Data Visualization

Add highlighters to convey information in a better way

Table 2. Filter recovery error in dB on the test set for various CRs (bold indicates successful recovery with error below -50 dB).

CR [%]	M_z	G-MBD	GS-MBD	FS-MBD	LS-MBD	LS-MBD-L
50	99	**-54.05**	-44.93	-43.96	**-53.27**	-26.54
40.4	80	**-55.07**	-40.55	-26.52	**-52.80**	-
35.35	70	**-52.43**	-40.00	-22.76	**-51.50**	-
31.31	62	**-53.63**	-37.13	-21.86	**-54.71**	-
25.25	50	**-53.36**	-28.57	-8.40	**-51.41**	-
23.74	47	**-50.60**	-26.11	-6.84	**-50.35**	-
22.72	45	**-52.98**	-23.17	-6.14	-43.61	-
20.20	40	-47.39	-14.75	-5.13	-17.07	-

Add highlighters to convey information in a better way

Table 2. Difference between a Regular Table and a Heat Map

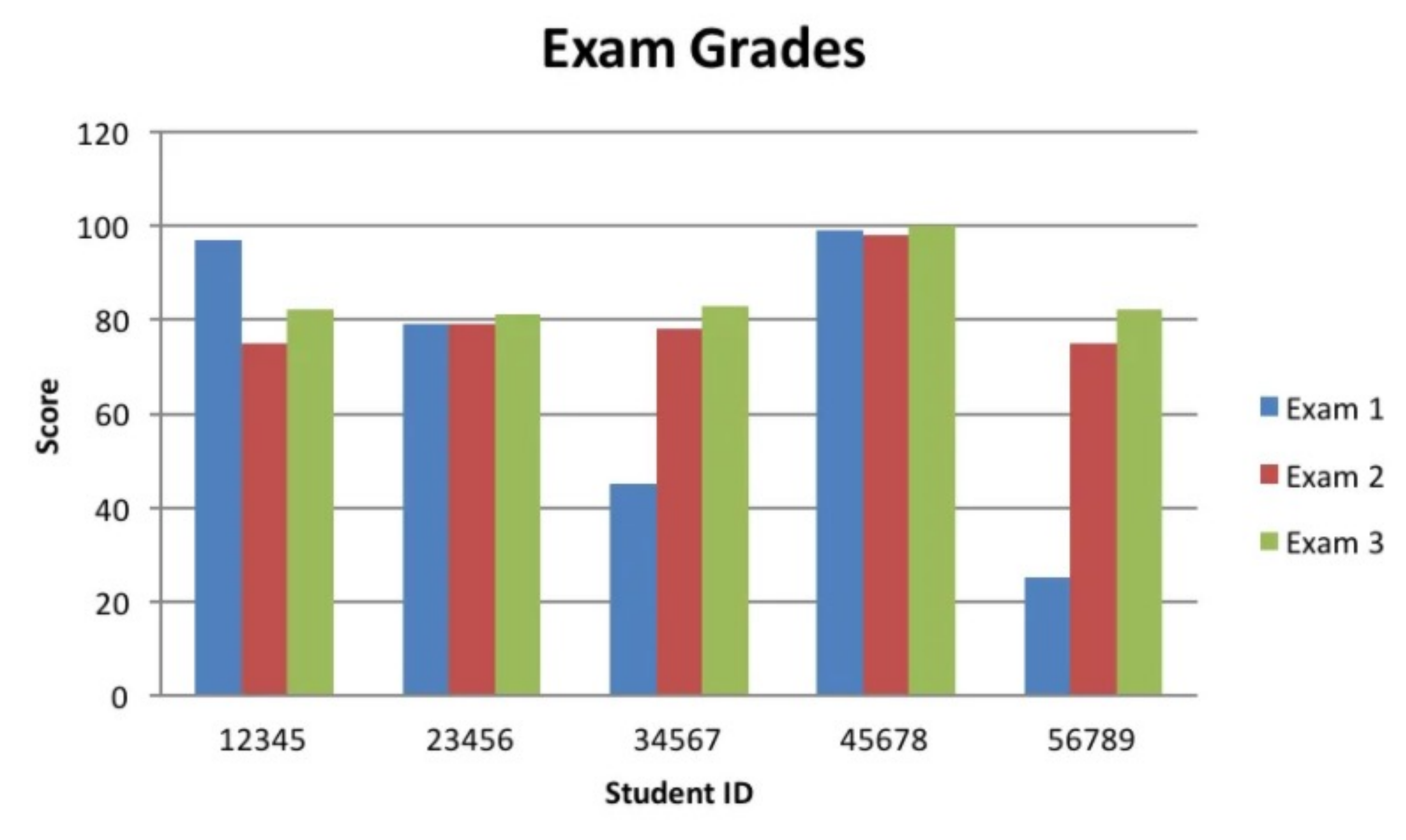
Example of a regular table				Example of a heat map			
SBP	DBP	MBP	HR	SBP	DBP	MBP	HR
128	66	87	87	128	66	87	87
125	43	70	85	125	43	70	85
114	52	68	103	114	52	68	103
111	44	66	79	111	44	66	79
139	61	81	90	139	61	81	90
103	44	61	96	103	44	61	96
94	47	61	83	94	47	61	83

All numbers were created by the author. SBP: systolic blood pressure, DBP: diastolic blood pressure, MBP: mean blood pressure, HR: heart rate.

