

PROJECTS for MCC3, MATLAB HGFei, Edinburgh, **Feb. 21st**

Version of Thursday, Feb. 21st, 2008, 9.a.m., by HGFei (only minor additions will come from now on!)

GENERAL HINT: Linear algebra is a *powerful*¹ yet quite *intuitive* tool of mathematics² The course and the assignments are supposed to *help* you (aside from providing you with grades for a certain course) to learn that with a piece of mathematical software, such as MATLAB (but of course also with others) one can check theoretical results and general methods (like the Gauss elimination, or the Gram-Schmidt process) numerically, combine *geometric intuition* (about how to determine the nearest point on a plane to a given point in R^3 by looking at the footing point under normal vector, describing the plane!) with concrete figures. It should be to you like using the pocket calculator in order to solve some trigonometric problem (using *cos* and *sin* etc.).

EVERYBODY PLEASE READ THE ABOVE PARAGRAPH CAREFULLY!

TOPICS for the PRESENTATIONS

1. **Polynomials from the point of view of linear algebra:** Polynomials of a fixed degree are nice finite dimensional vector spaces. The usual (standard) basis is the set of all monomials, i.e. the functions (x^k) , with $k = 0, 1, \dots, r$, the maximal degree (for MATLAB they should be taken in inverse order). There are a lot of linear mappings on the polynomials, e.g. the shifting-operators, or the dilation operators. Differentiation is a non-invertible operation on any of these spaces. Hence also $(x - x_0)^k$ is a basis for the polynomials of degree r .

I suggest to do most of the demonstrations for cubic polynomials ($r = 3$), but higher degrees might be suitable in some cases!

THIS IS A LIST OF SUGGESTIONS from which to CHOOSE a nice subset. So not everybody having this assignment should do exactly the same! (in particular, please choose your own numbers, or do random polynomials, for example, if this is possible, although then you have to be more careful with the window in which to plot the corresponding curves).

- Could one take also take $(x - x_k)^k$, for $0 \leq k \leq r$?
- for any four points determine a (linear independent) set of Lagrange interpolators, which form another set of linear independent cubic polynomials, hence another basis for the cubic polynomials;
- the *Hermite interpolation problem*, given sufficiently many function values and derivatives find the exact solution among the polynomials which satisfy the corresponding conditions (for example, giving the first three derivatives at four different points should determine a polynomial of degree 11, as it has 12 coefficients); but how can one obtain the polynomial from the data?

¹Due to its generality, essentially valid for arbitrary finite-dimensional vector spaces.

²It is of course also the basis for functional analysis, which is in a way the theory of not necessary finite-dimensional vector spaces, together with convergence considerations, which are necessary to handle infinite expansions as opposed to finite linear combinations.

- can one find a quadratic polynomial by determining the value of the derivative at two values? if the answer is no, under which conditions can it be done (and when not)?
- Plot a typical cubic polynomial (whatever you like, such as $p(x) = x^3 - 2x^2 - x + 5$ and determine (and mark) its local minima, maxima, turning point etc., and plot the graph and do a full curve discussion, for example, including perhaps a typical tangent to the curve in some given point (e.g. the turning point);
- Obviously shifting all the polynomials by a certain amount, e.g. replacing $p(x)$ by $T(p(x)) = q(x) = p(x - 2)$ is a *linear mapping* on the coefficients (from the coefficients of $p(x)$ to those of $q(x)$, which has a matrix, and its inverse (left shift by 2) must correspond to the inverse matrix. What are the matrices? Plot the polynomial before and after shift (using its coefficients) to see that you have found the correct matrix of transition.
- Same for the dilation matrix. Stretching the graph of a given polynomial (leaving it a polynomial of the same degree). Here we consider mapping such as $D(p(x)) = p(x/3)$ (D for dilation). Again, check by plotting the result.
- combining the last two operations and a standard example show that you can move the local maxima and minima to arbitrary positions, by first choosing concrete positions (like -1 and $+1$) and then applying affine transformations (combination of shift and scaling, as above, of the argument of the polynomial).
- If you are interested in Hermite interpolation (given values and the derivatives or tangents at those to points) show that you can either do it at the standard points, say $0, 1$ and then transform it to general position a, b in R (using the above transformations) or do it directly with the positions (say $a = 3, b = 7$, or $b = 5$, whatever you like or choose yourself for the sake of demonstration).
- Is a quadratic polynomial determined by any triple of values $p(x_1), p'(x_2)$ and $p''(x_3)$, whatever the values of (x_1, x_2, x_3) are. What about replacing $p''(x_3)$ by $p(x_3)$ (for $x_1 \neq x_3$)?

