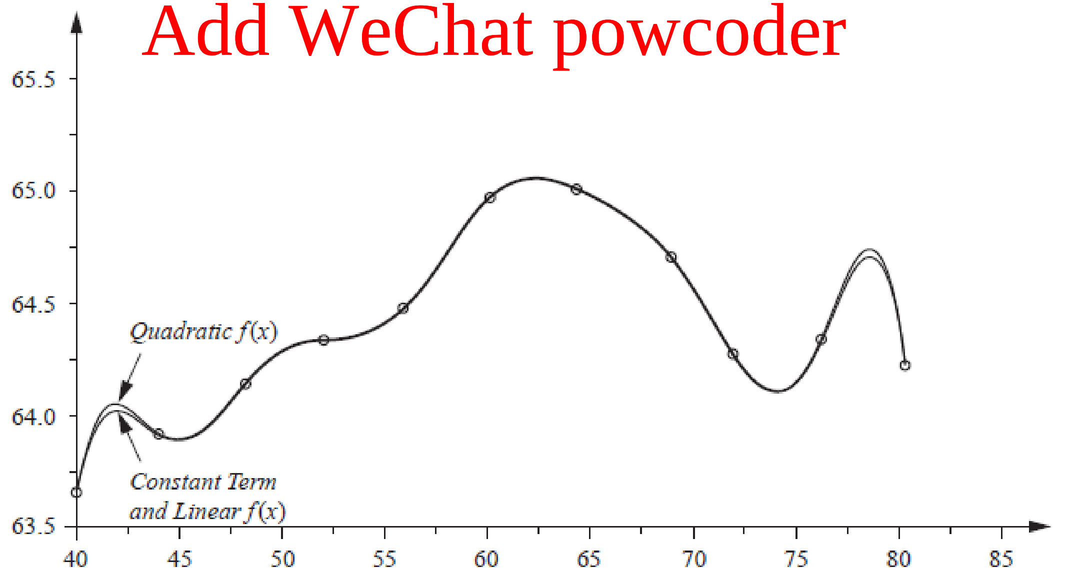


Figure 2.8. Kriging metamodels with different polynomials.

Surrogate modeling steps

1. Gather data

   Sample the design space by efficiently choosing which "experiments" to run

2. Choose a function structure

   E.g., linear regression, terms in a regression model, kriging, neural networks, etc.

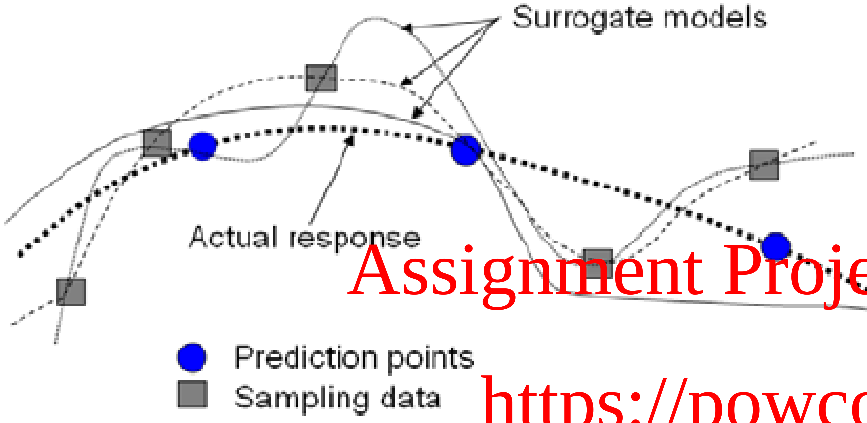
3. Fit a function to data

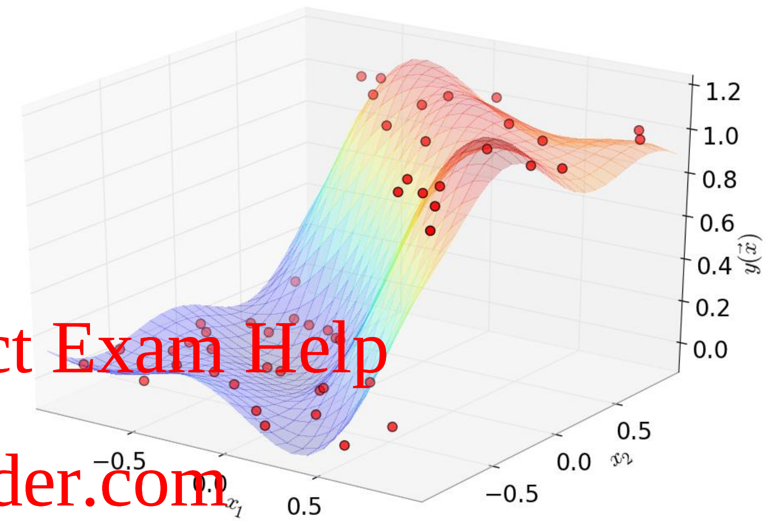
   Find best-fit parameters for function structure

4. **Assess fitness**

   Measure how well the surrogate model fits the data

Visualizing error





- With 3 or more variables, visualization is difficult
  - Can project to two dimensions
  - Can use eigenvectors to reduce dimensionality
- Fitness can be mathematically calculated
  - Standard error
  - Mean square error
  - Cross-validation

Right image: Samad, A., Choi, J. H., & Kim, K. Y. (2008). Blade Optimization of a Transonic Compressor Using a Multiple Surrogate Model. Transactions of the Korean Society of Mechanical Engineers B, 32(4), 317-326.

