Liceo Scientifico Statale M.Grigoletti, Pordenone

Modulo CLIL sui Numeri Complessi

Lezioni preliminari

Luciano Battaia - Mariateresa Esposito

Una serie di lezioni in seconda lingua (inglese), richiede anche una accurata preparazione degli allievi sul lessico specifico, ancora di più per il tipo di studenti destinatari (terzo anno di Liceo Scientifico) che si trovano per la prima volta ad essere coinvolti in un progetto di questo tipo. Le lezioni vere e proprie sui numeri complessi sono state, per questo motivo, precedute da:

- Interventi, di durata non superiore ai 15 minuti ma prolungati per un lungo arco di tempo (un mese), per apprendere i termini del linguaggio specifico. Questi interventi sono stati svolti solo dal docente di matematica.
- Una lezione in compresenza con l'insegnante di Lingua Inglese, su un argomento scientifico di carattere generale, avente lo scopo di facilitare i processi comunicativi e le dinamiche di gruppo.

Oltre a questo si è naturalmente proceduto ad un ripasso degli argomenti di matematica, trattati nel biennio, e la cui conoscenza è ritenuta indispensabile per affrontare con tranquillità il nuovo modulo. In questo ripasso è stata usata saltuariamente la lingua inglese, accanto alla madrelingua, per rendere l'impatto con la lezione in lingua straniera vera e propria meno traumatico possibile.

Questa commistione delle due lingue, su un argomento già noto, favorisce anche, a nostro avviso, il radicamento di una pratica di "bilinguismo" *spontaneo* che è uno degli obiettivi del progetto.

L'acquisizione del lessico specifico

Per questo lavoro è stato utilizzato il materiale prodotto, sempre nell'ambito del progetto CLIL, presso il Liceo Scientifico Grigoletti, negli anni passati. Tutto questo materiale è raccolto in una sezione, denominata *Saying maths*, del sito web del docente di matematica, precisamente alla pagina http://www.batmath.it/eng/say/say.htm.

Agli studenti sono state distribuite anche le stampe delle pagine fondamentali di questa raccolta, in particolare:

- The Greek alphabet and how to read it.
- Numbers: general remarks, elementary calculations, advanced calculations, useful expressions.
- Logic and sets, functions.
- Geometry

Poichè l'argomento dei numeri complessi richiede un buon numero di calcoli, non sempre elementari, si è dato largo spazio all'apprendimento della capacità di leggere formule anche articolate. Il lavoro è stato utile anche per la sua ricaduta specifica in matematica.

Si segnala, a mo' d'esempio, la seguente considerazione, emersa da una discussione con gli allievi.

> Nello scrivere il quadrato della funzione $f(x) = \sin x$, si usa, correntemente, la notazione $\sin^2 x$, al posto della più lunga (ma più precisa) scrittura $(\sin x)^2$. Quando si

vuole invece scrivere la composta tra la funzione $f(x) = \sin x$ e la funzione $g(x) = x^2$, si scrive $\sin(x^2)$, oppure, tralasciando le parentesi, $\sin x^2$. Questo tipo di scritture porta facilmente ad equivoci: poiché nei due casi si tratta, nella sostanza, solo di un diverso ordine nell'eseguire le operazioni, è facile fare confusione. Lo sforzo che si deve fare per leggere correttamente le formule in una lingua diversa dalla propria abitua lo studente a prestare la massima attenzione e lo aiuta a capire "al volo" la differenza tra le due scritture.

Questo lavoro preliminare è terminato con un test tipo *domino*, che si allega.

Lettura di un testo scientifico

In questa lezione agli studenti sono state distribuite le fotocopie del seguente articolo: *Word Processors: Stupid and Inefficient*, di Allin Cottrel. La scelta di questo articolo è legata al fatto che gli studenti, nel loro corso di informatica, stavano studiando l'uso di LaTeX come sistema di videoscrittura; inoltre l'articolo, pur essendo un testo di argomento scientifico, usa un linguaggio non infarcito di termini tecnici e quindi adatto agli allievi a cui è stato proposto.

Come già detto, scopo di questa lezione, svoltasi in compresenza con l'insegnante di lingua inglese, è stato quello di favorire lo sviluppo della capacità comunicative e delle dinamiche di gruppo, in vista del successivo lavoro sui numeri complessi. Gli allievi, a turno, hanno letto un paragrafo del testo, sul quale poi venivano poste domande orientate al versante linguistico da parte dell'insegnante di lingua e orientate al versante scientifico da parte dell'insegnante di matematica.

