Appendix A

Python - A Computational Tool for Biomedical Signal Processing

A.1 Introduction

Scientific computing is a key area of research which involves in developing platforms that cater the requirements of scientific community. It often requires modeling of complex systems, simulation of scientific processes, data mining, statistical interpretation of data etc. Large arrays of data need be processed with minimum programming difficulty in short time. Although there are many high level programming languages like C, C++ etc that are suitable for general computing, there are but a few that are usable for scientific computing purposes. The major requirements for a language for scientific computing are

- The language need be interpreted and not compiled. The line by line interpretation can easily return results, a feature which speeds up research.

- The language should have a simple syntax so that less time is spent on programming and more time is spent on scientific thinking.
A.2. OPEN SOURCE TOOLS FOR SCIENTIFIC COMPUTING

- The language should support a high quality plotting library that can generate publication quality plots, preferably in Postscript format.
- There need be easy syntax for easy vector-matrix multiplication, linear algebra, Fast Fourier transforms etc.
- Greater execution speed.

Historically, FORTRAN had many features of a scientific computing language, though it did not have its own plotting library. C remained as a fast language for general computing. Being compiler based it did not become popular in scientific computing. As C did not have easy array multiplication syntax, MATLAB, a meta language, that supported a lot of good features came to picture with an immediate success. MATLAB has many disadvantages besides the extremely high cost and licensing hassles. Being a closed source program one can never ascertain how the computations are done. It does all computations with arbitrary precision. Although MATLAB has a plotting library, it does not provide truly publication quality plots. So there was a need to look around for alternatives. With developments in open source software tools like Octave, Scilab, R etc came to picture the details of which are discussed in the next section.

A.2 Open source Tools for Scientific Computing

With the arrival of Linux and internet, distributed computing became popular and open source software developers around the globe could work as a team, resulting in improved software tools. In scientific computing clones of MATLAB like SCILAB, Octave came to picture. SCILAB relied on TCL/TK. Although it contains many toolboxes for
signal processing, image processing, communication, control systems etc. it lacked robustness. Furthermore the TCL/TK GUI is still not publication quality. Despite these facts it remains popular especially in academic circles. Octave has the same syntax as MATLAB but is a very slow in execution. There has been demand for an interpreted and modular language for scientific computing with a good plotting library. And Python entered the arena at this juncture.

Python is a modular, interpreted programming language with many resemblances with LISP and FORTRAN. Being modular, programmers could integrate the existing FORTRAN libraries for linear algebra such as LAPACK(Linear Algebra Package) and ATLAS(Automatically Tuned Linear Algebra package) with Python making an open source scientific computing package called Scipy. The details of Python and scipy are in the next section.

A.3 Python, Scipy and Matplotlib

Python language that came into existence just half a decade before, has very desirable features of a scientific computing language, the major ones being modularity, speed and robustness. An enhanced Python shell called ipython can be installed along with Python. Ipython resembles the command line in MATLAB and IDL but supports more features like shell scripting functions. The package scipy can be imported into Python for scientific computing. A Python library called MATPLOTLIB, popularly called pylab was developed for computing and for generating anti-aliased, publication quality plots. The two modules scipy and pylab forms the backbone of scientific computing with Python. These modules can be imported with the commands

```python
from scipy import *
from pylab import *
```
The two commands import all the modules in scipy and pylab. Selective importing can be done to keep the code light and robust as

```python
from scipy import math
```

Pylab offers good quality GUIs in addition to publication quality plots. Fig.A.1 shows random ellipses plotted with pylab. Python together with scipy and pylab has become a stable computing platform especially for signal and image processing. They offer greater speed and precision than MATLAB or IDL.

![Random ellipses plotted in pylab](image)

**Fig. A.1.** Random ellipses plotted in pylab

There are two main Python modules for image processing. The first one is the Python imaging library(PIL) and the second is a package in scipy called *ndimage*. Both contain modules for image acquisition, processing and image writing. Medical image processing comprises of reading, filtering, classifying, segmenting and interpreting medical images such as CT image, MRI image, retinal fundus image, angiograms.
etc. Understanding the data in these images are very critical in both
detection of diseases as well as in surgery and leaves no room for errors.
Python and its image processing tools can facilitate medical imaging
as detailed in the subsequent sections.

