CS 251: Presentation: LaTeX and gnuplot (Outlab)

- Due: 11:55 AM 17/09
- Please write (only if true) the honor code. If you used any source (person or thing) explicitly state it. You can find the honor code on the web page.

Overview

The inlab has given you an insight into some aspects of of LaTeX and gnuplot. We hope that everyone agrees LaTeX is pretty powerful although it might look a bit of a nuisance. Let us jump into something specific and pertaining to our daily life – mostly by this point of time in the year everyone wants to make a resume. Let us use this lab to get a styled resume. There is more though. Read on.

Given a chunk of data it is efficient to represent it as a function to rather than saving all the points. The function can be further used to extrapolate and generate new parts of the function space. As many of you might know that this is called regression. Let's use gnuplot to get the job done.

A0: Including LaTeX in Inkscape figures

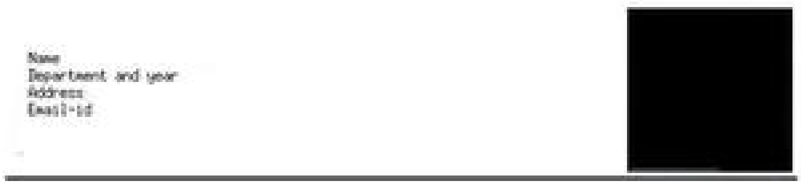
Earlier you had a brush with inkscape – a great open source drawing program. Here we connect LaTeX to inkscape – this is typical of open source programs. You will be really hard pressed to have beautiful math in your diagrams without LaTeX.

0. Open the problem statement of Lab 01 outlab and look at the image explaining refraction in page 2.
1. You will find an incomplete version of this image (`inkscape.svg`) in the support folder. Open the image in Inkscape and use LaTeX to fill in the missing parts of the image. You can add this completed image to a LaTeX file and see that this will look better than it did in the outlab of Lab 01. Filename has to be `inkscape_ssl.png`. Include this figure in your "beamer" document (see below) for Lab01.

A1. LaTeX for making resume

As an engineer, you get to not only use a racing car but fix the car (build a new one). As a code warrior, you get to not only use LaTeX but also change its behavior. Read about class (.cls) in LaTeX. This is a custom styling feature which can be programmed by the user as desired. You can find a basic overview here.

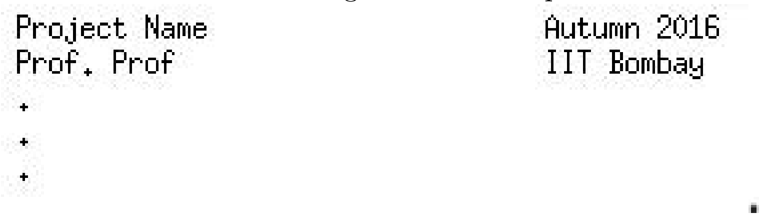
1. The task here is to develop the given `resume_ssl.cls`. Find it in the support folder and read it *carefully* to find important things about how to start off with `resume_ssl.tex`
   (a) Create a Header environment which takes in your name, program, and year (e.g. II year B.Tech), address, email-id and your photograph. The styling should be as follows:
      - Photo to the left side of the header name. The photo as `photo.jpg` and place it in the same working directory
      - Name, Degree and year, Address, Email-id should be in separate lines on the left side in parallel to the photograph, something like the image below


Name
Department and year
Address
Email-id

(this is just the header part cropped and shown). The image is just for showcase purpose, it might be different when you actually do it (fonts, margins etc.)

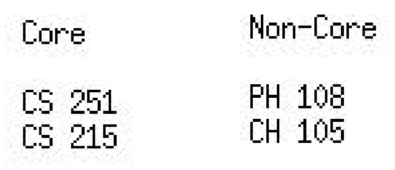
(b) The next environment is a Section environment with the following specifications:
- The heading should be always in uppercase (even if the argument is in lower case) eg. argument is **CompuTer**, the output in the pdf should be **COMPUTER**
- It should be underlined, any style of underlining can be used

(c) The next environment is a Subsection environment which should be able to handle your internships, projects etc.
- It should have 4 fields with different font styling and should be able to accommodate Project Name, Time period of project, Guide/Mentor/anything else, Place of work. These four fields can be used in many ways
- The Subsection should automatically produce 'bullet points' once the arguments are taken just by using \item tag (without using explicitly, e.g., \begin{itemize}) and their placement should be something like this example


```
Project Name                    Autumn 2016
Prof. Prof                      IIT Bombay
 •
 •
 •
```

(this is just the subsection cropped and shown). The image is just for showcase purpose, it might be different when you actually do it (fonts, margins etc.)