Bar Plots

Indicate and compare values in a discrete category

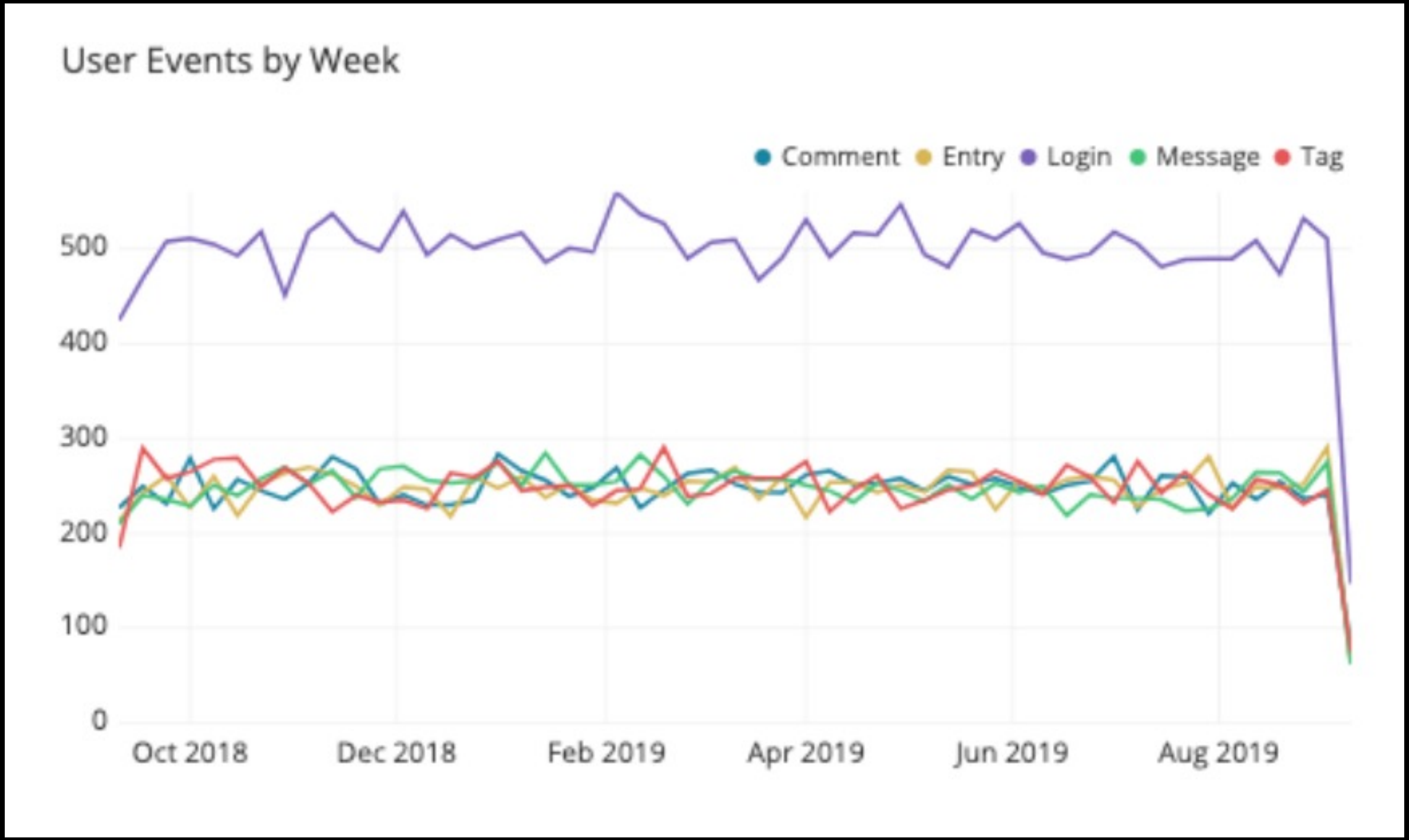
Compare multiple data sets



Source: https://writingcommons.org/article/data-visualizations

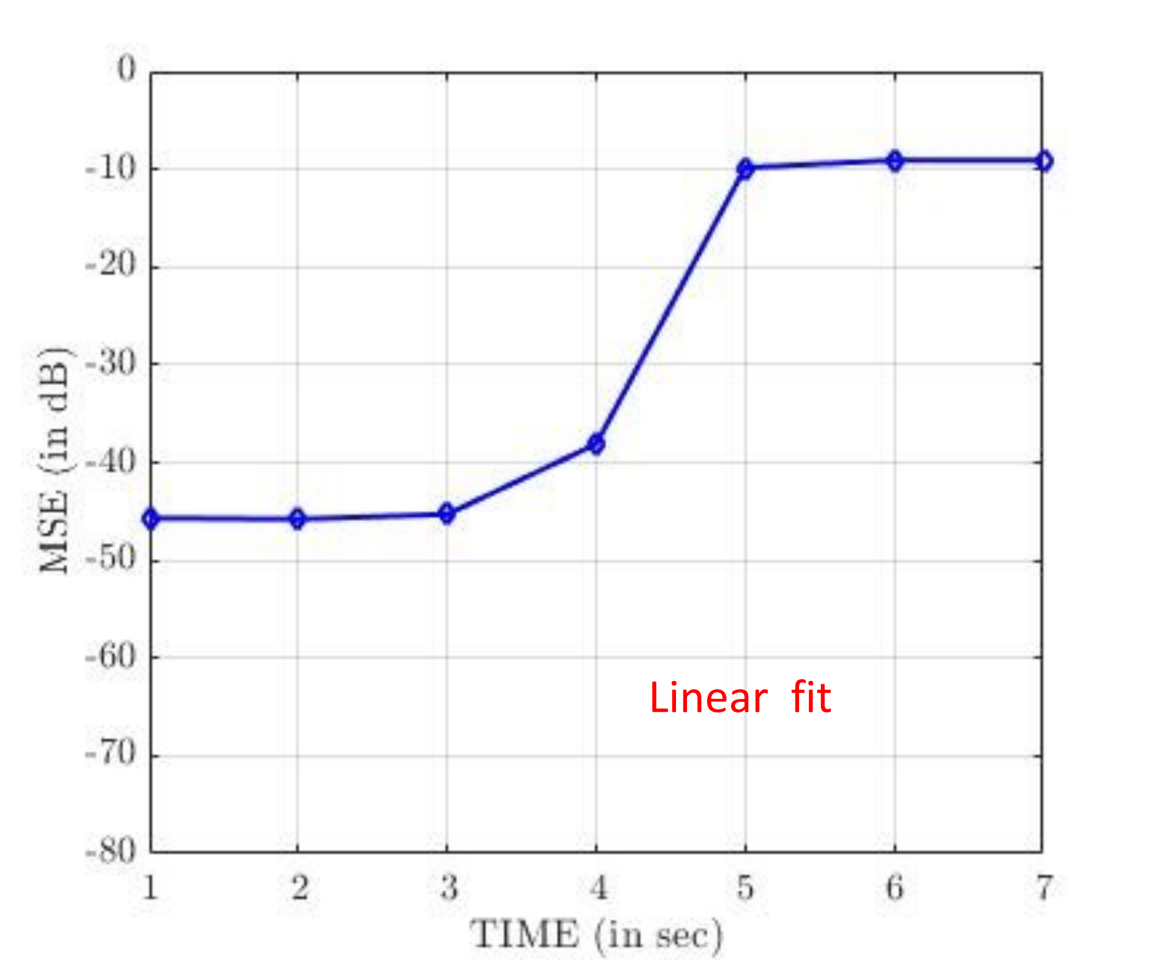
Line Graphs

Don't use too many of them



Source: https://chartio.com/learn/charts/line-chart-complete-guide/

Interpolation and fitting



Interpolation and fitting

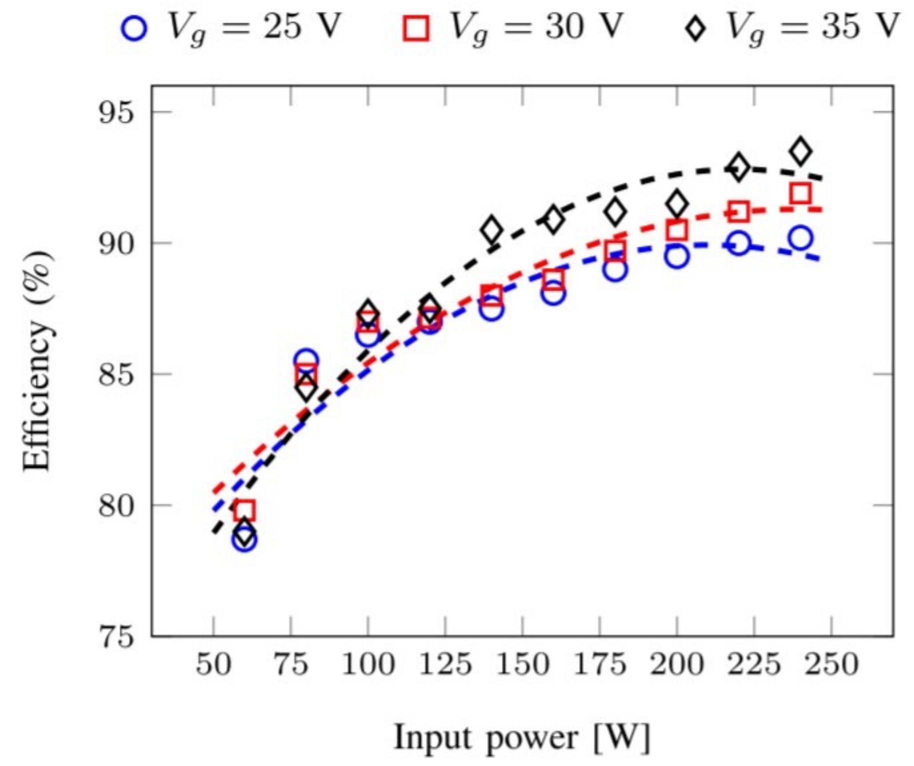
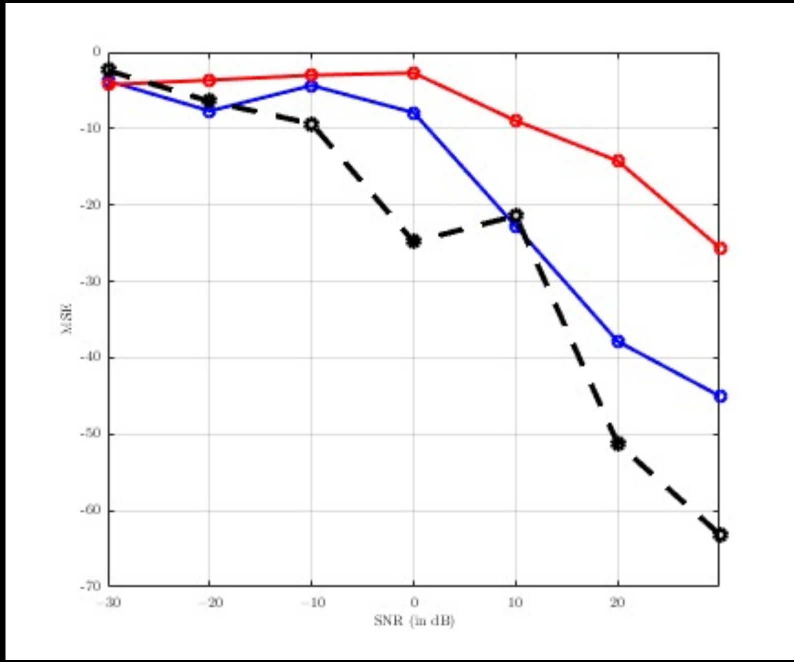
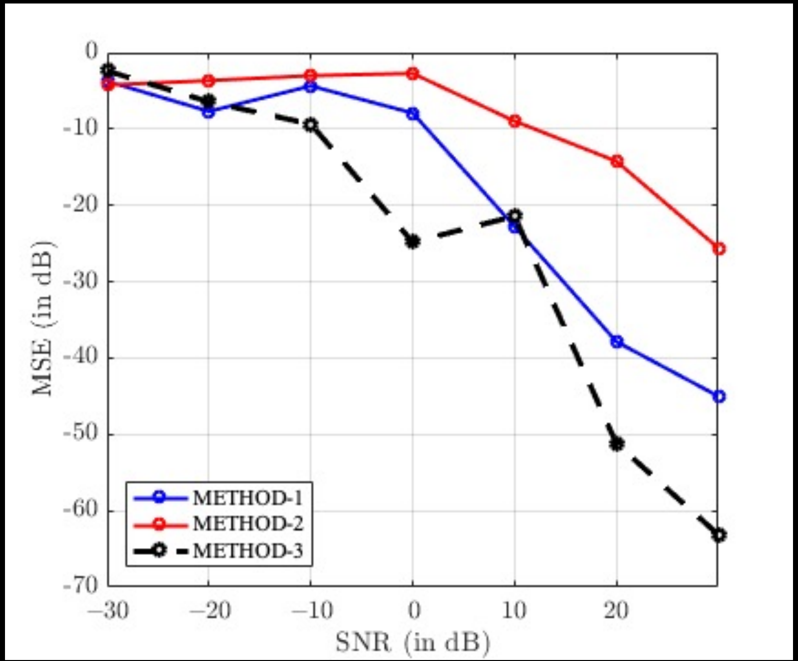


Fig. 26. Measured efficiency of the experimental prototype under different input voltage and load conditions. The dotted curves are corresponding quadratic best-fit curves.

Chakraborty and Chattopadhyay, "A Dual-Active-Bridge-Based Novel Single-Stage Low Device Count DC–AC Converter," *IEEE Trans. on Power Electron.*, 34 (3), Mar. 2019.

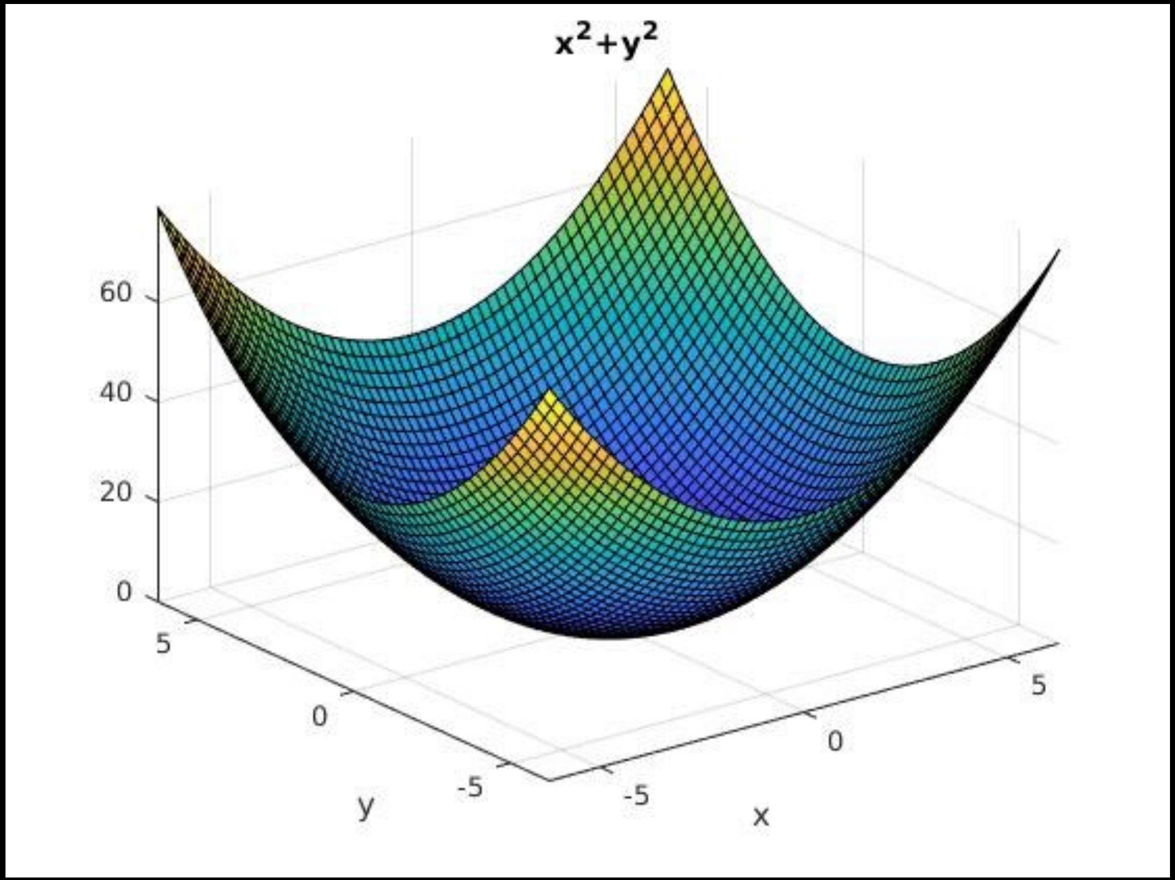




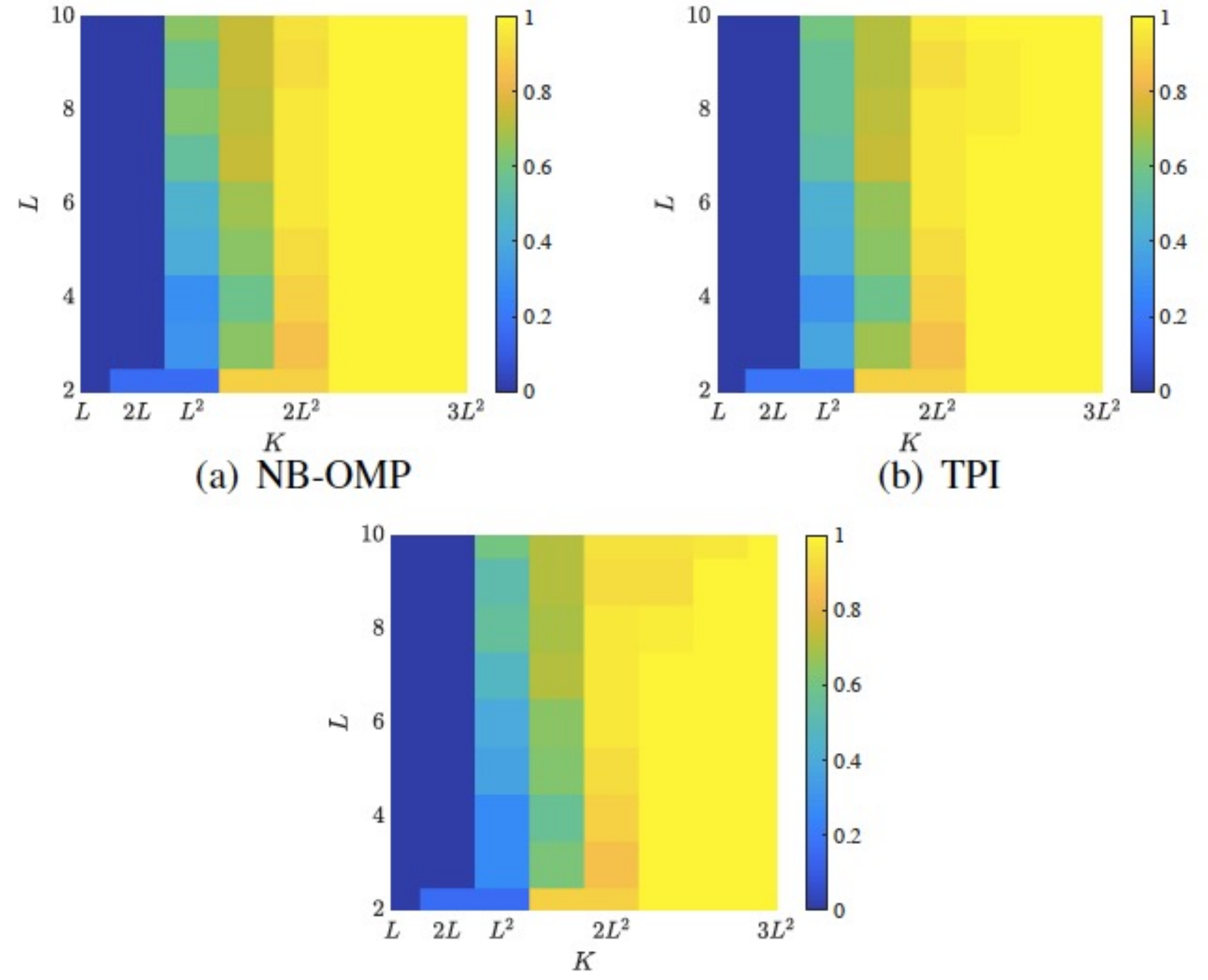
Comparing Method-1, Mehtod-2, and Method-3