A.4 Python for medical imaging

Both *ndimage* and *PIL* can be used with minimum amount of coding
for processing medical images as both support variety of image for-
mats like *.jpg*, *.png*, *.pgm* etc. The following code segment reads the
color fundus image “retina.jpg” from the hard disk, isolates the green
channel and displays it the pylab GUI as shown in Fig. A.2.

```
from scipy import*  
from pylab import* 
import Image 

inpu=imread("retina.jpg")  
x1=concatenate(concatenate(inpu))  
x2=x1[1:len(x1):3]
```

![Fundus image with acute diabetic retinopathy](image.png)

**Fig. A.2.** Fundus image with acute diabetic retinopathy
A.4. PYTHON FOR MEDICAL IMAGING

x3=resize(x2,(shape(inpu)[0],
    shape(inpu)[1]))
imshow(x3)
gray()

The image can be filtered for locating microaneurysms and swellings
by a variety of filters based on scipy. Fig. A.3 indicates the result of
median and Gaussian filtering as done the following code.

import scipy.ndimage as nd
subplot(121)
lapout=nd.gaussian_filter(x3,3)
imshow(lapout)
gray()
subplot(122)
medout=nd.median_filter(x3,3)
imshow(medout)
gray()

The histogram can be observed with the code given below. The bin
size can be incorporated with an argument to the hist command as,
say, hist(x,100). The grid command includes a grid on the GUI. The
savefig command saves the matplotlib GUI of the histogram in the
present working directory. The figure can be saved in .png, .pdf or.
.jpg in addition to the encapsulated postscript(.eps) format.

hist(x3)
grid("True")
savefig("hist.eps")

The resulting histogram is as in Fig. A.4.
Fig. A.3. Gaussian filtered output (left) and median filtered output (right)

Fig. A.4. Histogram of retinal image

A.4.1 Python for mammograms

Mammograms in .pgm and other formats can be read and processed as discussed in the previous section. Fig. A.5 shows mammogram processed by various filters.
A.4. PYTHON FOR MEDICAL IMAGING

![Input mammogram](image1) ![Output of Sobel filter](image2)

![Output of LoG filter](image3) ![Output of quadratic filter](image4)

Fig. A.5. Outputs of various filters for mammogram input

A.4.2 processing dicom files

Dicom files can be read and processed by the python module `pydicom`. The module can be imported as

```python
import pydicom
```

Dicom is a format that is a lot less esoteric that one may imagine. Much of it contains the patient’s information. The pathological data is contained as a $64 \times 64$ matrix which can be processed with scipy and pylab modules.
A.5 Working with Sound Files

**Wave Module**

The built *wave* module can be used for reading and writing sound waves in .wav, .raw etc. formats. The read waveforms can be easily converted into scipy arrays for further manipulations.

**Wavfile Module**

It is a more light and robust module than *wave*. Sound files can be read using the *wavfile* in scipy.io module. It supports both the read and write functions. A sound file in .wav format can be read as

![Signal from a wheezing lung](image)

**Fig. A.6.** Signal from a wheezing lung

```python
import scipy.io.wavfile as wav
z=wav.read("wheezinglungs.wav")
print z[0]
plot(z[1])
```
reads and displays the lung sound stored in the hard disk. The first element in the array z returns the sampling rate and the second array returns the values in the signal. The signal is as in Fig. A.6. The signal can be further processed using signal processing tools for feature extraction.

### A.6 Matplotlib Graphical User Interface

Matplotlib is Python library for publication quality outputs. This is invoked by importing the `pylab` module. This contains the `plot` function for continuous plots, `stem` function for discrete plots, `hist` for histograms etc. The plots are anti-aliased with \LaTeX\ fonts for axes and text within the plot. Animations of plot are possible with `pylab`. The outputs can be stored in postscript, encapsulated postscript and in pdf format. This makes integration of results with reports compiled with \LaTeX\ easily possible.