(d) (Extra Credit) Placing courses in the resume is a good option for sophomores so let us make an environment for the same naming it as a Course. Note: Course environment need not have arguments it can be like the inbuilt environments without any arguments (just like, e.g., \begin{center})
- The environment should place all your core courses on one side and other relevant non-core courses should be on the other side. Something like this


```
Core        Non-Core

CS 251      PH 108
CS 215      CH 105
```

(the courses cropped and shown). The image is just for showcase purpose, it might be different when you actually do it (fonts, margins etc.). Side here is left and right halves of the document. Note that there might not be a balance, you might have more core courses than non-core. (Core is used here in a figurative sense, not in the Institute definition of a core.)

(e) (Extra Credit) The last part of the resume is a mechanism to put your favourite inspirational quote (e.g., "*Experience is not what happens to a person. It is what a person does with what happens to him*" - Aldous Huxley) You should create a Quote command which take your favorite quote and its author as the only arguments and formats it as below

- It should have a special styling and different font size than the other things in the document (none of these should be given as arguments), should be centered and should be in " " (quote unquote)(these should not be given as part of the quote) followed by the author's name in a different style. Feel free to use a quote from your favourite Indian language.

2. All the four environments and one command should be in the same file `resume_ssl.cls` which when used in your .tex (`resume_ssl.tex`) should do the required actions.
   Note: Here is one more restriction. In all the cases, empty arguments must be handled elegantly. That is, it should not throw a compilation error if the user has missed one of the arguments. The format should be realistic: suppose your environment has two arguments which results in two lines. If one argument is missing you should not produce a space or a new line for the missing argument.

3. Using the above generated `resume_ssl.cls` write `resume_ssl.tex` (maximum 2 pages and uses the `resume_ssl.cls` class file) with (some of these are extra credit and can be omitted)

   - Header having details mentioned in the environment and properly sized photograph of one of the person whose resume is being made
   - A section containing academic achievements
   - A section about projects & internships. One project must have a Subsection described with 3 bullet points
   - A section with the courses done using your Course environment
   - A section having your favorite quote using the Quote command
   - Fill the rest of the resume with relevant content such as extra curricular activities.

4. You need to submit `resume_ssl.cls` and `resume_ssl.tex` . The tex file should be compiled using `pdflatex` using a terminal.

A2. LaTeX for efficient report generation

One of the major uses of LaTeX is to generate reports and research papers. But these reports can be extremely huge and writing them in one file will make the file very large. Here, we will learn a way to split a report into multiple parts to make it easier to work on. (This is kind of similar to the modularization of large C++ projects).

0. Refer to 3 algorithms in your Sahni book (inorder binary tree traversal, quicksort, shortest path). Write a textual description of the first two. Write the quicksort algorithm in algorithm style. cite Knuth vol 1 for inorder traversal, Hoare for quick sort and Dijkstra for shortest path. Use Google scholar (for example) to find the right way to cite the paper or book.

1. In each of the section when you are writing the summary `cite` the papers corresponding to that section using BibTex. Include the references (at the appropriate places). Use the file name `biblio.bib`. Compile main.tex to produce main.pdf as usual.

2. Now that you have your report (i.e. the so called big document), we are going to split this report into multiple parts. Make a new `part{i}.tex` for each i of the n sections. (That is, each of these files will have the content corresponding to a relevant section.)

3. Rewrite `main.tex` file by removing all content from it and introduce all `part{i}.tex` (Here i is the number of the part in serial order) files in it as `\include`. Now compile the `main.tex` file using `pdflatex` to get the `main.pdf` file. Your new `main.pdf` file should be identical to the previous one.

4. Now comes the major part of the task: Modify `main.tex` file so that it takes an input from the user and only the files mentioned in the user input are compiled and added to `main.pdf` file. If the user inputs "`all`", all the files have to be compiled and added to the `main.pdf` file.

The main idea here is conditional compilation (and somewhat different from the `makefile` concept.) Also to note that we won't be looking too seriously in your RSA content, only the form.

A3. LaTeX as programming language

LaTeX is a (so-called) Turing complete programming language — this simply translates to: theoretically it is as powerful as, say, C++. We will look at a special feature of LaTeX which allows it to be used like a normal programming language.