Left image: https://www.datadvance.net/product/macros/manual/6.5sp1/_images/initial_model_approximation.png

Fitness: R^2

Coefficient of determination

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

This is a measure of how well the model $\hat{y}_i$ fits compared to simply using the average value $\bar{y}$.

Adjusted R² *accounts for model complexity*

$$R^2{}_{adj} = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

where *n* is the number of data points and *p* is the number of parameters in the model

This is a common evaluation metric with regression models

Fitness: Cross-validation

1. Divide the data set into *n* subsets
2. Fit a model using all but the *i^th^* subset
3. Use the *i^th^* subset to test fitness
4. Repeat *n* times, average the results

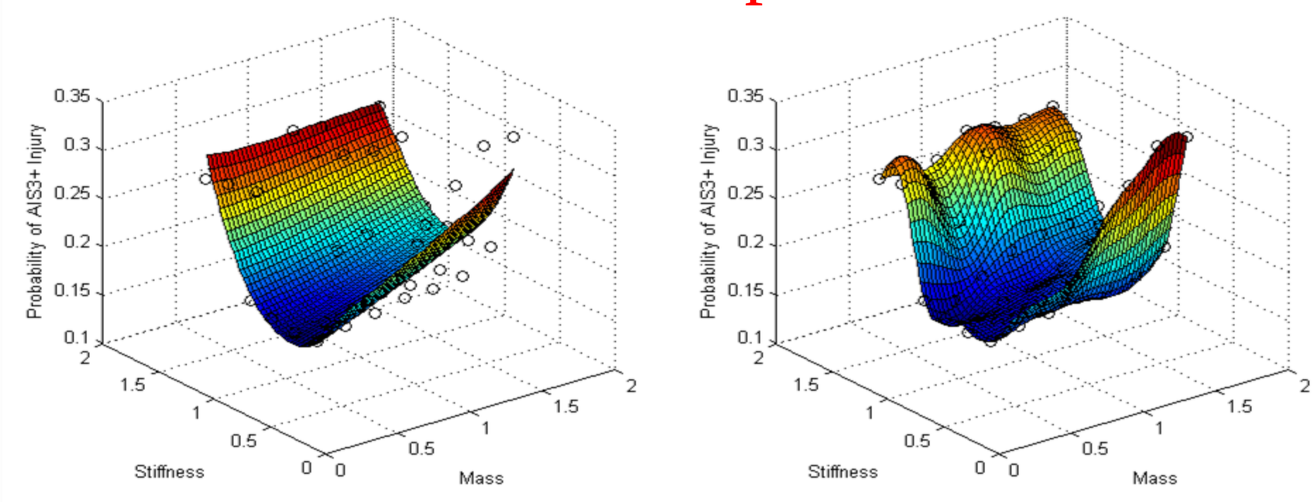
Fitness often measured in terms of ***mean square error***:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

