Computational Physics 301

Dr Jim Brooke

Office 4.52
james.brooke@bristol.ac.uk

Lecture Overview

- Course Outline
- Practical Advice
- Assignment 1 - Linear Algebra

Course Outline

Numerical Methods

- Using computers to solve scientific problems
- Most, if not all, real life physics problems do not have analytic solutions
- Numerical methods can give answers to a specific precision
- Computational methods are an essential tool in modern physics research

Learning Outcomes

- Write programs that can manipulate and analyse scientific data
- Simulate experimental data
- Evaluate theoretical problems
- Present results clearly, accurately, concisely

Your tasks

- Choose appropriate algorithms
  - Accuracy, robustness, efficiency
- Implement algorithms in code
- Use algorithms from appropriate libraries
  - Hard work has usually already been done !
- Use code to solve problems numerically
- Test and analyse your solution

Relationship to 2nd year course

- Exercises less prescribed, more open-ended
- Higher standard of analysis expected
- Complementary topics :
  - Linear algebra and matrices
  - Partial differential equations
  - Monte Carlo simulation

Course Structure

- Three assignments - that you work through in your own time

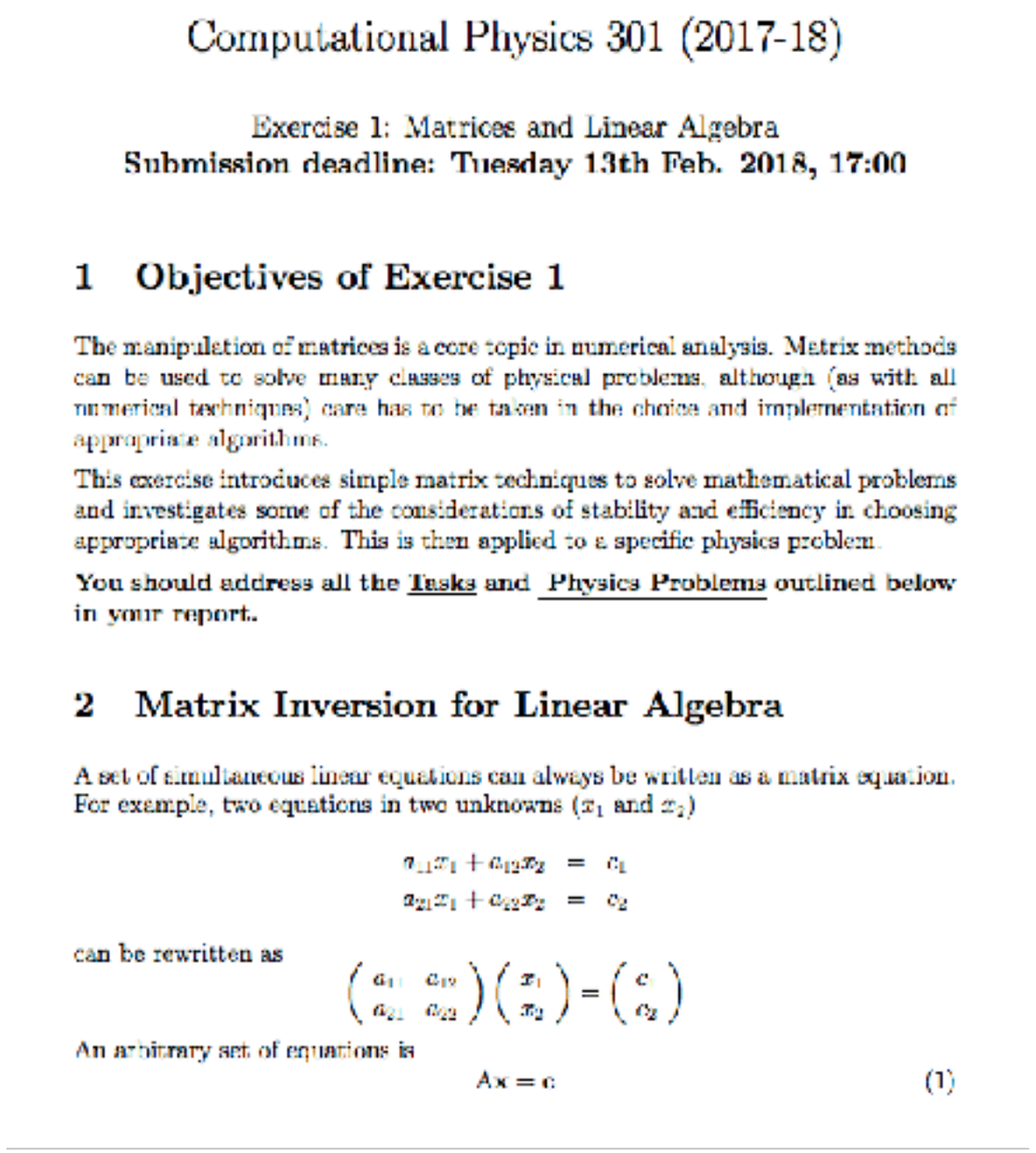
- Lectures - one (short) lecture at the start of each assignment

