Year 1 MT Vacation Work, Vectors and Matrices

Your college tutors will normally set vacation work. However, below are a few suggestion on what you might do to revise and consolidate the material as well as deepen your understanding.

1 Revision and Consolidation

As a normal course of action you should go over the problem sheets and focus on the questions which you haven't yet managed to solve or where you had problems the first time. Usually, this will point to a more general problem you have had with the material. Go back and read up on it in the lecture notes or in a book first, then come back to the question.

The lecture course has given you a coherent and logical exposition of the material with proofs for all essential statements. Of course, you will not be asked to proof theorem X or Y in a collection or in your prelims exam. Rather, collections and exams will focus on the specific applications we have discussed, typically in low dimensions (that is, dimensions 2,3 and 4). Concentrating on the un-starred questions on the problem sheets will give you some indication on what these are. To be specific, below is a checklist which should help you to get ready:

- basic vector manipulations: adding of vectors, scalar multiplication, linear combinations, checking linear independence, checking if vectors form a basis, finding coordinates of a vector relative to a basis
- scalar, vector and triple product: being able to manipulate expressions involving these products, definition of length of vectors and angles between vectors, applications to geometry such as finding perpendicular vectors
- lines and planes: describing lines and planes with vectors, Cartesian and vector form, how to set them up and convert between them, calculating intersections of lines and planes, finding minimal distances from lines/planes to points or between two lines in 3d
- basic matrix manipulations: addition and scalar multiplication of matrices, multiplication of matrices, transpose and hermitian conjugate
- more advanced matrix manipulations: row reductions, computing the rank of a matrix using row reduction, computing the inverse of a matrix using row reduction or the co-factor method
- systems of linear equations: understanding the structure of solutions, being able to find solutions using the various methods available: "explicit calculation", row reduction on augmented matrix, using inverse matrix, Cramer's method (Be sure you can apply all these methods but also to identify the method which works best and fastest for you on relative small systems of linear equations.), solving systems of linear equations with parameters
- determinants: knowing basic properties (multi-linearity, anti-symmetry), product law for determinants, determinant of transpose matrix, working out determinants for small matrices and expansion by row or column, determinant as a criterion for invertibility of a matrix, application to inversion of matrices (co-factor method) and solving systems of linear equations (Cramer's rule).
- scalar product: idea of an ortho-normal basis, carry out Gram-Schmidt procedure, definition and interpretation of orthogonal and unitary matrices, working with 2d rotations

- eigenvectors and eigenvalues: computing eigenvalues and eigenvectors for specific matrices and using these to diagonalise the matrices, application to computing functions of matrices and quadratic forms

2 A Mini-Project

This project is exploring some of the algorithmic and computational aspects of the subject. As a physicist, being computer-literate is part of the standard-repertoire so if you have had little experience so far this is an opportunity to make a start.

The basic task is: Write a code in an advanced programming language – preferably c or c++ – which calculates the rank of a matrix of arbitrary size (and not necessarily quadratic) using Gaussian elimination. To avoid having to deal with tricky numerical issues work over the field $\mathbb{F}_p = \{0, 1, \ldots, p-1\}$, where p is a prime number. In this field, addition and multiplication of two integers k, l are defined by

$$k + l := (k + l) \bmod p \,, \quad kl := (kl) \bmod p \,. \tag{1}$$

Here, $k \bmod p$ denotes the remainder of a division of k by p. So, in practice, your matrices will contain integers of limited size (from 0 to $p-1$) and whenever an addition or a multiplication leads to a result $\geq p$ or < 0 it is brought back into the range 0 to $p-1$ by the mod operation in Eq. (1). Keep the prime p as an arbitrary but fixed constant in your program.

Some steps along the way are:

- If you don't have a c compiler installed on your computer yet this is a good time to do so. Free, open-source compilers, such as gnu's c compiler, are available. If you are unfamiliar with the language, a quick, no nonsense introduction is, for example, B. W. Kernighan and D. M. Ritchie, "The C Programming Language".

- Understand the arithmetics of the finite field $\mathbb{F}_p$ with addition and multiplication as in Eq. (1) and implement this computationally.

- Go over Gaussian elimination and translate it into a detailed algorithm which can be programmed. Your code should work for matrices of arbitrary size. Devise some checks to make sure your code works correctly.

- Computing time should increase as n^α with the typical size n of the matrix and some characteristic power α. Find the power α for your code, check that it conforms with general expectations and think about any optimisations of your code which might decrease α.

3 A Challenging Mini-Project

This project provides an opportunity to practice many of the techniques of linear algebra which we have introduced in the context of a more advanced setting. It is, of course, entirely voluntary and for your amusement only. The underlying problem has arisen in the context of a "real-life" research project but it can be formulated in elementary terms. The problem is non-trivial and currently unsolved so don't despair if you don't get anywhere.

