3.09 Numerical Methods 1
Programming exercise

15 March 2012

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: the sequence of statements in the code must make logical sense.

Time limit

You may work on the problems until 1200 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 the file 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.

The Jacobi method is an *iterative* algorithm for solving matrix systems:

$$\mathrm{A}\mathbf{x} = \mathbf{b} \tag{1}$$

where A is *diagonally dominant*. A matrix is diagonally dominant if the diagonal entry on each row is larger in magnitude than the sum of all the other entries on that row. That is, for any i:

$$|A_{i,i}| > \sum_{i \neq j} |A_{i,j}| \tag{2}$$

The Jacobi method operates by splitting the matrix A into a matrix D consisting of the diagonal of A and matrix R consisting of the rest of A. That is:

$$\mathrm{D}_{i,j} = \begin{cases} \mathrm{A}_{i,i} & \text{where } i = j \\ 0 & \text{where } i \neq j \end{cases} \tag{3}$$

$$\mathrm{R}_{i,j} = \begin{cases} 0 & \text{where } i = j \\ \mathrm{A}_{i,j} & \text{where } i \neq j \end{cases} \tag{4}$$

The Jacobi method makes use of D^{-1}, the inverse of D. The inverse of a diagonal matrix is very easy to calculate, it is simply given by taking the reciprocal of each of the diagonal entries:

$$\mathrm{D}^{-1} = \begin{bmatrix} \mathrm{D}_{0,0} & 0 & \cdots & 0 \\ 0 & \mathrm{D}_{1,1} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \mathrm{D}_{n,n} \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{\mathrm{D}_{0,0}} & 0 & \cdots & 0 \\ 0 & \frac{1}{\mathrm{D}_{1,1}} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \frac{1}{\mathrm{D}_{n,n}} \end{bmatrix} \tag{5}$$

As an iterative method, the Jacobi method starts from an initial guess for $\mathbf{x}$ and calculates a series of improved values of $\mathbf{x}$. We will use an initial guess of the zero vector $\mathbf{0}$.

As the Jacobi method defines an iteration, it is also necessary to calculate when the iteration should terminate. For this, we will use the residual:

$$\mathbf{r} = \mathbf{b} - \mathrm{A}\mathbf{x} \tag{6}$$

When we start, with $\mathbf{x} = \mathbf{0}$, the residual will be equal to $\mathbf{b}$ (since $\mathrm{A}\mathbf{0} = \mathbf{0}$). We will stop the iteration when the residual has shrunk by a factor of ϵ where ϵ is a small positive parameter supplied by the user.

The algorithm for the Jacobi method is given by:

$\mathbf{x} \leftarrow \mathbf{0}$
$r_0 \leftarrow \sqrt{\mathbf{b} \cdot \mathbf{b}}$
repeat
  $\mathbf{x} \leftarrow \mathrm{D}^{-1}(\mathbf{b} - \mathbf{R}\mathbf{x})$
  $\mathbf{r} \leftarrow \mathbf{b} - \mathrm{A}\mathbf{x}$
until $\sqrt{\mathbf{r} \cdot \mathbf{r}}/r_0 < \epsilon$ or maximum iterations exceeded.

Copy the module `/numerical-methods-1/random_jacobi.py` to your working directory. It contains the routines `random_dominant`, `random_nondominant`, `random_diag` and `random_vec`, which you may find useful in testing your answers to the following as you go along.

1. Create a module `jacobi` containing a function `diagonally_dominant(A)` which returns `True` if the array `A` is square and diagonally dominant, and `False` otherwise.

   (20 marks)

2. Add a function `split_matrix(A)` to `jacobi` which returns a pair of arrays `D, R` where `D` contains is a the diagonal part of `A` and `R` is the remaining part, as given by equations 3 and 4, above.

   (20 marks)

3. Add a function `inverse_diag(D)` to `jacobi` which returns the inverse of the diagonal matrix `D`.

   (20 marks)

4. Add a function `solve(A, b, epsilon=1.e-6, N=100)` to `jacobi` which returns the solution to the matrix system $A\mathbf{x} = \mathbf{b}$ computed using the Jacobi method. `epsilon` should be used as the convergence criterion. Your routine should raise `ValueError` if A is not diagonally dominant, and `ArithmeticError` if the maximum number of iterations N is exceeded. You may find it useful to call the functions from the previous questions in composing your answer.

   (40 marks)