Anche di questo articolo si allega copia della parte che è stata oggetto della discussione in classe. L'intero articolo si può reperire all'indirizzo `http://ricardo.ecn.wfu.edu/~cottrell/wp.html`.

Richiami sui prerequisiti matematici

Poiché l'introduzione dell'insieme dei numeri complessi costituisce, in un certo senso, un capitolo conclusivo dello studio dell'algebra nella Scuola media Superiore, si è dato ampio spazio ad un richiamo, con numerosi approfondimenti e precisazioni, sui prerequisiti ritenuti prioritari. Gli argomenti fondamentali su cui si è incentrato il ripasso, che ha avuto una durata di 5 ore di lezione (una settimana nell'orario del corso di matematica PNI), sono:

L'insieme dei numeri reali Proprietà caratteristiche dell'insieme dei reali; le operazioni in $\mathbb{R}$.

Insiemi numerici e risoluzione di equazioni Risolubilità delle equazioni di primo grado in $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$. Il problema della risolubilità delle equazioni di secondo grado, con conseguente constatazione della insufficienza anche dell'insieme dei reali.

Cenni sulla cardinalità degli insiemi numerici Numerabilità dei razionali, non numerabilità dei reali.

Relazioni d'ordine negli insiemi numerici Proprietà dell'ordine introdotto sugli insiemi numerici $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$. Quest'ultimo argomento è particolarmente importante in considerazione del fatto che non è possibile introdurre un ordine 'naturale' nell'insieme dei numeri complessi.

Seguono gli allegati

1. Test tipo *domino* sulla lettura delle formule matematiche.
2. Una parte di un articolo di Allin Cottrell dal titolo *Word Processors: Stupid and Inefficient*.

Liceo Grigoletti Pordenone - Progetto CLIL

A warm-up activity:

Reading and Comprehension of Math Formulas
2005-01-20

Students are given a domino card each, then they are asked to follow the instructions below

Listen to your classmate reading.
If this is the correct reading of the formula you have on the left-hand column of your leaflet, put your hand up and read the formula.
Then read the text on the right-hand column.

While reading remember that ... is used to inform you that you must make a pause.

$\frac{1}{2}+\frac{1}{3}=\frac{5}{6}$	x cubed plus y squared, all over three
$\frac{x^3+y^2}{3}$	x cubed plus … y squared over three
$x^3+\frac{y^2}{3}$	the square root of … x plus y squared
$\sqrt{x+y^2}$	the square root of … x plus y all squared
$\sqrt{(x+y)^2}$	zero point double oh five, plus two point thirty seven all recurring
$0.005+2.\overline{37}$	zero point oh three, times thirty four point two three

0.03×34.23	three is less than four and less than five
$3 < 4 < 5$	x is between 3 and 7, the first bound included
$3 \leq x < 7$	x to the minus two plus the square root of 3
$x^{-2} + \sqrt{3}$	four choose 2 equals six
$\binom{4}{2} = 6$	one, plus two, plus three, and so on up to one hundred equals five thousand fifty
$1 + 2 + 3 + \cdots + 100 = 5050$	the n-th root of x plus the n-th root of y ... all squared

$\left(\sqrt[n]{x}+\sqrt[n]{y}\right)^2$	two over ... x cubed plus y squared
$\frac{2}{x^3+y^2}$	x plus y ... all over 2 plus the square root of three
$\frac{x+y}{2+\sqrt{3}}$	one, times two, times three, and so on up to ten is three millions six hundred twenty eight thousand and eight hundred
$1\times 2\times 3\times\cdots\times 10=3,628,800$	a minus b, times, open parenthesis, c plus d, close parenthesis
a-b(c+d)	two plus x, all over the fourth root of three
$\frac{2+x}{\sqrt[4]{3}}$	one half plus one third is equal to five over six

Word Processors: Stupid and Inefficient

Allin Cottrel

Estratto dell'articolo originale, reperibile all'indirizzo http://ricardo.ecn.wfu.edu/~cottrell/wp.html. L'estratto contiene solo le parti che hanno formato oggetto della discussione in classe durante le lezioni di un Modulo CLIL svolto al Liceo Grigoletti di Pordenone.

