FRTN65 - Exam solutions 2022 Jan

1 Dimensional Analysis

a) We will apply Buckingham's Pi-theorem. Dimensions of involved variables are

Quantity	Unit	Dimension
ω	Hz	T^{-1}
τ	kg/s^2	MT^{-2}
L	m	L
ρ	kg/m^3	ML^{-3}

To find dimensionless variables we need to find exponents a, b, c and d so

$$\begin{aligned}\Pi = \omega^a \tau^b L^c \rho^d = const \quad &\Leftrightarrow \quad (T^{-1})^a (MT^{-2})^b L^c (ML^{-3})^d = M^0 L^0 T^0 \\ &\Leftrightarrow \quad T^{(-a-2b)} M^{(b+d)} L^{(c-3d)} = M^0 L^0 T^0.\end{aligned}$$

This can be written as a linear system of equations

$$\begin{array}{c} \\ T \\ M \\ L \end{array} \begin{array}{c} \begin{array}{cccc} \omega & \tau & L & \rho \end{array} \\ \begin{bmatrix} -1 & -2 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -3 \end{bmatrix} \end{array} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = 0$$

It is easy to see that the first three columns of the matrix are independent, and hence the rank is 3. There will therefore be a 4-3 = one-parametric null-space, so that one dimensionless Π-variable can be formed. This null space can be found either by `null(A)` in matlab or by hand calculations, leading to

$$\begin{cases} a &= 2t \\ b &= -t \\ c &= 3t \\ d &= t. \end{cases}$$

With $t = 1$ we get a solution represented by integers $a = 2$, $b = -1$, $c = 3$ and $d = 1$, indicating the physical relation

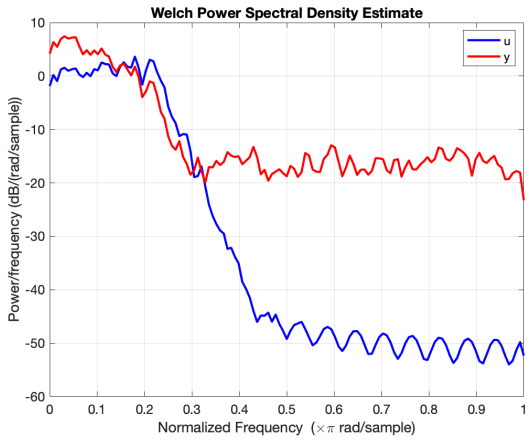
$$\tau = \text{const} \cdot \omega^2 L^3 \rho$$

which means that $(n_1, n_2, n_3) = (2, 3, 1)$.

2 System Identification Hands-on

A quick look on the data reveals no obvious outliers. It also seems that the mean values of u and y are close to zero, so we need not consider any signal offsets to get a linear model.

It is also a good idea to check excitation by plotting the spectra of input(lower plot) and outputs(upper plot). We see that we only have excitation up to about 0.25 of the Nyquist frequency, i.e. $0.25\pi/0.01 \approx 75$ rad/s. There is hence bad high frequency excitation of the system using the u,y data, and it is healthy to be skeptical to the system identification results obtained above 75 rad/s.



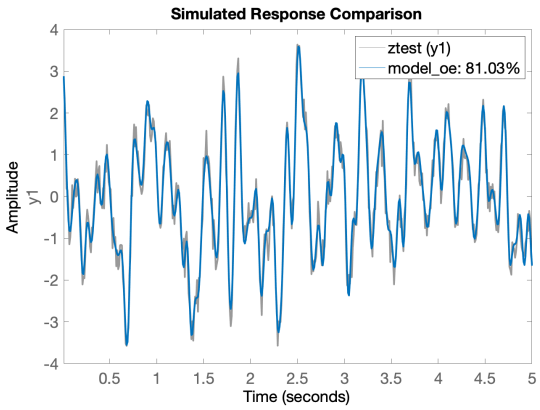
Since we have special test data (the step response) available, it is not absolutely necessary to split the (y,u) data into train and test sets. But it is still a good idea to do it. The results below are obtained by splitting the data into equally long train and test sets (500 time points each).