3D Plots

- Two continuous-valued independent variables



3D Plots



(a) NB-OMP (b) TPI (c) BDC


object to mirror
irror_mod.mirror_object
peration == "MIRROR_X":
irror_mod.use_x = True
irror_mod.use_y = False
irror_mod.use_z = False
operation == "MIRROR_Y":
irror_mod.use_x = False
irror_mod.use_y = True
irror_mod.use_z = False
operation == "MIRROR_Z":
irror_mod.use_x = False
irror_mod.use_y = False
irror_mod.use_z = True
election at the end -add
_ob.select= 1
er_ob.select=1
ntext.scene.objects.acti
"Selected" + str(modifie
irror_ob.select = 0
bpy.context.selected_ob
ata.objects[one.name].se
int("please select exactl
OPERATOR CLASSES
types.Operator):
X mirror to the selected
ject.mirror_mirror_x"
rror X"

Algorithms

- An algorithm is a *set of steps* to solve a problem
- Independent of programming or coding language

We are NOT discussing about algorithms used in CS courses

- An algorithm is a *set of steps* to solve a problem

$$\min_{x \in S} f(x)$$

- An algorithm is a *set of steps* to solve a problem

$$\min_{x \in S} f(x)$$

At k-th iteration:

$$x_k = x_{k-1} - \alpha \nabla f_{k-1}(x)$$
$$x_k = S(x_k)$$

- An algorithm is a *set of steps* to solve a problem

$$\min_{x \in S} f(x)$$

1. Output:
2. Input:
3. Initialization
4. Steps
When to stop

At k-th iteration:

$$x_k = x_{k-1} - \alpha \nabla f_{k-1}(x)$$
$$x_k = S(x_k)$$

- An algorithm is a *set of steps* to solve a problem

$$\min_{x \in S} f(x)$$

1. Output:
2. Input:
3. Initialization
4. Steps
When to stop

1. Output: optimum solution
2. Input: Gradient function
3. Initialization: x_0
4. Steps: A. B.
When to stop

At k-th iteration:

$$x_k = x_{k-1} - \alpha \nabla f_{k-1}(x)$$
$$x_k = S(x_k)$$

- An algorithm is a *set of steps* to solve a problem

Algorithm 1 BDC for solving (15).

Output: $\mathbf{s}$ and $\mathbf{X}$

Input: $\mathbf{Y}$, $\bar{\mathbf{A}}$ L, and the initial estimate $\mathbf{s}^{(0)}$

Let $i \leftarrow 1$

repeat

  Estimate $\mathbf{X}^{(i)}$ by applying OMP to $\mathrm{diag}(\mathbf{s}^{(i-1)})^{-1}\mathbf{Y}$ columnwise

  $\mathbf{s}^{(i)} \leftarrow \mathrm{argmin}_{\mathbf{s}} \|\mathbf{Y} - \mathrm{diag}(\mathbf{s})\bar{\mathbf{A}}\mathbf{X}^{(i)}\|_2^2$

  $i \leftarrow i + 1$

until convergence criterion is reached

- Should be self contained

Algorithm 2 Joint Subsampling and Recovery Algorithm

Inputs: Data $\mathcal{D}$ and full sample indices $\mathcal{N}$

Initialize: $\mathcal{K}^{(0)} = \emptyset$

for $k = 1$ to K **do**

[S1] **for all** $i \in \mathcal{N} \backslash \mathcal{K}^{(k-1)}$ **do**

(a) For a binary-valued vector $\mathbf{c}_i \in \{0, 1\}^{|\mathcal{N}|}$, set $\text{supp}\{\mathbf{c}_i\} = \mathcal{K}^{(k-1)} \cup \{i\}$

(b) $\boldsymbol{\theta}_i^{(k)} = \arg\min_{\boldsymbol{\theta}} \frac{1}{Q} \sum_{q=1}^{Q} \|\mathbf{x}_q - r_{\boldsymbol{\theta}}(\text{diag}(\mathbf{c}_i)\mathbf{f}_q)\|_2^2$ where $r_{\boldsymbol{\theta}}$ is a LISTA-based reconstruction.

(c) $\mathbf{x}_{q,i} = r_{\boldsymbol{\theta}_i^{(k)}}(\text{diag}(\mathbf{c}_i)\mathbf{f}_q)$ for $q = 1, \cdots, Q$

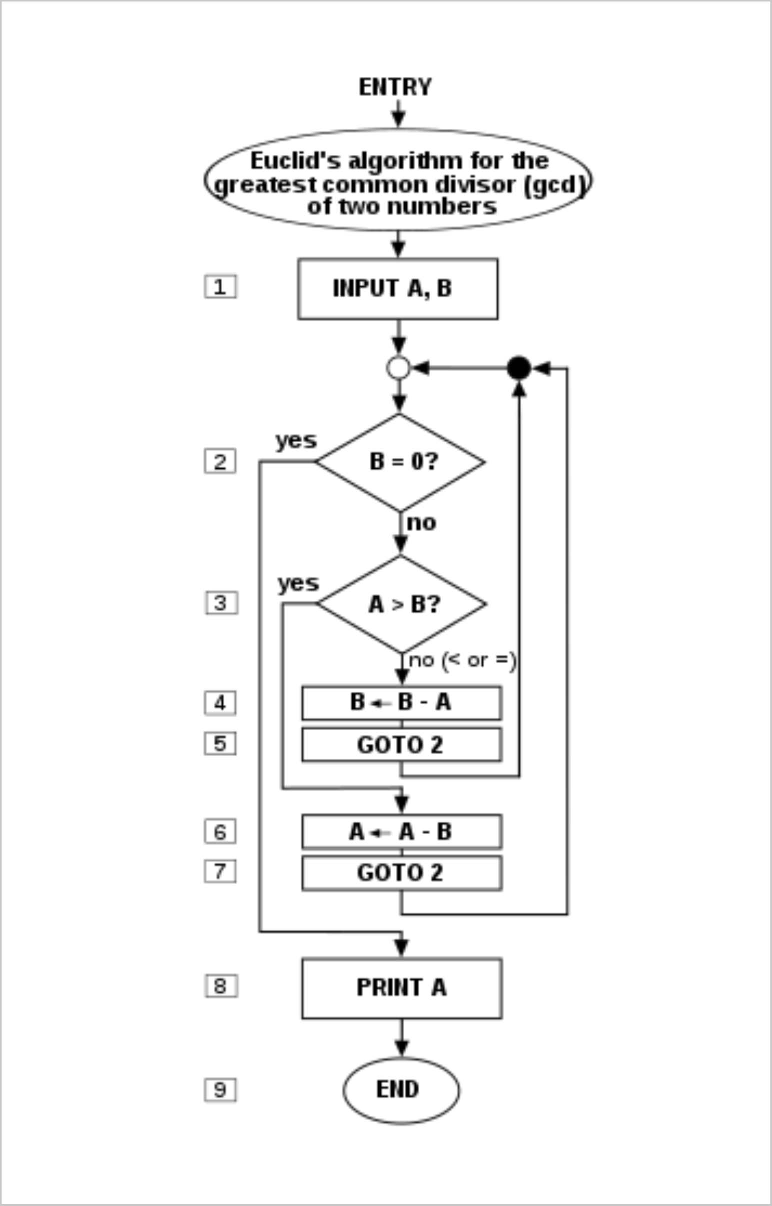
end for

[S2] $i_*^{(k)} = \underset{i \in \mathcal{N} \backslash \mathcal{K}^{(k-1)}}{\arg\min} \quad \frac{1}{Q} \sum_{q=1}^{Q} \|\mathbf{x}_q - \mathbf{x}_{q,\mathcal{K} \backslash \{i\}}\|_2^2$

[S3] $\mathcal{K}^{(k)} = \mathcal{K}^{(k-1)} \cup \{i_*^{(k)}\}$

end for

Output: Optimal sampling set $\mathcal{K}^K \subseteq \mathcal{N}$ with $|\mathcal{K}^K| = K$ and corresponding reconstruction parameters $\boldsymbol{\theta}_{i_*^{(k)}}$



- Flowcharts are alternative representations

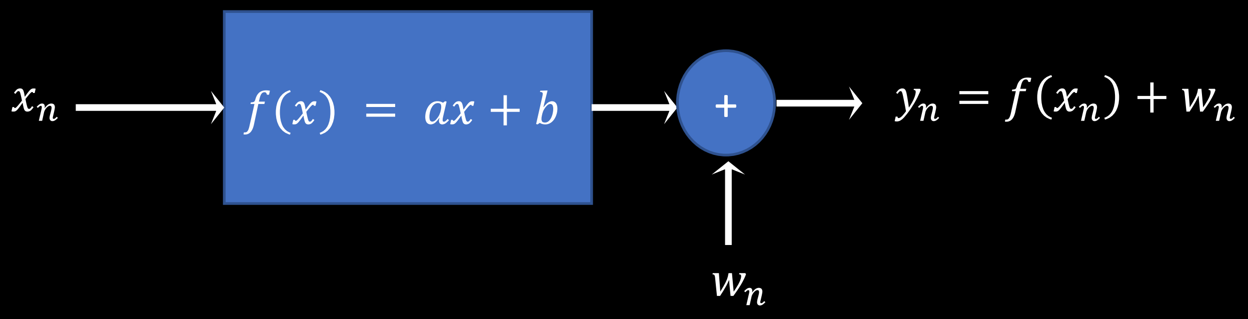
How to Design Simulations?

Problem: Consider a linear system $f(x) = ax + b$. The objective is to estimate the system from a set of input and output pairs in the presence of noise. The data may also contain outliers.

Given: $y_n = f(x_n) + w_n, n = 1, \dots, N$, estimate a and b

Problem: Consider a linear system $f(x) = ax + b$. The objective is to estimate the system from a set of input and output pairs in the presence of noise. The data may also contain outliers.

Given: $y_n = f(x_n) + w_n, n = 1, \dots, N$, estimate a and b



Problem: Consider a linear system $f(x) = ax + b$. The objective is to estimate the system from a set of input and output pairs in the presence of noise. The data may also contain outliers.

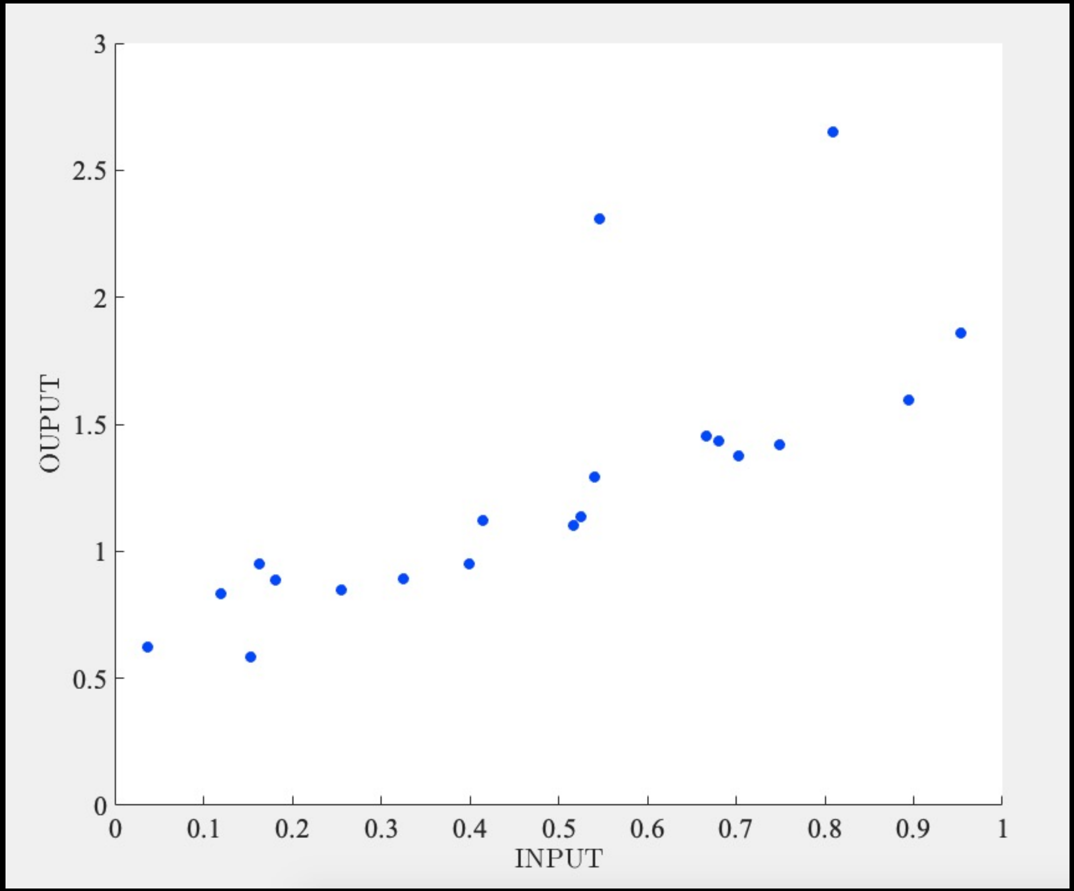
Given: $y_n = f(x_n) + w_n, n = 1, \ldots, N$, estimate a and b

x	y
0.0366	0.6242
0.1202	0.8307
0.1536	0.5836
0.1636	0.9504
0.1807	0.8876
0.2554	0.8489
0.3258	0.8935
0.3989	0.9487
0.4151	1.1229
0.5166	1.1021
0.5250	1.1364
0.5409	1.2915
0.5464	2.3104
0.6660	1.4555
0.6797	1.4341
0.7027	1.3734
0.7486	1.4218
0.8092	2.6520
0.8944	1.5956
0.9535	1.8582

Problem: Consider a linear system $f(x) = ax + b$. The objective is to estimate the system from a set of input and output pairs in the presence of noise. The data may also contain outliers.