- Drop-in sessions - get help and advice from demonstrators

- 10 credit point unit - expect to spend ~100 hours on this course !

Assignments

- 1st assignment is on Blackboard
  - **Task 1** : Write a general routine for matrix inversion
  - **Task 2** : Use some functions from ***scipy.linalg***
  - **Task 3** : Solve a physics problem
- Assignments 2 & 3 will be similar



Computational Physics 301 (2017-18)

Exercise 1: Matrices and Linear Algebra
Submission deadline: Tuesday 13th Feb. 2018, 17:00

1 Objectives of Exercise 1

The manipulation of matrices is a core topic in numerical analysis. Matrix methods can be used to solve many classes of physical problems, although (as with all numerical techniques) care has to be taken in the choice and implementation of appropriate algorithms.

This exercise introduces simple matrix techniques to solve mathematical problems and investigates some of the considerations of stability and efficiency in choosing appropriate algorithms. This is then applied to a specific physics problem.

You should address all the Tasks and Physics Problems outlined below in your report.

2 Matrix Inversion for Linear Algebra

A set of simultaneous linear equations can always be written as a matrix equation. For example, two equations in two unknowns (x_1 and x_2)

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &= c_1 \\ a_{21}x_1 + a_{22}x_2 &= c_2 \end{aligned}$$

