CMAT USER GUIDE

K. R. MATTHEWS

5th July 2010

1 A brief description of `cmat`

`cmat` is an exact arithmetic calculator program, written in C and designed to run on computers which support UNIX, or 386/486 PC's. `cmat` performs many of the standard arithmetical operations that can be carried out exactly, without approximations, on matrices and polynomials whose coefficients are either rational numbers, complex rational numbers, or elements of $\mathbb{Z}_p$, the finite field of p elements, where p is a prime less than $2^{16} = 65536$.

There are three calculator programs within `cmat` and these can be accessed directly:

`cmatr`, `cmatcr` and `cmatm` (UNIX).

`cmatr.exe`, `cmatcr.exe` and `cmatm.exe` (MSDOS).

`cmatr` works over the field $\mathbb{Q}$ of rational numbers, `cmatcr` over the field $\mathbb{Q}(i)$ of complex rational numbers – numbers of the form $a + ib$, where a and b are rationals, while `cmatm` works over $\mathbb{Z}_p$.

The programs use multiple precision arithmetic routines based on those in [4, pages 342–357, 175–185].

The program allows for the creation and storage of up to $M0$ objects of the following type ($M0$ is usually set to 30).

1. cR: rational numbers: $R[0], \ldots, R[M0-1]$;
2. cRM: matrices with rational coefficients: $RM[0], \ldots, RM[M0-1]$;
3. cPR: polynomials with rational coefficients: $PR[0], \ldots, PR[M0-1]$;
4. cPRM: matrices with polynomial rational coefficients: $PRM[0], \ldots, PRM[M0-1]$;
5. cCR: complex rational numbers: $CR[0], \ldots, CR[M0-1]$;
6. cCRM: matrices with complex rational coefficients: $CRM[0], \ldots, CRM[M0-1]$;
7. cPCR: polynomials with complex rational coefficients: $PCR[0], \ldots, PCR[M0-1]$;

8. cPCRM: matrices with polynomial complex rational coefficients: $PCRM[0], \ldots, PCRM[M0-1]$;
9. cmM: matrices with mod p coefficients: $mM[0], \ldots, mM[M0-1]$;
10. cPm: polynomials with mod p coefficients: $Pm[0], \ldots, Pm[M0-1]$;
11. cPmM: matrices with polynomial mod p coefficients: $PmM[0], \ldots, PmM[M0-1]$;

These arrays always contain the appropriate zero object until replaced by either an object entered by the user, or one resulting from a calculation.

Storage limitations and execution times of algorithms are much less when calculating with matrices over $\mathbb{Z}_p$.

2 Routines available

- The familiar arithmetical operations of addition, subtraction, multiplication, inverse, ratio and exponentiation, are available for rationals, complex rationals and elements of $\mathbb{Z}_p$. Complex conjugation of complex numbers is also available. The m–th root of a rational number can be computed to a given number of decimal places.

- For polynomials, there are the usual operations of addition, subtraction, multiplication, scalar multiplication, exponentiation, derivative, evaluation, quotient, remainder, greatest common divisor, expressing the greatest common divisor as a linear combination of the polynomials involved.

  The resultant of two polynomials over $\mathbb{Z}[x]$ and the discriminant of a polynomial over $\mathbb{Z}[x]$ can be calculated.

- There is modular exponentiation for polynomials in $\mathbb{Z}_p[x] \,(\mathrm{mod}\, P[j])$. The squarefree decomposition of a polynomial can be found. Polynomials over $\mathbb{Q}, \mathbb{Q}(i)$ or $\mathbb{Z}_p$ can be factorized into irreducibles. The Berlekamp matrix of a squarefree polynomial $f \in \mathbb{Z}_p[x]$ is available. This is the transpose of the usual Berlekamp matrix that is described in [6, pages 420–429] and is the $n \times n$ matrix $Q = [q_{ij}]$ over $\mathbb{Z}_p$ defined by the congruence

$$x^{pj} \equiv \sum_{i=0}^{n-1} q_{ij} x^i \,(\mathrm{mod}\, f),\ 0 \le j \le n-1,$$

where $n = \deg f$.

An irreducible polynomial of given degree over $\mathbb{Z}_p[x]$ can be constructed.

The cyclotomic polynomial $\Phi_n(x)$ can be calculated over $\mathbb{Z}[x]$ and $\mathbb{Z}_p[x]$.