Given: $y_n = f(x_n) + w_n, n = 1, \ldots, N$, estimate a and b

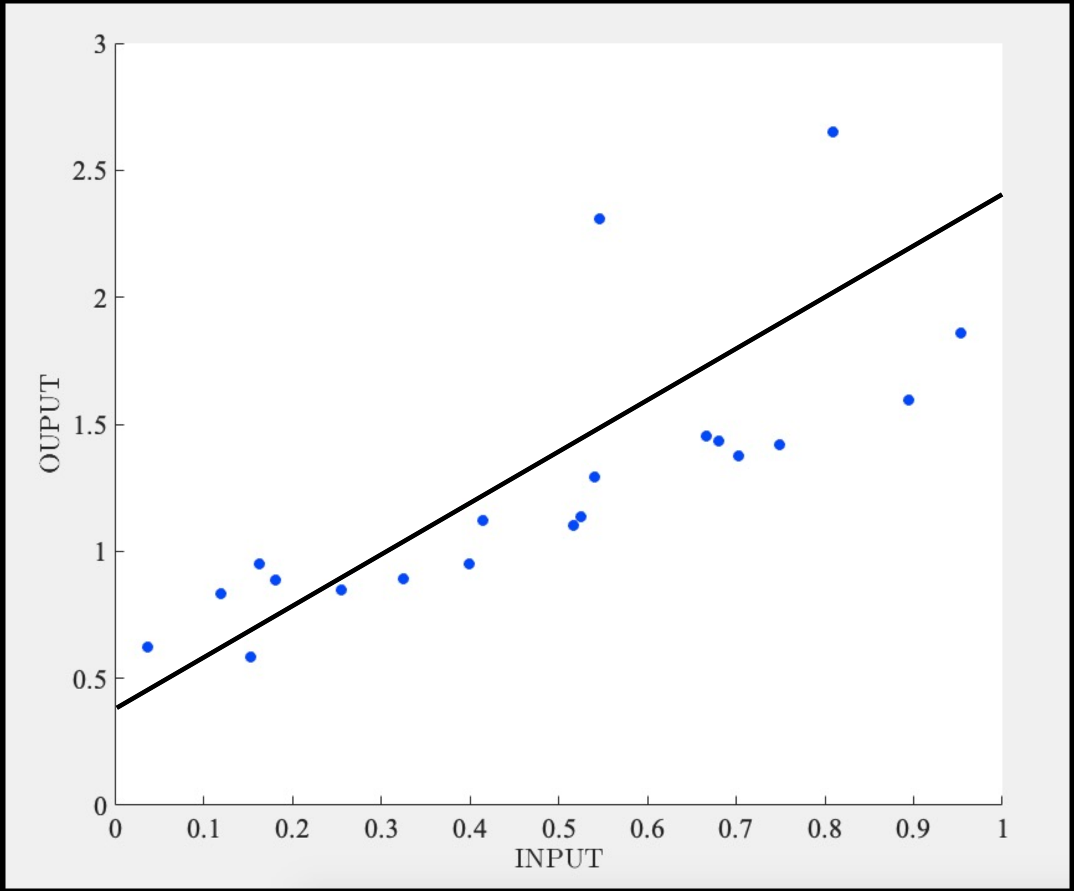
x	y
0.0366	0.6242
0.1202	0.8307
0.1536	0.5836
0.1636	0.9504
0.1807	0.8876
0.2554	0.8489
0.3258	0.8935
0.3989	0.9487
0.4151	1.1229
0.5166	1.1021
0.5250	1.1364
0.5409	1.2915
0.5464	2.3104
0.6660	1.4555
0.6797	1.4341
0.7027	1.3734
0.7486	1.4218
0.8092	2.6520
0.8944	1.5956
0.9535	1.8582



Problem: Consider a linear system $f(x) = ax + b$. The objective is to estimate the system from a set of input and output pairs in the presence of noise. The data may also contain outliers.

Given: $y_n = f(x_n) + w_n, n = 1, \ldots, N$, estimate a and b

x	y
0.0366	0.6242
0.1202	0.8307
0.1536	0.5836
0.1636	0.9504
0.1807	0.8876
0.2554	0.8489
0.3258	0.8935
0.3989	0.9487
0.4151	1.1229
0.5166	1.1021
0.5250	1.1364
0.5409	1.2915
0.5464	2.3104
0.6660	1.4555
0.6797	1.4341
0.7027	1.3734
0.7486	1.4218
0.8092	2.6520
0.8944	1.5956
0.9535	1.8582



Problem: Consider a linear system $f(x) = ax + b$. The objective is to estimate the system from a set of input and output pairs in the presence of noise. The data may also contain outliers.

Given: $y_n = f(x_n) + w_n, n = 1, \ldots, N$, estimate a and b

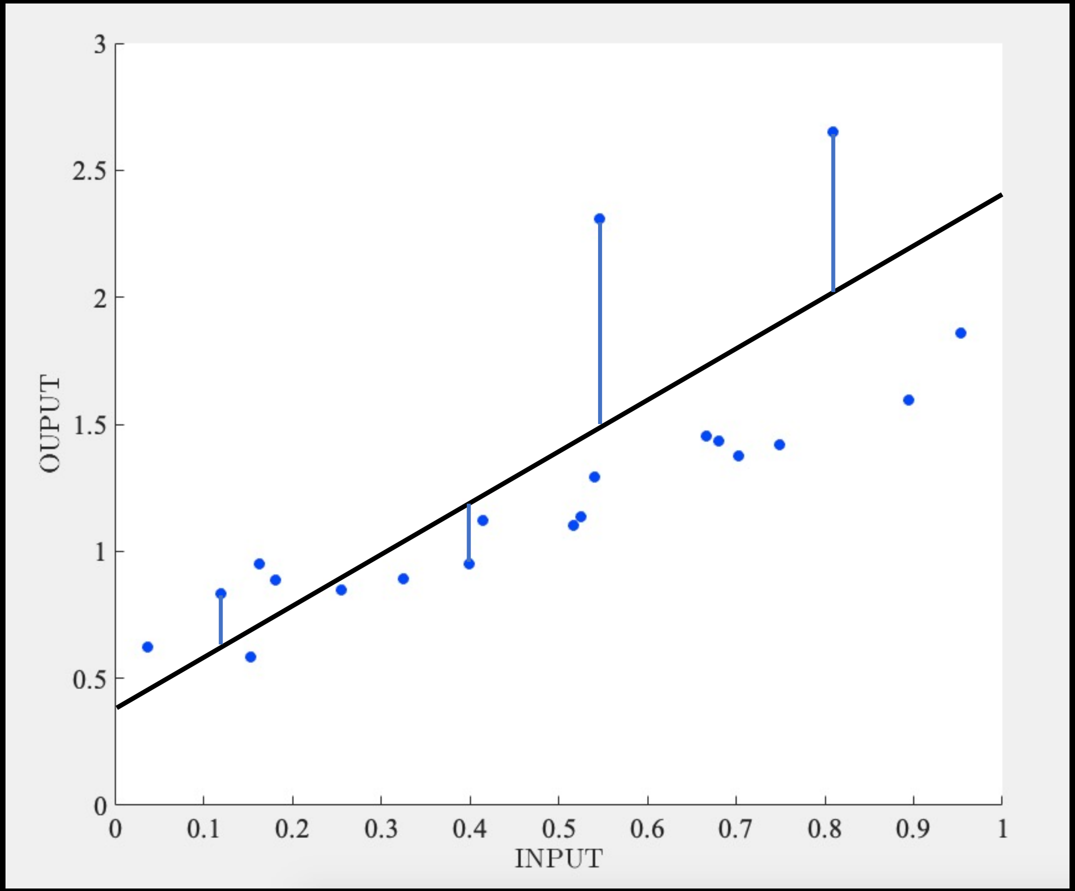
Existing solution?

Least-squares solution

$$\min_{a,b} \sum_{n=1}^{N} |y_n - (ax_n + b)|^2$$

The outliers result in inaccurate fit!

Can we leave out the outliers?



Problem: Consider a linear system $f(x) = ax + b$. The objective is to estimate the system from a set of input and output pairs in the presence of noise. The data may also contain outliers.

Given: $y_n = f(x_n) + w_n, n = 1, \ldots, N$, estimate a and b

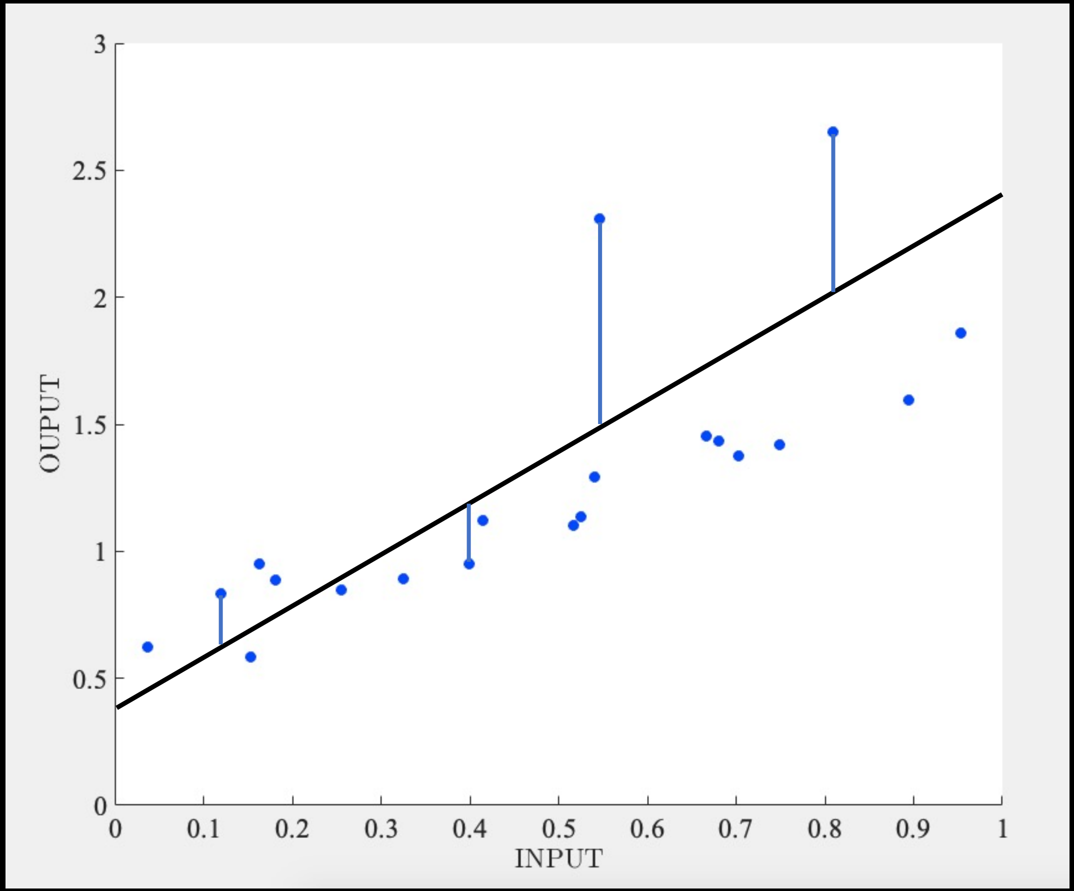
Existing solution?

Weighted least-squares solution

$$\min_{a,b} \sum_{n=1}^{N} v_n \left|y_n - (ax_n + b)\right|^2$$

Keep the weights low for the outliers

How do we the outliers?