can be rewritten as

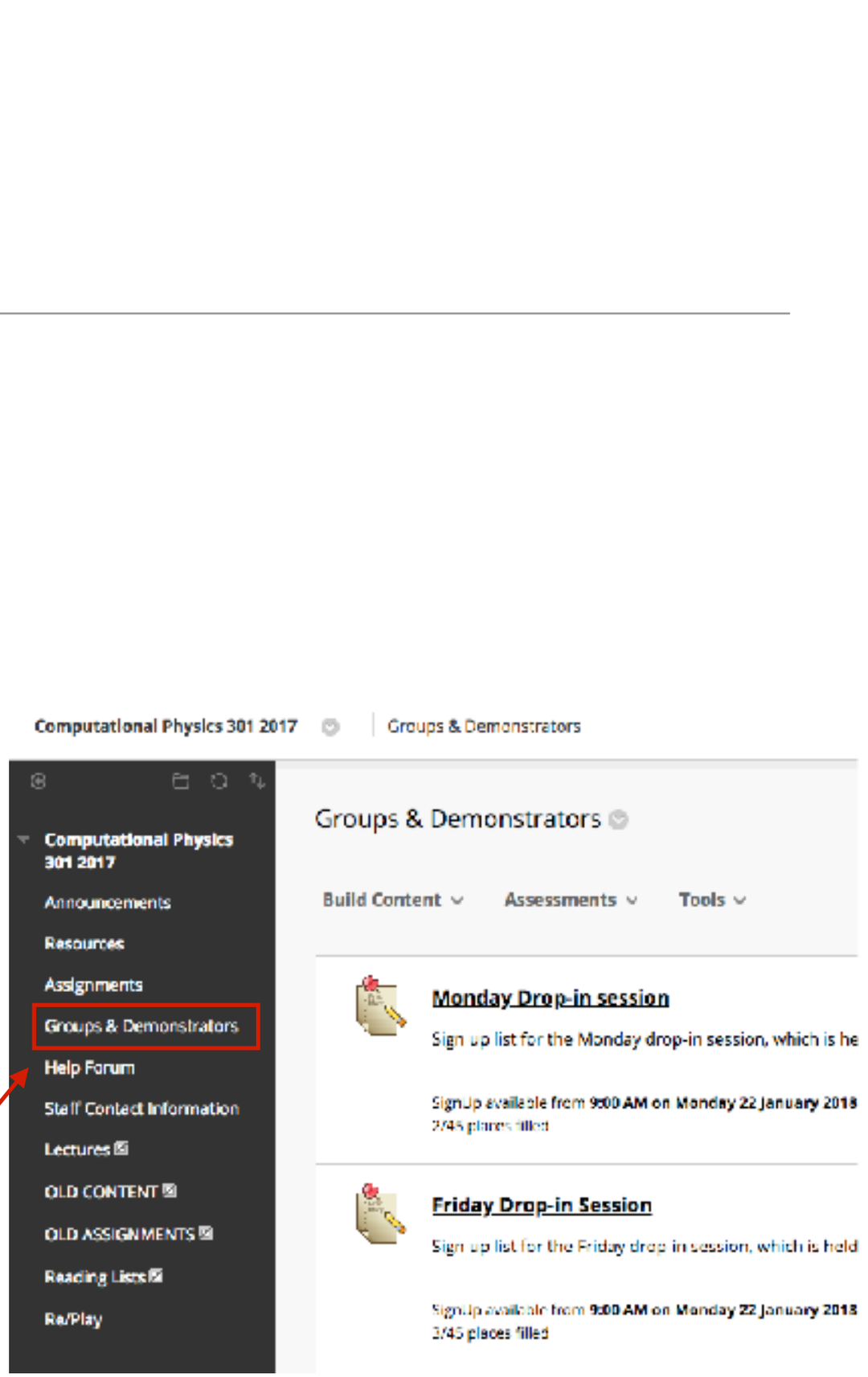
$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

An arbitrary set of equations is

$$\mathbf{A}\mathbf{x} = \mathbf{c} \qquad (1)$$

Drop-in Sessions

- **Friday 14:00-17:00, in 1.14**
  - Advice and help from teaching staff
  - Discuss progress with your demonstrator
  - Get feedback on work submitted
- Space in 1.14 is limited; two sessions
  - **14:00-15:30** & **15:30-17:00**
  - Please sign up for one session on blackboard



Deadlines

- Deadlines are all **Tuesday 5pm**
  - Assignment 1 - Week 16 (13th February, 5pm)
  - Assignment 2 – Week 19 (6th March, 5pm)
  - Assignment 3 – Week 22 (17th April, 5pm)

Assessment Procedure

- Submit report & code ***via Blackboard*** by deadline - late penalties apply !
  - Submit code as a single text file - change .py to .txt
  - Include instructions for running the code in comments !
  - Report as Word doc or PDF
- Your demonstrator will ***run your code*** and mark your work
- You will receive feedback, with a preliminary mark ~1 week after submission
- Your marks are moderated by me and finalised
- Contact me if you have any questions about assessment or feedback

Assessment

- **Code - 50%**
  - Does it solve the problems set out in the assignment ?
  - Appropriate use of methods, loops, etc. ?
  - Are test functions included ?
  - Is it clear and well commented ?
- **Report - 50%**
  - Are the results requested presented ?
  - Are the results analysed and discussed ?
  - Are the limitations of the code understood ?

