Week 2: Report Writing with Lyx, C programming with pointers

Dr. H. Ramachandran
EE Dept
IIT Madras

August 10, 2015

Abstract

In this lab session, you will learn about the technical wordprocessor, LyX. You will use LyX to create a simple report. You will also try out some of the things you learned about last week.

Contents

1 This Week's Assignment

- Learn to use the *LyX* editor described in this assignment.
- Study the structure of the given example article. Your scientific reports and articles should follow this format, **including all submissions in this course.**
- Program the C assignment and include the source code in a LyX report as LyX-code. Include output obtained in the LyX report.
  - Note that the frequency counts of different word lengths should be presented in a table.
  - The code taken from the stub given below should be grey in blue, while the code you have entered should be black. For example:

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc,char **argv){
  if(argc < 2){
    printf("Usage ./a.out n1 n2 ...");
    exit(1);
  }
  int sum=0,i;
  printf("The sum of ");
  for(i=1 ; i<argc ; i++){
    sum += atoi(argv[i]);
    if( i==1 )
      printf("%s",argv[i]);
    else
      printf(" and %s",argv[i]);
  }
  printf(" is %d\n",sum);
}
```

- Zip the LyX files and submit the archive to the moodle site.

2 The C Program

Please review the material presented last week and posed on the moodle site.

The task is to create a C program that goes through a text file and prints the statistics of number of words with different lengths. We will consider words of length of 3 to 10 for our statistics.

The stub code for this program is as follows:

```
#include<stdio.h>
#include<stdlib.h>
#define MAXLENGTH 512
int main(int argc, char **argv)
{
  /* Program expects a filename. Check that argument was passed */
  if(argc != 2){
    printf("Usage ./a.out <filename>");
    exit(1);
  }
  /* Open file while checking for existence */
  FILE *fp = fopen(argv[1], 'r');
  if(fp == NULL){
    printf("File could not be opened");
    exit(2);
  }
  /* Read in lines from file and process. Note the use of fgets() which is */
  /* more secure than gets() */
  char buf[MAXLENGTH];
  /* also allocate variables to hold counting information */
  while(fgets(buf, MAXLENGTH, fp)){
    \\Fill in the counting logic.
  }
  /* Here print out the results of your analysis */
}
```

The program reads each line in an char array. We have to go through the array and check for word separators i.e. ' ', '\t', ',' . **You should not use functions from the string library to do the assignment.**

2.1 C assignment 1

From your CS1100 classes you know to index char arrays by using buf[i]. You have to loop through the whole buffer array until you encounter a '\0' and calculate the number of words of different lengths.

- Complete the above program using this logic, without the use of pointers.
- Create a LyX report explaining the purpose of the code and include the code itself using the "LyX-code" style.

The code should be well commented.

2.2 C assignment 2

2.2.1 Arrays as implicit pointers

When you type something like buf[i], you are 'indexing' into the array buf. Another way to think about it is you are trying to access the char in buf which is at an address i. So if i is that index, then what is "buf"?. "buf" is a location in computer memory where the char array buf starts. So the buf array name is actually an address in your computer memory. C thinks that your whole computer memory is just an array of bytes. C also implements the concept of types and sizes which allows us to abstract the sizes of chars, integers, floats and doubles. So C is basically doing the following with your arrays.

- Creating a block of memory inside your computer.
- 'Pointing' the name buf at the beginning of this block.
- When you index buf[i], it is actually using the size and type of the variable and obtaining the data present at i distance away from the memory buf is pointing to.

2.2.2 Pointers

A pointer is simply an address pointing somewhere inside your computer's memory, with a type specifier, so that you can get the data of the right size. When you opened the file using a file pointer, you have a file pointer pointing to the start of the file.

2.2.3 Pointer syntax

type *ptr A pointer of type named ptr.

***ptr** The value of the data ptr is pointing to.

***(ptr+i)** The value of data at (whatever ptr is pointed at plus i)

&data The address of "data"

- Now that you know how pointer arithmetic works, do the above assignment using pointers. You can use `*(ptr + i)` to index the ith char in the array or you can increment pointer `(ptr++)` and point to the next char in the array. Even though it appears from the above discussion that pointers and arrays are the same, they are not. Try printing sizeof(array) and sizeof(arraypointer) to see one of the differences. Try figuring out more differences.
- Create a LYX report explaining the purpose of the code and include the code itself using the "LYX-code" style.

The code should be well commented.

2.3 Extra Credit assignment

Now that you have the statistics relating to the length of words, create a text histogram of the statistics as shown below with your C program.