Instead of trying even higher ARX model order, we decide it is better to consider OE, ARMAX and BJ models of different orders. Testing OE models with structure

$$y(t) = \frac{B(z)}{F(z)}u(t - nk) + e(t)$$

of low order, we quickly find a good fit with model structure (3,3,1). (Some quick tests indicate that ARMAX or BJ models do not improve the performance)

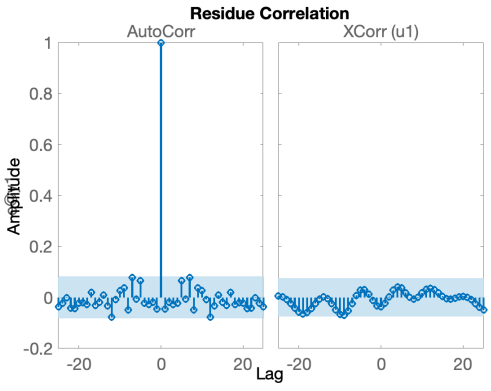
```
oe331 = oe(z,[3 3 1])
compare(ztest,oe331,arxbestInf)
resid(z,oe331)
plot(t,ystep),t,step(oe331)
bodeplot(oe331,{0.01,pi/h})
```

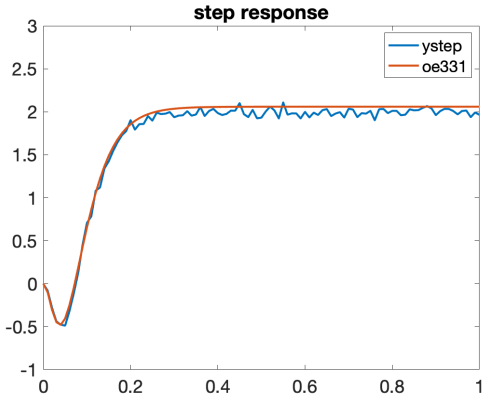


The explained variance(81 percent) is similar as for the 10th order ARX model, but we now use significantly fewer parameters (6 instead of 20), so our model should generalize better to new data.

Checking the reported estimated coefficients and their variances show that they have low uncertainty, which is a good sign.

```
model_oe =
Discrete-time OE model: y(t) = [B(z)/F(z)]u(t) + e(t)
B(z) = -0.1004 (+/- 0.01733) z^-1 + 0.008812 (+/- 0.04889) z^-2 + 0.1405 (+/- 0.03549) z^-3
F(z) = 1 - 2.123 (+/- 0.04968) z^-1 + 1.5 (+/- 0.08484) z^-2 - 0.3537 (+/- 0.03697) z^-3
```

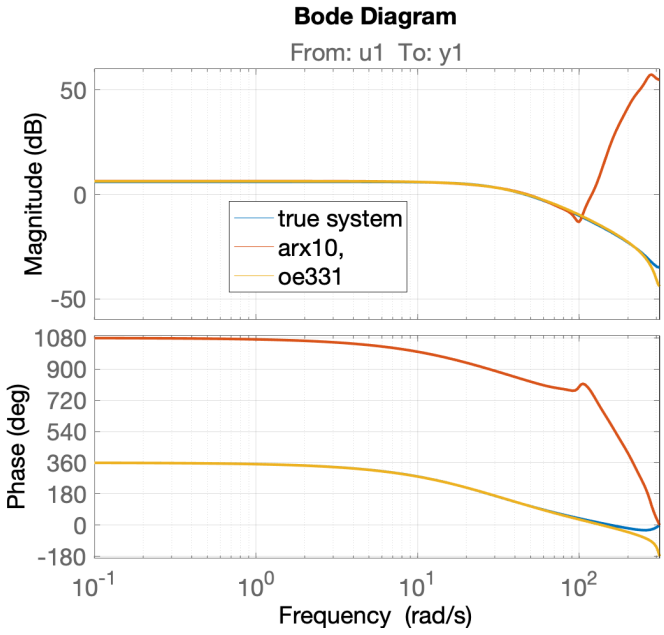




The plots of residuals and step response, also look fine. There is a clear indication of non-minimum phase response in the step response, visable if one zooms in on the initial part. One also sees that there is a small error in the static gain of a couple of percent, since the red curve stays slightly above the blue. (The model fit could therefore probably have been improved slightly at low frequency by also using the step response during system identifcation, since the step response give good excitation at low frequencies).

