DF(H): Exam guidance

There have been some concerns over what is examinable in DF(H). This guide is intended to put your mind at ease and provide guidance on studying effectively. A full sample paper will be made available early in the second semester to help you revise, as there are no past papers for this course. The questions in this document are *roughly* indicative of the style of questions that might be asked.

What will be the format of the exam?

There will be three questions. You answer any two, of your choice. These questions will *typically* be from separate Units of the course, but some questions might require knowledge of several Units to answer fully. The exam will be marked out of 40; 20 for each question. You will have 90 minutes in the exam, as with most other Honours courses.

What is examinable?

You **should** know the written material (*excluding* the code blocks) and diagrams in the lecture notes.

You **should** know the numbered equations, including being able to write them down and clearly understand what they mean.

You **should** know the definition of **bolded** terms in the notes, although I don't expect precise formal definitions. You should hopefully remember the things I said in lectures, although anything which isn't written down in the lecture notes is not examinable for this course.

To do well, you **should** have a strong understanding of the concepts; an intuition as to how they would be applied in practice; and an understanding of the technical implications of using the concepts.

What is not examinable?

You are **not** expected to have read *any* of the supplementary material; either my supplements or the external resources recommended.

You are **not** expected to remember equations that are not numbered, but you should be able to discuss them in context if they were presented to you -- though this is not likely to be a major component of the exam.

You are definitely **not** expected to remember the code in the notes, but you should understand the gist of what is going on (unless the notes say that you don't need to understand it, in which case you can ignore it entirely). The code is provided to help you understand the concepts and let you experiment with examples -- and to show you how the abstract ideas work in practice. If you don't find this helpful, just look at the results the code produce and try and understand how the results are arrived at.

You are **not** expected to remember the details of the lab exercises, *except how to apply the concepts in the course to real-world problems*.

How should I study for this course?

When you come to study for this course, these are my recommendations:

When revising, read the Unit summaries at the start of each Unit's lecture, and check off the items as you work through the notes. This covers the key points you need to know.

If things are still unclear, look at the resources at the end of each Unit, particularly if there are any recommended resources. In particular try watching the recommended videos for the Linear Algebra section of the course if you found that part challenging. The resources aren't required, but they do explain things from another perspective.

Take note of **bolded** terms. These are important.

If any of the mathematical notation seems daunting or unfamiliar, buy the recommended book on notation and keep it by your side when studying. It really makes things easier. Whatever you do, don't panic.

It may be helpful to use the Jupyter notebook version of the notes, so you can run code examples and check whether your understanding of the concepts is correct. It's not necessary to do this, but the code is provided in full for that purpose.

If you didn't complete the labs, you may wish to try them when studying, but don't spend too long on them. The lab material is not directly examinable, but it greatly helps in understanding how the ideas are applied to problems. If you are confused, find someone who did the lab successfully and have them explain it to you!

What will I be asked to do in the exam?

You will **not** have to:

Write extensive code, more than a line or two: see below for examples.

Where any code is requested, it is **not** marked requiring perfect syntax or memory of the (very extensive) APIs. You just need to show you understand the operations that can be performed.

> For example, if a question asked for code for the mean vector of data stored as a matrix X of row vectors, the exact answer would be `np.mean(X, axis=0)`; but `mean(X, axis=0)`, or `mean(X,0)` or `sum(x,dimension=0)/X.shape[0]` or anything else reasonable would be fully acceptable.

Do any non-trivial algebra or other non-trivial mathematics (e.g. **definitely** no matrix algebra by hand). But you might be asked to explain what variables mean in equations; or how data would be used in an formula; or to translate simple expressions into NumPy; or vice-versa.

Remember formulae other than those numbered in the notes.

Do any numerical calculations by hand, *unless* they are trivial to do mentally, like:

> **Q**: What is the $L_{infinity}$ norm of the vector [1 2 3]?
> **A**: 3 (maximum element)
> **Q**: If P(x)=0.1 and P(y)=0.3, and x and y are independent, what is the probability P(x and y)?
> **A**: 0.1 * 0.3 = 0.03

You **will** have to:

Answer basic knowledge questions, similar to the quiz questions, but not in multiple choice format.

Q: Write down the definition of an **eigenvalue** of a matrix **A**: [3 marks]
A: Ax=λx; λ is the eigenvalue [Note: this is a numbered formula in the notes]

Q: What is a **coord** in the grammar of graphics? [2 marks]
A: a coord is a coordinate system and maps data attributes to spatial locations.

Q: What does *f(x)* being **Lipschitz continuous** mean? [2 marks]

A: the derivatives of f(x) are smaller than some fixed constant, or any equivalent statement. [Note: we haven't covered this yet!]

Answer questions about the properties and uses of concepts in the course.

Q: In a strided array memory representation, what is the time complexity of transposition of a tensor, and how is it implemented? [3 marks]

A: The time complexity of tensor transpose is O(1) [or O(D) if you are being completely correct], and it is implemented by reversing the order of the strides.

Q: Give a reason why a faceted graph might be preferred over a layered graph? [3 marks]

A: A faceted graph allows different coordinate systems to be used for different attributes, while a layered graph requires all of the attributes to be plotted on the same coordinate system; alternatively an argument about reducing clutter would be acceptable.

Q. If the *exponent* of a IEEE 754 floating point value is all ones (111111111111…), what **three** values could this float possibly represent, and how are they distinguished? [4 marks]

A: `inf`, `-inf` and `NaN`. If the mantissa is all zero, then the number is inf, with the sign bit indicating the sign. Otherwise, the number is NaN.