- For matrices, there are the usual operations of addition, subtraction, multiplication, scalar multiplication, exponentiation, Kronecker product, direct sum of two matrices, evaluation of a matrix polynomial.
- The standard elementary row and column operations can be performed on all types of matrices.
- The entries of a stored matrix can be changed, either individually, or by row or column. A single row or column can be deleted.
- A single row or column vector can be selected from a stored matrix. More generally, a submatrix of a stored matrix can be formed by selecting a subfamily of rows or columns, respectively.
- Two matrices can be joined by rows or columns. The partitioned matrix $[A|I]$ can be formed from A. A matrix can be unwrapped into a column vector (a process useful for finding the minimum polynomial of a matrix).
- The matrix $A - \lambda I$ can be formed from A and λ; this is important when finding a basis for the eigenspace of A corresponding to the eigenvalue λ.
- Special matrices such as band matrices, the identity matrix, the elementary Jordan matrix and the companion matrix of a polynomial, can be created.
- Files of matrices (rational, floating point, complex rational or mod p), prepared by the user, can be read into the `cmat` program. The floating point numbers can only be used in the rational program and are converted internally to rational numbers in lowest terms.
- Matrices whose (i, j)th element has real and imaginary parts given by simple arithmetical expressions such as `i^2+3*j` for the numerators and denominators, may be created wholly within `cmat`.
- The inverse, adjoint, determinant, characteristic polynomial and minimum polynomial of a matrix of scalars can be found. The adjoint and determinant can

also be found for matrices with polynomial elements. Also $P^{-1}AP$ can be calculated.

and minimum polynomial of a matrix of scalars, $P^{-1}AP$. The Cholesky decomposition of a positive definite matrix is found: $A = L^t DL$, where L is upper unit triangular and D is a diagonal matrix. The output has the diagonal elements of D on its diagonal and its above diagonal elements are those of L.

The inverse of a matrix M of integers mod m can be found, when m is a positive integer relatively prime to $\det M$.

- The reduced row–echelon form of a matrix can be found. Also systems of linear equations can be solved.

- Bases can be found for the null–space $N(A)$ and column–space $C(A)$ of a matrix A.

  Both bases are associated with the reduced row–echelon form B of the given matrix A: if the non–zero rows of B have leading entries 1 in columns $c_1, \ldots, c_r$, then X in $N(A)$ is found by expressing the unknowns $x_{c_1}, \ldots, x_{c_r}$ in terms of the remaining unknowns $x_{c_{r+1}}, \ldots, x_{c_n}$, which are taken to be arbitrary. This results in expressing X as a linear combination

  $$X = x_{c_{r+1}} X_1 + \cdots x_{c_n} X_{n-r}$$

  and $X_1, \ldots, X_{n-r}$ form the basis for $N(A)$ that is selected by `cmat`.

  The column vectors of A in columns $c_1, \ldots, c_r$ form a basis for $C(A)$ that is selected by `cmat`. This basis is known as the *left–to–right basis* for the column–space.

  The left–to–right basis is especially useful for extending a given linearly independent family of column vectors $X_1, \ldots, X_s$ in a vector space V, to a basis for V, where a spanning family $Y_1, \ldots, Y_t$ is given. For

  $$V = \langle X_1, \ldots, X_s, Y_1, \ldots, Y_t \rangle$$

  and applying the left–to–right algorithm to the spanning family will select $X_1, \ldots, X_s$, together with a subfamily of $Y_1, \ldots, Y_t$ as a basis for V. This procedure is very useful in one of the algorithms for finding the Jordan canonical form or more generally, the rational canonical form of a square matrix. (See[10].)

- The dot product and length of vectors with rational or complex rational entries can be calculated. The Gram–Schmidt process is also available. The cross–product of two vectors with rational components can be found. There is also a generalisation to the cross–product of the rows of an $(n-1) \times n$ matrix of rationals.

- The Smith canonical form of a matrix B with polynomial elements can be found. When applied to the matrix $B = xI - A$, where A is a matrix of scalars, this gives the similarity invariants of A; in particular, we obtain the minimum polynomial of A as the invariant of highest degree. Polynomial matrices P and Q, whose determinants are constant polynomials, are also found such that PBQ is in Smith canonical form.

- All data currently stored in the above arrays can be printed either on the screen or on a line printer. If desired, matrices of rational and complex rational coefficients can be printed to a file or the screen, to a nominated number of decimal places.

3 Calculating with `cmat`

- The user can successively descend deeper into the program by selecting various displayed screen options. To ascend through the various levels, type `q` when this is an option. When leaving the program, the user is given the option of saving data for re–use in future sessions.

