1. Syllabus

- **Course description**

In this course, we learn how to use *Mathematica* in various mathematical problems. *Mathematica* is great for visualizing mathematical objects (such as functions, sets, polytopes etc.), for collecting empirical data and for testing conjectures. In particular, we focus on 3D graphics tools provided by *Mathematica*. These tools are ideal for drawing high precision pictures for mathematical papers.

- **Grading scheme**

Each student must complete a course project. The final course grade is equal to the project grade.

The list of topics for projects will be posted on the course webpage soon. Students might also choose a topic not from the list (for instance, a topic related to their term papers). In particular, students are encouraged to use *Mathematica* for drawing pictures for their term papers. The deadline for choosing topics is May 13 (after this deadline the topics will be assigned by instructor’s choice). The deadline for submitting a project is June 21. If the project is submitted before or on June 14, then its author will receive feedback together with an opportunity to correct the project. No feedback will be given on projects submitted after June 14, they will be graded as is.

A good project should contain not only *Mathematica* code but also a complete exposition of underlying mathematics. Such an exposition (in Russian or in English) should be neatly typed using built-in *Mathematica* text editor. Guidelines on preparing a good project will be posted on the course webpage soon.

- **What is Mathematica?**

*Mathematica* combines a computer algebra system and a high-level programming language. It is a commercial product developed by Wolfram Research. There are free alternatives to *Mathematica* such as Sage and other commercial alternatives such as Maple. There are two reasons for choosing *Mathematica* for our course: first, *Mathematica* is already installed on many department computers, second, *Mathematica* has special features (such as manipulator) that can be used to display results nicely.

If you master *Mathematica* you will have no difficulty working with other computer algebra systems since they are all built on the same principles.

2. Quick Start

- **How to use Mathematica?**

Write a sequence of *Mathematica* commands in a notebook (.nb). This will be your Input. To evaluate the command press *Enter* while holding *Shift*. You will get Output in the same notebook. Here is an example.

```
In[1]:= 2 + 2
Out[1]= 4
```
Inputs and Outputs are labelled by Mathematica in the order they are evaluated (not in the order they are written in a notebook). Symbol % denotes the previous output reused in the next evaluation.

\begin{verbatim}
In[2]:= % + 2
Out[2]= 6
\end{verbatim}

You can use inputs and outputs of all previous evaluations as follows.

\begin{verbatim}
In[3]:= Out[1] + 2
Out[3]= 6
\end{verbatim}

However, it might be more convenient to use your own notation for inputs.

\begin{verbatim}
In[4]:= a = 2 + 2
Out[4]= 4
\end{verbatim}

To keep your code short you can ask Mathematica not to display intermediate outputs by placing semicolon ; after inputs. Here is an example.

\begin{verbatim}
In[5]:= a = 2 + 2;
a + 2
Out[6]= 6
\end{verbatim}

- **Syntax and Grammar**

  Built-in Mathematica commands and functions (as well as built-in constants) begin with capital letters. Their arguments are put in square brackets and separated by commas. Here are some examples.

  - \(\sin \pi\)

    \begin{verbatim}
    In[7]:= Sin[Pi]
    Out[7]= 0
    \end{verbatim}

  - \(\log e\)

    \begin{verbatim}
    In[8]:= Log[E]
    Out[8]= 1
    \end{verbatim}

  - \(\sqrt{2}\)

    \begin{verbatim}
    In[9]:= Sqrt[2]
    Out[9]= \(\sqrt{2}\)
    \end{verbatim}

  - Built-in function FactorInteger\(n\) computes all prime factors of an integer \(n\) together with their multiplicities.
In[10]:=
FactorInteger[2013]

Out[10]=
{(3, 1), (11, 1), (61, 1)}

Built-in function 
\text{N}[\text{a}, n] \text{computes}
the first \( n \) digits in the decimal fraction of \( a \).