```
7|
6| *
5| * * *
4| * * * *
3| * * * *
2| * * * * *
1| * * *
0+--------------
```

Better still, have the code generate the following:

```
 |     *
 |     *  *  *
 |  *  *  *  *  *
 |  *  *  *  *  *
 |  *  *  *  *  *  *
 +-----------------------
 0  1  2  3  4  5  6  7
```

We will learn better ways to do plotting, but text plots are actually very useful since you can generate them in systems that lack graphics support.

- Create a LYX report explaining the purpose of the code and include the code itself using the "LYX-code" style.

The code should be well commented.

3 Submitting the assignment

Submit the LYX files along with the figure file in a zip archive to the moodle site.

4 LyX

4.1 Introduction

One of the commonest tasks any student faces, and indeed any engineer faces, is that of writing a clear, concise and complete report. This week, we introduce a tool that makes technical report writing easy. Not only easy, but extremely elegant and professional in appearance.

The tool in question is called TEX. TEXwas invented by the famous Stanford mathematician, Donald Knuth, to make mathematical typesetting simpler. He studied the arcane intricacies of professional typesetting and discovered the rules that govern the spacing, size and arrangement of characters on a page that produce the appearance we find in textbooks and other professionally produced manuscripts. And he put all these rules into a program, gave it a programming interface and released it to the scientific and engineering community.

It is not easy to use TEX. To create a document, you actually have to write a program. The program describes the logical structure of your report. When compiled, the result is a professional looking layout of the report. Clearly not everyone is competent enough to create complex reports using TEX. So a set of routines (or macros) were developed and called LATEX. Using these macros, writing reports became very easy, and indeed a generation of scientists and engineers have used LATEX to write their papers. Most scientific and engineering journals accept LATEX documents and publish "templates" (basically more *macros*) that put the articles in the format they require.

But even LATEX requires that you write a program. A simpler program, true, but still a program. You do not get to see what you have created till you compile the program using the "latex" command. In this day of WYSIWYG, that is very challenging for most users. So, a set of visual editors have developed, that look like Microsoft Word or Wordperfect, but which save the file as LATEX documents. The advantage of this approach is that you get the ease of use that a visual editor provides, while retaining the unmatched quality that LATEX offers. LyX is one such visual editor. In this week's session, we will learn to use LyX and will learn to create a simple report.

All reports in this course must be submitted as LyX documents.

Note: This document itself (and all other lab manuals for this course) have also been written in LyX.

All said and done, LyX is only a convenience. You will have to learn LATEX and quite soon. This is because the *Python* assignments will include the need to put mathematical expressions in plots and that is done by including TEX expressions. By your final year, you should all be experts in creating reports with LATEX.

4.2 WYSIWYG versus Typesetting Languages

What differentiates a visual editor such as *Microsoft Word* from LyX?

When we create a document in *Word*, we specify everything about the document. Essentially, the editor provides us with a blank page, and some tools with which to write on that page. All editorial control is with us. We can use whatever font we wish, in whatever colour and make it underlined, bold and italics at the same time. This is similar to the power that *assembly language* programming gives us.

When we create a document in LyX however, we specify relatively little about the layout. We just type. The font selection and style is mostly determined by the environment we choose. A section heading has a font that goes with it. It has a style as well.

The advantage of the WYSIWYG approach is that one can create arbitrary documents. We can put the table of contents in the middle of page 3, with page numbers on the left.

The advantage of the LyX approach is that we make very few typesetting decisions, and all our decisions are related to our own material. Thus, document creation can be extremely fast.

4.3 Starting LyX

LyXneeds to be installed on your system if it is not already there. It is, ofcourse, installed on the department lab machines. However, on your hostel PC, you will need to install it. The instructions for this are given in Appendix **??**.

Once installed, LyX can be launched from the command line by executing the command:

