Introduction to Pari/GP

Marine Rougnant

19/02/2024 - 23/02/2024

1 Basic commands

Exercise 1 (Basic commands)**.**

Play with the following basic commands. You run commands by typing them and then pressing Enter.

1. Arithmetic operations:

```
2+2  4-7   5*7  3^4   4/3  17\3  128%7
```

2. Floating-point numbers:

```
2.3  3.9E6  Pi
\p 100
Pi
2^Pi
```

3. Complex numbers:

```
I^2  (1+2*I)*(4-3*I)  (-1)^(1/2)
```

4. Add a semicolon if you do not want to display the output, or to separate commands on the same line. Compare:

```
N = 2022!
N = 2022!;
N = 2022!; N%2027
```

5. Elements of basic quotient rings:

```
Mod(2,7)^3
Mod(x+7,x^4+1)^10
Mod(Mod(3*x+1,x^3+x+1),11)^100
```

Exercise 2 (Variables, functions, symbolic expressions)**.**

1. Assignment to a variable is `=`, and can be combined with a basic operation:

```
a = 19; b = 11;
a*b
a += b
a
```

2. Every non-assigned variable can be used as a polynomial variable:
```
P = x^3 + 13*x + 1
P'
```
The polynomial variable can be recovered with a quote (') **before** the name of the variable:
```
a^2+2*a+3
a = 'a
a^2+2*a+3
```

3. Function call:

```
sqrt(2)  exp(2*I*Pi/3)  abs(3+4*I)  polcoef(P,1)   P
```

Some functions have optional arguments:

```
polcoef((x+1)*(y+2*y^2)*z,1,y)
```

4. Define a function with a symbolic expression:

```
f(x)=x^2+x+1
f(2)
(x-1)*f
(x-1)*f(x)
g(x,y) = x^2+2*y  g(1,y)   g(x,1)   g(1,1)  g(1)
```

5. Evaluate a symbolic expression:

```
subst(P,x,1+T)
```

Exercise 3 ((Vectors and matrices))**.**

1. Row vectors are created with square brackets, with values (of any type) separated by commas.

```
[1,3,2]
v = [-1,y,"bla",1/u]
[-3..10]
```

Their components are numbered from 1 to the length `#v`.

```
v[#v]
```

2. Matrices are created with square brackets, with components of a row separated by commas and rows separated by semicolons.

```
M = [1,2;3,4]
M^2
```

3. You can type a multiline command using backslash as follows:

```
N = [7,9;\
     1,8;\
     3,0]
N*M
M*N
```

4. You can transpose with `~`:

```
v~
N~
```

5. You can access entries of matrices with square brackets:

```
N[3,1]
N[,2]
N[,1] = v[2..4]~
N
```

6. You can create vectors and matrices with formulas for the entries:

```
vector(5)
vector(5,i,i^2+1)
matrix(3,5,i,j,1/(i+j))
```

7. You can concatenate two matrices with compatible number of rows:

```
x = [1,2];  y = [3,4]; concat(x,y)
Mat([x,y]~)
matconcat([x;y])
A=2*matid(2); B=[1,-1]~; matconcat([A,B])
```

8. Use the fonctions `matid` and `matdiagonal` to create $A = I_4$ and the diagonal matrix B of size 8 with diagonal coefficients equal to $1, \ldots, 8$.

9. With `matconcat`, define :
$$C = \left(\begin{array}{c|c} & a \\ B & \vdots \\ & h \end{array} \right)$$

10. What should be the result of the concatenation of A and B ? Try.

11. Define the following bloc matrix :
$$D = \left(\begin{array}{c|c|c} \frac{I_4}{I_4} & \text{B} & \text{C} \end{array} \right)$$

12. What is the size of D ? Use `matsize`.

2 Programming

Exercise 4 (Reading files)**.**

Once you start working on exercises requiring more than a few commands (or working on your own projects), you should write your code in a file and read the file in Pari/GP.