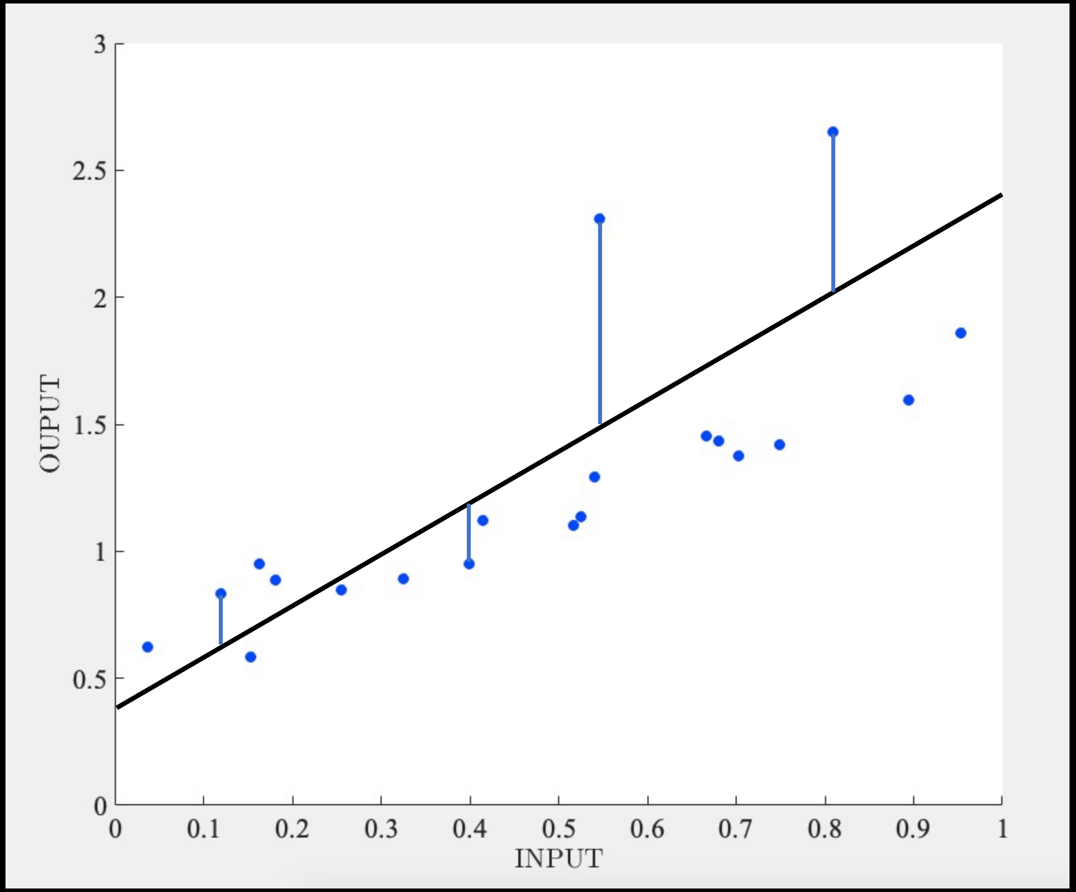
Problem: Consider a linear system $f(x) = ax + b$. The objective is to estimate the system from a set of input and output pairs in the presence of noise. The data may also contain outliers.

Given: $y_n = f(x_n) + w_n, n = 1, \dots, N$, estimate a and b

Weighted least-squares solution

$$\min_{a,b} \sum_{n=1}^{N} v_n \, |y_n - (ax_n + b)|^2$$

Our solution: Use an algorithm called iterative reweighted least-squares that automatically adjusts the weights



Problem: Consider a linear system $f(x) = ax + b$. The objective is to estimate the system from a set of input and output pairs in the presence of noise. The data may also contain outliers.

Given: $y_n = f(x_n) + w_n, n = 1, \ldots, N$, estimate a and b

Weighted least-squares solution

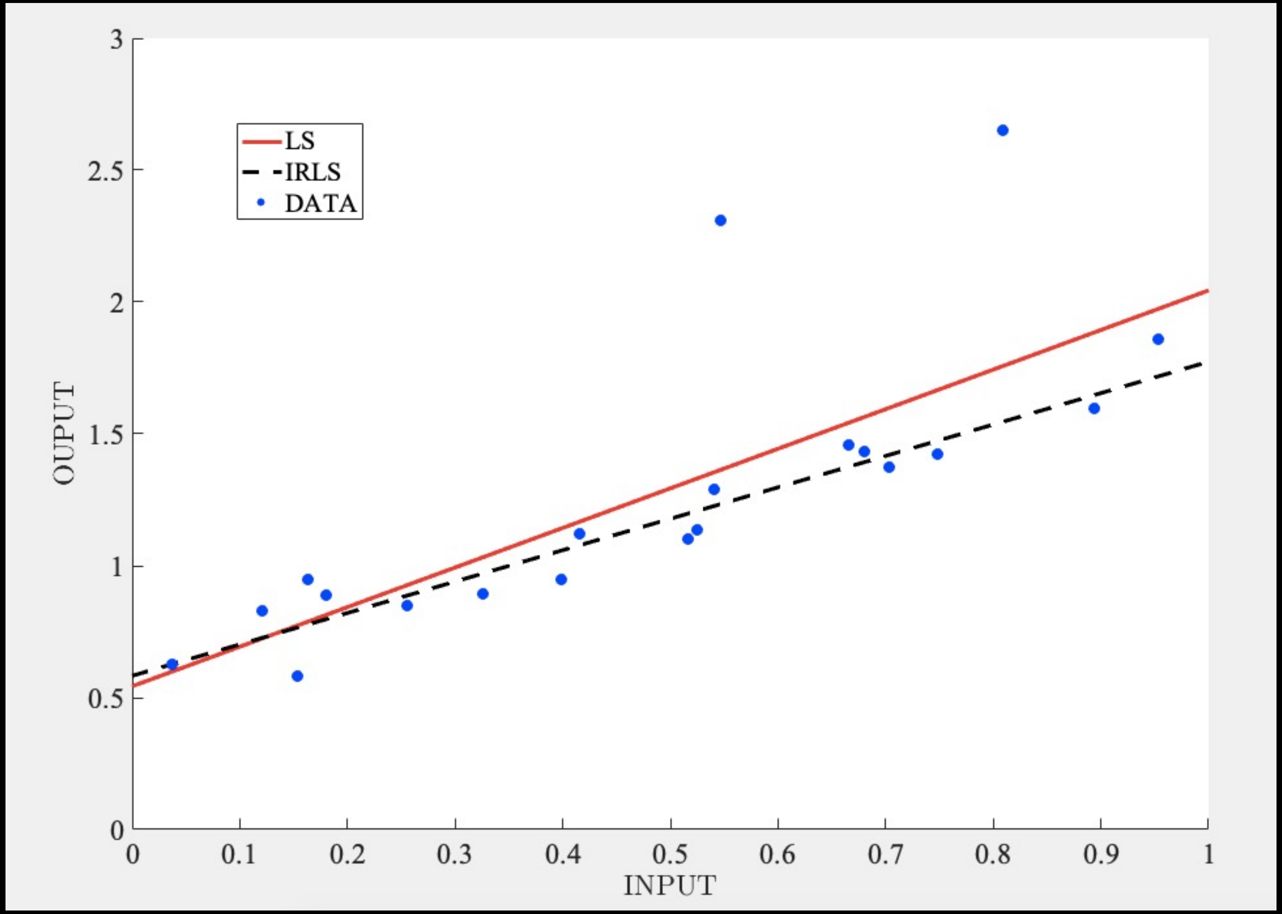
$$\min_{a,b} \sum_{n=1}^{N} v_n \left| y_n - (ax_n + b) \right|^2$$

Our solution: Use an algorithm called iterative reweighted least-squares that automatically adjusts the weights

IRLS Algorithm
1. Start with some initial weights
2. Estimate a, b
3. Update weights
4. Repeat 2 and 3 until convergence

What/how do you show that IRLS is better than LS?

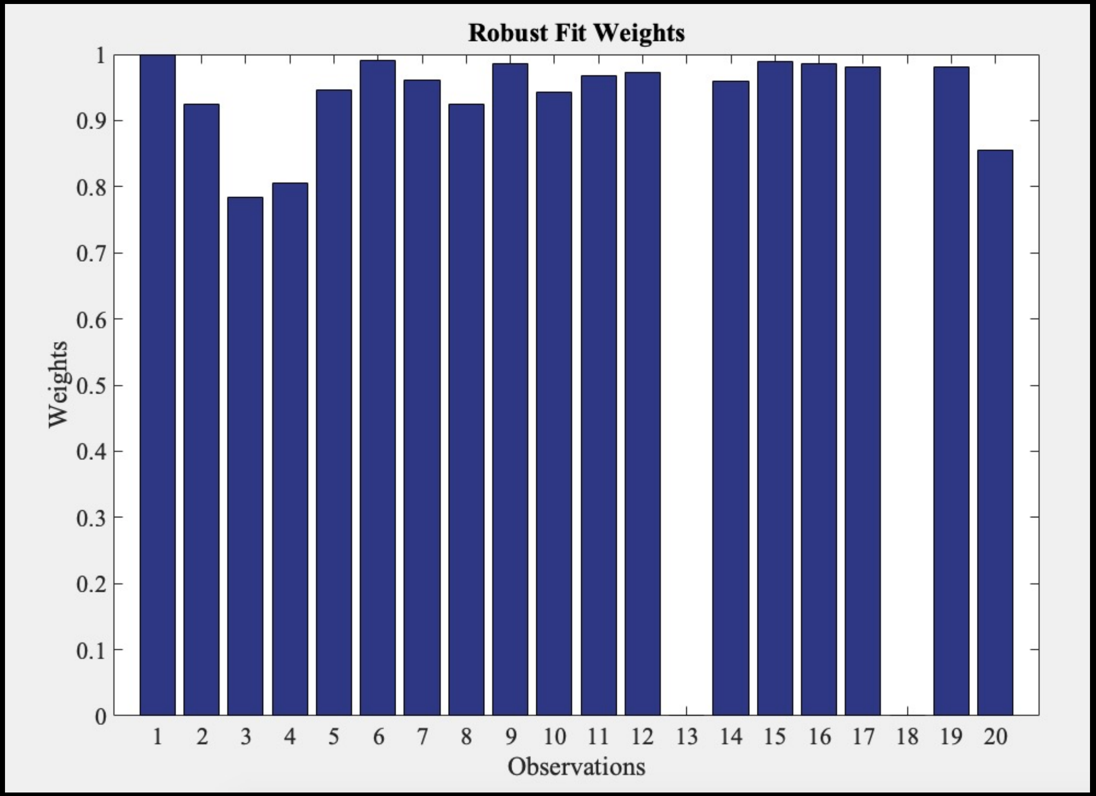
1. Visually show that your approach is better than others

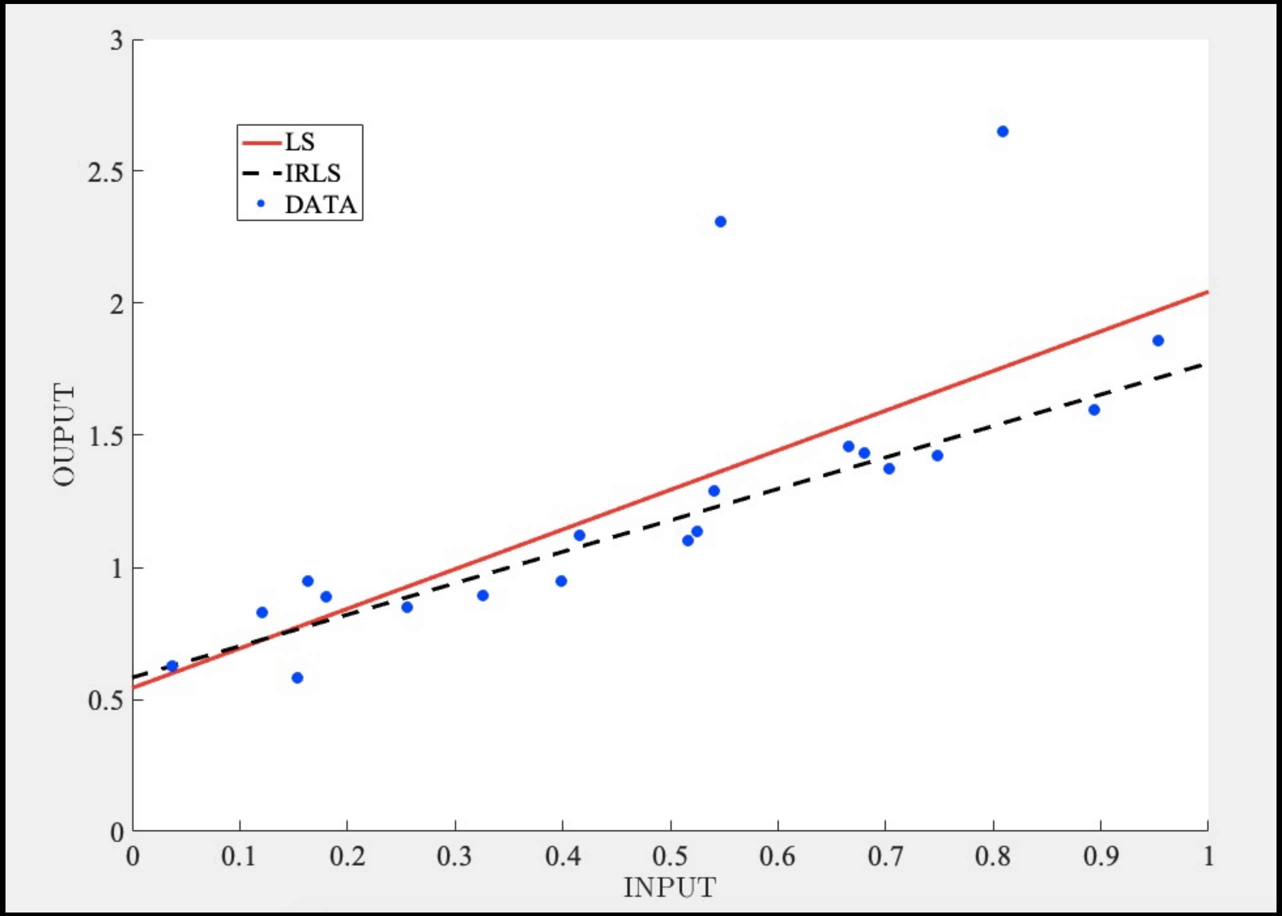


What/how do you show that IRLS is better than LS?