2. Spline functions (basic theory)

They are typically defined to be piecewise polynomial functions with “knots” at the integers. Since polynomials of higher order are not such a good idea for the purpose of (uniformly) approximating functions, even if they are smooth (cf. Runge’s example) the so-called spline-functions have been invented. The most important case is the case that of cubic splines, which are in fact functions which have a continuous first and second derivative (but possible jumps in their third derivative at the node points, let us consider only integer nodes). Hence over the interval $[0, 3]$ the space of splines consists a priori of the space of cubic polynomials over each of the subintervals $[0, 1]$, $[1, 2]$ and $[2, 3]$, but with the ‘gluing conditions’ that the left hand limit (of the function and its first and second derivative) is matching the right hand limit of the corresponding derivative on the right hand side. This defines a 12-dimensional space (since we have 4 coefficients over each of the 3 intervals), but within that we have $2 \times 3 = 6$ coupling conditions, so overall one may expect $12 - 6 = 6$ dimensions of the spline functions over $[0, 3]$. So we ask questions like this one:

- Given this space, how can we find a basis for this space, or in other words, what is the dimension of the space? Can we find for a typical choice of data, e.g. by requiring 6

specified values, 2 per interval at arbitrary choosen points, to be exactly interpolated by some element of the spline type space?

- If we choose in each interval the basis such that the ‘Hermite interpolation conditions are satisfied’ one can in fact show that the coupling conditions mean that one can remove some of the variables (it is like taking the plane in R^3 described by the condition $y = z$, which obviously has a basis of the form $[1, 0, 0]$ and $[0, 1, 1]$).
- An important property of the spline functions over the integers is that one can find a so-called basis of B-splines, which is one function (a standard cubic polynomial) which is a cubic spline, supported on $[-2, 2]$ and symmetric around 0.

3. The **FFT, the Fast Fourier Transform**, i.e. the realization of the DFT via some fast algorithm, is one of the most interesting and most important algorithms for signal processing applications.

It can be (first of all) interpreted as an orthonormal change of bases, to a basis which consists of functions which are eigenvectors of the cyclic shift matrix.

At the same time it describes the (linear) mapping from the coefficients in C^n of a complex polynomial to its values over the unit-roots of order n . From this property we can gain a number of consequences:

- (a) The so-called convolution theorem: doing either the Cauchy-product of two polynomials or the pointwise product of their Fourier transform. Of course one has to consider the degree of the product polynomial, because in order to properly gain all the coefficients correct of the product polynomial one needs at least $r + 1$ values in the complex plane, so at least some FFT of length $r + 1$. One does so by adding zeros to the product polynomial.
- (b) The same applies to powers of polynomials, which can be realized not by an inductive FOR-loop using CONV but instead much faster by taking a pointwise power on the Fourier transform side. For example, in order to determine the coefficients of $(1 + x)^{53}$ one observes that one needs at least 54 coefficients, hence one should extend the sequence $[1, 1]$ describing $1 \cdots x + 1$ by 62 more zeros, in order to obtain a sequence of length $64 = 2^6$, and work with that sequence;
- (c) The interpretation of polynomials allows to also demonstrate the interpolation property that one can get by zero-padding on the IFT-side. Given for example 8 data points, interpreted as values of some polynomial of order 7 over the unit-roots of order 8 we can interpolate and obtain the same polynomial over the unit roots of order 32
- (d) It is not too difficult to go from 1D-FT to the 2D-version, simply by applying the *FFT* first row-wise, and then columnwise (or vice versa). That this is the same comes out if one realizes that $fft2(A)$ is the same as $fft(fft(A)')'$, or with $F = fft(eye(n))$ and expressed via matrix multiplication $F * A * F$ (which can be read as $(F * A) * F$ or alternatively as $F * (A * F)$).

