3.09 Numerical Methods 1
Programming exercise

November 26, 2013

Instructions to candidates

Answering questions

Each question consists of instructions for a piece of Python code which you are to write. In each case, the question will specify both the *algorithm* to be employed and the *interface* you are to write. Marks will be awarded for all of the following:

1. Correct implementation of the algorithm.
2. Correct implementation of the specified interface.
3. Quality of commenting. The comments should be such that another NM1 student would understand the code without difficulty.
4. Logical structure of code: does the sequence of statements in the code must make logical sense.

Time limit

You may work on the problems until 1700 if you wish, however the programming exercise should take much less time than this.

Submitting your answer

You must submit your answer by uploading a .tar or .zip archive to ESESIS. This archive must contain all of the files containing your answers. I suggest you create a new folder in your home directory for this exercise. From your home directory, you can then use a command like:

```
tar cfvz answers.tgz dir_name
```

This creates the archive `answers.tgz` for you to upload. Of course you should use the name of your directory, not `dir_name`. To double check the archive before you upload it, type:

```
tar tfvz answers.tgz
```

This will list all the files in your archive for you to check.

Allowed materials

This is an open book exercise. You may use the course text, lecture notes, your own notes and any other written material which you choose.

Network access

You may use the web to, for example, access Python documentation. However, you may **not** use any communication protocol including, but not limited to, email, any chat program, posting on internet fora, or social network sites. In essence you may use the web as a reference but you may not use it to communicate with anyone during the exercise.

You must ensure that any communication programmes are shut down for the duration of the exercise.

Violation of this rule is cheating and may constitute an examination offence under college rules with very serious consequences.

1. Create a module `function.py` which contains a function `sin_cos(x)` which implements the following function:

$$f(x) = \sin(x^2)\cos(|x|)$$

2. The Taylor series for $\sin(x)$ evaluated near zero is given by:

$$\sin(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!}$$

   (a) Create a module `series.py` which contains a function `taylor_sin(x,n)`. This function should evaluate the first n entries in the Taylor series above and return the result. Count the $k = 0$ entry as the first entry. You may find the function `factorial` function from the `math` module useful.

   (b) Create a Python program `plot_sin_series.py` which uses your `series` module to plot a graph of `sin(x)` for $-\pi < 0 < \pi$. Use only the first 3 entries in the Taylor series (ie set $n = 3$).

   Your program should also plot $\sin(x)$ on the same axes for the same range of x values using the `sin` function from `numpy`. Note that the two lines will diverge as x gets further from zero. This is a programming exercise so you are to hand in the program which makes the plot, not the plot itself.

3. Write a module `linear_algebra.py` containing a function `matvec(A,b)` which takes a rank 2 Numpy array `A` and a rank 1 numpy array `b` and returns the corresponding matrix-vector product. Your programme should raise `ValueError` if the shapes of `A` and `b` are incompatible. **In answering this question, you may not make use of the `dot` function from `numpy`, `scitools` or elsewhere**.