```
$ lyx [filename] &
```

The "$" is the shell prompt (it depends on your choice of shell prompt, and may look different for each person). "lyx" is the command name. "[filename]" is the (optional) filename. If no filename is given, LyX starts with an empty window. The "&" is a directive to the shell that the program should be started as a child process and control should return to the shell. This permits you to enter further commands in that console session.

For the purpose of the next few paragraphs, start LyX as

```
$ lyx report1.lyx &
```

This starts LyX with an empty file called *report1.lyx*. We shall put things into this report as we go along.

The LyX window is very similar to that of other GUI wordprocessors. There is a set of drop-down menu buttons. Below that is a list of icons that offer a quick way to perform common actions. Below that is the text window itself. At the bottom is the status line which gives information about the LyX editing session.

However, the LyX window is quite different from your standard wordprocessor too:

- There is no font selection menu.
- There is no point size selection menu.
- There is no ruler.
- There are no tab stops.

Instead there is an all important menu, just below the *File* menu button. Click on the down arrow beside that menu. You will see a large number of "environments" to choose from.

The reason for the absence of the font and size menus is that there are clear rules about which font and which size to use where. The environment pretty much dictates the choice of font and size. The "ruler" and "tab stops" are made obsolete by the presence of environments.

4.4 Document Settings

First we set up the report as a report. Click on *Document→Settings*. This opens up a form with many complicated settings.

- The most important of these is the "Document Class" field. Click on the $\updownarrow$ arrow and select the "article" class (this is actually the default class so you didn't need to do anything). The "Class" setting sets all the major features of the document. It selects which macros are predefined, and largely determines the layout of your document.
- Click on "Fonts" and select "12" for "size". By default the font size is "10", which is too small for reports. "10" point font is used in books, and is the standard for the publishing industry. For reports, "12" point is much more readable.
- Click on the "Page Layout" and select "A4" for the paper size.
- Click on "Page margins", and set the margins to 1.25 cm on all sides. This makes better use of the page and does not waste paper.
- Select "OK" to close the *Document* form.

The "article" class is used in most scientific and engineering communications. For example, articles submitted to scientific journals use the "article" style. In this style, different sections start on the same page. This is unlike the "book" style, for example, where different chapters start on fresh pages. The article style, in fact, does not have chapters. It starts with sections, sub-sections, etc.

4.5 Adding the Title, Author, Date and Abstract

The first thing to do when creating a report is to put a title, add your name as author and add an abstract of what the report is about. A date is added automatically. Go to the "environment" drop-down menu and select title (alternatively, press *Alt-p t*, i.e., press the *Alt* and *p* keys simultaneously, release, and press the *t* key). Type in a title for the lab - put in the week corresponding to this assignment, and the subject. Press the *Enter* key. The *Enter* key separates paragraphs, just as it does in *Word*. Any environment that consists of a single paragraph is terminated by this key. So, when you press the *Enter* key, LyX takes you out of the *Title* environment and puts you in the default *Standard* environment.

Now select the "Author" environment (*Alt-p Shift-A*). Enter your name, roll number and department. To put these one under the other, use *Ctrl-Enter*. *Enter* would take you out of the

"Author" environment. *Ctrl-Enter* inserts a line break while keeping you in the same environment. Press *Enter* when done.

Finally, select the "Abstract" environment (*Alt-p a*). Type in the text on page 13.