4. Linear Algebra from a geometric/numerical point of view

Given a matrix, how can one determine the relevant sub-objects:

- a basis for the row or the column space (using Gauss elimination)
- obtain orthonormal bases by applying
- built-in MATLAB commands (such as ORTH)
- do it using the PINV-command as described in the course
- show that the result are equal, either for a particular random matrix (which should be allowed to be *not of full rank*, i.e. with $\text{rank}(A) < \min(m, n)$. (recall that $\text{rand}(8, 3) * \text{rand}(3, 5)$ is likely to give some random 8×5 matrix of rank 3!

5. **Orthogonalization methods:** Given a basis of a vector space, how can one obtain an orthogonal basis for the same space in a way which is maximally fair. Standard *Gram-Schmidt* is *not* fair, in the sense that it gives the first vector the chance to stay unchanged, while the last one has to adapt maximally to all the other ones. Hence the result of Gram-Schmidt (and also the stability of the process) depends heavily on the order in which the basis vectors are chosen. The symmetric or Loewdin orthogonalization is the most important one (NuHAG: Orth-Best command). It can be best explained using the *SVD*, the *singular value decomposition* of a general matrix. One can say, that one “takes out the rescaling part” and leaves just a change of basis of vectors, from the vectors originally taken making up the column space, to the orthonormal system U which occurs in the factorization $A = U * S * V'$ (obtained within MATLAB by calling `[U,S,V] = svd(A);`). Then the best approximation to A (in the sense of the so-called Frobenius norm, `norm(A, 'fro')`), i.e. considering A as an element of C^{n^2} (with the Euclidean norm) we get the best approximation by a unitary matrix by choosind $U * V'$.

There is an interesting connection between the so-called biorthogonal system and the (rows) of the inverse of a matrix. Write $B = \text{inv}(A)$; then $B * A = Id$ (the unit-matrix), resp. `norm(B*A - eye(n))`; Hence (at least for real matrices) the row-vector number 3 has scalar product 1 with the third column of A and zero scalar product with respect to all the other column vectors. In this sense the rows of B are biorthogonal to the column vectors of A .

Such a biorthogonality condition also occurs if we look at the LAST vector in the Gram Schmidt system. So let us take the vectors of A in some random order, just making sure that the third (to give an example) vector is put in last position. Apply then the Gram-Schmidt procedure. Then obviously the last vector will be just the third column of A stripped of al its components in the direction of all the other ones. This means, that we can obtain the rows of the inverse matrix (in fact their normalized versions only) by applying Gram-Schmidt repeatedly to the matrix, for example by the command, applied to some random 6×6 -matrix A :

```
for jj = 0:5;AA = A(:, 1 + mod( -jj+(0:5),6)), pause, end.
```

6. Image Compression Algorithms (lossy versus lossless compression)

The goal of this project is to give a short summary over various compression schemes, applicable to digital images. Many of these algorithms are based on the following idea: Take a suitable transform (such as an orthonormal change of basis) and then determine in some way the “most important coefficients” that have to be stored.

It would be relatively stupid and simple-minded to reduce the storage space for a large image by a factor of 9 by just taking every third pixel in each direction. The result would depend very much on the offset, and not be very representative for the full image.

I hope that there will be a way to download/obtain the basic routines for image processing in MATLAB without having the license for the image analysis (and signal analysis) toolbox. See `dct2`, `fft2`, `idct2`, `ifft2`.

7. The Spectrogram or Short-time Fourier Transform

It is available as a built-in function of MATLAB (please test) `spectrogram`, `periodogram`, `pwelch`, `spectrum`, `goertzel` but these routines might be part of the signal processing toolbox, so I recommend to use the NuHAG routines instead, to be found/downloaded from www.nuhag.eu (go to DB+tools > MATLAB).

There will be much more material on this for those who get this part assigned.

8. Diagonalization of matrices and eigenvectors of matrices

Recall the definition of eigenvalues and eigenvectors (from your favorite book in linear algebra), and try to see how the program described in that book works out in order to obtain first the eigenvalues and then the eigenvectors.

We start by building the characteristic polynomial, which is described by the MATLAB coefficients `poly(A)`. Recall that this characteristic polynomial has roots (you get all of them by the command `roots(poly(A))`) and check that these roots are indeed eigenvalues, because for any λ which is among the roots of $p_A(z)$

There is also a nice numerical methods whose action can be demonstrated easily (without going into the proof, by the following simple experiment:

```
for kk = 1 : 100; [Q,R] = grmsfei(A); A= R*Q; imagesc(abs(A)); shg; end;
```

9. Provide background on the **determinant rule**:

$$\det(A * B) = \det(A) \cdot \det(B)$$

for 3×3 matrices, making use of the cross product and the interpretation by volumes. Recall the MATLAB command first:

MATLAB: see

help cross

CROSS Vector cross product.