0. Read about the LaTeX commands `\newcommand` and `\newcount` (in the light of arithmetic here, in a naive translation *newcommand = function* and *newcount = integer*) and Fibonacci numbers.
1. Write a LaTeX file `fibonacci.tex` which when compiled using `pdflatex` asks the user to input a number `n` and then produces a `fibonacci.pdf` file which contains the first n Fibonacci integers. Function call : \fibonacci{number}
2. Write a LaTeX file `factorial.tex` which produces the file `factorial.pdf` having the factorial of the number n taken from the user input as mentioned in the previous part. Function call : \factorial{number}
3. **CHALLENGE:** Similar to the above two questions, write a file `ackermann.tex` which gives the value of the Ackermann function for two given values (again the values are to be taken from the user input). You can read about the Ackermann function on Wikipedia. Do not attempt this part until you have completed all the questions in the lab

B. Gnuplot

The data we come across everyday is most of the time governed by a known distribution along with predictable noise added. We shall use gnuplot's function fitting to get the functions. We will provide the data in the support area as `data.dat`.

0. Read about standard noise distributions (examples are poisson, gaussian.)
1. Plot the given data and try to make a wise guess of a probable distribution going into it.
2. The noise is a well-known noise distribution which was scaled by a factor of `0.05` and is added to the actual distribution. Currently `D(x) = A(x)+0.05*N(x)`. Most of the times in nature we will be finding noise distributions which have the mean of the Noise being zero, keep this in mind before you make an assumption in models. You have to report the distributions `A` and `N` along with the free parameters in them (e.g.., like λ in case of poisson distribution)
3. Your fit should have a cumulative error less than **e-10** and the code should be in `fit.gnu`
4. Apart from the above functionalities `fit.gnu` should generate a plot which shows the plot of original data vs the fit obtained and this plot has to be saved a `plot.png` (it should have relevant legend, axis naming and title)
5. Write the distributions and the free parameters obtainied after curve fitting in gnuplot in `report.txt` and submit both `fit.gnu, report.txt`

Submission Guidelines

1. When you submit, please document individual percentages such as Student 1: 80%, Student 2:100%, Student 3:10% in the readme.txt. In this example, the second student will get full marks (10/10) and the first student will receive 8/10.
2. Do include a readme.txt (telling us whatever you want to tell me). (The reflection essay is not required for inlab, but is required for outlab.) Do include group members (name, roll number), group number, honor code, citations etc.
3. The folder and its compressed version should both be named `lab06_groupXY_outlab` for example folder should be named `lab06_group07_outlab` and the related `tar.gz` should be named `lab06_group07_outlab.tar.gz`
4. The submission folder should contain 5 sub-folders `taskA0`, `taskA1`, `taskA2`, `taskA3` and `taskB`
5. The subfolder taskA0 should contain the inkscape_ssl.png file.
6. The subfolder taskA1 contains 3 files resume_ssl.cls, resume_ssl.tex and the photo.jpg file.
7. The subfolder taskA2 should contain all the part{i}.tex, main.tex, biblio.bib and main.pdf files.
8. The subfolder taskA3 should contain the fibonacci.tex and factorial.tex files. If you have done the extra credit also, ackermann.tex file should also be included.
9. The subfolder taskB should contain fit.gnu, plot.png, report.txt
10. Your submission folder should look something like this:

```
lab06_groupXY_outlab
 |__taskA0
 |   |__inkscape_ssl.png
 |__taskA1
 |   |__resume_ssl.cls
 |   |__resume_ssl.tex
 |   |__photo.jpg
 |__taskA2
 |   |__main.tex
 |   |__biblio.bib
 |   |__main.pdf
 |   |__part1.tex
 |   |__part2.tex
 |   |__..
 |__taskA3
 |   |__fibonacci.tex
 |   |__factorial.tex
 |   |__ackermann.tex, if done
 |__taskB
 |   |__fit.gnu
 |   |__plot.png
 |   |__report.txt
 |__readme.txt
```

How We will Grade You [60 + 10 Marks]

Extra credit points are available to you only if you get the basics right.

1. Task A0 [**5 Marks**]
2. Task A1 [**15+10=25 Marks**]
   - Each of the environment/command with relevant usage in .tex (5 marks for each) : 25 Marks
3. Task A2 [**15 Marks**]
   - Correct implementation of task : 5 Marks
   - Encryption algorithm and minimum 3 sections : 5 Marks
   - Bibliography : 5 Marks
4. Task A3 [**15 Marks**]
   - Fibonacci : 7 Marks
   - Factorial : 8 Marks
   - Ackermann [CHALLENGE]
5. Task B [**10 Marks**]
   - Prediction of A + params : 3+1 Marks
   - Prediction of N + params : 3+1 Marks
   - plot.png : 2 Marks