4.6 Adding an Introductory, Theory Section

Start a section (*Alt-p 2*), and name it "Introduction". Under it enter the text on page 13.

4.6.1 Handling Mathematical Equations

LaTeX is at its best when handling mathematical equations. Using LyX, you can get those perfect equations with relatively little effort. There are two ways of entering equations. The first is to use the menus. The "math" submenu in the "insert" menu contains everything you need. The only problem is that it is clumsy, and only suitable for very beginning users. Far better is to use the keyboard. The *Alt-m* key sequence gives you pretty much everything you need to create equations. Let us try to create the following equation:

$$\frac{\partial A}{\partial z} + v_G \frac{\partial A}{\partial t} + iD \frac{\partial^2 A}{\partial t^2} = \gamma |A|^2 A \tag{1}$$

1. First enter the "Descriptive Math Mode" by pressing *Alt-m d* which starts an equation on a separate line.

2. The terms on the left side involve fractions. A fraction is entered by typing *Alt-m f* ("f" for fraction). To enter the ∂ symbol, type *Alt-m p* ("p" for partial). So type

   Alt-m d Alt-m f Alt-m p A <down arrow> Alt-m p z <space>

   The <down arrow> takes you to the denominator field, while the <space> leaves the fraction and allows you to enter the next term.

3. The second term involves a subscript. This is done by typing "_". So the second term is entered as:

   + v_G Alt-m f Alt-m p A <down arrow> Alt-m p t <space>

   The "v_G" entry creates v_G.

4. The third term involves superscripts. This is done by typing "^". So the third term is entered as follows:

   + iD Alt-m f Alt-m p^2 A <down arrow> Alt-m p t^2 <space>

   Notice the "Alt-m p^2" and the "t^2" entries. These create the second derivitives.

5. The term on the right involves a Greek letter and vertical bars. These are entered as follows:

= *Alt-m g g* |*A*|^2 *A*

Here the "Alt-m g" sequence selects the Greek keyboard, where "abcde…" become "αβχδε…". The vertical bar is just directly typed in as seen above.

6. Finally, we want to give the equation a number. By default, LYX does not number equations. If you want to add a number to an equation, just put the cursor into the equation and type *Alt-m n*. The equation number is automatically generated, and is guaranteed to be in proper sequence, with proper respect paid to style. If you want to remove an equation number, just type *Alt-m Shift-n*.

   However, the only real reason to number an equation such as Eq. (1) is to refer to it in the text. In that case, we can't just add a number, we have to give that number a meaningful label. This is done by placing the cursor in the equation, and typing *Alt-i l*, which opens up a dialog box where you can give the name of the label, say "eq:maineq" (by default, LYX will put "eq:" as part of an equation label to keep it from being confused with a section label or a figure label or any other label). Once you have done that, you can refer to that equation elsewhere by typing

   Eq. *Alt-i r* and selecting "eq:maineq"

That is pretty much it. There is much more you can do, like creating matrices, integral signs etc. But the essence of the math mode in LYX is what we just did. But look at the result (type *Alt-v v*) and see the quality of the typesetting that we have painlessly obtained. The *Alt-m* keyboard is summarized below for quick reference:

LYX	Description	LATEX	LYX	Description	LATEX
Alt-m f	fraction	\frac	*Alt-m s*	square root	\sqrt
Alt-m u	$\sum$ symbol	\sum	*Alt-m i*	$\int$ symbol	\int
Alt-m d	start eqn on fresh line		*Alt-m m*	start eqn inline	
Alt-m p	∂ symbol	\partial	*Alt-m r*	$\sqrt{}$ symbol	\root
Alt-m n	add eqn number		*Alt-m S-n*	remove eqn number	
Alt-m 8	∞ symbol	\infty	*Alt-m g*	Greek keyboard	\alpha etc

4.7 Enumerated Text

Very often in reports, we have to present information as a list. For example, in the text given in the next section, the four algorithms are described in an enumerated list. The way to do this is to use the "enumerate" environment, or *Alt-p e*. This puts a number in front of every paragraph. Often, we require a sub-list within a list (again see the example text below). To increase the "depth" of the enumeration, click on the "increase environmental depth" icon. This will create a sub-list, enumerated by (a), (b), etc.