Consider m vectors $\mathbf{x}_r = (x_{r,0}, \ldots, x_{r,n_r})$ of variables $x_{r,i}$, each with dimension $n_r + 1$, where $r = 1, \ldots, m$, and the (ring of) polynomials $R = \mathbb{C}[\mathbf{x}_1, \ldots, \mathbf{x}_m]$ in all so-defined variables. Then focus on the two vector spaces $V = R_{\mathbf{k}}$ and $W = R_{\mathbf{l}}$ of polynomials with multi-degrees $\mathbf{k}$ and $\mathbf{l}$ in those variables. A polynomial has multi-degree $\mathbf{k} = (k_1, \ldots, k_m)$ if its total degree in the variables $x_{r,0}, \ldots, x_{r,n_r}$ is k_r. For example, let $m = 2$, $n_1 = n_2 = 1$ and denote the two vectors by $\mathbf{x}_1 = (x_0, x_1)$ and $\mathbf{x}_2 = (y_0, y_1)$, so that $R = \mathbb{C}[x_0, x_1, y_0, y_1]$. Then, the multi-degree $(2, 1)$-polynomials are given by

$$V = R_{(2,1)} = \text{Span}\{x_0^2 y_0, x_0 x_1 y_0, x_1^2 y_0, x_0^2 y_1, x_0 x_1 y_1, x_1^2 y_1\} \ , \tag{2}$$

so they define a six-dimensional vector space.

Now introduce a family $f(\mathbf{c})$ of polynomials with multi-degree $\mathbf{l} - \mathbf{k}$ by writing

$$f(\mathbf{c}) = \sum_i c_i m_i \ , \tag{3}$$

where m_i are fixed polynomials (for example monomials) of multi-degree $\mathbf{l} - \mathbf{k}$ and $c_i \in \mathbb{C}$. There is a subtlety if a component, $l_r - k_r$, of the multi-degree $\mathbf{l} - \mathbf{k}$ happens to be negative. In this case, the coordinates $x_{r,i}$ in f should be replaced by the partial derivatives $\partial_{x_{r,i}} = \partial/\partial x_{r,i}$. With this refinement the polynomials $f(\mathbf{c})$ define linear maps

$$f(\mathbf{c}) : V \to W \tag{4}$$

between our two polynomial vector spaces V and W, simply by virtue of mapping $v \in V$ into $f(\mathbf{c})v \in W$. For example, if

$$W = \mathbb{C}[x_0, x_1, y_0, y_1]_{(1,2)} = \text{Span}\{x_0 y_0^2, x_0 y_0 y_1, x_0 y_1^2, x_1 y_0^2, x_1 y_0 y_1, x_1 y_1^2\} \ , \tag{5}$$

then the map f should have degree $(-1, 1)$ and the most general such map can be written as

$$f(\mathbf{c}) = c_1 y_0 \partial_{x_0} + c_2 y_0 \partial_{x_1} + c_3 y_1 \partial_{x_0} + c_4 y_1 \partial_{x_1} \ . \tag{6}$$

Pick a basis (for example of monomials) for each of V and W. Relative to this basis the maps $f(\mathbf{c})$ can be represented by matrices $M(\mathbf{c}) : \mathbb{C}^{\dim(V)} \to \mathbb{C}^{\dim(W)}$ whose entries are linear in the coefficients c_i. As an illustration, for the example defined by Eqs. (2), (5), (6), and with the monomial basis given there, the matrices $M(\mathbf{c})$ take the form

$$M(\mathbf{c}) = \begin{pmatrix} 2c_1 & c_2 & 0 & 0 & 0 & 0 \\ 2c_3 & c_4 & 0 & 2c_1 & c_2 & 0 \\ 0 & 0 & 0 & 2c_3 & c_4 & 0 \\ 0 & c_1 & 2c_2 & 0 & 0 & 0 \\ 0 & c_3 & 2c_4 & 0 & c_1 & 2c_2 \\ 0 & 0 & 0 & 0 & c_3 & 2c_4 \end{pmatrix} \ . \tag{7}$$

This is of course just a small example. For higher polynomial degrees and more variables the matrices become very large (think hundreds times hundreds). Also note, although the above example has led to a quadratic

matrix, in general $M(\mathbf{c})$ will not be quadratic.

The problem that needs to be solved concerns the rank of the matrix $M(\mathbf{c})$. Of course this rank depends on the values of the coefficients c_i. The main question is:

What are the special loci in $\mathbf{c}$ *space where* $\mathrm{rk}(M(\mathbf{c})) \leq p$ *for any given integer* $p = 0, 1, 2, \ldots$ *?*

Let me illustrate this question for the example (7). The "generic" rank of $M(\mathbf{c})$ in Eq. (7) is six. Its rank is less or equal than 5 on the locus in $\mathbf{c}$ space defined by

$$\det(M(\mathbf{c})) = -16c_2^3c_3^3 + 48c_1c_2^2c_4c_3^2 - 48c_1^2c_2c_4^2c_3 + 16c_1^3c_4^3 \stackrel{!}{=} 0 \; . \tag{8}$$

This locus is three-dimensional (in four-dimensional $\mathbf{c}$ space) and, it turns out, it can equivalently and more simply be defined as the locus where $c_2c_3 - c_1c_4 = 0$. Actually on this locus the rank equals 4. Further, a rank lower than 4 can only be achieved in a trivial way by setting all $c_i = 0$, in which case the matrix $M(\mathbf{c})$ is entirely zero and has vanishing rank.

Everything is reasonably straightforward for the small example above but what if the size of the matrix $M(\mathbf{c})$ is 20×20 or even larger, depending on $\mathcal{O}(10)$ coefficients c_i? In this case the determinant (or anything similar) is completely useless as it would lead to 20! or more terms. Can anything be said in general about the special loci in $\mathbf{c}$ space for such large cases? Is there an effective algorithm which allows their computation?