1. Create a file `test` with a text editor, and write 10 lines of Pari/GP commands, including some variable assignments. Make sure that there are no spaces in the file name (for instance, **do not** use `my test` as a file name).

   Warning: the lines will be executed one by one, as if you hit Enter between the lines. If you want to write a multiline instruction, you should use backslash, or enclose the block in curly braces. However, you will then need to separate instructions with semicolons, as if they were on the same line.

```
{M = matrix(10,10,i,j,
  a = 1+i+j;
  b = 2+i^2+j^2;
  a/b
)};
```

2. To read the file:

- If you are a Linux user, type

  `\r test`

  possibly adding a path if the file is not in folder where you started Pari/GP.
- If you are a Windows user, type

  `\r`

  but do not press Enter. Then, click on the file icon of `test` and drag it to the Pari/GP windows, and drop it on the command line. The complete path to your file should appear. Then, press Enter.
- If you are a MacOS user, you can use either of the options above.

You need to make sure that the file is a plain text file (TXT / `.txt` file), not a file with an enriched format (RTF / `.rtf`, DOC / `.doc`, etc). On MacOS, you can try Format → Make Plain Text.

3. Do you see the output of your commands?
4. What are the values of the variables you that assigned in the file?
5. Add some print statements: `print(something)` to your file (save the changes!).
6. Read it again by simply typing `\r` and Enter. You should see the output of the print statements.

Exercise 5 (Writing in a file)**.** You may want to save the outputs of your functions.

1. Type

   `\w`

   then add the path of the file in wich you want to write. This file has to be a text file, and use a forward slash rather than a back slash to separate parts of the path. That will print the last output from pari to the document.

   (a) Try to write "Hello world" in a text document.

   (b) In the same document, print the identity matrix of size 4. Can you then read this matrix from PARI/GP ?

2. Another way to write in a document is to use the `write( file, str)` function : it appends a new line to your `file` and print `str`. Write in a document the identity matrix of size 4 in such a way that you can read it with the

   `\r`

   command.

Exercise 6 (Programming)**.**

1. The syntax to define functions is as follows:

```
f(a,b,c) =
{
  instruction1;
  instruction2;
  ...
  value_to_be_returned
}
```

   You can also return a value in the middle of the function with `return(value)`.

   Write a function `f(n)` that returns n^n mod 5. Test it.
   Be careful : use `my ( )` to store the local variables.

2. Boolean values are represented by 0 (false) and 1 (true).

```
2 < 3
4 >= 4.
P == 1
a != 3
```

3. Comprehension

```
[n^2|n<-[1..10]]
```

`isprime(x)` says if a given integer is prime or not. Give the list of all prime numbers between 10 and 55.

4. The syntax of the conditional control structure is as follows:

```
if(condition, instructions_if_true, instructions_if_false)
```

It is better to indent your code as follows:

```
if(condition,
  instruction1_if_true;
  instruction2_if_true;
  etc
,/*else*/
  instruction1_if_false;
  instruction2_if_false;
  etc
)
```

Write a function `parity(n)` that tests the parity of `n`, and prints "even" or "odd" accordingly. Test it.

5. The boolean operators are `&&` (and), `||` (or), `!` (not). Write a function `isgood(n)` that prints "good" when `n` is (not divisible by 10) or (congruent to 1 mod 3 and strictly greater than 30), and "bad" otherwise. Test it.

6. The syntax of the for loops is as follows:

```
for(i=start_value,end_value,
  instruction1;
  instruction2;
  etc
)
```

Write a function `squares(n)` that prints all the squares of integers between 1 and `n`. Test it.

7. The syntax of the while loops is as follows:

```
while(condition,
  instruction1;
  instruction2;
  etc
)
```

Determine the lowest positive integer N such that $e^{-N} < 10^{-6}$.