It will not be required in this assignment, but we sometimes need to continue text in a second paragraph within a single enumeration. For an example, see item 6 of section 4.6.1. When we press <*Enter*> to start the next paragraph, a new number gets added, as in the below example:

First attempt:

1. A sample line. Ended by pressing <*Enter*>
2. The next para corresponds to item 2.
3. This para corresponds to item 3.

To prevent this happening, click on the "increase environmental dept" icon. This will still put a number, but it will look like (a):

Second attempt

1. A sample line. Ended by pressing <*Enter*>
    (a) The next para corresponds to item 1(a).
2. This para corresponds to item 2.

Now, change the sub-environment to "standard" (*Alt-p s*). Type in the line. When you now press <*Enter*>, you go back to the outer nesting level, but without enumeration:

Third attempt:

1. A sample line. Ended by pressing <*Enter*>
   The next para corresponds to text under item 1.

This para is in standard text.

To get it all right, type *Alt-p e* once again in the next paragraph:

Final attempt:

1. A sample line. Ended by pressing <*Enter*>
   The next para corresponds to text under item 1.
2. This para is item 2.

With a little practice, lists and enumerations are extremely convenient and easy to do under LYX.

4.8 Typing in the Introduction

Type in the text on page 13 following the above guidelines.

4.9 Entering the Experimental Details

Select the "section" environment (*Alt-p 2*) and give a heading. Press *Enter*. Now add text below to describe the simulation runs.

Further sections can be added as needed. To add a new section at any point, just press *Enter* (this is the universal means of starting a new environment), and select the "section" environment (*Alt-p 2*). Numbering is automatically adjusted to keep the sections numbered appropriately. To add a subsection, such as the one below for adding a table, select the "subsection" environment.

4.10 Adding a Table

The main thing that the report has is a table containing the measurements. Insert a table via *Insert→float→table.* The effect of this command to insert a place for a table to be added. Add a caption by typing in the space after "Table #:". Press *Enter.* Now insert the table itself by clicking on the table icon, or via *insert→tabular material.* Type in the data from Table 1.

To enter the numbers, go into Math mode (*Alt-m m*) and then type in the numbers. The $\times$ symbol is obtained by typing "\times", or selecting it from the "math panel".

You will notice that the inserted table is "left aligned" rather than centered. To correct this, you have to change the alignment. Do this by placing the cursor just outside the table and selecting *Edit→paragraph Settings*, and then clicking on "centered" in the paragraph form. By default, LyX works in "block" mode, which creates justified text. For objects such as tables and figures, this usually needs to change to "centered", which is what we did above.

A peculiarity of the "floating" table is that LyX decides where to place it. LyX, or rather TeX, has a very sophisticated set of algorithms that decides hyphenation, line- and page-breaking and float placement. By and large, you should not tinker with these algorithms. In almost every situation, Knuth's default arrangement is superior to our taste. We are not professional typesetters, and our tinkering rarely produces an improved document. There *are* ways to force our preferences on LyX, but these are to be used with care.

4.11 Adding Figures

The table is also plotted, as shown in Fig. 1. To create a figure, you must have a graphic file. Download *fig1.ps* from the site. Insert a floating figure using *Insert→float→figure*. Insert a label in the caption using *Insert→label*. Type in a caption. Press *Enter*, and insert the file using *Insert→graphics*. The figure is now inserted. It may or may not show in LyX itself - that depends on the graphics rendering in your version of LyX. But it will show up in the compiled output.