Answer technical questions in context

Q: Given a 80,40,3 tensor `img` representing an RGB image in the format (rows, cols, colors) how would you compute the mean RGB colour over the entire image? [3 marks]

A: `np.mean(np.mean(img, axis=0), axis=0)` or `np.mean(img, axis=(0,1))` or anything plausible which indicated that the mean could be computed over the first two axes.

Q: Which floating point exception would the operation 0.0/0.0 invoke, if any? [1 mark]

A: Invalid operation (or undefined operation, or similar words)

Q: How would you implement the equation: $A^{-1}X^{T} + dY$ in NumPy? Assume d is a scalar and A, X and Y are matrices. Remember that the inverse is `np.linalg.inv()` in NumPy. [3 marks]

A: `np.linalg.inv(A) @ X.T + d * Y`

Q: Explain four key steps in vectorising problems. Give a **brief** example of the type of operation that might be performed for each of these steps. [8 marks]

A:
Collecting input data into suitable numerical arrays; e.g. reshaping data
Generating auxiliary arrays, such as index arrays; e.g. using arange()
Performing vectorised operations on these arrays; e.g. using sum()
Selecting and collating results; e.g. using slicing

Apply ideas to solve real world problems. **This is where most of the marks will be.**

A client wants you to find a way to make light bulbs last as long as possible. She is considering whether to hire a PhD student to mathematically model the lifetime of bulbs given the pressure of the gas inside of them and the type of gas (argon, xenon or radon). The client already has a collection of experimental data about the light bulb lifetime, in the form of measurements in three columns of a matrix:

```
[[lifetime, pressure, gas_type]
 [lifetime, pressure, gas_type]
 [lifetime, pressure, gas_type]
 [...]]
```

`gas_type` is a number that can be 0=argon,1=xenon or 2=radon.
`lifetime` is a measurement in hours (h).
`pressure` is a measurement in kilopascals (kPA).

You don't need to know anything about lightbulbs to answer this question.

(a) Recommend a visualisation strategy to show the lifetime of the bulbs at the next board meeting. Draw a diagram to show how the figure should look. You can assume the caption for the figure is: "*This chart shows effect of gas pressure on lightbulb lifetimes for xenon, argon and radon.*" [5 marks]

Answer: A layered scatterplot or line plot would be appropriate here. The diagram should have labels showing pressure in kPA on the x axis, and the lifetime on the y axis. The geoms should be identifiable in some way (e.g. dashed styles or marker shapes) and there should be a legend to indicate the layers. Faceted is a reasonable alternative, if labeled correctly.

(b) What type of problem is this, and should the client hire the PhD student? Justify your hiring choice. [2 marks]

This is an optimisation (or maximisation) problem. Hiring the PhD student would make it possible to run the optimisation without doing real-world tests, using the model instead, and accelerate the optimisation process. She should hire the student.
[any argument that references the idea of a model speeding up the optimisation is valid here]

(c) Write down the parameters of this problem, and the objective function. [2 marks]

The objective function is the the measured lifetime of the bulb, assuming we are maximising.
The parameters are the gas type and the pressure.
[note: reading ahead here would help with part (b)!]

(d) Focusing on just one gas type, the PhD student has managed to build a model for argon filled bulbs. He has modeled the problem as a function of the pressure using some NumPy code. He has decided to use *grid search* to find the best solution.

Explain how automatic differentiation could be used here, and, with this in mind, suggest an algorithm that would be better than grid search, and why it would be better. [3 marks]

Answer: Automatic differentiation could (potentially) find the gradient of the objective function directly from the model code. This would allow the use of first-order optimisation methods like gradient descent. This is much faster than grid search, in the sense that it would require many fewer evaluations of the objective function to converge.

(e) The PhD student has ignored your advice and proceeded to use grid search in the final product. The client has issued new requirements that the *pressure, operating temperature, dopant quantity, glass density, and light colour temperature* be taken into account. While the mathematical model is very easily extended to include these variables, explain why the PhD student now looking very unhappy about the grid search choice. [2 marks]

Answer: the curse of dimensionality means that grid search will be exceptionally inefficient. The number of objective function evaluations for grid search is exponential in the number of parameters.

(f) As part of a background survey, data has been collected from a large collection of real-world light bulbs, which includes the five variables above, measured for each bulb: `pressure, operating temperature, dopant quantity, glass density, and light colour temperature`

Describe how the PhD student should find the two most important ways in which real-world lightbulbs vary, in terms of **weighted sums** of the variables above. Along with an explanation, your answer should provide NumPy code which will print these out, assuming the data is stored in a matrix `lightbulbs`, where every *row* is the five measurements for one bulb. Include a line of code that will print out the average (in the sense of the arithmetic mean) bulb measurements -- the "typical" bulb. [6 marks]

Answer: This can be computed from the first two principal components, which are the eigenvectors of the covariance matrix.

```
evals, evecs = np.linalg.eig(np.cov(lightbulb_vars, rowvar=True))
print(evecs[0:2])
print(np.mean(lightbulb_vars, axis=0))
```

*[Note: it is **not** necessary to get the return values from np.linalg.eig correct, nor to to specify rowvar=False to get full marks, and it is fine to have some variation of np.linalg.eig which is not quite exact, like np.eigenvecs() -- but there should be an eigenvector computation and a covariance matrix computed somewhere, even if the syntax is a bit wonky]*