Marks

- Assignment 1 - 20%
- Assignment 2 - 35%
- Assignment 3 - 45%

Plagiarism

- Plagiarism rules apply to code as well as written work
- The code you submit should be your own work
  - Not copied from other sources e.g. the web
- Don’t do it, you’ll end up in serious trouble !

Practical Advice

Computers & Language

- Use your laptop, PCs in 1.14, UoB remote desktop
- We expect you to complete the assignments using Python
- Other languages by prior agreement ONLY

Recommended Software

- Recommend using Anaconda installation which includes :
  - Jupyter Notebook for editing
  - Matplotlib for plots
  - Numpy for arrays, and math operations
  - Scipy for algorithms (expect when explicitly asked to write your own!)
- LaTeX for report writing
- But these are not compulsory…

Books

- Useful books are the same as for the 2nd year course
  - "*A Student's Guide to Python for Physical Modeling*", Jesse M. Kinder, Princeton University Press
  - "*Learning Scientific Programming with Python*", Christian Hill, Cambridge University Press
- Also the definitive work on the subject :
  - "*Numerical Recipes - The Art of Scientific Computing*". Press, et al. Cambridge University Press.
  - Also available online at http://numerical.recipes/

Online Resources

- Simon Hanna's Python tutorial
  - In particular, familiarise yourself with Section 5 on advanced Python and Numpy
- Exemplar reports from previous years
  - Page limits were different then ! You are expected to fill 3-4 pages with results & analysis
- General advice on code and report writing from past demonstrators
- Help Forum
  - Please use it !
- General programming websites eg. stackoverflow

Practical Tips

- **Writing the code**
  - Start with simplified versions, or special cases
  - Make sure you test that your results are correct - think about how you will do this !
  - The code you submit to Blackboard should produce results for all tasks and tests
  - Debug from the outset - don’t write 100s of lines before you run it
- **Writing the reports**
  - Typically 3-4 pages
  - No need for lengthy introductions, restating the problem etc.
  - Focus on your code, the results and their analysis
  - Don’t forget to mention how you know your code works… ie. testing !

Practical Tips

- Don’t leave programming to the last minute - debugging and testing takes time !

- Prioritise - don’t waste time on things that are not important

- Seek help from teaching staff - advice with code structure/design, reports

- Come to a drop-in session every week - not just the one before a deadline !!!

What Now ?

- Download Assignment 1 and get started
- Sign up for either Monday or Friday drop-in sessions
- Make sure you have a Python/numpy/scipy installation
- Come to the drop-in session on Friday

Assignment 1

Linear Algebra

Linear Algebra

- Many applications in physics
  - geometry, mechanics, electromagnetism, relativity, quantum mechanics, …
- And computing
  - graphics, image processing, cryptography, machine learning, optimization, graph algorithms, information retrieval, web search, …
- Even useful for later assignments in this course…

Simultaneous Equations

- Consider a set of simultaneous equations

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 &= c_1 \\
a_{21}x_1 + a_{22}x_2 &= c_2
\end{aligned}
$$

$$
\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}
$$

- Many methods to find a solution for (x_1, x_2) :
  - Matrix inversion
  - Gaussian Elimination
  - Matrix decomposition

Matrix Inversion

- A solution becomes clear if we use the matrix form :

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &= c_1 \\ a_{21}x_1 + a_{22}x_2 &= c_2 \end{aligned} \quad \rightarrow \quad \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

$$\mathbf{Ax} = \mathbf{c} \quad \rightarrow \quad \begin{aligned} \mathrm{A}^{-1}\mathbf{Ax} &= \mathrm{A}^{-1}\mathbf{c} \\ \Leftrightarrow \mathbf{x} &= \mathrm{A}^{-1}\mathbf{c} \end{aligned}$$