4.12 Typing in the Results and Discussion Section

Create Table 1 and Figure 1, and type in the text on page 14.

4.13 Creating the output

Having created the report, you need to compile and view it. That is done by *View→pdf* (or *Ctrl-x g*). LyX generates a *pdf* file, which can be viewed by *xpdf.* LyX automatically calls the pdf viewing application and shows you the output. Note that you can do this at any stage, not only at the end of report creation.

A printable version of the output can be obtained by *File→Export→postscript.* If you want to get *pdf* output instead, use *File→Export→pdf.*

5 A Sample Report

Below is a report by a student on a minor project study he had done. See if you can understand how to create the math equations in that report.

5.1 Abstract

This report presents a study of various split-step fourier methods applied to the problem of wave propagation down a dispersive, optical fibre. It is found that the algorithms that centre the dispersion and the nonlinearity computations about each other converge much better, i.e., the cumulative error scales as Δz^4. Of such algorithms, the algorithm that takes a first half step (and last half step) of dispersion is found to be superior to the algorithm that starts and ends with nonlinearity half steps.

5.2 Introduction

This report presents four different algorithms to solve the following equation

$$\frac{\partial A}{\partial z}+i\frac{\partial^2 iA}{\partial t^2}=|A|^2A \qquad (2)$$

which appears in the theory of optical waveguides. In this equation, z represents position in the "wave frame", and both z and t have been scaled to eliminate the constants usually present. Here, A is the complex amplitude of the transverse Electric field of the wave.

Equation (2) is a nonlinear partial differential equation (PDE) of the hyperbolic, and is therefore difficult to solve. In the absence of the nonlinear term (the term on the right), Eq. (2) is a wave equation that can be solved by fourier methods. In the absence of the time derivitive term, Eq. (2) reduces to an ordinary differential equation, though a nonlinear one. Such are easy to solve using a standard differential equation solver such as the fifth order, adaptive step, Runge Kutta solver.

To make progress, we consider solving the equation over a very small step in z. It is easy to show that the two terms are additive in their effect in this limit. That is to say, first solve the linear equation (which represents "dispersion") from z to $z+dz$. Use that as initial condition and solve the nonlinear ODE from z to $z+dz$. In the limit of dz going to zero, this is equivalent to solving the full equation exactly.

In this study, we try four different ways of carrying out this approach:

1. Perform the following steps, using the result of each as the initial condition of the next.

   (a) Solve the linear equation from z to $z+dz$.

   (b) Solve the nonlinear ODE from z to $z+dz$.

   (c) Solve the linear equation from $z + dz$ to $z + 2dz$.

   (d) etc.

2. Perform the following steps:

   (a) Solve the nonlinear ODE from z to $z + dz$. Save the result.

   (b) Solve the linear equation from z to $z + dz$. Save the result.

   (c) Average the results of (a) and (b) to obtain the initial condition for the next step.

   (d) etc.

3. Perform the following steps, using the result of each as the initial condition of the next.

   (a) Take a step of $dz/2$ first, on which the nonlinear ODE is solved.

   (b) Now solve the linear equation from z to $z + dz$.

   (c) Solve the ODE from $z + dz/2$ to $z + 3dz/2$.

   (d) Solve the linear equation from $z + dz$ to $z + 2dz$.

   (e) etc.

   (f) Solve the ODE from $L - dz/2$ to L.

4. Perform the following steps, using the result of each as the initial condition of the next.

   (a) Take a step of $dz/2$ first, on which the linear equation is solved.

   (b) Now solve the nonlinear ODE from z to $z + dz$.

   (c) Solve the linear equation from $z + dz/2$ to $z + 3dz/2$.

   (d) Solve the ODE from $z + dz$ to $z + 2dz$.

   (e) etc.

   (f) Solve the linear equation from $L - dz/2$ to L.