The following figure shows the Bode diagrams of the actual (unknown) true system (blue), oe331 (yellow) and arx10 (red). The blue and yellow curves are on top of each other, indicating a good

model fit. The ARX model has significant model errors at higher frequencies where the input excitation is poor and the wrong model structure has resulted in a significant bias due to influence of the noise.



Remark: The true system was actually given by the following 3rd order system with a time delay. There was 20 percent white noise added on the output.

```
sysc = 2*(1-0.05*s)/(1+0.03*s)^3;
sysd = c2d(sysc,h)

  -0.1231 z^2 + 0.06769 z + 0.1009
  --------------------------------
  z^3 - 2.15 z^2 + 1.54 z - 0.3679
```

Our identified OE model is hence close to the true system.

3 Supervised Learning - Classification

The data set describes features obtained from images of two types of raisins.

a) It is obvious from looking at the data that there is a big difference in the numerical sizes of the different features. The features 'eccentricity' and 'extent' have numerical values smaller than 1, whereas 'area' and 'convexarea' are 100000 times larger.

Therefore it is important to scale the features. The given code was prepared for this, since a

StandardScaler was imported in the beginning of the code, though never used. Introducing scaling is the most important change to improve the solution.

Some ideas of feature importances can be obtained from the correlation matrix, where we see that features 0,1,4 and 6 has a correlation with the target ("Class") above 0.6, whereas features 0,3,4 are lower correlated. Note though that correlation values measures linear relations, so it is only a rough indication, and also note that if two features are highly correlated with each other it might not be informative to pick both.

From the pairplots, one can get some ideas of good feature-pairs.

A solution using features 1,4,6 achieves 86-87 percent accuracy. It should be noted that similar results can be achieved with many other choices of features. Actually, using only one or two features can give results in the 80-85 percent range.

```
scaler = StandardScaler()
chosenfeatures = [1,4,6];
x_chosen = x_data[:,chosenfeatures]
x_train, x_test, y_train, y_test = train_test_split(x_chosen, y_data, test_size=0.4, random_state = 1)
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
KNNgood = KNeighborsClassifier(n_neighbors = 7)
KNNgood.fit(x_train,y_train)
y_predicted = KNNgood.predict(x_test)
print("accuracy =", accuracy_score(y_test, y_predicted.round())*100, "%")
accuracy=  87.2%
```

A suitable value of neighbors can be found through cross-validation.

Verifying performance on the test data gave accuracy of 87.2 percent (this number can vary depending on state of random generator used)

It is somewhat difficult to avoid data leakage during training if scaling is to be combined with cross-validation, since the scaling needs to be done on different parts of the data during CV. A solution to this problem is presented in this google colab code

The main thing is however, to evaluate the final performance on untouched data, and using an algorithm where all parameters have been obtained using only the training data.

It should be noted that similar results can be achieved with many other choices of features. Actually, using only one or two features can give results in the 80-85 percent range.

b) A random forest can report how important the different features have been for reducing the classification error. This can be achieved by

```
RF = RandomForestClassifier(n_estimators=500)
RF.fit(x_data,y_data);
RF.feature_importances_
```

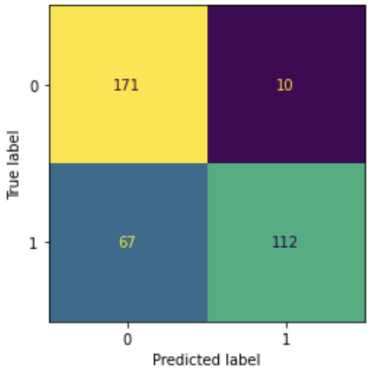
```
[0.13009528, 0.23972331, 0.07145606, 0.08999266, 0.1593534 ,
       0.06418065, 0.24519865]
```

The high numbers for features 1 and 6 indicate that these are most important, followed by feature 4 and 0.

c) A tradeoff between false negatives and false positives can be achieved by changing the probability threshold used in the binary classifier. (The KNN probabilities for each class is calculated simply counting the votes of the neighboors.) The following examples uses the probability threshold 0.8 (from the default value of 0.5)

```
cm = confusion_matrix(y_test,(probs>0.8).astype(int), labels=KNN.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=KNN.classes_)
disp.plot();
```

and results in this confusion matrix, where a low number of only 10 false positives have been achieved.



