Generic Linear Algebra and Quotient Rings in Maple

Simon Lo Michael Monagan Roman Pearce

Computational Algebra Group
Centre for Experimental and Constructive Mathematics
Department of Mathematics, Simon Fraser University

Introduction

Our goal is to do linear algebra computations over arbitrary rings and fields. We have written a Maple package, *GenericLinearAlgebra*, that implements the following commands:

```
> with(GenericLinearAlgebra);
```

$$[BareissAlgorithm, BerkowitzAlgorithm, CharacteristicPolynomial, Determinant, GaussianElimination, GenericCheck, HermiteForm, HessenbergAlgorithm, HessenbergForm, LinearSolve, MatrixInverse, MatrixMatrixMultiply, MatrixVectorMultiply, MinorExpansion, NullSpace, RREF, ReducedRowEchelonForm, SmithForm, StronglyConnectedBlocks]$$

The algorithms in this package are coded in such a way as to allow the user to define a field, Euclidean domain, integral domain, or commutative ring over which the computations take place.

Constructing a Domain

To demonstrate, we will construct $\mathrm{GF}(16)$ as $\mathbb{Z}_2[x]/\langle x^4+x+1\rangle$ as a table of Maple constants and procedures. One can also use a Maple module.

```
> f := x^4+x+1:
> GF16['-'] := proc(a,b) a-b mod 2 end:
> GF16['+'] := proc() add(i,i=[args]) mod 2 end:
> GF16['*'] := proc() Rem(mul(i,i=[args]),f,x) mod 2 end:
> GF16['/'] := proc(a,b) local q; Gcdex(b,f,x,'q') mod 2:
                  Rem(a*q,f,x) mod 2; end:
> GF16['='] := proc(a,b) evalb((Rem(a-b,f,x) mod 2)=0) end:
> GF16['0'] := 0:
> GF16['1'] := 1:
```

This domain can now be used with GenericLinearAlgebra. Below we solve a linear system over $\mathrm{GF}(16)$ and check the result. The domain is passed as an index to each command of the package.

$$\begin{bmatrix} x^3+x+1 & x^3+x^2+x \\ x^2+x & x^3 \end{bmatrix} \mathbf{X} = \begin{bmatrix} 1 \\ x^3+1 \end{bmatrix}$$

```
> A := Matrix([[x^3+x+1, x^3+x^2+x], [x^2+x, x^3]]):
> B := Vector([1,x^3+1]):
> X := LinearSolve[GF16](A,B);
```

$$X := \begin{bmatrix} x+1 \\ x \end{bmatrix}, [\ ]$$

```
> MatrixVectorMultiply[GF16](A,X[1]);
```

$$\begin{bmatrix} 1 \\ x^3+1 \end{bmatrix}$$

Coding Style

To illustrate how the routines are coded, we show the source code for matrix–matrix multiplication.

```
MatrixMatrixMultiply := proc(A::Matrix, B::Matrix)
 local D, n, p, m, C, i, j, k;
  D := GenericCheck(procname,MatrixMatrixMultiplyOperations);
  if op(1,A)[2] <> op(1,B)[1] then error end if;
  n, p := op(1,A);
  m := op(1,B)[2];
  C := Matrix(n,m);
  for i to n do
   for j to m do
    C[i,j] := D['+'](seq(D['*'](A[i,k], B[k,j]), k=1..p))
   end do
  end do;
  C
end proc
```

Strongly Connected Blocks

The use of the strongly connected blocks was motivated by the following question: For what matrix structures can we compute the characteristic polynomial quickly?

Consider a matrix of the form

$$A = \begin{pmatrix} h & 0 & g & 0 \\ z & d & y & c \\ f & 0 & e & 0 \\ x & b & w & a \end{pmatrix}$$

If we permute rows 1,4 and columns 1,4, we would get a upper triangular matrix, where the computation of the characteristic polynomial (and also determinant) will be quick. One can show that simultaneous row and column interchanges do not modify the characteristic polynomial.

In order to apply the above observation, we need to efficiently compute the permutation that would make the matrix block upper triangular.

Let G be a directed graph with n vertices and m edges such that vertex j is adjacent to vertex i iff $A_{ij} \neq 0$. The strongly connected components of G, which can be determined in $O(n+m)$ time, correspond to the strongly connected blocks of A.

```
> Q['+'],Q['-'],Q['*'],Q['0'],Q['1'] := '+','-','*',0,1:
> Q['='] := proc(a,b) evalb(a=b) end proc:
> A := Matrix([[8,0,7,0],[26,4,25,3],[6,0,5,0],[24,2,23,1]]);
```

$$\begin{bmatrix} 8 & 0 & 7 & 0 \\ 26 & 4 & 25 & 3 \\ 6 & 0 & 5 & 0 \\ 24 & 2 & 23 & 1 \end{bmatrix}$$

```
> k,B := StronglyConnectedBlocks[Q](A);
```

$$0, [\begin{bmatrix} 8 & 7 \\ 6 & 5 \end{bmatrix}, \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix}]$$

Quotient Rings

We have also written a Maple package for computing in polynomial quotient rings. Below we construct $\mathbb{Q}[x,y,z]/\langle x^2+y, z^3-x, xy^2-2\rangle$ and test whether this domain is a field, before computing a determinant.

```
> with(QuotientRings);
> Q := QuotientRing(x^2+y, z^3-x, x*y^2-2);
```

$$Q := Q[x,y,z]/\langle x^2+y, z^3-x, xy^2-2\rangle$$

```
> IsField(Q);
```

$$true$$

```
> A := Matrix([[x*z^2 + 1, y^2 + x], [x*y-2, x+z+5]]);
```

$$A := \begin{bmatrix} xz^2+1 & y^2+x \\ xy-2 & x+z+5 \end{bmatrix}$$

```
> d := Determinant[Q](A);
```

$$5+3x+z-z^2y+5z^2x-3y+3y^2$$

The package can also perform computations over an integral domain. It includes an exact division algorithm, and an algorithm to simplify fractions which we illustrate.

```
> Q := QuotientRing(x*z-y^2, x-y-1);
```

$$Q := Q[x,y,z]/\langle xz-y^2, x-y-1\rangle$$

```
> IsField(Q), IsIntegralDomain(Q);
```

$$false, true$$

```
> GaussianElimination[Q](A);
```

$$\begin{bmatrix} 1 & \dfrac{z+1}{z^2+z-y+1} \\ 0 & 1 \end{bmatrix}$$