5.3 Results and Discussion

The four algorithms described in the Introduction were implemented in *C*. The simulation was carried out for a fixed length of waveguide, using different dz. As this particular choice of coefficients is theoretically tractable, the exact solution was also known.

Table 1 presents the error in each algorithm, for different dz values. Figure 1 presents the same data in a log-log plot. Each series was also fitted to a power law fit of the form

$$a\,dz^b$$

and the corresponding best fit a and b were obtained. These are indicated in the legend in the figure.

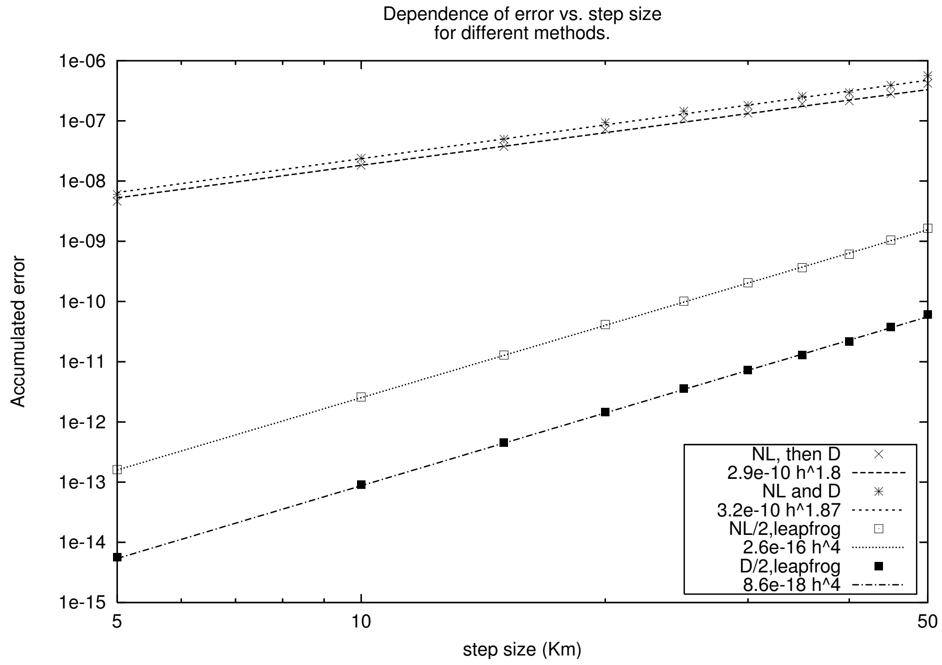
All the algorithms performed better as dz is reduced. This was expected, since the algorithms are all exact in the limit of dz going to zero. It is clear from the results, however, that the first two algorithms are inferior in that the error improves slowly as dz is decreased. The third and fourth algorithms improved as dz^4, while the first two improved only as dz^2. This is a clear indication that the interleaving of nonlinearity and dispersion manages error in a much better fashion.

While both algorithms (3) and (4) scaled the same, there was a clear two orders of improvement in error when using algorithm (4).

Table 1: Cumulative error due to different split-step algorithms for an optical link of 100 Km. Algorithm 1 calculates the contribution from nonlinearity (NL) first, and then uses the result to compute the contribution from dispersion (D). Algorithm 2 calculates NL and D based on the original signal. Algorithm 3 computes NL for a half step, and then computes D and NL, each centered about the other, with a final half step of NL. Algorithm 4 computes D for a half step, and then computes NL and D, each centred about the other, with a final half step of D.

dz (Km)	ε_1	ε_2	ε_3	ε_4
10	1.824×10^{-8}	2.393×10^{-8}	2.589×10^{-12}	9.099×10^{-14}
20	7.17×10^{-8}	9.396×10^{-8}	4.150×10^{-11}	1.467×10^{-12}
30	1.341×10^{-7}	1.812×10^{-7}	2.052×10^{-10}	7.260×10^{-12}
40	2.137×10^{-7}	2.959×10^{-7}	6.111×10^{-10}	2.143×10^{-11}
50	4.206×10^{-7}	5.651×10^{-7}	1.643×10^{-9}	6.069×10^{-11}
60	4.229×10^{-7}	6.063×10^{-7}	2.662×10^{-9}	8.902×10^{-11}
70	4.930×10^{-7}	7.524×10^{-7}	5.306×10^{-9}	1.768×10^{-10}
80	6.545×10^{-7}	1.000×10^{-6}	1.033×10^{-8}	3.636×10^{-10}
90	9.507×10^{-7}	1.500×10^{-6}	1.813×10^{-8}	7.003×10^{-10}
100	1.500×10^{-6}	2.200×10^{-6}	2.830×10^{-8}	1.235×10^{-9}

Figure 1: Scaling of Error for different split-step algorithms. The points correspond to the data in Table 1. The lines correspond to best power-law fits as indicated in the legends. As can be seen, algorithms 1 and 2 scale as dz^2, while algorithms 3 and 4 scale as dz^4.



The difference between the two algorithms is merely that the third algorithm uses a first and last half-step of nonlinearity, while the fourth algorithm uses a first and last half-step of dispersion. It is clear that the nonlinear term is a delicate term that is susceptible to error if it is carried out in a “non-centered” manner. Thus it always has to be sandwitched between dispersion computations.

To conclude, it is clear that not all split-step fourier methods are equal. Great care has to be taken to implement the algorithm correctly. The reward is an enormous reduction in computational time. For example, if an accuracy of 10^{-12} is required, algorithms 1 and 2 would require extremely small steps of the order of 0.06 Km to hope to achieve the result. Algorithm 3 would require a step size of between 5 and 10 Km. But algorithm 4 would require a step size of 20 Km, thus reducing the computation by a further factor of 3.

To conclude, it is clear that not all split-step fourier methods are equal. Great care has to be taken to implement the algorithm correctly. The reward is an enormous reduction in computational time. For example, if an accuracy of 10^{-12} is required, algorithms 1 and 2 would require extremely small steps of the order of 0.06 Km to hope to achieve the result. Algorithm 3 would require a step size of between 5 and 10 Km. But algorithm 4 would require a step size of 20 Km, thus reducing the computation by a further factor of 3.