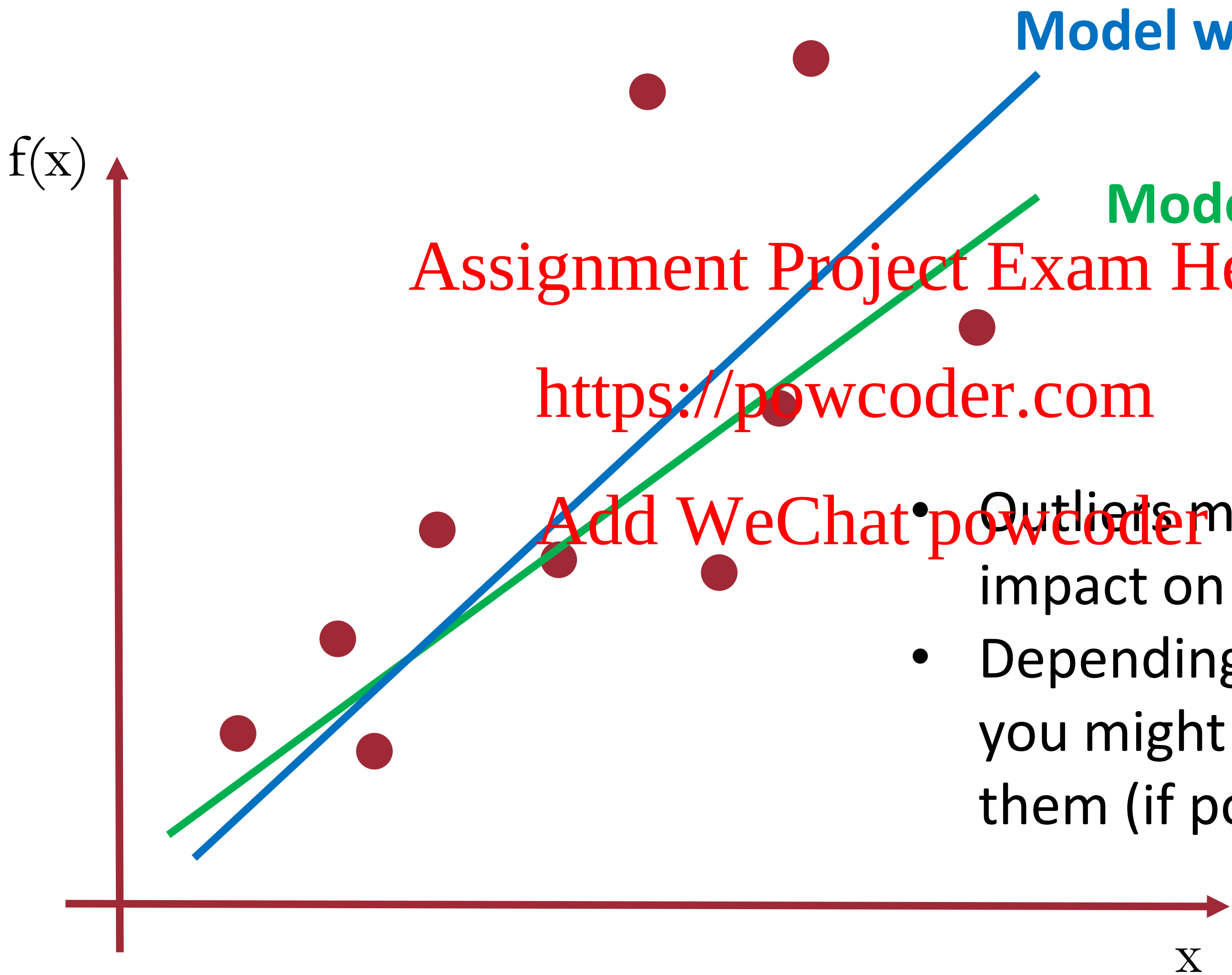
This is a common evaluation metric with exact-fit models

Some metamodeling considerations

- Outliers
- Underfitting
- Overfitting
- Training vs validation data
- Regularization

