Let's change a little the initial system of 4 equations by adding to the right side by numbers -1, 1, or 0 at random.

1*x + 2*y + 3*z = 14	(1)	1*x + 2*y + 3*z = 14 - 1	1*x + 2*y + 3*z = 13 = s1e	15+	
2*x + 1*y + 2*z = 10	(2)	2*x + 1*y + 2*z = 10 + 0	2*x + 1*y + 2*z = 10 = s2e	11	
3*x + 2*y + 2*z = 13	(3)	3*x + 2*y + 2*z = 13 +1	3*x + 2*y + 2*z = 14 = s3e	13+	
3*x + 3*y + 1*z = 12	(4)	3*x + 3*y + 1*z = 12 - 1	3*x + 3*y + 1*z = 11 = s4e	12+	

(1) + (3) + (4) = 7   7*x + 7*y + 6*z = 7   =7 mod 11

Then we introduce the erroneous vector for the system corresponding to (1), (2), (3) equations denoting it by s123e, and   the erroneous vector for the system corresponding to (2), (3), (4) equations denoting it by s234e.
The corresponding matrices M123 and M234 are the same.

Then the solution of the first system of equations can be found by computing the inverse matrix to the matrix M123 we denote by M123i, and
      the solution of the second system of equations can be found by computing the inverse matrix to the matrix M234 we denote by M234i

```
>> M123=[1 2 3; 2 1 2; 3 2 2]
M123 =
   1  2  3
   2  1  2
   3  2  2
```

```
>> s123e=[13;10;14]
s123e =
   13
   10
   14
```

```
>> M234=[2 1 2; 3 2 2; 3 3 1]
M234 =
   2  1  2      x
   3  2  2      y
   3  3  1      z
```

```
>> s234e=[10;14;11]
s234e =
   10
   14
   11
```

```
>> M123i=inv(M123)
M123i =
  -0.4000   0.4000   0.2000
   0.4000  -1.4000   0.8000
   0.2000   0.8000  -0.6000
```

```
>> M234i=inv(M234)
M234i =
  -4   5  -2
   3  -4   2
   3  -3   1
```

```
>> w123e=M123i*s123e
w123e =
   1.6000
   2.4000
   2.2000
```

```
>> w234e=M234i*s234e
w234e =
   6.0000e+00
  -2.0000e+00
   7.1054e-15
```

As we see solutions differs.

Learning With Errors - LWE Encryption

0

Time is computed modulo 12.
We will perform computations modulo 11.

0

Time is computed modulo 12.
We will perform computations modulo 11.

In classical cryptography computations are performed modulo *p*,
when *p* is a large prime number.
Let *p* is prime then for all positive integers *z*, expression *z* mod *p*
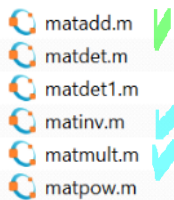is a reminder of *z* divided by *p*.

Let *p*=11.
Then computations of equations (1), (2), (3) and equations (2), (3), (4) we perform modulo 11.
Matrices M123, M234 and s123, s234 we leave the same and operations we perform modulo 11.
Matrix operations mod p are realized in .m files.
Library of .m files

matadd.m
matdet.m
matdet1.m
matinv.m
matmult.m
matpow.m

We are using functions **matinv.m** and **matmult.m**
To find inverse matrix **Mi** mod *p* to matrix **M** we use
>> Mi=matinv(M,p)
To multiply inverse matrix **Mi** with vector **s** mod *p* to obtain vector **w** we use command
>> w=matmult(Mi,s,p)

Let's **again** change a little the initial system of 4 equations by adding to the right side the numbers -1, 1, or 0 at random.

Initial equations   Equations with errors   Final equations with errors
1*x + 2*y + 3*z = 14   (1)   1*x + 2*y + 3*z = 14 - 1   1*x + 2*y + 3*z = 13 = s1e
2*x + 1*y + 2*z = 10   (2)   2*x + 1*y + 2*z = 10 - 0   2*x + 1*y + 2*z = 9 = s2e
3*x + 2*y + 2*z = 13   (3)   3*x + 2*y + 2*z = 13 + 1   3*x + 2*y + 2*z = 14 = s3e
3*x + 3*y + 1*z = 12   (4)   3*x + 3*y + 1*z = 12 + 1   3*x + 3*y + 1*z = 13 = s4e

```
>> M123=[1 2 3; 2 1 2; 3 2 2]        >> s123e=[13;10;14]       >> M234=[2 1 2; 3 2 2; 3 3 1]        >> s234e=[10;14;11]
M123 =        w123e=                 s123e =                   M234 =        w=234e                s234e =
  1  2  3        x                      13                       2  1  2        x                     10
  2  1  2        y                      10                       3  2  2        y                     14
  3  2  2        z                      14                       3  3  1        z                     11
```

```
>> M123i=matinv(M123,p)                                        >> M234i=matinv(M234,p)
M123i =                                                        M234i =
  4  7  9                                                        7  5  9
  7  3  3                                                        3  7  2
  9  3  6                                                        3  8  1
```

```
>> I123=matmult(M123i,M123,p)                                  >> I123=matmult(M123i,M123,p)
I123 =                                                         I123 =
  1  0  0                                                        1  0  0
```

>> I123=matmult(M123i,M123,p)
I123 =
  1 0 0
  0 1 0
  0 0 1

>> w123e=matmult(M123i,s123e,p)
w123e =

  6
  9
  0

>> w234e=matmult(M234i,s234e,p)
w234e =

  8
  7
  10

As we see solutions differs as well since right sides of equations are corrupted by errors.
System of equations (1), (2), (3), (4) is inconsistent since if solution w123e satisfies one part of equations (1), (2), (3) it does not satisfies the other part of equations (2), (3), (4).