1. Visually show that your approach is better than others

2. Add some more details of your algorithm





What/how do you show that IRLS is better than LS?

3. Will IRLS works better than LS in different scenarios?

Given: $y_n = ax_n + b + w_n, n = 1, \ldots, N$, estimate a and b

Parameters that effect the estimation: Number of samples N and noise level

Noise level: $w_n \sim Normal(0, \sigma^2), n = 1, \ldots, N$

How to measure efficiency of the two algorithms as a function of N and σ ?

Given: $y_n = ax_n + b + w_n, n = 1, \ldots, N$, estimate a and b

$$Err(a) = (a - \tilde{a})^2 \qquad Err(b) = \left(b - \tilde{b}\right)^2 \qquad Err(a, b) = (a - \tilde{a})^2 + \left(b - \tilde{b}\right)^2$$

$$RErr(a) = (a - \tilde{a})^2/a^2 \qquad RErr(b) = \left(b - \tilde{b}\right)^2/b^2 \qquad RErr(a, b) = (a - \tilde{a})^2/a^2 + \left(b - \tilde{b}\right)^2/b^2$$

True error or relative error?

	True value	Estimate	Error	R. Error
a	0.5	0.6	0.01	0.04
b	10	11	1	0.01

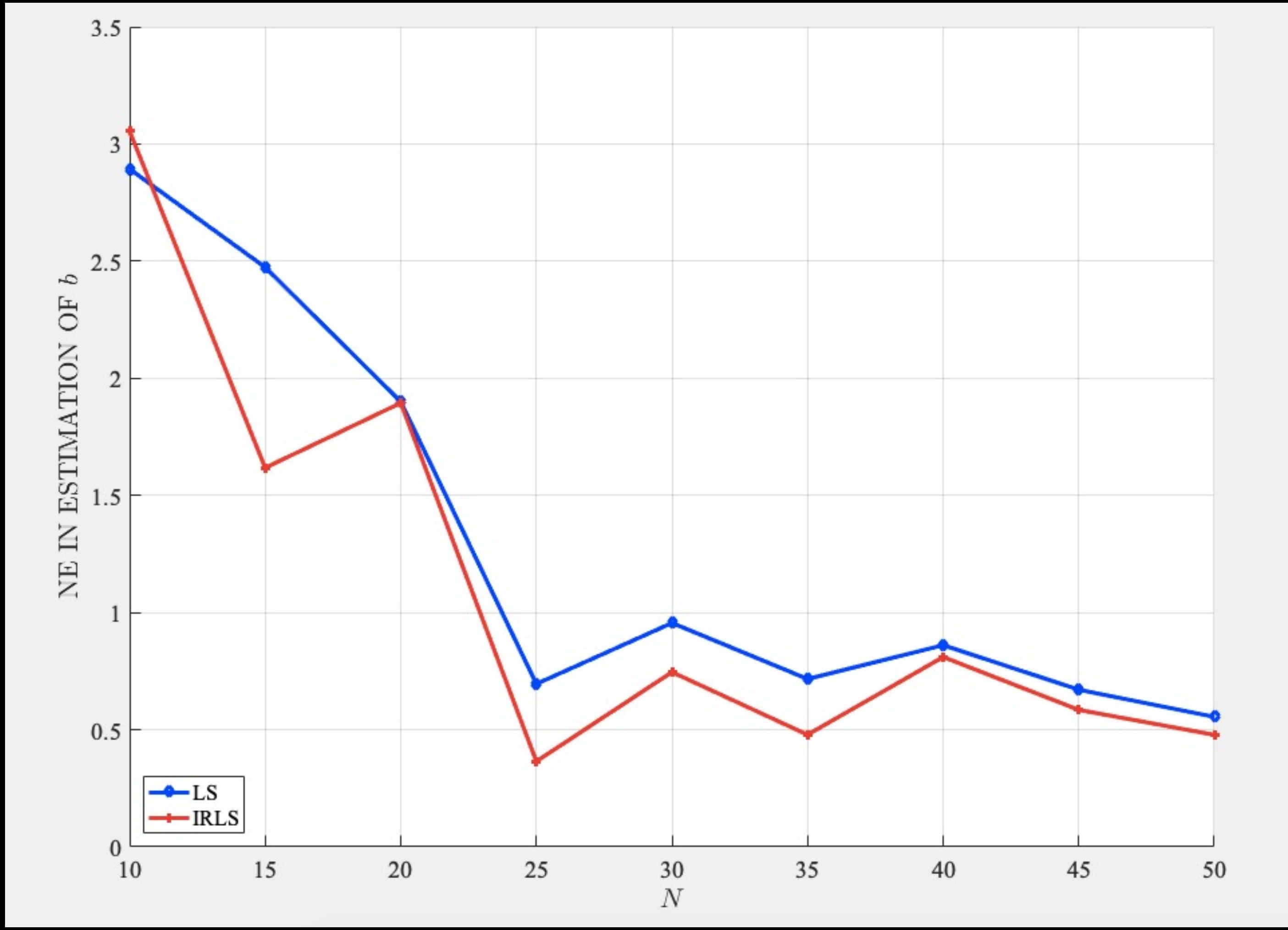


Fig. Comparison of LS and IRLS for estimation estimation of b: the value of b is 5. Conclusion?

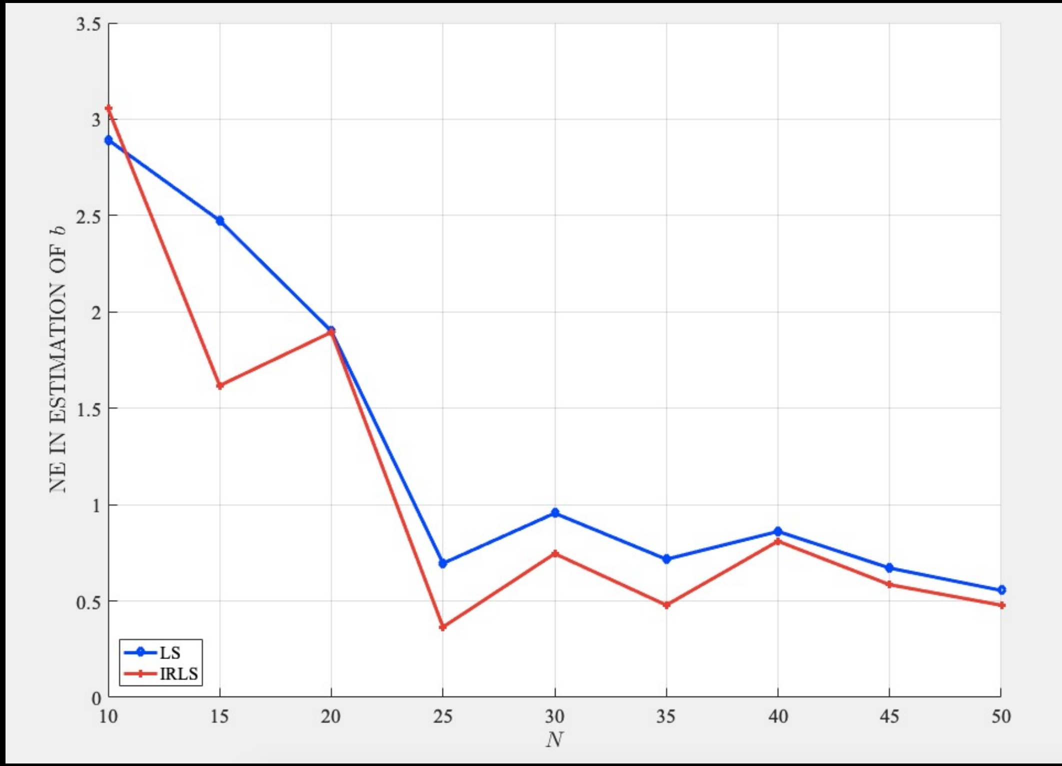


Fig. Comparison of LS and IRLS for estimation estimation of b: the value of b is 5. Conclusion?

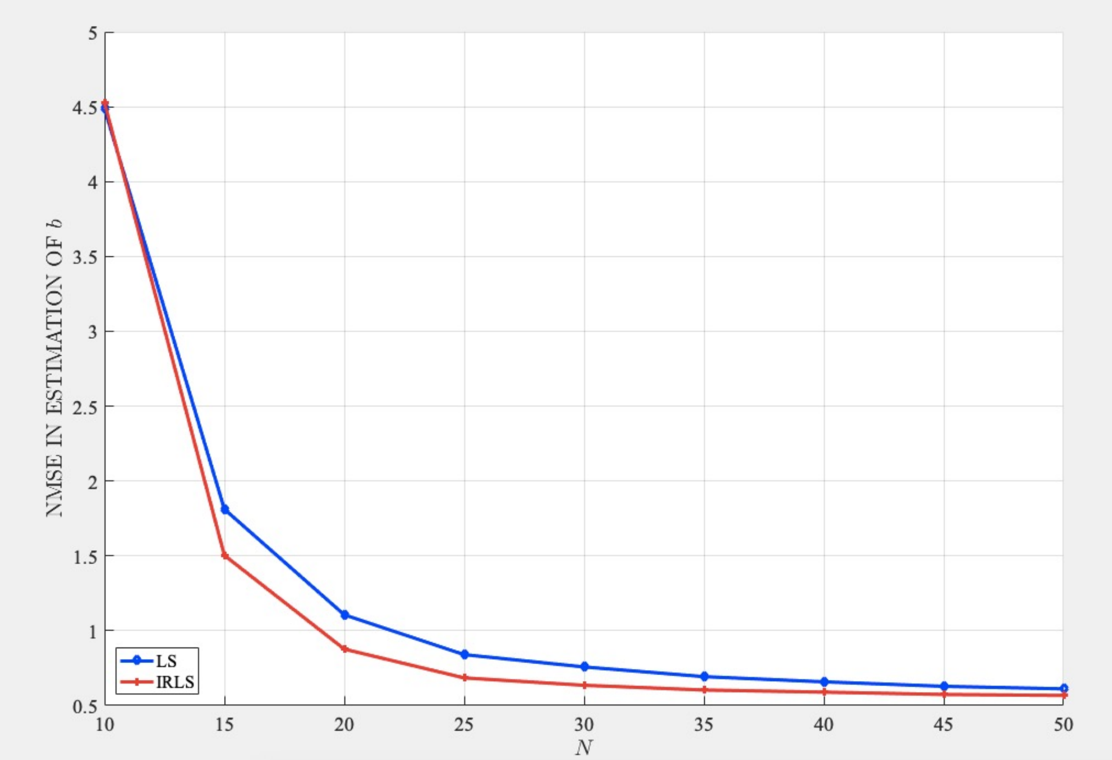


Fig. Comparison of LS and IRLS for estimation estimation of b: the value of b is 5. IRLS performs better than LS for $N = 10 - 50$

Monte-Carlo Simulations: For each N, fix a, b, and x values, estimate a, b for 1000 independent realizations of noise and then Compute the normalized mean-square-error (NMSE)

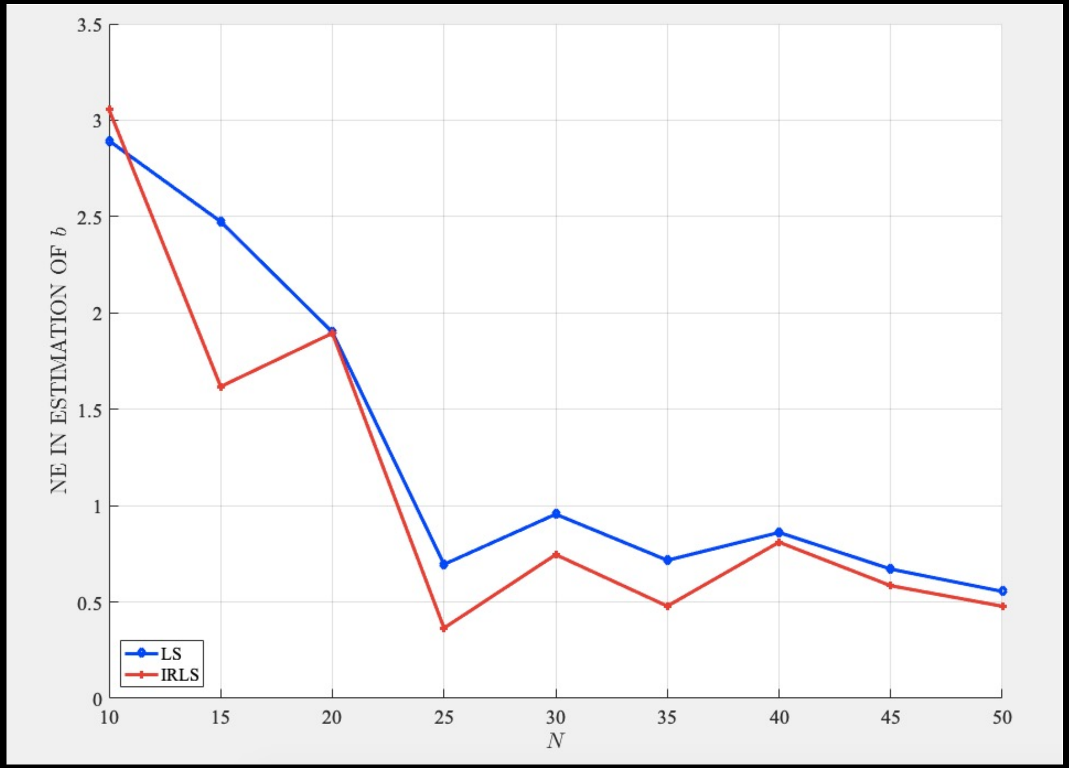


Fig. Comparison of LS and IRLS for estimation estimation of b: the value of b is 5. Conclusion?