In[11]:=
N[Sqrt[2], 3]

Out[11]=
1.41

\textit{Function shorthand.} For any built-in function \( F[x] \), you may write \( x // F \) instead. You may also write \( F[a + b] \) instead of \( F[a + b] \).

\textbf{Lists and Tables}

Set-theoretic notation is used for lists: they are put in braces and their elements are separated by commas. To retrieve the \( i \)-th element of a list \( L \) write \( L[[i]] \). Here is an example.

In[12]:=
L = {1, 2, 5};
L[[3]]

Out[13]=
5

Matrices can be encoded by lists as follows.

In[14]:=
M = {{1, 3}, {2, 5}};
MatrixForm[M]

Out[15]/MatrixForm=
\[
\begin{pmatrix}
1 & 3 \\
2 & 5
\end{pmatrix}
\]

To retrieve the \( (i,j) \)-th element of a matrix \( M \) write \( M[[i]][[j]] \) or \( M[[i,j]] \).

In[16]:=
M[[1, 2]]

Out[16]=
3

To create a list you may use the built-in command \texttt{Table} as follows.

In[17]:=
Table[x^2, {x, 1, 10}]

Out[17]=
{1, 4, 9, 16, 25, 36, 49, 64, 81, 100}

The second argument of \texttt{Table} specifies the values of \( x \). If the second argument has form \{\( x, a, b, c \)\} then \( x \) runs over all terms of the arithmetic progression \( a + cn \) in the segment \([a, b] \).

\textbf{Manipulator}

Manipulator is an excellent tool for visualizing outputs depending on several discrete or continuous parameters. Here are some examples.
Manipulate[Plot[Sin[n x], {x, 0, 6}], {n, 1, 10}]

Manipulate[Factor[x^n - 1], {n, 10, 100, 1}]

Manipulate[Plot[Sin[a x + b], {x, 0, 6}],
{{a, 2, "Multiplier"}, 1, 4}, {{b, 0, "Phase Parameter"}, 0, 10}]

(-1 + x) (1 + x) (1 - x + x^2 - x^3 + x^4) (1 + x + x^2 + x^3 + x^4)
Manipulate[Nest[Subscript[#, #] &, x, n], {n, 1, 6, 1}]

Manipulate[{x^2, Button["reset", x = 0]}, {x, 0, 10}]

Manipulate[Plot[f[x], {x, 0, 2 Pi}], {f, {Sin, Cos, Tan, Cot}}]
Manipulate[
Graphics[
{color, Disk[]}
], {color, Purple}]

Manipulate[Graphics[Polygon[pt], PlotRange \[Rule] 2],
  {{pt, {{0, 0}, {1, 0}, {1, 1}, {0, 1}, {1, -1}}}, Locator}]

Manipulate[Graphics[
  ParametricPlot[
    {{t + Sin[t], 1 + Cos[t]}, {a + Cos[t], 1 + Sin[t]},
      {a + Sin[a] + 0.1 Sin[t], 1 + Cos[a] + 0.1 Cos[t]},
      {t, -10, 10}}, {a, -10, 10}]
  ]}
Manipulate[Graphics3D[{Opacity[0.7],
Polyon[{-a, 0, -a}, {-a, b, -a}, {-a, b, b}, {-a, 0, 0}],
Polyon[{-a, 0, -a}, {-a, 0, 0}, {0, 0, 0}],
Polyon[{-a, b, b}, {-a, 0, 0}, {0, 0, 0}, {0, b, b}],
Polyon[{-a, 0, -a}, {0, 0, 0}, {0, b, 0}, {-a, b, -a}],
Polyon[{0, 0, 0}, {0, b, 0}, {0, b, b}],
Polyon[{0, b, 0}, {-a, b, -a}, {-a, b, b}, {0, b, b}]],
PlotRange -> {{-3, 0}, {0, 3}, {-3, 3}}, {a, 1, 3}, {b, 1, 3}]

a
b