1 The claim

The word processor is a stupid and grossly inefficient tool for preparing text for communication with others. That is the claim I shall defend below. It will probably strike you as bizarre at first sight. If I am against word processors, what do I propose: that we write in longhand, or use a mechanical typewriter? No. While there are things to be said in favor of these modes of text preparation I take it for granted that most readers of this essay will do most of their writing using a computer, as I do. My claim is that there are much better ways of preparing text, using a computer, than the word processor.

The wording of my claim is intended to be provocative, but let me be clear: when I say word processors are stupid I am not saying that you, if you are a user of a word processor, are stupid. I am castigating a technology, but one that is assiduously promoted by the major software vendors, and that has become a de facto standard of sorts. Unless you happen to have been in the right place at the right time, you are likely unaware of the existence of alternatives. The alternatives are not promoted by the major vendors, for good reason: as we shall see, they are available for free.

Let's begin by working back from the end product. Text that is designed to communicate ideas and information to others is disseminated in two main ways:

1. As "hard copy", that is, in the form of traditional printed documents.
2. By digital means: electronic mail, web pages, documents designed to be viewable on screen.

There is some overlap here. For instance, a document that is intended for printing may be distributed in digital form, in the hope that the recipient has the means to print the file in question. But let us consider these two modes of dissemination in turn.

2 Printed documents

You want to type a document at your computer keyboard, and have it appear in nicely printed form at your computer's printer. Naturally you don't want this to happen in real time (the material appearing at the printer as you type). You want to type the document first and "save" it in digital form on some storage medium. You want to be able to retrieve the document and edit it at will, and to send it to the printer when the time is right. Surely a word processor-such as the market leader, Microsoft Word-is the "natural" way to do all this? Well, it's one way, but not the best. Why not?

2.1 Composition versus typesetting

Preparing printable text using a word processor effectively forces you to conflate two tasks that are conceptually distinct and that, to ensure that people's time is used most effectively and that the final communication is most effective, ought also to be kept practically distinct. The two tasks are

The composition of the text itself. By this I mean the actual choice of words to express one's ideas, and the logical structuring of the text. The latter may take various forms depending on the nature of the document. It includes matters such as the division of the text into paragraphs, sections or chapters, the choice of whether certain material will appear as footnotes or in the main text, the adding of special emphasis to certain portions of the text, the representation of some pieces of text as block quotations rather than as the author's own words, and so on.

The typesetting of the document. This refers to matters such as the choice of the font family in which the text is to be printed, and the way in which structural elements will be visually represented. Should section headings be in bold face or small capitals? Should they be flush left or centered? Should the text be justified or not? Should the notes appear at the foot of the page or at the end? Should the text be set in one column or two? And so on.

The author of a text should, at least in the first instance, concentrate entirely on the first of these sets of tasks. That is the author's business. Adam Smith famously pointed out the great benefits that flow from the division of labor. Composition and logical structuring of text is the author's specific contribution to the production of a printed text. Typesetting is the typesetter's business. This division of labour was of course fulfilled in the traditional production of books and articles in the pre-computer age. The author wrote, and indicated to the publisher the logical structure of the text by means of various annotations. The typesetter translated the author's text into a printed document, implementing the author's logical design in a concrete typographical design. One only has to imagine, say, Jane Austen wondering in what font to put the chapter headings of Pride and Prejudice to see how ridiculous the notion is. Jane Austen was a great writer; she was not a typesetter. You may be thinking this is beside the point. Jane Austen's writing was publishable; professional typesetters were interested in laying it out and printing it. You and I are not so lucky; if we want a printed article we will have to do it ourselves (and besides, we want it done much faster than via traditional typesetting). Well, yes and no. We will in a sense have to do it ourselves (on our own computers), but we have a lot of help at our disposal. In particular we have a professional-quality typesetting program available. This program (or set of programs) will in effect do for us, for free and in a few seconds or fractions of a second, the job that traditional typesetters did for Shakespeare, Jane Austen, Sir Walter Scott and all the rest. We just have to supply the program with a suitably marked-up text, as the traditional author did.

I am suggesting, therefore, that should be two distinct "moments" in the production of a printed text using a computer. First one types one's text and gets its logical structure right, indicating this structure in the text via simple annotations. This is accomplished using a text editor, a piece of software not to be confused with a word processor. (I will explain this distinction more fully below.) Then one "hands over" one's text to a typesetting program, which in a very short time returns beautifully typeset copy.