- The solution is found using the inverse of the matrix

Task 1 - Matrix Inversion

- Write your *own* routine to invert a NxN matrix, for at least N <= 4
- Use the standard formula (Cramer's rule) : $$\mathrm{A}^{-1} = \frac{1}{\det \mathrm{A}} \mathrm{C}^{T}$$
- C is the matrix of *co-factors*, with the *co-factor* $C_{1,2}$ given in terms of the *minor* $M_{1,2}$:

$$C_{1,2} = (-1)^{1+2} M_{1,2} = -1^3 \times 36 = -36$$

$$M = \begin{pmatrix} & & \\ 4 & 0 & \\ -1 & 9 & \end{pmatrix} \rightarrow M_{1,2} = \det \begin{pmatrix} 4 & 0 \\ -1 & 9 \end{pmatrix} = (4 \times 9) - (-1 \times 0) = 36$$

No linear algebra libraries allowed in this task!
(But numpy arrays are OK)

Matrix Inversion

- However, there are several potential problems..
  - Singular matrices may have no solution !
  - Inverting large matrices can be time-consuming
  - Roundings errors can accumulate
    - Large matrices, or near-singular matrices…

- How can you test and quantify these things in your assignment ?

Gaussian Elimination

LU Decomposition

- Solving a matrix equation is straightforward if it is triangular
  - ie. the entries above or below the diagonal are all zero
- LU decomposition involves writing a general matrix as the product of two triangular matrices

$$\begin{pmatrix} a_{11} & 0 & 0 & \dots \\ a_{21} & a_{22} & 0 & \dots \\ \vdots & & & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \end{pmatrix}$$

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}, L = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix}, U = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

- And solving the resulting two triangular equations $\mathbf{Ly} = \mathbf{c}$ $\mathbf{Ux} = \mathbf{y}$

Singular Value Decomposition

- Sometimes we have more unknowns than equations
    - eg. MxN matrix with M<N, or M=N but equations are degenerate
        - There may be no solution, or a space of solutions
- Or we have a square matrix with determinant zero
- Such matrices are called singular, and not invertible

Singular Value Decomposition

- SVD useful for singular matrices
    - And near-singular, eg. when 1/det(A) is close to floating point precision, and rounding error becomes a problem
- Write general matrix A as : $A = U\Sigma V^T$
- Where U and V are orthonormal matrices, and D is a diagonal matrix of the singular values
- Then we can calculate : $\bar{x} = V\Sigma^+ U^T c$
- If A is non-singular : $\bar{x} = x$
- And if it is, $\bar{x}$ is interesting for a variety of reasons…

Σ^+ is obtained by taking the inverse of each non-zero element on the diagonal of Σ and then transposing the result.