Exercises

Public parameters - **PP** is a system of 4 linear equations
with errors in the right-side computed mod 11:

$1*x + 2*y + 3*z = 13 \bmod 11 = 2 = s1e$
$2*x + 1*y + 2*z = 9 \bmod 11 = 9 = s2e$
$3*x + 2*y + 2*z = 14 \bmod 11 = 3 = s3e$
$3*x + 3*y + 1*z = 13 \bmod 11 = 2 = s4e$

Learning with errors - LWE encryption is performed for a single bit $b=\{0, 1\}$ at once.

The private key is a vector of unknown but **exact** wages per hour: **PrK** = **w'** = (x, y, z) = (1, 2, 3),
where **w'** is written as a row vector and is a transpose vector of vector column **w**.
w' = [1,2,3] = **PrK**.

To encrypt bit **b** a random sub-system of equations is selected, e.g. let's take (1), (2), (3) sub-system of equations.
Then erroneous vector for the sub-system corresponding to (1), (2), (3) equations we denote by s123e.
The corresponding matrix M123 is the same.
The equations and erroneous vectors are of the form.

$1*x + 2*y + 3*z = 2 = s1e$ (1)
$2*x + 1*y + 2*z = 9 = s2e$ (2)
$3*x + 2*y + 2*z = 3 = s3e$ (3)

>> M123=[1 2 3; 2 1 2; 3 2 2]
M123 =
  1 2 3   x
  2 1 2   y
  3 2 2   z

>> s123e=[13;10;14]
s123e =
  2
  9
  3

Encryption

Let **Alice** is intending to encrypt bit **b**=\{0, 1\} and send it to **Bob.**
By having public parameters **PP**, consisting of system of 4 equations, **Alice** chooses **at random** some sub-system consisting of 3 equations.

Let this sub-system consist of equations (1), (2), (3).
Having this sub-system, **Alice** adds (1), (2), (3) equations together mod 11:

$$+ \begin{cases} 1*x + 2*y + 3*z = 2 = s1e \ (1) \\ 2*x + 1*y + 2*z = 9 = s2e \ (2) \\ 3*x + 2*y + 2*z = 3 = s3e \ (3) \end{cases}$$

$$6*x + 5*y + 7*z = 14 \bmod 11 = 3$$

⌐⌐⌐⌐

If **Alice** intends to encrypt bit *b*=0 then she sends original equation $6*x + 5*y + 7*z = 3$ corresponding to ciphertext c_0 to **Bob**.

If **Alice** intends to encrypt bit *b*=1 then she sends equation $6*x + 5*y + 7*z = 3 + 5 = 8$ corresponding to ciphertext c_1 to **Bob**.
This means that for *b*=1 encryption **Alice** adds to the right side of equation number **5**.
This number corresponds to the rounded down number **p**/2 = 11/2 = 5.5, i.e. it is equal to **5**.

<div align="center">

Decryption

</div>

For decryption **Bob** is using his **PrK** = **w'** = (x, y, z) = (1, 2, 3).

Let **Bob** received ciphertext c_0: $6*x + 5*y + 7*z = 3$.
Then **Bob** puts his **PrK** values (1, 2, 3) to the left side of c_0 equation and obtains:
$$6*1 + 5*2 + 7*3 = 6 + 10 + 21 = 37 \bmod 11 = 4$$
The number **4** is close to the right side number **3** of the received ciphertext c_0: $6*x + 5*y + 7*z = 3$
This means that **Bob** decrypted bit *b*=0.

Let **Bob** received ciphertext c_1: $6*x + 5*y + 7*z = 8$.
Then **Bob** puts his **PrK** values (1, 2, 3) to the left side of c_1 equation and obtains:
$$6*1 + 5*2 + 7*3 = 6 + 10 + 21 = 37 \bmod 11 = 4$$
The number **4** is far from to the right side number **8** of the received ciphertext c_1: $6*x + 5*y + 7*z = 8$.
The difference is **8** - **4** = **4** and it is more than the errors values fluctuating in the range [-1, 1].
This means that **Bob** decrypted bit *b*=1.

$7*x + 7*y + 6*z = 7 = c_0$

w' = [1,2,3] = **PrK**.
$7*1 + 7*2 + 7*3 = 42 \bmod 11 = 9$

```
>> mod(15+13+12,11)
ans = 7
>> mod(42,11)
ans = 9
```

15+
11
13+
12+