Exercise 7 (Break loop)**.**

1. Run the following code:

```
{for(i=1,oo,
  for(j=-i,i,
    if(!random(1000), 1/0)
  )
)}
```

2. The code will trigger an error, and GP will end up in a state called the break loop. In the break loop, you can run GP commands, and in particular you can inspect the values of the variables when the error was triggered. Try it with `i` and `j`.

3. You can exit the break loop by tying `break` or pressing `Ctrl+D`.

4. Replace `1/0` by `print(j)`. Run the code again. Press `Ctrl+C`. You enter the break loop again. Here, since there was no error, you can either continue running the code by pressing Enter, or exit the break loop as before.

3 Linear algebra

Exercise 8.
Let A and Y be the matrices

$$A = \begin{pmatrix} -3 & 1 \\ 6 & -2 \end{pmatrix} \qquad Y = \begin{pmatrix} 9 \\ -18 \end{pmatrix}$$

1. You can compute the determinant of A with `matdet`. Is A invertible ?

2. Give a basis of the image of A (`matimage`).

3. Give a basis of the kernel of A (`matker`)

4. Find a solution of the system $AX = Y$ with `matinverseimage`.

5. Deduce the set of the solutions of $AX = Y$.

6. Define an invertible 3×3 matrix A' and a column vector Y'. Solve the system $A'X = Y'$ with both `matsolve` and `matinverseimage`. Is it possible to use `matsolve` in the previous example ? Why ?

Exercise 9 (Inverse)**.**
Consider the matrix

$$A = \begin{pmatrix} 1 & 2 & -1 & 4 \\ 7 & 0 & 1 & 3 \\ -1 & 0 & 5 & 1 \end{pmatrix}$$

1. Take you favourite 3×3 invertible matrix. Compute the inverse.

2. Compute the rank of the matrix A. Is A invertible ? left-invertible ? right-invertible ? Try to compute A^{-1}.

3. Same question with ${}^tA$.

Exercise 10 (Linear algebra over $\mathbb{Z}/N\mathbb{Z}$)**.**
Consider the matrix A of the previouos exercise.

1. Compute a basis of the image of A mod 5 with `matimagemod`. Compare with `matimage(Mod(A,5))`.

2. Compute the kernel of A mod 5 with `matkermod`.

3. Extract from A the matrix formed by the three first columns. Compute its determinant. What should it be mod 5 ? Check with `matdetmod` and compute the inverse using `matinvmod`. Compare with `Mod(M^-1,5)`.
   Deduce the solutions of $MX = Y$ mod 5. Find the same result with `matsolvemod`.

Exercise 11. Consider the matrix

$$A = \begin{pmatrix} 0 & 4 & 3 \\ 2 & -2 & -3 \\ -2 & 4 & 5 \end{pmatrix}$$

1. Show that $P(X) = X^3 - 3X^2 + 4$ is the characteristic polynomial of A (`charpoly`).
2. Factor P and deduce the eigenvalues of A. Check your result with `mateigen`.
3. Compute the minimum polynomial (`minpoly`). Is A diagonalisable ?
4. Give a basis and the dimension of each eigenspace.
5. Give an upper triangular (or diagonal) matrix Δ such that Δ and A are similar.

Exercise 12. Consider the following vectors :

$$u_1 = (0, 1, -2, 1), \quad u_2 = (1, 0, 2, -1), \quad u_3 = (3, 2, 2, -1), \quad u_4 = (0, 0, 1, 0)$$

1. Define these four vectors as column matrices. (`Mat`). Check that it has the correct type with `type`.
2. Give a basis β of the subspace $\mathrm{Vec}\,(u_1, u_2, u_3, u_4)$.
3. Complete β to get a basis of $\mathbb{R}^4$ (`matsupplement`).
4. Compute the intersection of $V_1 = \mathrm{Vec}\,(u_1, u_2)$ and $V_2 = \mathrm{Vec}\,(u_3, u_4)$ (`matintersect`) and check wether it's equal to $\mathrm{Vec}\,(1, 1, 0, 0)$ or not.