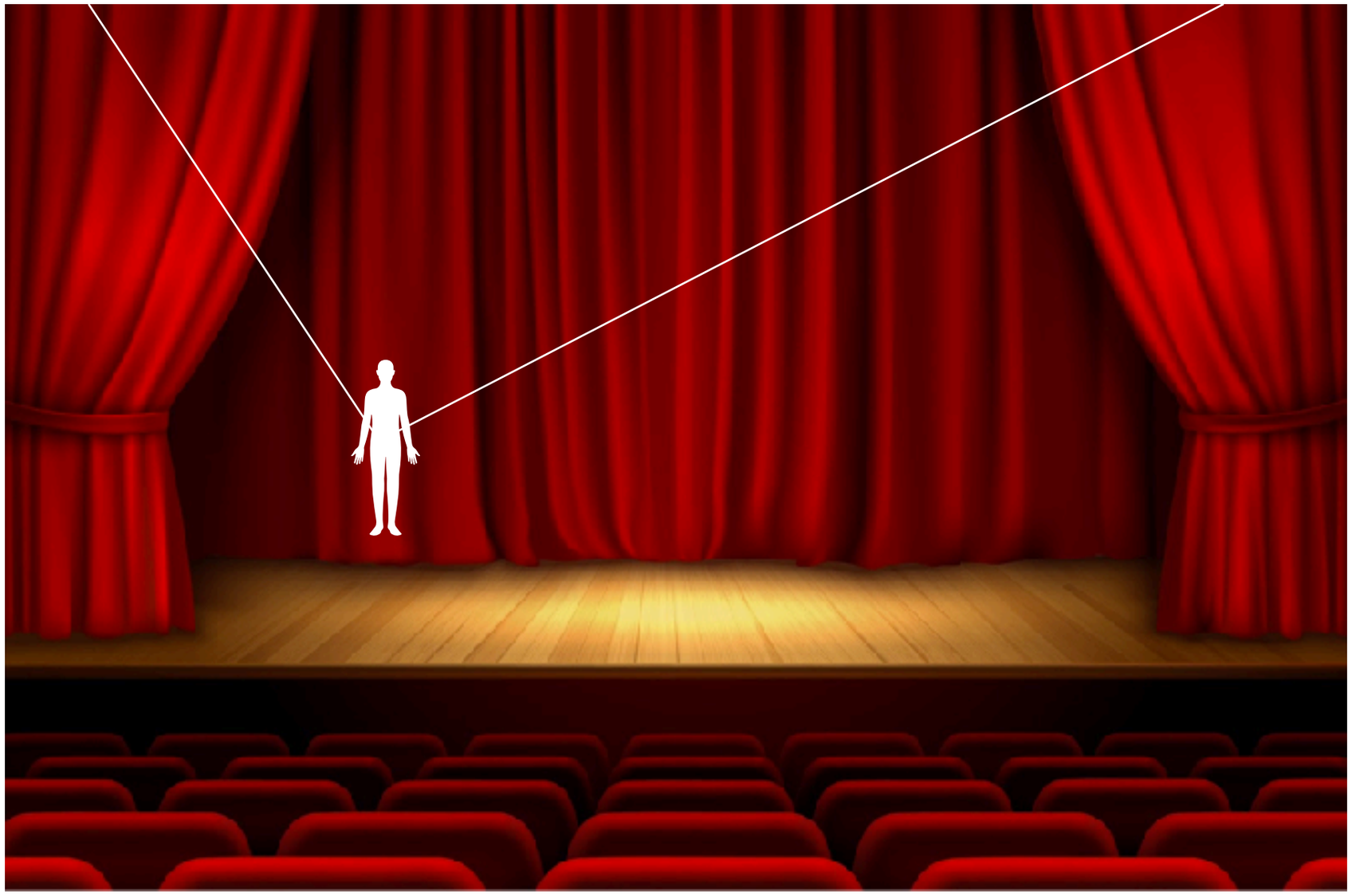
Task 2

- Compare LUD and SVD methods for solving simultaneous equations with your own analytic matrix inversion method.
  - Look at a range of equation sets
    - Including special cases, eg. near singular matrices
  - How fast are the different methods for solving them ?
  - How do the results compare ?
- Read more on LUD, SVD and other matrix factorisation techniques, in Numerical Recipes

