Lab: Cuerpos
Tratamiento de señales

Cuerpos en aritmética modular Sabemos que $(\mathbb{Z}_p, +, \cdot)$ es cuerpo si y sólo si p es primo.

- Vamos a construir la función `tablas` que nos de las tablas de sumar y multiplicar en aritmética modular módulo n. Por supuesto podríamos construir cada una de las tablas con dos lazos `for` anidados... pero vamos a tratar de ser un poco más elegantes (Hay que leer la ayuda de Matlab sobre qué hace la función `kron` y ojear el producto de Kronecker en wikipedia o similar para entender bien cómo se comporta esta función...):

```
function [sum, mul] = tablas(n)
    mul = mod(kron(1:n-1, [1:n-1]'), n);
    sum = mod(kron(0:n-1, ones(n, 1)) + kron([0:n-1]', ones(1,n)), n);
end
```

- Utilizamos la función para echar un vistazo a las tablas de aritmética modular para varios módulos:

```
[sum, mul] = tablas(3)
[sum, mul] = tablas(4)
[sum, mul] = tablas(5)
[sum, mul] = tablas(6)
[sum, mul] = tablas(7)
```

  y, ya puestos, vamos a ver un par de casos especiales:

```
[sum, mul] = tablas(2)
[sum, mul] = tablas(1)
```

Inverso multiplicativo Vamos a ver tres formas alternativas de calcular el inverso multiplicativo en $(\mathbb{Z}_p, +, \cdot)$. Es decir, dado $a \in \mathbb{Z}_p$, queremos encontrar un $b \in \mathbb{Z}_p$ tal que $a \cdot b \equiv 1 \pmod{p}$. A este número b lo podremos denotar como a^{-1}.

- La forma más inmediata, podría ser una búsqueda exhaustiva mediante un lazo `for` del que salimos en cuanto encontramos el inverso (sabemos que tiene que existir si estamos trabajando con p primo):

```
function b = inv1(a, p)
    for b=1:p-1
        if mod(a*b, p) == 1
            break
        end
    end
end
```

- Alternativamente, podríamos apoyarnos en la función `tablas` para no tener que utilizar un lazo `for` (a expensas de tener que construir y manejar tablas muy grandes si p es muy grande):

```
function b = inv2(a, p)
    [sum, prod] = tablas(p);
    b = find(prod(a,:) == 1);
end
```

- Por último, podríamos calcular explícitamente el inverso apoyándonos en el denominado "pequeño teorema de Fermat"; que establece que $a^p \equiv a \pmod{p}$. Por lo tanto, tendremos estas dos formulaciones alternativas, de las que utilizaremos la última para calcular el inverso de a: $a^{p-1} \equiv 1 \pmod{p}$ y $a^{p-2} \equiv a^{-1} \pmod{p}$:

```
function b = inv3(a, p)
    b = mod(a^(p-2), p);
end
```

- Utilice las funciones para calcular algunos inversos:

```
a=11; p=13; [inv1(a, p), inv2(a, p), inv3(a, p)]
a=11; p=31; [inv1(a, p), inv2(a, p), inv3(a, p)]
```

- ¿Observa algún problema? Investigue su causa y piense (no necesariamente ahora, puede que le lleve un poco de tiempo) una forma de solventarlo...

Inversos de matrices con elementos en $\mathbb{Z}_p$ Sabemos que una matriz de $A \in \mathbb{R}^{m\times m}$ tiene inverso si y sólo si $|A| \neq 0$. Vamos a ver qué ocurre con las matrices de $\mathbb{Z}_p^{m\times m}$. Por sencillez, vamos a trabajar con $m \times m = 2 \times 2$ y vamos a empezar con $p = 3$.

- En primer lugar vamos a estudiar la matriz
$$A = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix},$$
cuyo determinante es $|A| = 1$ y cuya matriz inversa nos proporciona matlab como `B=inv(A)`:
$$B = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}.$$
Si reducimos módulo 3 en matlab con `C=mod(B, 3)` obtenemos
$$C = \begin{bmatrix} 1 & 2 \\ 2 & 2 \end{bmatrix}$$
y si calculamos `D=A*C` y `E=mod(D, 3)` tenemos
$$D = \begin{bmatrix} 4 & 6 \\ 3 & 4 \end{bmatrix} \qquad E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I;$$
por lo que efectivamente $C = A^{-1}$.

- Estudiemos ahora la matriz
$$A = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix},$$
cuyo determinante es $|A| = 4 \equiv 1 \pmod 3$ y cuya matriz inversa (en $\mathbb{R}$) nos proporciona matlab como `B=inv(A)`:
$$B = \frac{1}{4}\begin{bmatrix} 2 & -1 \\ 0 & 2 \end{bmatrix}.$$
Si tenemos en cuenta que $4 \equiv 4^{-1} \pmod 3$ y que $-1 \equiv 2 \pmod 3$, podríamos decir que una $C \equiv B \pmod 3$ sería
$$C = \begin{bmatrix} 2 & 2 \\ 0 & 2 \end{bmatrix}.$$
Si realizamos dicho producto y reduciendo módulo 3, tendríamos `D=A*C` y `E=mod(D,3)`
$$D = \begin{bmatrix} 4 & 6 \\ 0 & 4 \end{bmatrix}, \qquad E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I;$$
por lo que efectivamente $C = A^{-1}$.

- Hacemos una función `inversa` para calcular la inversa de cualquier matriz $A \in \mathbb{Z}_p^{m\times m}$ con $|A| \neq 0$:

```
function C=inversa(A, p)
    d = det(A);
    assert(d ~= 0, 'Determinante nulo');
    A = mod(A, p);
    invd = inv1(d, p);
    B = inv(A);
    C = mod(d*invd*B, p);
    D = A*C;
    E = mod(D, p);
    assert(isequal(E, eye(size(A))), 'Algo va mal...');
end
```

- Por último, a modo de curiosidad, vamos a contar cuántas matrices hay en $\mathbb{Z}_p^{m\times m}$ y cuántas de ellas tienen inversa. Para ello, podemos observar que podemos numerar los elementos de matriz $m \times m$ como los elementos de un vector de m^2 componentes (por ejemplo colocando las columnas una después de otra en una única columna o las filas...). Si hacemos esto, habrá tantas matrices como números de m^2 cifras de entre el alfabeto de $\mathbb{Z}_p$, que consta de p elementos $0, 1, \ldots, p-1$. Como tenemos m^2 posiciones en las que colocar uno de entre p símbolos, tendremos p^{m^2} matrices distintas. La función `cuenta` se encarga de enumerar todas estas matrices y de contar cuántas de ellas tienen determinante no congruente con 0

```
function [N, I] = cuenta(p, m)
    N = p^(m^2);
    I = 0;
    for k=0:N-1
        cod = dec2base(k, p, m^2);
        vec = cod - '0';
        A = reshape(vec, m, m);
        D = mod(det(A), p);
        if D ~= 0
            I = I + 1;
        end
    end
end
```

- Haz una función que represente la variación del porcentaje de matrices invertibles en $\mathbb{Z}_p^{2\times 2}$ con p (representar p en el eje x y el porcentaje de matrices invertibles en el eje y para los primos $p \leqslant 20$, que se pueden generar en matlab con `P = primes(20);`