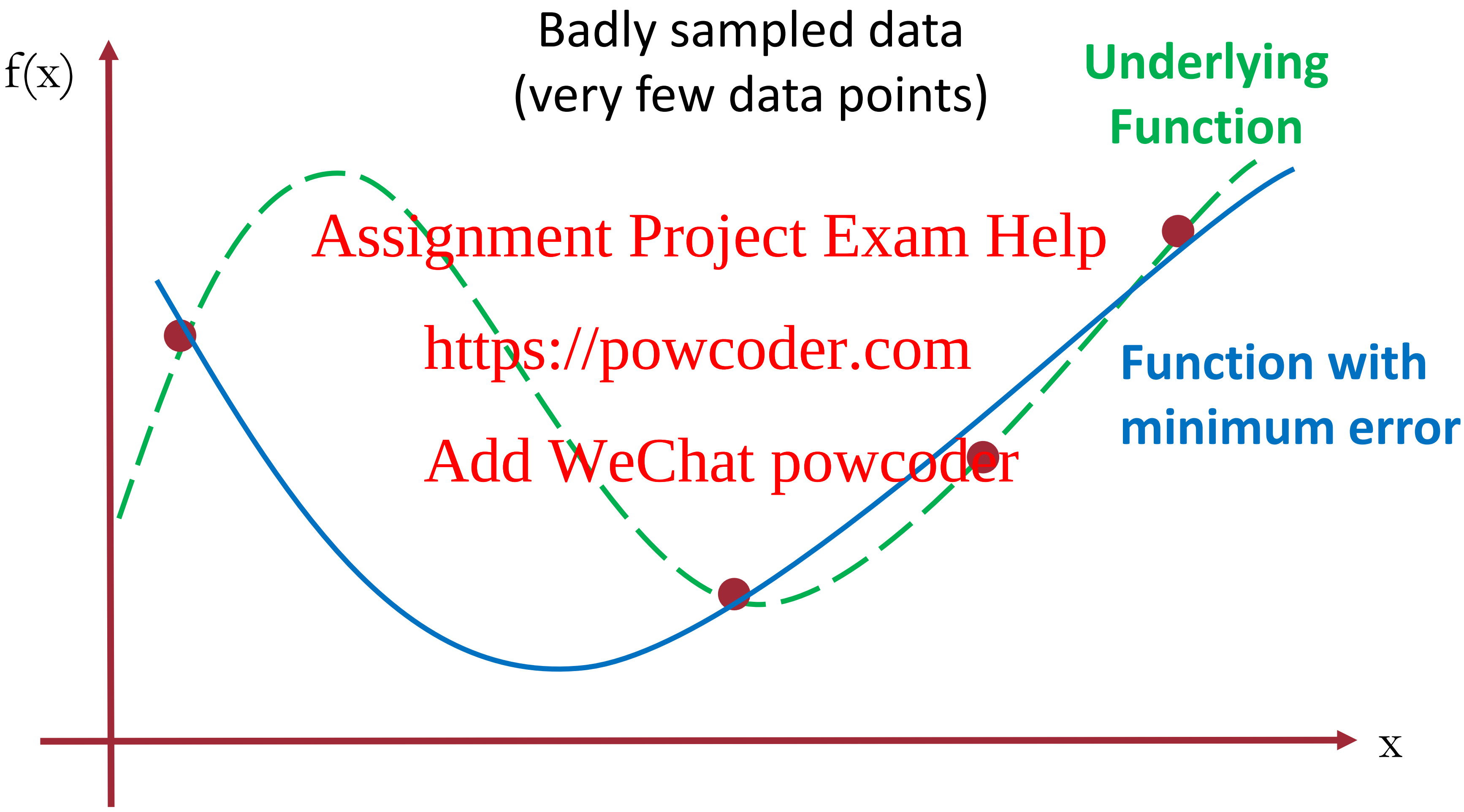


Outliers



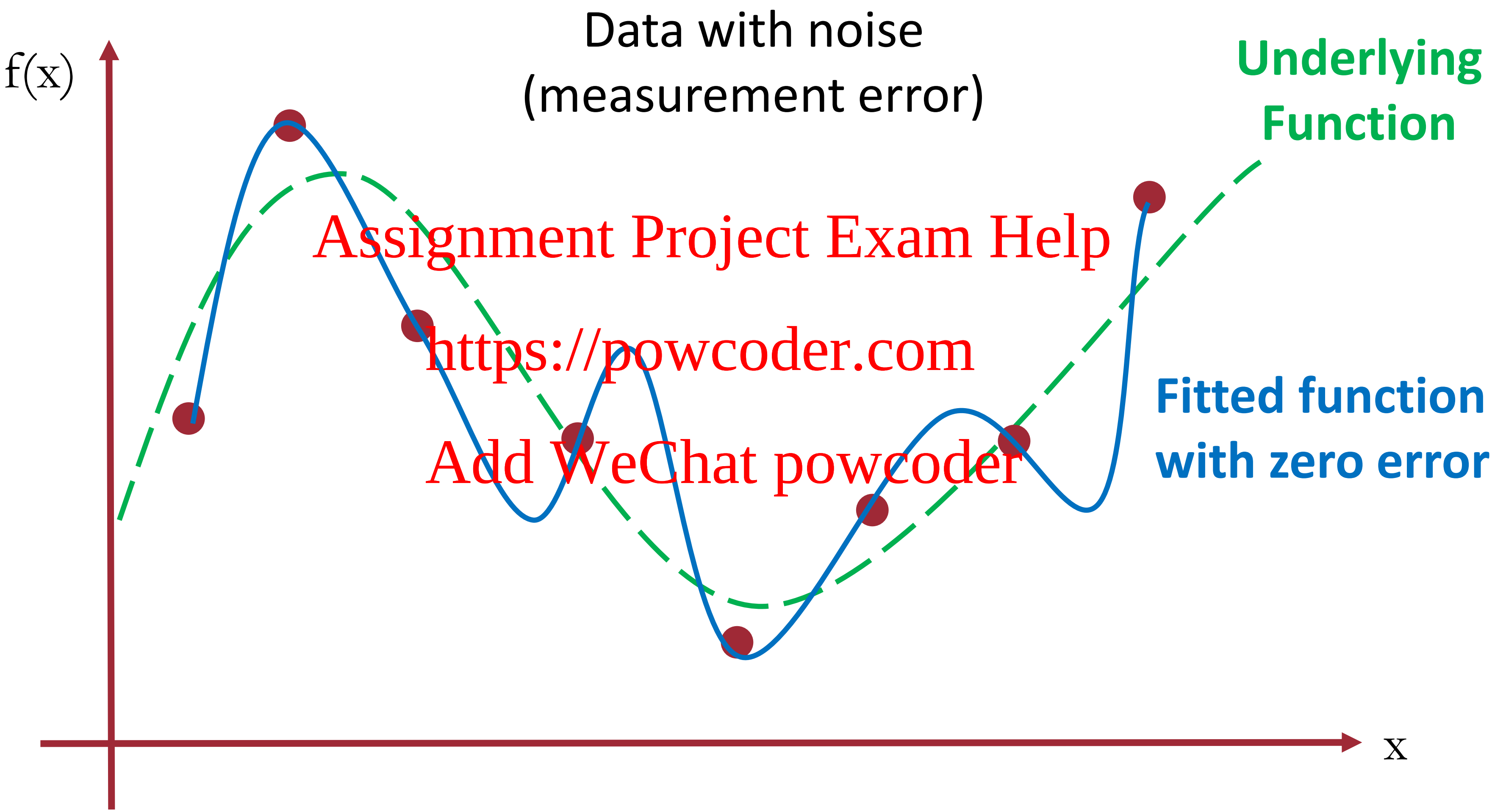
- Outliers might have a big impact on the model
- Depending on the problem you might want to remove them (if possible)