Physics Problem

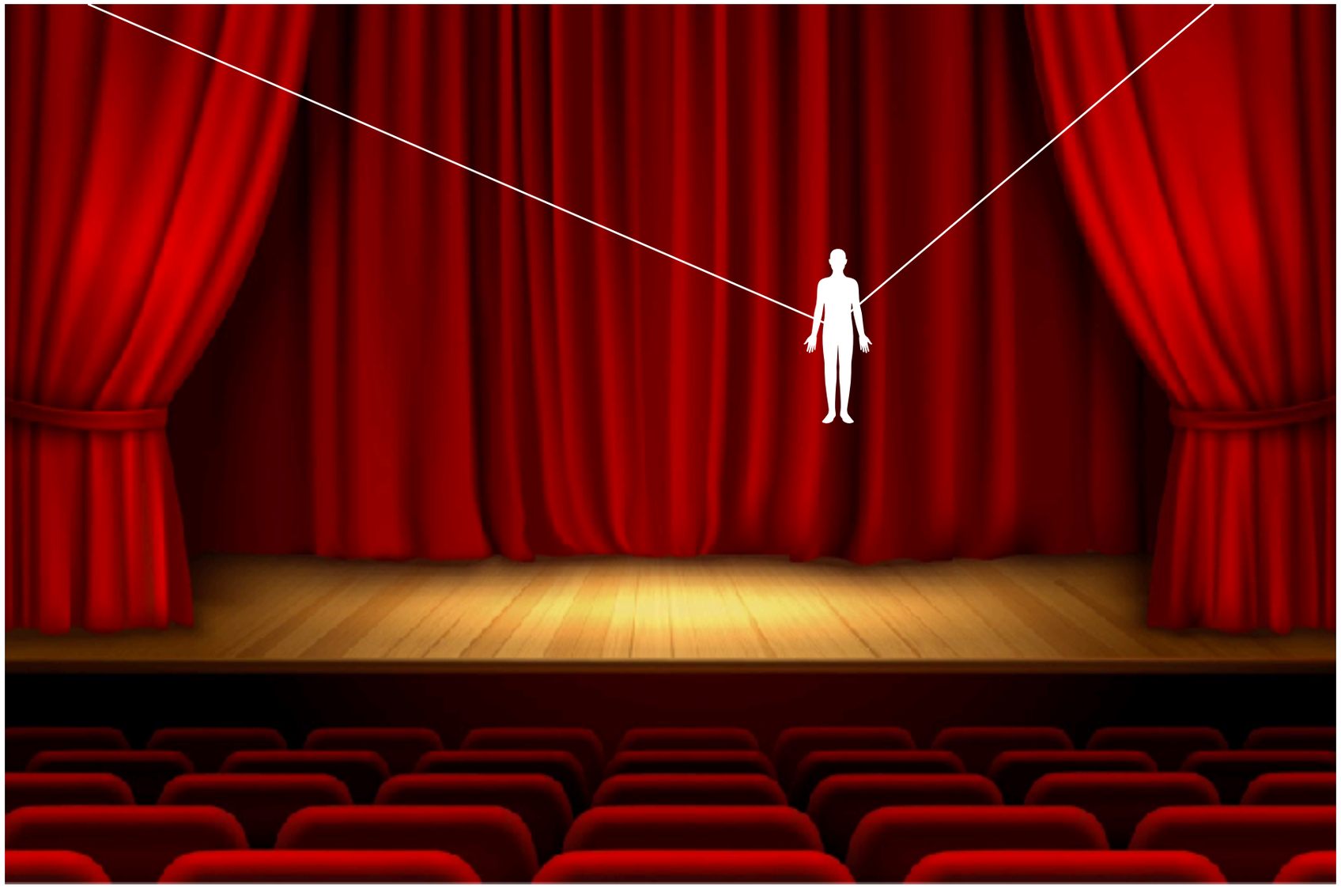
A trapeze artist is suspended above a stage by a system of 3 wires. Each wire is fed from a motorised drum, such that the artist appears to fly around above the stage by controlling the length of the wires. The stage is 15 m wide, 8 m deep and the wire drums are 8 m above the stage. Two drums are fixed above the front corners of the stage, and the third is in the centre at the rear. The artist has a mass of 70kg - you can ignore the mass of the wires.