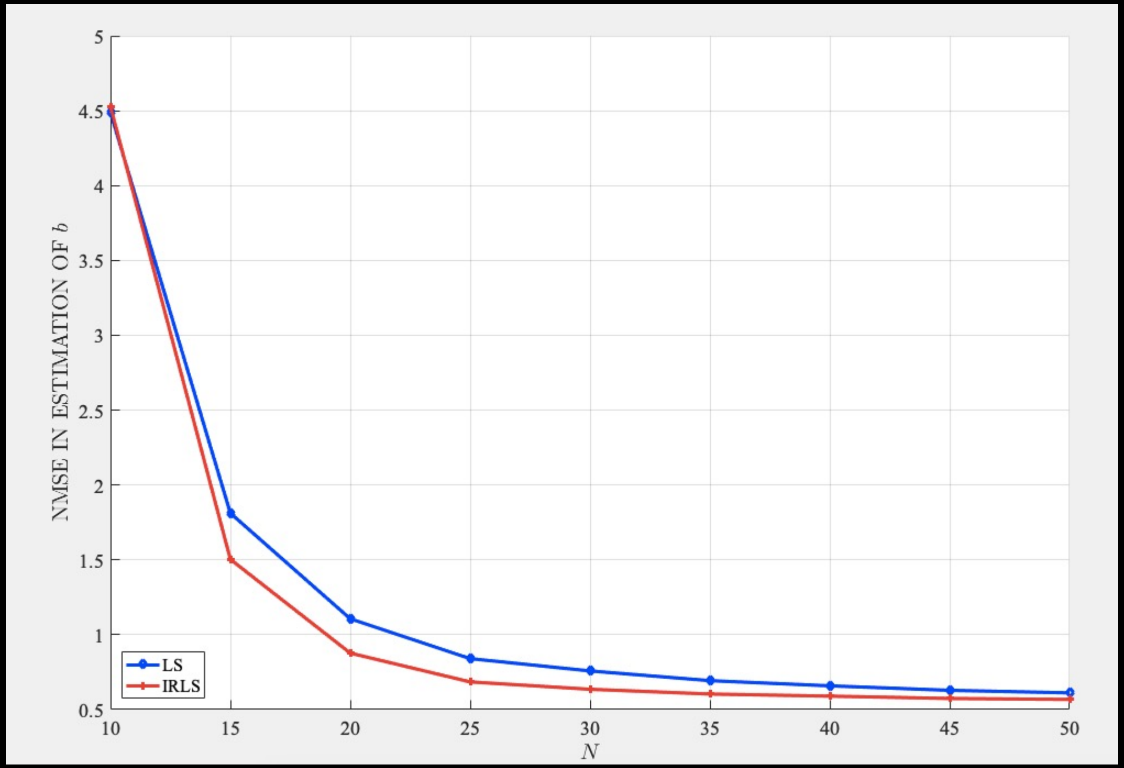


Fig. Comparison of LS and IRLS for estimation estimation of b: the value of b is 5. IRLS performs better than LS for $N = 10 - 50$

Monte-Carlo Simulations: For each N, fix a, b, and x values, estimate a, b for 1000 independent realizations of noise and then Compute the normalized mean-square-error (NMSE)

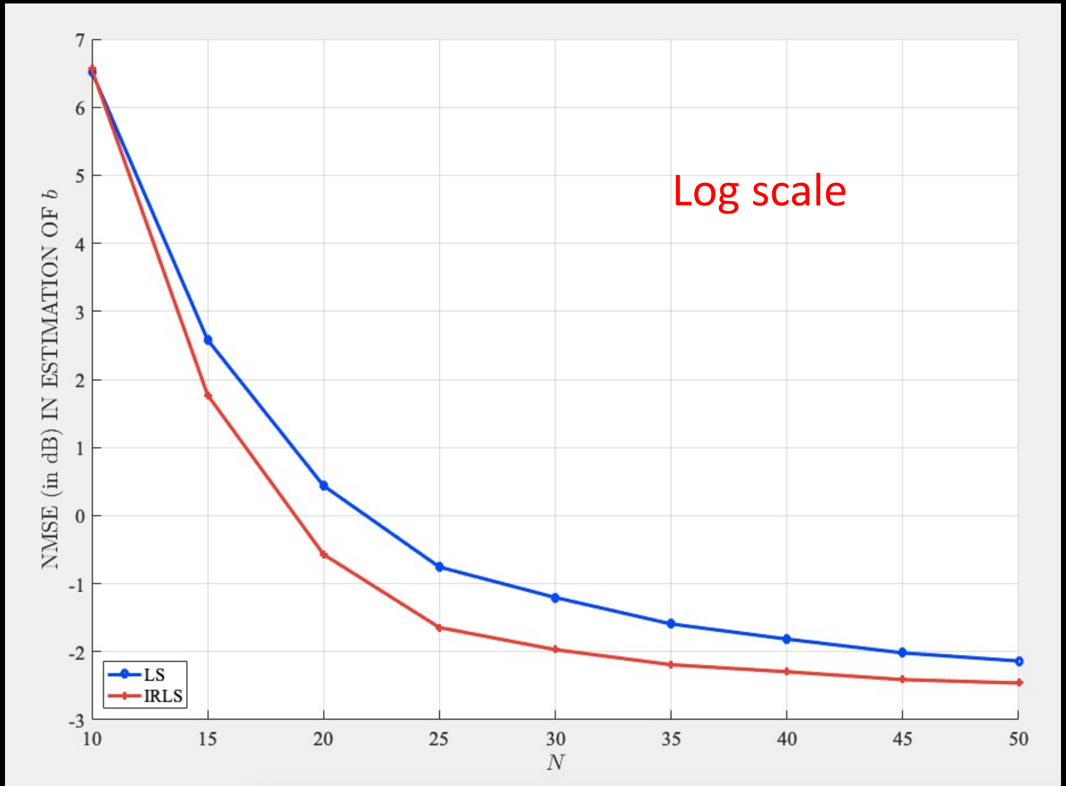


Fig. Comparison of LS and IRLS for estimation estimation of b: the value of b is 5. IRLS has 0.5-2 dB lower NMSE compared to LS for $N > 15$

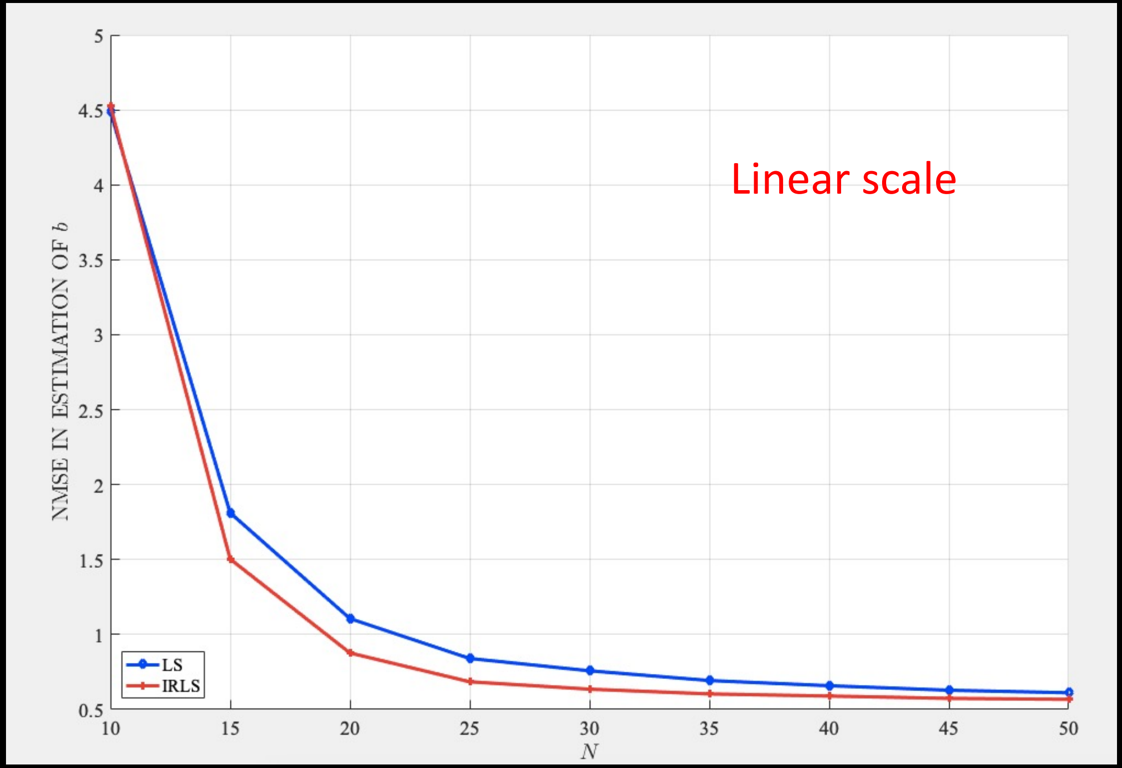


Fig. Comparison of LS and IRLS for estimation estimation of b: the value of b is 5. IRLS performs better than LS for $N = 10 \; to \; 50$

Monte-Carlo Simulations: For each N, fix a, b, and x values, estimate a, b for 1000 independent realizations of noise and then Compute the normalized mean-square-error (NMSE)

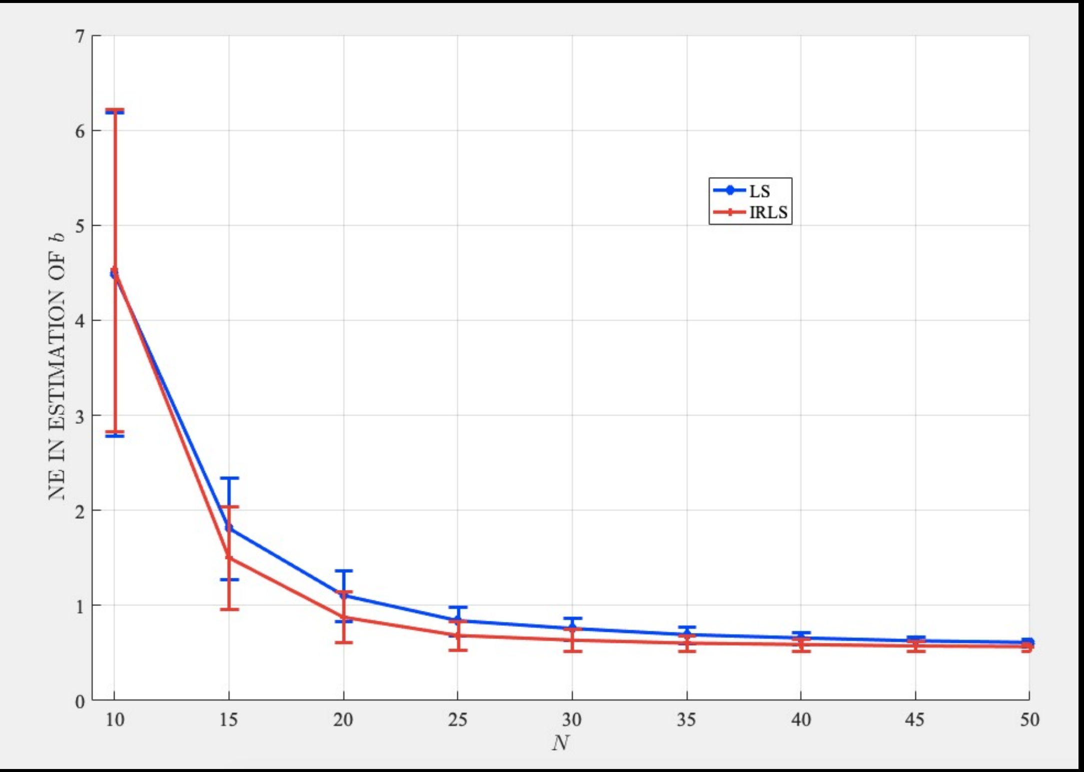


Fig. Comparison of LS and IRLS for estimation estimation of b: the value of b is 5. The estimates are inconsistent for $N < 20$