Underfitting



Sampling size should be sufficient to capture the function behavior

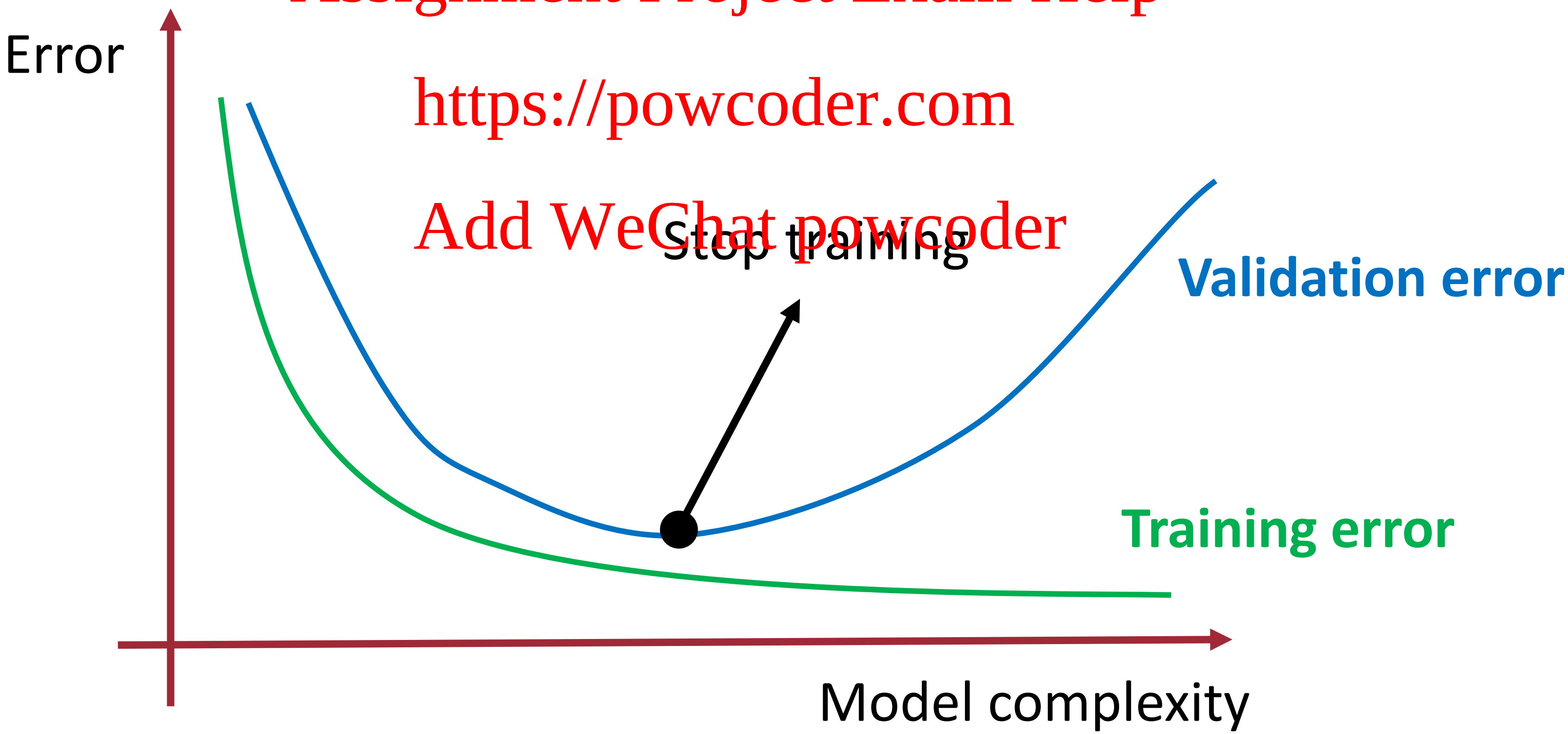
Overfitting



- Overfitting captures the noise instead of the underlying function
- Unnecessary model complexity might also cause overfitting

Training vs Validation Data

- To overcome overfitting entire data set is separated into **training** and **validation** data
- Model is developed for training data



Regularization

Can we find an optimal complexity for our model?