C = CROSS(A,B) returns the cross product of the vectors A and B. That is, C = A x B. A and B must be 3 element vectors.

C = CROSS(A,B) returns the cross product of A and B along the first dimension of length 3.

The point of your work should be to give a geometrical interpretation for the product rule for determinants, i.e. why is it, that $\det(AB) = \det(A) \cdot \det(B)$ for any 3×3 -matrices A, B . Hence $\det(\text{inv}(A)) == 1/\det(A)$. WHY is it like that, given the somewhat complicated or strange definition of the determinant (following Sarrus' rule, if you want)?

The cross product helps to determine the volume of a parallel-epiped, i.e. the body generated in 3D by three given vectors. Hence it is zero if and only if the vectors are within one plan (or even multiple of each other), and nonzero if and only if they are a basis for $\mathbb{R}^3$.

Verify that the Volume of any such parallel-epiped (not just the unit cube) after applying the linear mapping $x \mapsto A * x$ changes exactly by the scalar factor $\det(A)$, i.e. triples (and changes orientation) in case of $\det(A) = -3$, to give an example (don't care about orientation matters in your demonstration).

Clearly composition of two linear mappings, say one which amplifies volumes by the factor of 3, the other shrinking by a factor of 5 gives the effect of multiplying the volume of arbitrary parallel-epipeds by the factor $3/5$ under the composite mapping.

10. **SVD, the singular value decomposition** Special assignment for some colleagues already discussed at the PC-Lab.

Goal: explain the features of SVD and how it is used to get the PINV, the pseudo-inverse, or in other words, how to solve the MNLSQ-problem (minimal norm least squares problem).

There is a link to material from Gilbert Strang (and his book on Linear Algebra):

11. Data fitting

We have seen during the course that one can use the PINV (or if you want the so-called normal equation, obtained by replacing the original linear equation $A * x = b$ by $A' * A * x = A' * b$ to obtain the so-called MNLSQ solution $x = inv(A'A) * (A' * b)$ for the solution of the problem, with the slight (but !unavoidable) disadvantage that $A * x$ is just the projection of b onto the column space of A (and not b), but this is fine, because if the system is *consistent*, i.e. if (and only if) b is in the column space of A this is a correct solution (even the one with minimal length), while otherwise we have changed to original problem in a minimal way (by doing an orthonormal projection onto the column space, which is the only way to get the problem consistent).

Now when we apply this to “real life situations” we can say that this idea is at the basis of all scattered data approximation tasks. One has in a way “to many” but probably “noisy” (and therefore typically inconsistent) data. You find examples in the course material, where the problem of having say 7 inexact vales of a cubic polynomial is discussed. Either work out a similar example, maybe with polynomials of higher degree, also check the sensitivity of the process on data errors, i.e. check out, what the influence is in the reconstruction process of data errors. For example, you may want to fix all but one data points (out of many) and vary its value over a certain reasonable range, and then plot the corresponding resulting polynomials, in dependence of that particular value. Mark the specific value (e.g. with a red *) to clearly indicate what happens, and how much (or how little) the overall curve is following such a point when it moves around. Think of alternative settings that you would like to understand or demonstrate yourself.

Ideally I would like to see alternative settings, such as the best approximation by trigonometric polynomials (contact me if you have interest in this and cannot get started). I am suggesting to think of the *trigonometric polynomials* of degree at most 10. For e.g. $n = 256$ you may interpret this as the system of vectors which is composed from the real-values of $F(:, 1 : 11)$, where $F = fft(eye(n))$; (plot those functions). Obviously one can give a minimum of at least 11 (but say typically a couple more points, or some true sequence of values at some choice of coordinates, obtained by adding (random) “noise” to the coordinates of a linear combinations of those 11 functions), and try to identify (and then plot) the particular linear combination which does the best fit over the given set of coordinates. For simplicity you may take the coordinate sequence $1 : 10 : n$ (so that you will have 25 points then).