Physics Problem



Consider front two wires only - plot the tension in one of them as a function of position (x, z)

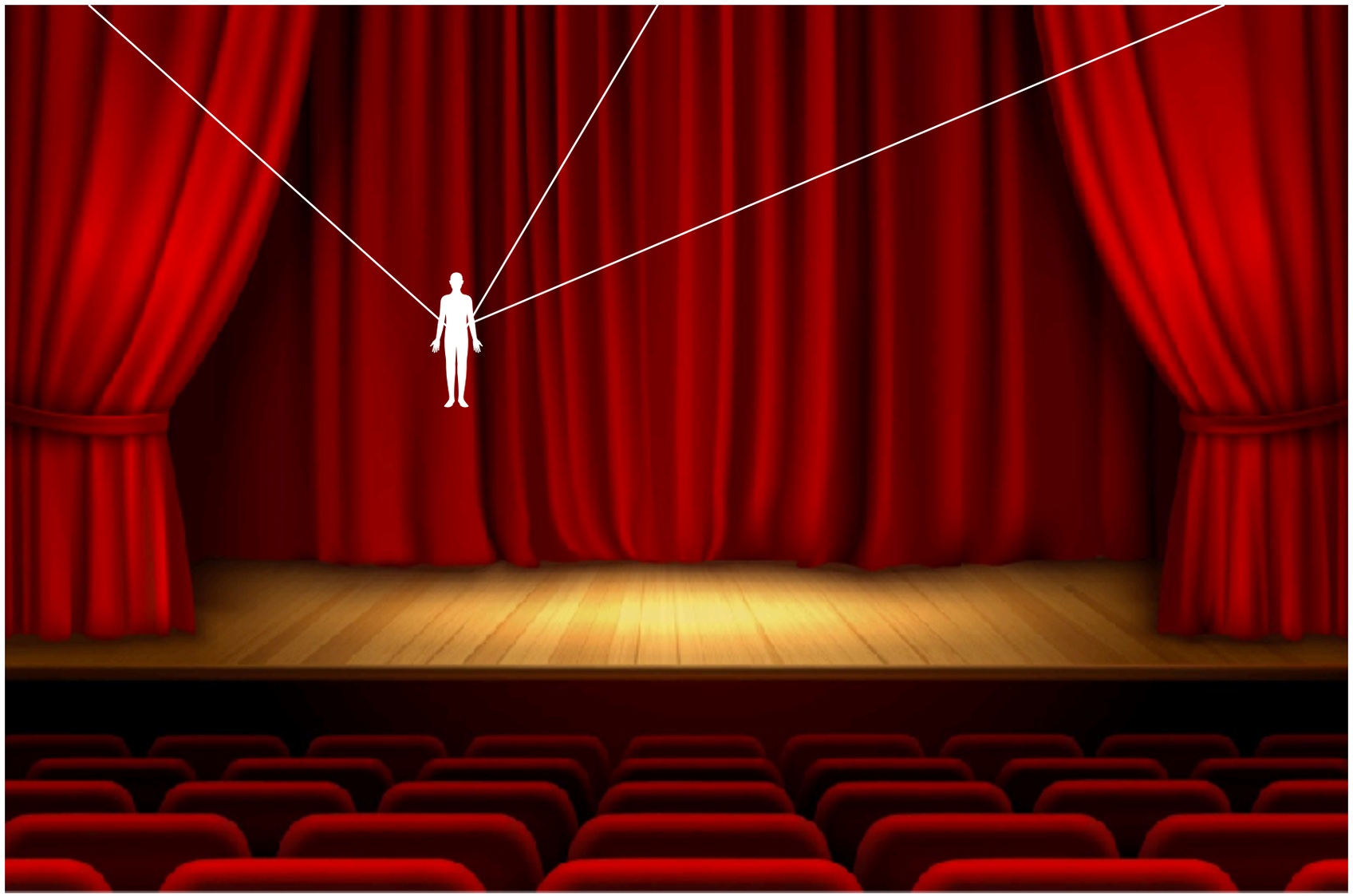
Physics Problem



Consider front two wires only - plot the tension in one of them as a function of position (x, z)

What is the maximum tension, and where does this occur ?

Physics Problem



Then consider all 3 wires - where is the maximum tension now ?

Can you think of ways to represent the tension as function of position ?