Consider the following linear regression problem

$$Error(\boldsymbol{\beta}) = \sum_{i=1}^{k} (\boldsymbol{\beta}^T \mathbf{x}^{(i)} - y^{(i)}) + \lambda \|\boldsymbol{\beta}\|$$

Penalty for complexity

where $\mathbf{x} = [1, x, x^2, x^3, x^4 \ldots, x^n]^T$

Increasing λ will force to find simpler models

Metamodeling Steps (for ANNs)

1) If you do not have data already, sample the space using any sampling method, e.g., Latin hypercube
2) Plot data to see if outliers exist; remove if you can
3) Normalize your data
4) Separate your data into training and validation/test (90% training, 10% validation is a good start)
5) Train your model using a metamodeling method, e.g. neural networks
6) If error is not good enough, either add more data or increase model complexity

Additional resources

Model fitting and regression in MATLAB (9-min video; includes importing Excel data, plotting 2-d, examining goodness of fit, using the "basic fitting" plot tool):

http://www.youtube.com/watch?v=NsT5BAofRN0

MATLAB Neural Network Toolbox:

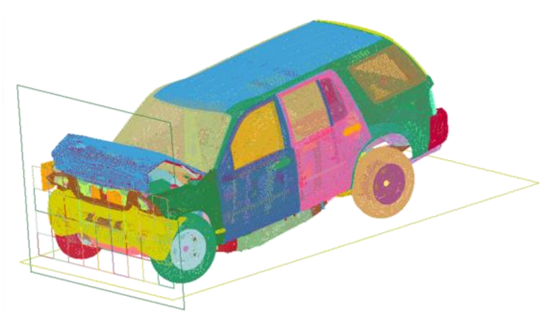
http://www.mathworks.com/videos/getting-started-with-neural-network-toolbox-68794.html

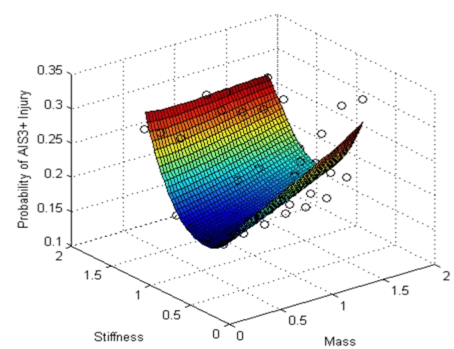
Kriging:

http://www2.imm.dtu.dk/~hbni/dace/

Summary

- **Metamodeling** is fitting a mathematical function to your data to speed up evaluations and optimization
- This involves **four general steps**:
  - Gather data (e.g., using a DOE)
  - Choose a function structure (e.g., linear, polynomial, kriging, ANN)
  - Fit a function to the data
  - Assess fitness
- **Watch out** for outliers, underfitting, and overfitting