- The menus are somewhat tersely organized and are listed at the end of this guide. The menus are interpreted as follows:

  To enter a rational number and store the number as $RM[0]$ say, one selects the appropriate menu (number R14 at the end of this guide) and types `n 0`.

  NOTE: Use CONTROL-H, not the backspace key, to backspace over characters.

- Integers are entered as usual, while non–integer rationals are entered with a slash as integer/positive integer: e.g. `-71/2`, while complex numbers with non–zero imaginary part must end with an i: e.g. `1/2i` represents $(1/2)i$. No brackets are to be used when entering numbers. Rational numbers are outputted in 'lowest terms', with positive denominator.

- Some latitude is allowed as regards spacing. For example, `1-i, 1- i, 1 - i` represent the same complex rational. However `1 -i` represents the two numbers `1` and `-i`.

- Matrices are entered row by row, the end of a row being marked by a carriage return. Spaces separate entries of a row.

- To discontinue entering a polynomial or matrix from the keyboard, type `q` or any non–alphanumeric character when entering a coefficient. A default polynomial or matrix is then stored.

- To perform calculations with stored data, one has to select the appropriate menu. For example, having entered matrices $RM[0]$ and $RM[1]$, the product $RM[0] * RM[1]$ is sent to $RM[2]$ (or $RM[0]$ or $RM[1]$) by selecting the multiplication option `t` from menu R7, reproduced below, typing `t 0 1 2` :

```
            Rational Matrix Arithmetic                    R7
a  j  k  l  :          RM[j] + RM[k]          ->  RM[l]
t  j  k  l  :          RM[j] * RM[k]          ->  RM[l]
s  j  k  l  :          RM[j] - RM[k]          ->  RM[l]
m  j  k     :              -RM[j]             ->  RM[k]
f  j  k  l  :           R[j](RM[k])           ->  RM[l]
*  j  k     :             (RM[j])*            ->  RM[k]
e  n  j  k  :             RM[j]^n             ->  RM[k]
z  j  k  l  :          RM[j] - R[k]I          ->  RM[l]
p           :  print numbers and matrices
x           :          to enter data
q           :             to exit
```

- To calculate the 10th power of $RM[0]$, select the exponentiation option `e` from the above menu, typing `e 10 0 3` to send the output to $RM[3]$.

- The situation is somewhat simpler with modular arithmetic calculations. Here no numbers are stored and non–negative numbers less than the relevant modulus are entered directly, instead of being stored. For example to multiply the matrix $mM[0]$ by $5 \pmod 7$ and to store the result as $mM[1]$, select menu $m7$ below and type

`f 5 0 1 7.`

- It is important to remember that when using the modular arithmetic section of `cmat`, operations should be carried out only on stored objects having the same modulus.

- To terminate input prematurely after entering the first letter of a menu option, type `q` followed by **RETURN**.

- To input a file of r ($<= M0 - j$) prepared matrices of the same type (rational, floating point, complex rational or modular) into array positions $j, \ldots, j+r-1$, the first line of the file should contain the number r; the next line consists of the row and column number of the first matrix, followed by the respective rows of the matrix.

  For example, the following file represents two rational matrices, the first being 3×3, the second being 2×2:

```
/* rational matrix file */
2
3 3
2/3 5 -7/8
1/2 12 -5
7 6 4
2 2
1 0
0 1
```

4 Comments on algorithms used in `cmat`

- The tricks of P. Henrici mentioned in [1, pages 200–201] are used to speed up addition and multiplication of rationals.

- The Gauss–Jordan method is used to find the reduced row–echelon form and to solve system of linear equations over all three fields.

- The adjoint, inverse, determinant and characteristic polynomial of a matrix of rationals or complex rationals are found by the Faddeeva–Leverrier method. See the book [3, pages 177–181].

  The method is also valid for matrices over $\mathbb{Z}_p$ or $\mathbb{Z}_p[x]$ if $p > n$, the size of the matrix.

In fact, for matrices over $\mathbb{Z}_p$, we use instead a modification of an algorithm due to Frobenius (see [11, pages 168–175]) to find the characteristic polynomial. Also over $\mathbb{Z}_p$, the Gauss–Jordan algorithm is used to find the inverse and determinant of a matrix. The adjoint is found using the fact that $\operatorname{adj} A = 0$ if $\operatorname{rank} A \leq n-2$, $\operatorname{rank}(\operatorname{adj} A) = 1$ if $\operatorname{rank} A = n-1$ and $\operatorname{adj} A = (\det A) \cdot A^{-1}$ otherwise.

- The inverse of an integer matrix mod m is found by using the adjoint matrix and multiplying mod m by the inverse of determinant mod m.