### A.7 Mayavi – Python Tool for Data Visualization

Mayavi is popular open source tool for scientific data visualization and 3-D plotting written in Python, currently hosted by `enthought`. It can be used as a stand alone application or as module that can be imported into a Python code. As a stand alone program it reads data in `vtk` format and displays in three dimensions. This make easy visualization of internal organs possible. Mayavi is used in 3-D visualization of MRI data. For using `mayavi` the Python session should be invoked as

```
ipython -wthread
```

Mayavi as a module can be imported as
import enthought.mayavi

Additional modules like scipy, pylab etc. can be imported and used along with mayavi.

A.8 Installation of Various Modules

All the major Python modules can be installed using the `yum` or `apt` utility in Linux. The .exe files can be freely downloaded and installed on Windows machines.

A.9 Conclusion

The chapter outlines the usage of Python as a stable replacement of commercial scientific computing languages like MATLAB, IDL etc. Scipy is a collection of signal processing and linear algebra routines written in FORTRAN that can be imported into Python for performing scientific computations quickly and reliably. MATPLOTLIB alias pylab offers publication quality plotting library. Modules like pydicom, ndimage etc. can be used for medical imaging. Wavfile, wave can be used for processing medical audio files. Medical data visualization can be done effectively using Mayavi.
Index

ATLAS, 198

Bi-impulse response method, 51

Central limit theorem, 43

Diabetic retinopathy, 43, 94
  microaneurysm, 43, 94, 97, 118
  nonproliferative, 94
  proliferative, 94

Differential pulse code modulation, 143, 157

Edge crispness, 129, 133

Edge detection, 27, 42, 43, 100
  canny filter, 126
  Laplacian filter, 96
  LoG filter, 96
  noise performance, 112
  Prewitt filter, 126
  quadratic filter, 112

Fingerprint, 43

Finite impulse response filter, 17, 26, 39, 60

FORTRAN, 197

Fourier Transform
  Fast, 50

Fourier transform, 17, 35
  Fast, 197

Frobenius norm, 110, 128

Histogram, 201

Impulse response, 34

Impulsive noise, 125

Infinite impulse response filter, 17, 26

Ipython, 198

Isotropy, 53, 106

Kronecker product, 32

LAPACK, 198

Linux, 197, 206

LISP, 198
INDEX

LTI systems, 25
Maclaurin series, 29
Mammogram, 202
MATLAB, 197
Matrix decomposition
LU method, 61
singular value decomposition, 61 109 120 128 130
Mayavi, 205
Mean square error, 53
MRI, 44 124
cardiac, 124

Noise
Gaussian, 114
impulsive, 44 114
Johnson noise, 43
shot noise, 43
Nyquist rate, 157

Octave, 197
Optimization, 52
conjugate gradient method, 53
Powell method, 99 107 109 126

PCM, 143 157
Postscript, 197
Prediction, 143
linear, 45
Pydicom, 203
Pylab, 132 198

Python, 132 196 198
imaging library, 199
ndimage, 199
wave module, 204
wavfile module, 204

Quadratic filter, 69
design, 51
isotropy, 33
two dimensional, 32 51

Random process, 143
Random variable, 45 143

Scientific computing, 196 197
Scilab, 197
Scipy, 198
Signal to noise ratio, 44 53 99 117 126
peak, 99 117 126
Stationarity, 17

Taylor series, 29 30
Teager filter, 98 101 110 114 118
design by least squares, 103
design by MMSE method, 106

Unsharp masking, 126 129 137

Volterra series, 96 143 147

Yellow dog updates manager, 206
About the Author

Hari V S received the B Tech. degree in Electronics and Communication from University of Kerala in 1994 and the M Tech. degree in Electrical Engineering with specialization in Communication Systems from the Department of Electrical Engineering, Indian Institute of Technology, Madras in 2006. He spent 17 years in teaching in various engineering colleges under the Institute of Human Resources Development (IHRD) and is currently Associate Professor in charge of Principal, College of Engineering, Karunagappally.

He carried out his research work leading to PhD at School of Engineering, Cochin University of Science and Technology in Quadratic Filters based on Volterra series. The areas of interest are signal processing, polynomial systems and digital communication. The author supports open source software and has a passion for programming in Python for scientific computing and in promoting \LaTeX for documentation. On a lighter side, he likes literature and uses it as an antidote for work.

Address
Punnamoottil, Manappally P O
Karunagappally, Kollam
Kerala, India – 690 574
Ph.- 91-8547463422
E-mail - hari_cec@yahoo.com