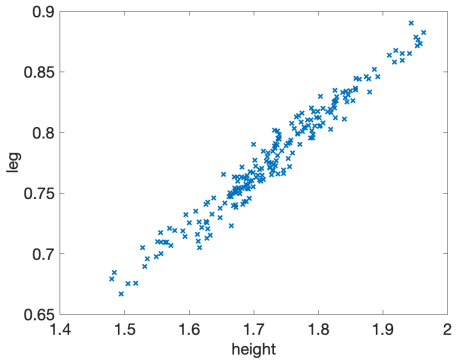
4 Linear Regression

a) The prediction should be done on data where mean values have been subtracted, and we need to add back the mean value ms to the resulting prediction:

```
Ypred(k) = ms + [height(k)-mh leg(k)-ml] * thetahat;
```

b) The features 'height' and 'leglength' are highly correlated, as can be seen from plotting the features, or by calcuating their correlation, in matlab done by: `corrcoef([height leg])`, resulting in a correlation value of 0.97. In fact leg $\approx 0.45 \cdot$ height for this data.

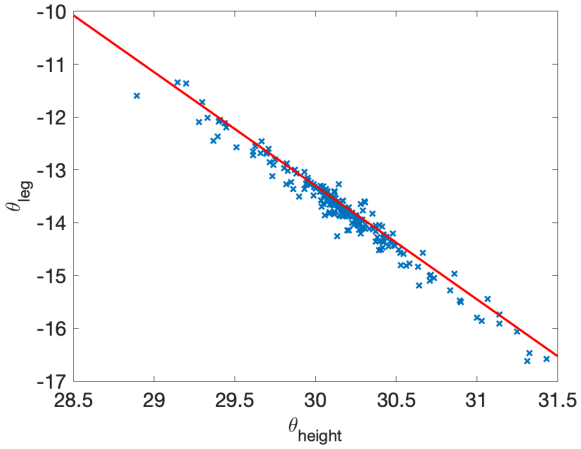
Another way to find the high correlation between the features is to calculate the singular values of the regressor matrix X using svd(X).



The result is that a linear combination of the two coefficients can be accurately identified, but that their individual values can be highly varying. In fact, a plot of θ_2 vs θ_1 shows that the parameter estimates are all located close to the line

$$\theta_1 + 24/51.2\ \theta_2 = 24$$

where the results of the previous one-feature estimators also lie.



c) The parameter `gamma` introduces so called Tikhonov regularization to the regression. This leads to a reduction of the size of the estimated parameters. This is a general way to reduce overfitting. A good side effect is that the normal equations also get better conditioned.

d) All the three different investigated estimators give close to equal performance. (A careful study will indicate that it is slightly better to only use the height feature, since the leg feature only introduces additional noise. The difference is however small).

5 Causal Inference and DAGs

a) After the intervention $X = x$ we get

$$y = (c_6 + c_4 c_7)x + \text{noise}$$

where the noise term is independent of x and has mean value zero. The best value for x is therefore $x = 15/(c_6 + c_4 c_7)$.

b) There are some backdoor paths from X to Y confounding the relation between X and Y. A standard LS regression $Y \sim X$ will therefore not give the correct result.

c) It is true that $\{B\}$ is a valid adjustment set, since it satisfies the backdoor condition. (In fact it is an example of parent adjustment.) Since both backdoor paths go via the variable B we can close them by adjusting by B, i.e. including B in the LS estimation. Since the relations are linear this will give us the true causal impact of X on Y. The correct OLS is hence

$$Y \sim X + B - 1$$

where the -1 symbol indicates that we do not want to estimate any constant bias. We can read of the causal impact of X on Y from the coefficient for X in this OLS. This gives the value 2. This means we should put $x = 15/2 = 7.5$.

The OLS estimate $Y \sim X + A - 1$ fails since it leaves one backdoor path (X-B-C-Y) open.

The OLS estimate $Y \sim X + C - 1$ fails since it closes a part of the causal path from X to Y that goes via C, it also leaves a backdoor path open.

Experimenting with the code will show that the OLS $Y \sim X + B - 1$ will give the correct value for x also for other values of the coefficients c_1 to c_7.