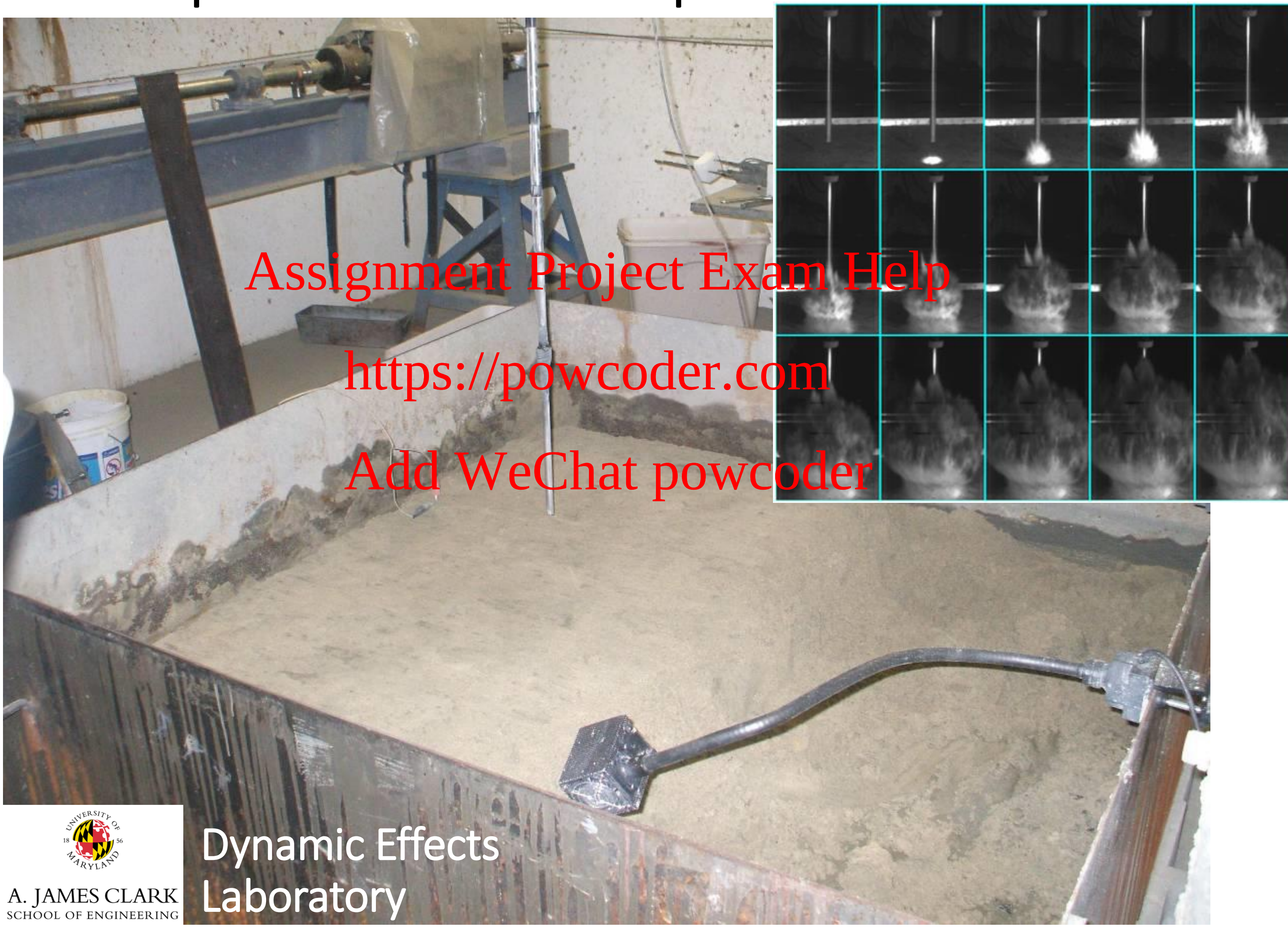


Workshop task

Use the DeJong function from the Yang 2010 paper: http://arxiv.org/pdf/1008.0549.pdf (if n-dimensional, choose n = 2)

1. Sample the function
   a. Using full factorial sampling (e.g., 4 x 4)
   b. Using an optimal Latin hypercube (e.g., 16 points)
2. Fit regression metamodels to each sample (try different function structures and sample sizes) and measure the R^2 value and MSE
3. Plot the surfaces to show which sampling method and which metamodeling technique seems best
4. If you have time, also fit metamodels using neural nets or kriging and measure the cross-validated MSE
5. Summarize your findings – at the end we will meet as a class and hear from each of you: (1) which test functions you tried, (2) which sampling methods and (3) metamodels you used, and (4) what seemed to work