- The minimum polynomial of an $n \times n$ matrix A is found by finding the least $m \geq 1$ such that A^m is expressible as a linear combination of $I_n, \ldots, A^{m-1}$. This is done by unwrapping the matrices $I_n, \ldots, A_r$ into column vectors for $1 \leq r \leq m$ and solving the equation

$$x_0 I + \cdots + x_{r-1} A^{r-1} = A^r$$

  as a system of n^2 equations in r unknowns. The system will be inconsistent for $1 \leq r < m$, but will have a unique solution when $r = m$, giving the minimum polynomial $m_A = x^m - x_{m-1}x^{m-1} - \cdots - x_0$.

- The Smith canonical form is found by the algorithm in [5, pages 111–113]. Faster algorithms exist (see [8]).

- Factorization of polynomials in $\mathbb{Z}_p[x]$ is accomplished using a method of finding a non–trivial factor due to P. Camion (see [2]). This is used in conjunction with the Berlekamp matrix of a polynomial. For information on this matrix see [6, pages 420–429]. (Our Berlekamp matrix is the transpose of Knuth's.)

- Factorization of a polynomial in $\mathbb{Z}[x]$ is accomplished using an algorithm outlined in [12]. The algorithm uses the degree–set testing procedure which sometimes uncovers irreducibility before more complicated tests are employed. The Hensel–Zassenhaus quadratic and linear liftings are used to obtain a factorization of $f(x)$ modulo a high power of a prime. The degree–set test also has the effect of reducing the number of test divisions needed to determine possible factors over $\mathbb{Z}[x]$ after the liftings are completed.

- Factorization of a polynomial in $\mathbb{Q}(i)[x]$ is accomplished using an algorithm outlined in [13].

- A squarefree decomposition algorithm for polynomials due to D.Y.Y. Yun and described in [6, pages 631–632] is used.

- Irreducible polynomials of given degree (mod p) are constructed using a probabilistic algorithm from [8, pages 145–149].
- The algorithm used for calculating the resultant of two polynomials with integer coefficients is outlined in the article by R. Loos in [1, page 130].
- The cyclotomic polynomial $\Phi_n(x)$ is calculated using an algorithm from [8, pages 354–356], which in turn is based on an account in [7, pages 58–63]. Factoring $\Phi_{p^n-1}(x)$ over $\mathbb{Z}_p[x]$ gives the $\phi(p^n-1)/n$ monic primitive polynomials of degree n over $\mathbb{Z}_p$.
- The m–th root of a rational number is calculated using the algorithm from [9].

Any suggestions for improvement, as well as uncovering of bugs, should be communicated to Keith Matthews, http://www.numbertheory.org/keith.html

References

[1] B. Buchberger, G.E. Collins and R. Loos (Editors). *Computer Algebra, Symbolic and Algebraic Computation.* 1972, Springer–Verlag Wien, New York.

[2] P. Camion. *A Deterministic Algorithm for Factorizing Polynomials of $F_q[x]$.* Annals of Discrete Mathematics 17 (1983) 149–157.

[3] V.N. Faddeeva. *Computational Methods in Linear Algebra.* 1959, Dover, New York.

[4] H. Flanders. *Scientific Pascal.* 1984, Reston Publishing Company, Reston, Virginia.

[5] B. Hartley and T.O. Hawkes. *Rings, Modules and Linear Algebra.* 1970, Chapman and Hall, London.

[6] D.E. Knuth. *The Art of Computer Programming, Volume 2.* Second Edition 1981, Addison–Wesley Publishing Company, Reading, Massachusetts.

[7] H. Lüneburg. *Galoisfelder, Kreisteilungskörper und Schieberegisterfolgen.* 1979, B.I. Wissenschaftsverlag, Mannheim/Wien/Zürich.

[8] H. Lüneburg. *On the Rational Normal Form of Endomorphisms.* 1987, B.I. Wissenschaftsverlag, Mannheim/Wien/Zürich. J.A.C.M. 22 (1975) 291–308.

[9] K.R. Matthews. *Computing mth roots.* College Mathematics Journal 19 (1988) 174–176.

[10] K.R. Matthews. *A Rational Canonical Form Algorithm.* Mathematica Bohemica 117 (1992) 315–324.

[11] W.A. McWorter, Jr. *An Algorithm for the Characteristic Polynomial.*

[12] D.R. Musser. *Multivariate Polynomial Factorization.* Mathematics Magazine 56 (1983) 168–175.

[13] B.M. Trager. *Algebraic Factoring and Rational Function Integration.* Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computation, 219–226.