2.2 The evils of WYSIWYG

These two jobs are rolled into one with the modern WYSIWYG ("What You See Is What You Get") word processor. You type your text, and as you go the text is given, on the computer screen, a concrete typographical representation which supposedly corresponds closely to what you will see when you send the document to the printer (although for various reasons it does not always do so). In effect, the text is continuously typeset as you key it in. At first sight this may seem to be a great convenience; on closer inspection it is a curse. There are three aspects to this.

The author is distracted from the proper business of composing text, in favor of making typographical choices in relation to which she may have no expertise ("fiddling with fonts and margins" when she should be concentrating on content).

The typesetting algorithm employed by WYSIWYG word processor sacrifices quality to the speed required for the setting and resetting of the user's input in real time. The final product is greatly inferior to that of a real typesetting program.

The user of a word processor is under a strong temptation to lose sight of the logical structure of the text and to conflate this with superficial typographical elements.

The first two points above should be self-explanatory. Let me expand on the third. (Its importance depends on the sort of document under consideration.)

2.3 Document structure

Take for instance a section heading. So far as the logical structure of a document is concerned, all that matters is that a particular piece of text should be "marked" somehow as a section heading. One might for instance type `\section{Text of heading}`. How section headings will be implemented typographically in the printed version is a separate question. When you're using a word processor, though, what you see is (all!) you get. You are forced to decide on a specific typographical appearance for your heading as you create it.

Suppose you decide you'd like your headings in boldface, and slightly larger than the rest of the text. How are you going to achieve this appearance? There's more than one way to do it, but for most people the most obvious and intuitive way (given the whole WYSIWYG context) is to type the text of the heading, highlight it, click the boldface icon, pull down the little box of point sizes for the type, and select a larger size. The heading is now bold and large.

Great! But what says it's a heading? There's nothing in your document that logically identifies this little bit of text as a section heading. Suppose at some later date you decide that you'd actually prefer to have the headings in small caps, or numbered with roman numerals, or centered, or whatever. What you'd like to say is "Please make such-and-such a change in the setting of all section headings." But if you've applied formatting as described above, you'll have to go through your entire document and alter each heading manually.

Now there is a way of specifying the structural status of bits of text in (for instance) Microsoft Word. You can, if you are careful, achieve effects such as changing the appearance of all section headings with one command. But few users of Word exploit this consistently, and that is not surprising: the WYSIWYG approach does not encourage concern with structure. You can easily-all too easily-"fake" structure with low-level formatting commands. When typing one's text using a text editor, on the other hand, the need to indicate structure is immediately apparent.

2.4 Text editors

OK, it's probably time to explain what a text editor is, and how it differs from a word processor. A modern text editor looks a bit like a word processor. It has the usual apparatus of pull-down menus and/or clickable icons for functions like opening and saving files, searching and replacing, checking spelling, and so on. But it has no typesetting functionality. The text you type appears on screen in a clear visual representation, but with no pretense at representing the final printed appearance of the document.

When you save your document, it is saved in the form of plain text, which in the US context usually means in "ASCII" (the American Standard Code for Information Interchange). ASCII is composed of 128 characters (this is sometimes referred to as a "7-bit" character set, since it requires 7 binary digits for its encoding: 2 to the seventh power = 128). It includes the numerals 0 through 9, the roman alphabet in both upper and lower case, the standard punctuation marks, and a number of special characters. ASCII is the lowest common denominator of textual communication in digital form. An ASCII message will be "understandable" by any computer in the world. If you send such a message, you can be sure that the recipient will see precisely what you typed.

By contrast, when you save a file from a word processor, the file contains various "control" characters, outside of the ASCII range. These characters represent the formatting that you applied (e.g. boldface or italics) plus various sorts of internal "business" relating to the mechanics of the word processor. They are not universally "understandable". To make sense of them, you need a copy of the word processor with which the document was created (or some suitable conversion filter). If you open a word processor file in a text editor, you will see (besides the text, or bits of it) a lot of "funny looking stuff": this is the binary formatting code.

