Project Description

VBA and Python

Expanding the capabilities of Excel's Optimization.

Project Description

- In this project you will overcome some of the limitations of VBA by using Python. In particular we are going to focus on the limitations that the `Solver` add-in has in Excel (maximum number of variables, constraints, etc.).
- You have to create a VBA macro that will communicate with a python script. The python script will solve the optimization problem and return the results to be analyzed in Excel.
- First the communication between VBA and python will be done using text files.
- In a second stage you should use APIs to communicate between the two languages (Bonus)

Implementation

- You are free to come up with your own implementation. However, I will provide a general outline of how you could implement this project.
- A standard workflow would look like this:
    i. The user defines an optimization problem by adding an objective function, and a list of constraints. This should have either a VBA interface or a well defined structure in Excel.
    ii. The user runs the VBA macro that will write the optimization problem to a text file that Python can read and understand.
    iii. The Python script reads the text file, converts it into an optimization problem that can be solved in python (e.g. with scipy).
    iv. The Python script solves the optimization problem and writes the results to a text file.
    v. The VBA macro reads the results from the text file and displays them in Excel.

General Modeling of an optimization problem

- Create a VBA module that the user can use to model a convex optimization problem. A general convex optimization problem can be written in terms of equations and inequalities.

$$\begin{aligned} \text{minimize} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{aligned}$$

Where $x \in \mathbb{R}^n$ is the optimization variable, $f_0 : \mathbb{R}^n \to \mathbb{R}$ is the objective function, $f_i : \mathbb{R}^n \to \mathbb{R}$ are the inequality constraints, and $h_i : \mathbb{R}^n \to \mathbb{R}$ are the equality constraints.

Mathematical functions

- Functions in VBA are not necessarily objects, which means that you cannot pass them into other functions. Eg. if you want to minimize the logarithm of a function, how would you tell python what is the function you want to minimize?
- To overcome this limitation, we are going to restrict the universe of functions that the user can use to model the optimization problem. This means that the functions f_0, f_i, h_i will be functions that you can parametrize and both languages could agree on.

First alternative

- One way to do this is to create commands that python can easily interpret. For example if the user in VBA wants to model the general polynomial function $f(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$, you can create a string of the form:

```
"Polynomial([a0, a1, a2, ..., an], x)"
```

This string can then be passed to python and python can interpret it as a function (e.g. using OOP).

Second alternative

- Pass the exact python code. For example if the user wants to model the polynomial function $f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$, the vba module can create a string with the python code:

```
"def f(x, a0, a1, a2, ..., an):\n return a0 + a1*x + a2*x**2 + ... + an*x**n"
```

This string can be written to a file that python can import and use as a function.

Functions

- By default you should be able to implement at least the following functions:
    i. Polynomials
    ii. Exponential
    iii. Logarithm
    iv. Absolute Values
    v. Addition, Substraction, Multiplication, Division of functions

Step 1B: Constraints

- The constraints can be modeled in a similar way, VBA will pass a string to python that will be interpreted as a constraint. For example, if the user wants to model the constraint $f(x) \leq 0$, the string could be:

```
"Polynomial([a0, a1, a2, ..., an], x)<=0.0"
```

Or something more Pythonic using OOP

```
"LessThan(Polynomial([a0, a1, a2, ..., an], x), 0.0)"
```

Partial Overview of the workflow

Consider the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & f_0(x) = 10x^2 + 5.2x + 4 \\ \text{subject to} \quad & f_1(x) = x^2 - 1 \leq 0 \end{aligned}$$

Step 1: Model the optimization problem in VBA

- The user should be able to input both for the objective function and the constraints, the type of function and the parameters. Your program should then create a text file with the following (or similar) content:

```
min Polynomial([10, 5.2, 4])
s.t.
LessThan(Polynomial([1, 0, -1]), 0.0)
```

Or

```
min Polynomial([10, 5.2, 4])
s.t.
LessThan(my_function, 0.0)
```

where `my_function` is a function that was defined in a separate file.

```
def my_function(x):
    return x**2 - 1
```

Python Module

- The python script should read the text file, interpret the functions and constraints, solve the optimization problem and write the results to a text file.
- You need to define how the text file that VBA creates should look like. E.g. the first line could be the objective function, the second line the constraints etc.
- Depending on how you write the text file the parsin of the strings will be different. You should rely on the string functions in python to parse the strings. E.g. to identify if the symbol '<=' is in the line.
- You can rely on the standard File I/O functions seen in class, in particular when reading line by line.
- Moreover you can use the function `eval` to evaluate the strings as python code.

The `eval` function

- The `eval` function in python evaluates a string as a python expression. For example:

```
x = eval("10 + 5")
print(type(x))
print(x)
```

- The output will be:

```
<class 'int'>
15
```

Example

If the text file looks like this:

```
min Polynomial([10, 5.2, 4])
s.t.
LessThan(Polynomial([1, 0, -1]), 0.0)
```

- Your code should identify in the first line that the objective is to minimize a polynomial function.
- You could have defined a class `Polynomial` that can be evaluated with the parameters given in the list. `obj_function = Polynomial([10, 5.2, 4])`
- Your code can then traverse the file line by line, adding new constraints to the optimization problem.

Results to Excel

- The python script should write the results to a text file that the VBA macro can read.
- Investigate how to read from VBA a text file, and then display the results in Excel.
- You can also investigate how to call the python script directly from VBA, this would mean that you dont need to run the python script from the command line.

API communication (Bonus)

- In a second stage you should use APIs to communicate between the two languages.
- Create a Flask API that receives as text the optimization problem, solves it and returns the results.
- The VBA macro should send the optimization problem to the API and receive the results.

How to call an API from VBA (Bonus)

```
' Set the endpoint URL
url = "http://localhost:5000/endpoint?var1=" & var1 & "&var2=" & var2

' Create the XMLHTTP object
Set xmlhttp = CreateObject("MSXML2.XMLHTTP")

' Set up the request
xmlhttp.Open "GET", url, False
xmlhttp.setRequestHeader "Accept", "application/json"

' Send the request
xmlhttp.Send

' Text
MsgBox xmlhttp.responseText
```

- If you have time you can also investigate how to send the optimization problem as a JSON object.

Deliverables

- A zip file containing everything you need to run the project. This includes the VBA module, the python script, and any other files that are necessary.
- A README.md file with the instructions on how to run the code, together with any special considerations that the user should take into account when running the code.
- Share the file using https://wetransfer.com/ and send the link to my email juan.imbet@dauphine.psl.eu with the subject "VBA and Python Project NAME+LASTNAME".
- I will provide office hours on the following dates (please send me an email in advance):
  - Thursday Dec 12 2024 - 13:00-15:00
  - Tuesday Jan 07 2025 - 10:00-12:00
- Deadline is January 10th, 2025.