Example: Buried explosive tests



A. JAMES CLARK SCHOOL OF ENGINEERING

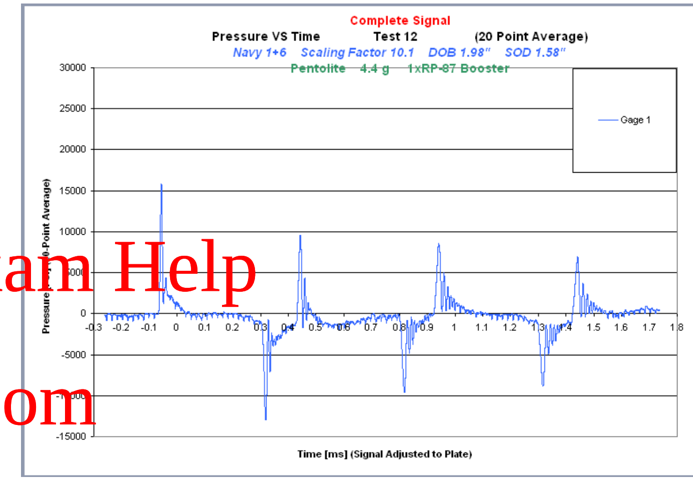
Dynamic Effects Laboratory

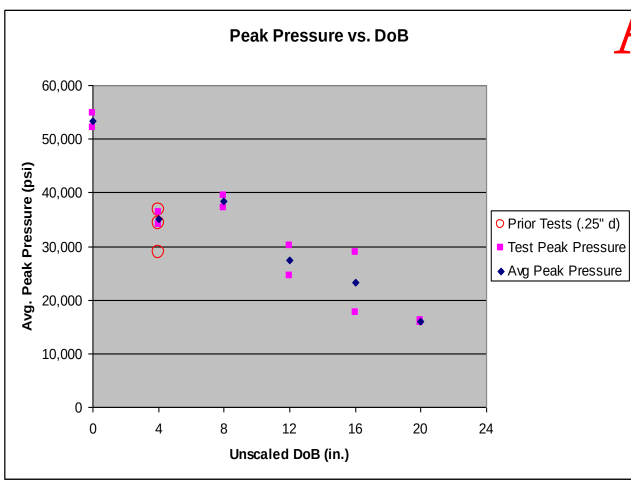
Example: Gather data

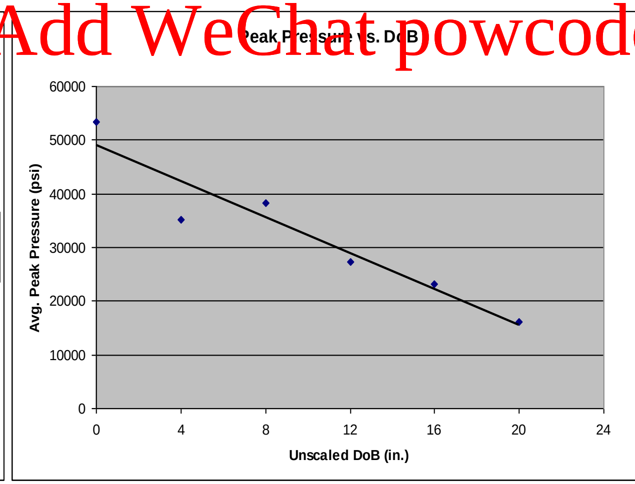
Test No.	Date	Charge DoB (in.)	Rod SoD (in.)	Peak Pressure (psi)	Pressure Integrated Impulse (lbs)	Rod Velocity Impulse (lbs)	Rod Peak Height Impulse (lbs)	Max Sand Vel. (in./s)	Min Sand Vel. (in./s)
1	02/17/07	0.39	1.58	-	-	-	-	33,643	29,381
2	02/23/07	0.39	1.58	-	-	-	-	35,310	31,155
3	02/26/07	0.39	1.58	-	-	-	-	37,995	30,529
4	03/05/07	0.39	1.58	-	-	-	-	30,564	28,625
5	03/07/07	0.39	1.58	-	-	-	-	30,592	29,963
6	03/12/07	0.39	1.58	[illegible]	[illegible]	-	-	36,392	32,246
7	03/28/07	1.19	1.58	24,548.61	0.094182	0.077731	0.070941	-	-
8	03/30/07	1.19	1.58	30,075.39	0.079394	0.095737	0.087992	-	-
9	04/04/07	1.58	1.58	[illegible]	[illegible]	[illegible]	0.068442	-	-
10	04/06/07	1.58	1.58	28,879.74	0.059595	0.070374	0.06222	-	-
11	04/06/07	1.98	1.58	16,279.34	0.051739	0.08998	0.069285	-	-
12	04/09/07	1.98	1.58	15,819.47	0.049109	0.052141	0.074404	-	-
13	04/11/07	0	1.58	56,425.86	0.095317	0.132825	0.13815	-	-
14	04/13/07	0	1.58	53,399.48	0.151439	0.134871	0.137307	-	-
15	04/16/07	0.39	1.58	-	-	0.117417	0.135319	-	-
16	04/18/07	0.79	1.58	39,509.14	0.070097	0.067683	0.092077	-	-
17	04/19/07	0.39	1.58	36,232.06	0.095693	0.098512	0.124753	-	-
18	04/20/07	0.79	1.58	37,101.63	0.067398	0.089442	0.084167	-	-

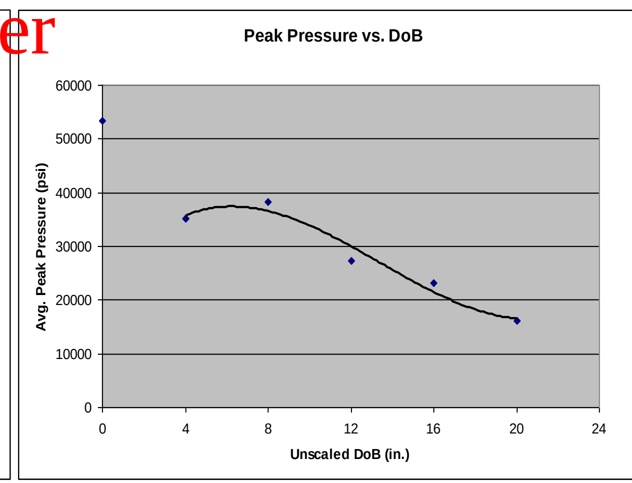
Example: Choose and fit function

- Strain gauges output pressure plot
- First spike is peak pressure
- Plotted depth of burial (DoB) vs. peak pressure









Acknowledgements

- These slides are modified from Steven Hoffenson's originally compiled slides for this course in 2017
- Much of this material came from Chapter 2 of the textbook, *Principles of Optimal Design*
- Some of the slides and examples came from Dr. Emrah Bayrak, Dr. Alex Burnap, and Dr. Namwoo Kang while they were at the University of Michigan
- Some of the slides and examples came from Dr. Michael Kokkolaras at McGill University