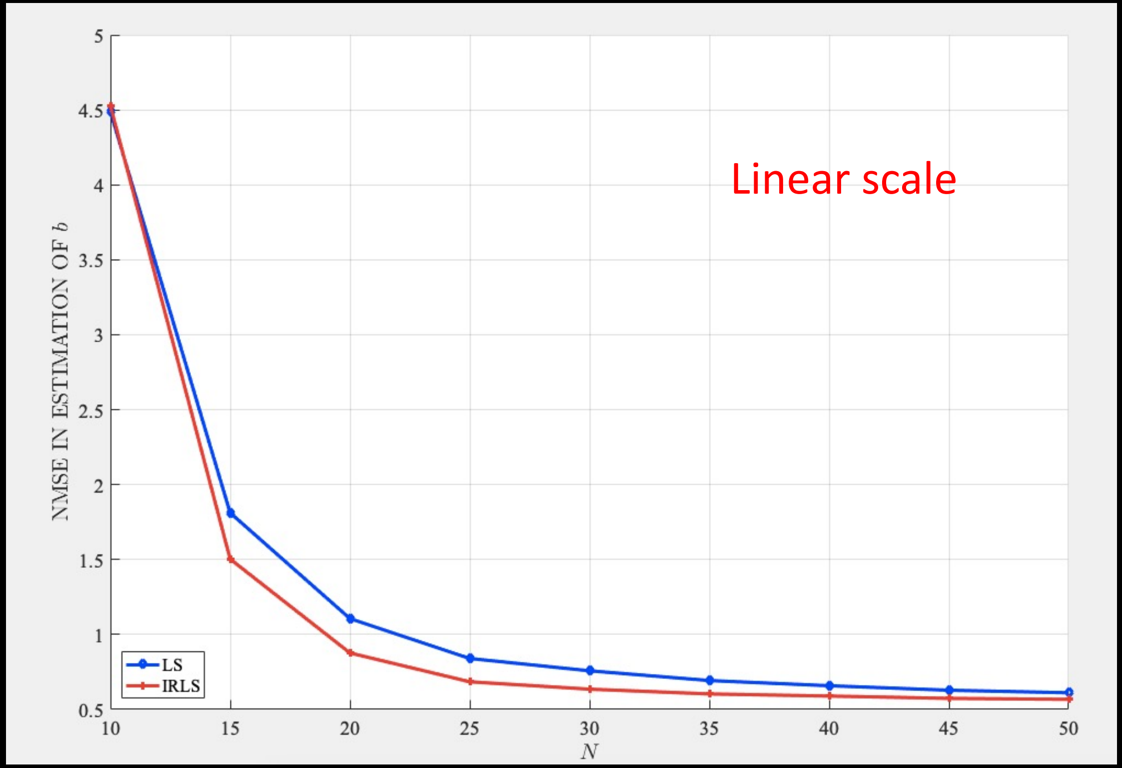


Fig. Comparison of LS and IRLS for estimation estimation of b: the value of b is 5. IRLS performs better than LS for $N = 10 \; to \; 50$

Monte-Carlo Simulations: For each N, fix a, b, and x values, estimate a, b for 1000 independent realizations of noise and then Compute the normalized mean-square-error (NMSE)

For a deep-learning problem…

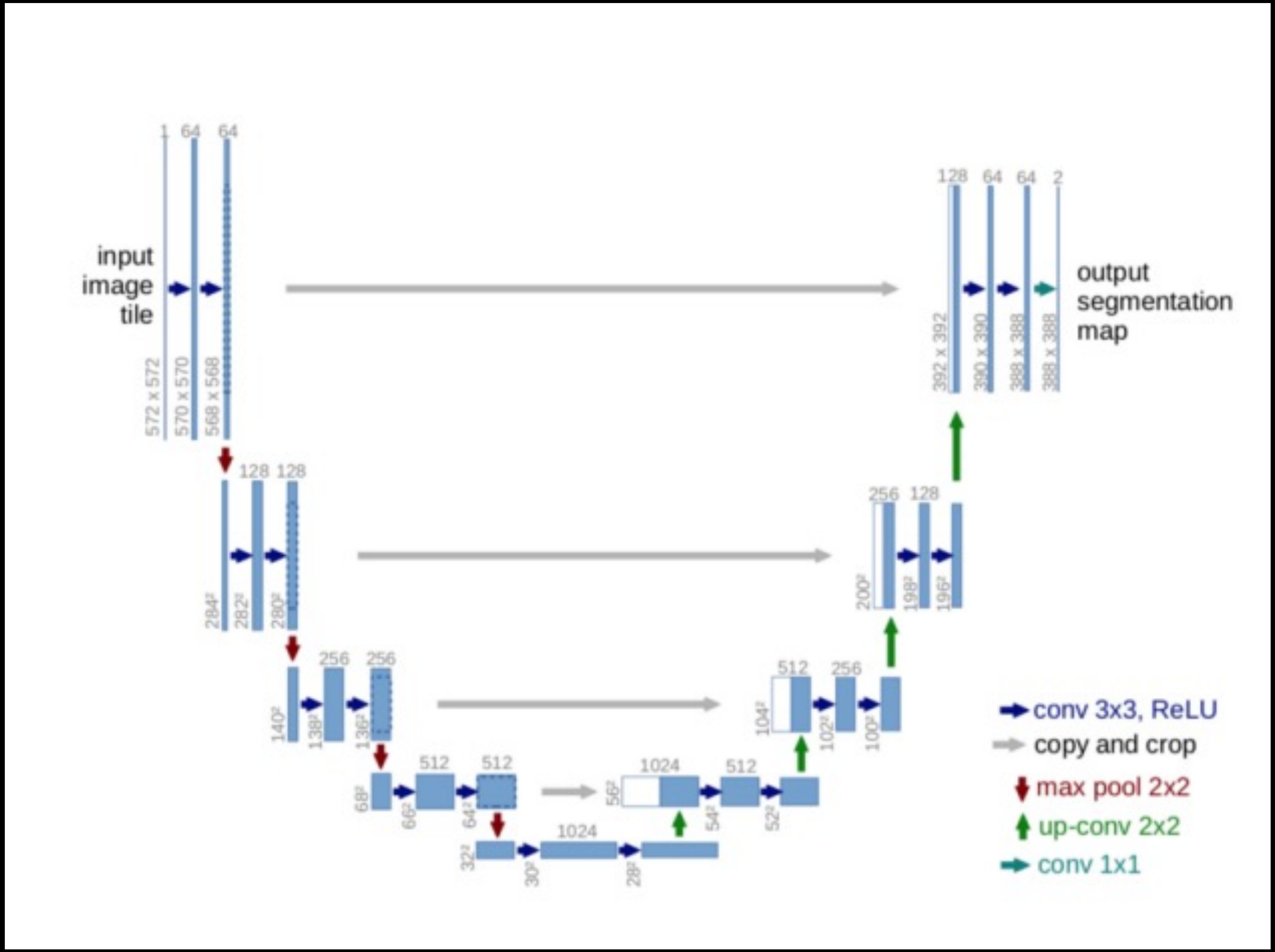
1. Add a block-diagram with all the details
2. For newly designed networks, add convergence plots
3. Chose correct error metrics for comparison and define them explicitly
4. Identify critical parameters and show comparative plots

Some pointers on presentation:

1. Understand your audience
2. Start with a generic setup
3. Use less text and more visuals
4. Be consistent with font style, size and highlighters
5. Use fewest possible equations
6. Avoid detailed proofs and algorithms
7. Ensure that fonts and colors are distinctly visible from a distance
8. Add references

U-net for image recosntruction

- U-net has a unique architecture where it takes an image as input and has an ability to output an entire image
- The architecture is symmetric and consists of two major parts — the left part is called contracting path, which is constituted by the general convolutional process; the right part is expansive path, which is constituted by transposed 2d convolutional layers



Results

The results are using MSE on each pixel on the generated image

```python
mse = tf.keras.losses.MeanSquaredError()
mae = tf.keras.losses.MeanAbsoluteError()
rmse = tf.keras.metrics.RootMeanSquaredError()
model.compile(loss=mse,
              optimizer='adam',
              metrics=rmse)
model.fit(X,Y,batch_size=1,epochs=5)
```

[8] ✓ 1259.3s

```
Epoch 1/5
200/200 [==============================] - 218s 896ms/step - loss: 0.0174 - root_mean_squared_error: 0.1423
Epoch 2/5
200/200 [==============================] - 245s 1s/step - loss: 0.0119 - root_mean_squared_error: 0.1223
Epoch 3/5
200/200 [==============================] - 261s 1s/step - loss: 0.0107 - root_mean_squared_error: 0.1146
Epoch 4/5
200/200 [==============================] - 266s 1s/step - loss: 0.0091 - root_mean_squared_error: 0.1111
Epoch 5/5
200/200 [==============================] - 269s 1s/step - loss: 0.0081 - root_mean_squared_error: 0.1069

<keras.callbacks.History at 0x29fdb619940>
```

+ Code + Markdown

Results

The results are using MSE on each pixel on the generated image

```
mse = tf.keras.losses.MeanSquaredError()
mae = tf.keras.losses.MeanAbsoluteError()
rmse = tf.keras.metrics.RootMeanSquaredError()
model.compile(loss=mse,
              optimizer='adam',
              metrics=rmse)
model.fit(X,Y,batch_size=1,epochs=5)
```

[7]

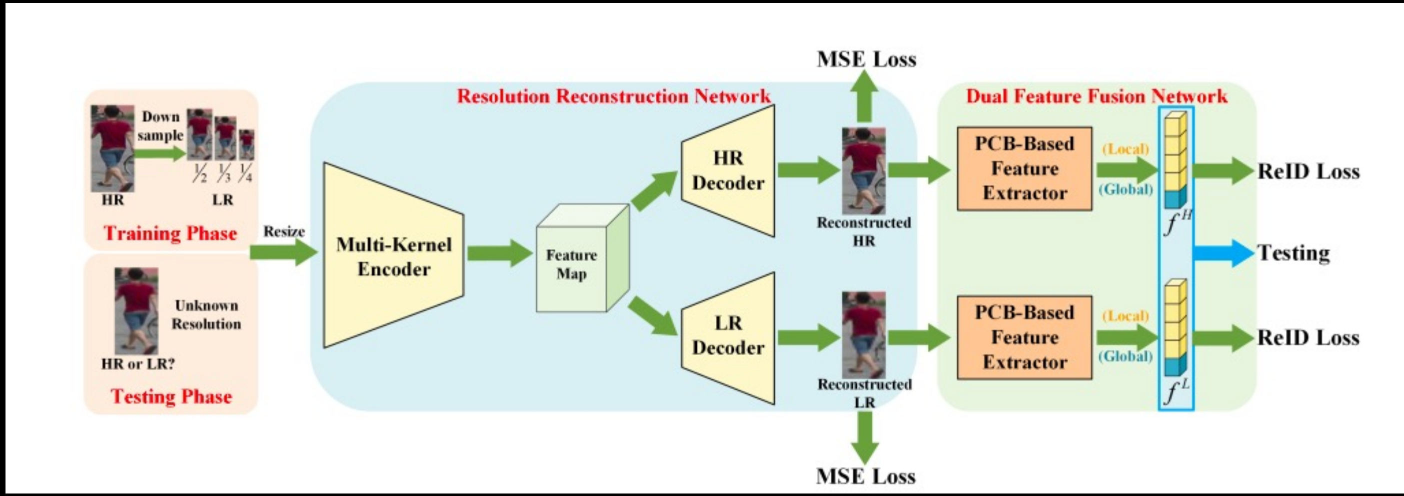
```
Epoch 1/5
200/200 [==============================] - 223s 953ms/step - loss: 0.0161 - root_mean_squared_error: 0.1360
Epoch 2/5
200/200 [==============================] - 252s 1s/step - loss: 0.0112 - root_mean_squared_error: 0.1178
Epoch 3/5
200/200 [==============================] - 285s 1s/step - loss: 0.0100 - root_mean_squared_error: 0.1121
Epoch 4/5
200/200 [==============================] - 296s 1s/step - loss: 0.0069 - root_mean_squared_error: 0.1066
Epoch 5/5
200/200 [==============================] - 300s 1s/step - loss: 0.0071 - root_mean_squared_error: 0.1024
<keras.callbacks.History at 0x1fb5fc3c670>
```

Comparison

- Lesser training time
- Better results on training and testing data
- Sharper recinstructed images

Low Resolution Information Also Matters: Learning Multi-Resolution Representations for Person Re-Identification

- Method consists of a Resolution Reconstruction Network (RRN) and a Dual Feature Fusion Network (DFFN).
- The RRN uses an input image to construct a High Resolution version and a Low Resolution version with an encoder and two decoders
- The DFFN adopts a dual-branch structure to generate person representations from multi-resolution images.



Dos and Don'ts

- Don’t copy paste figures (block diagrams and results) from any other paper
- In a technical presentation you can use graphics from other sources with proper citation
- Technical writing: Visual communication is not a replacement of text
- Technical presentation: Advisable to use visuals instead of text
- Don’t alter your results to show advantage
- Ensure that the codes are available online