

Linear Algebra over $\{0,1\}$

I've got two sets of routines for you to do linear algebra over the field of two elements--0 and 1. The code written in C that can be called in matlab. The set of routines that you should use were written by Arvind Sankar, the TA for the class the last time I taught it. They programs are:

- [FormDualMod2.c](#) and
- [RankMod2.c](#).

These programs are faster than mine and have cleaner code. They also work with logical matrices in Matlab. However, since these did not appear until Matlab 6.5, you might not be able to use them if you have an old version. In this case, I recommend my own code:

- [FormDual.c](#) and
- [Rank.c](#).

To turn these into ".mex" files that matlab can execute, type

```
>> mex Rank.c
>> mex FormDual.c
```

Note that this command produces binary files that will differ on different architectures. So, if you are running longjobs and you do not control which machine you job runs on, you could easily wind up running on a machine that doesn't understand the binary you created. Fortunately, matlab gives different extensions for different architectures. So, if you generate mex files on each architecture, you should be safe. (or, your longjob script could begin by running mex). The program Rank is designed to compute the rank of a 0/1 matrix. Here is an example

```
>> m = round(rand(4,4))
```

```
m =
```

```
    1    0    1    1
    1    1    1    0
    0    1    0    1
    0    0    1    1
```

```
>> Rank(m)
rank is 3
```

```
ans =
```

```
    3
```

```
>> rank(m)
```

```
ans =
```

```
    4
```

Note that the ordinary rank over the reals is 4. FormDual will help you compute the generator matrix of a code from its check matrix. However, it will not generate output for all input matrices. If its input matrix has rank less than its number of rows, or has all-zero columns, or if all-zero columns appear during elimination, then FormDual will return an error and the empty matrix. (the reason for this is that such matrices will not yield useful codes).

```
>> m = round(rand(3,6))
```

```
m =
    1    1    0    1    1    1
    1    0    0    0    0    0
    1    1    1    0    0    0
```

```
>> FormDual(m)
All-zero columns detected.
ans =
```

```
    []
```

```
>> G = round(rand(3,6))
```

```
m =
    0    0    0    1    0    1
    1    1    0    1    0    0
    1    0    1    1    1    1
```

```
>> H = FormDual(G)
```

```
H =
    1    1    1    0    0    0
    1    1    0    0    1    0
    0    1    0    1    0    1
```

To check that this really is a dual, you can verify that both matrices have full rank, and that the inner product of a row from one with a row from another is 0 mod 2.

```
>> Rank(G)
rank is 3
```

```
ans =
```

```
    3
```

```
>> Rank(H)
rank is 3
```

```
ans =
```

```
    3
```

```
>> G * H'
```

```
ans =
```

```
    0    0    2
    2    2    2
    2    2    2
```

You'll note that FormDual does give its output in systematic form, but that the embedded identity matrix might appear in interlaced columns. (in this last example it is in the 3rd, 5th and 6th). If we ask FormDual to output two arguments, it will also output a permutation that we can use to make the output matrix systematic with the identity matrix at the end. If we apply this permutation to the H matrix, we should also apply it to the G matrix or they might not be duals:

```
>> [H,p] = FormDual(G)
```

```

H =
    1    1    1    0    0    0
    1    1    0    0    1    0
    0    1    0    1    0    1

p =
    1    2    4    3    5    6

>> H = H(:,p)

H =
    1    1    0    1    0    0
    1    1    0    0    1    0
    0    1    1    0    0    1

>> G * H'    %% oops, still need to permute G

ans =
    1    0    1
    3    2    1
    2    2    2

>> G = G(:,p)

G =
    0    0    1    0    0    1
    1    1    1    0    0    0
    1    0    1    1    1    1

>> G * H'    %% much better

ans =
    0    0    2
    2    2    2
    2    2    2

>>

```

Arvind's code has the advantage that it works with logical matrices. These are matrices just of 0s and 1s, and I suspect that they are kept in a compressed format in Matlab 6.5. To generate a random 0/1 logical matrix, you could type

```

>> G = rand(5,10) > .5

G =
    0    0    1    0    1    0    0    1    0    0
    1    0    1    0    1    1    1    0    1    0
    1    1    0    1    1    0    0    1    1    0
    1    1    1    0    0    1    0    1    0    0
    1    0    0    1    0    1    1    0    1    0

>> class(G)

ans =

```

```
logical
```

```
>>
```

You cannot multiply a logical matrix by a logical matrix. But, you can multiply an ordinary vector by a logical:

```
>> (rand(1,5) > .5) * G
??? Error using ==> *
Function '*' is not defined for values of class 'logical'.
```

```
>> mod(round(rand(1,5)) * G,2)
```

```
ans =
```

```
1 1 0 0 1 0 0 0 1 0
```

Arvind's FormDualMod2 code produces the dual matrix and a vector indicating which rows contain the message bits.

```
>> [H,ind] = FormDualMod2(G)
```

```
H =
```

```
0 1 1 0 0
1 1 1 1 0
1 0 1 0 0
0 0 1 0 0
1 0 0 0 0
0 0 0 1 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
```

```
ind =
```

```
0 0 0 0 1 0 1 1 1 1
```

You can convert between logical and double matrices with the commands `logical` and `double` in Matlab. However, I recommend working with logicals whenever possible. To add two columns of H modulo 2, I recommend using `xor`:

```
>> c = xor(H(:,1),H(:,3))
```

```
c =
```

```
1
0
0
1
1
0
0
1
0
0
```