Since a text editor does not insert any binary formatting codes, if you want to represent features such as italics you have to do this via mark-up. That is, you type in an annotation (using nothing but ASCII), which will tell the typesetter to put the specified text into italics. For example, for the LaTeX typesetter (more on this below) you would type *stuff you want in italics*. Actually, if you are using a text editor which is designed to cooperate with LaTeX you would not have to type this yourself. You'd type some kind of shortcut sequence, select from a menu, or click an icon, and the appropriate annotation would be inserted for you; the mechanics of typing an ASCII document suitable for feeding to LaTeX are not much different from typing in a modern word processor.

2.5 The virtues of ASCII

The approach of composing your text in plain ASCII using a text editor, then typesetting it with a separate program, has several "incidental" virtues.

Portability: as mentioned above, anybody, using any computer platform, will be able to read your marked-up text, even if they don't have the means to view or print the typeset version. By contrast your Snazz 9.0 word processor file can be completely incomprehensible to a recipient who doesn't have the same brand and version of word processor as you-unless he or she is quite knowledgeable about computers and is able to extract the actual text from the binary "garbage". And this applies to you over time, as much as to you and a correspondent at one time. You may well have difficulty reading Snazz 8.0 files using Snazz 9.0, or vice versa, but you'll never have any trouble reading old ASCII files. Compactness: an ASCII file represents your ideas, and not a lot of word processor "business". For small documents in particular, word processor files can be as much as 10 times as large as a corresponding ASCII file containing the same relevant information. Security: the "text editor to typesetter" approach virtually guarantees that you will never have any problem of corruption of your documents (unless you suffer a hard disk crash or some comparable calamity). The source text will always be there, even if the typesetter fails for some reason. If you regularly use a word processor and have not had a problem with file corruption then you're very lucky! (Further reading: Sam Steingold's page No Proprietary Binary Data Formats.)

2.6 The typesetter

By this time I owe you a bit more detail on the typesetter part of the strategy I'm advocating. I won't go into technical details here, but will try to say enough to give you some idea of what I'm talking about.

The basic typesetting program that I have in mind is called TEX, and was written by Donald Knuth of Stanford University. TEX is available for free (via downloading from many Internet sites), in formats suitable for just about every computer platform. (You can if you wish purchase a CDROM with a complete set of TEX files for a very modest price.) Knuth started work on TEX in 1977; in 1990 he announced that he no longer intended to develop the program-not because of lack of interest, but rather because by this time the program was essentially perfected. It is as bug-free as any computer program can be, and it does a superb job of typesetting just about any material, from simple text to the higher mathematics.

I referred above to LATEXİf TEX is the basic typesetting engine, LATEX is a large set of macros, initially developed by Leslie Lamport in the 1980s and now maintained by an international group of experts. These macros make life a lot easier for the average user of the system. LATEX is still under active development, as new capabilities and packages are built on top of the underlying typesetter. Various "add-ons" for TEX are also under development, such as a system which allows you to make PDF (Adobe's "Portable Document Format") files directly from your ASCII source files. (I say "under development" but by this I just mean that they are continuously being improved. The programs are already very stable and full-featured.)

As mentioned above, you indicate the desired structure and formatting of your document to LATEX in the form of a set of annotations. There are many books (and web-based guides) that give the details of these annotations, and I will not go into them here. The common annotations are simple and easily remembered, besides which LATEXfriendly text editors (of which there are many) offer you a helping hand.

One very attractive feature of LATEX is the ability to change the typeset appearance of your text drastically and consistently with just a few commands. The overall appearance is controlled by

The "document class" that you choose (e.g. report, letter, article, book). The "packages" or style files that you decide to load. You can, for instance, completely change the font family (consistently across text, section headings, footnotes and all) and/or the sizes of the fonts used, by altering just one or two parameters in the "preamble" of your ASCII source file. Similarly, you can put everything into two-column format, or rotate it from portrait to landscape. It may be possible to accomplish something similar using a word processor, but generally it's much less convenient and you are far more likely to mess up and introduce unintended inconsistencies of formatting.

You can get as complex as you care to, typesetting with LaTeX. You can choose a "hands off" approach: just specify a document class and leave the rest up to the default macros. Generally this produces good results, the typesetting being of much higher quality than any word processor. (Naturally, things like numbering of chapters, sections and footnotes, cross-references and so on, are all taken care of automatically.) Or you can take a more "interventionist" approach, loading various packages (or even writing your own) to control various aspects of the typography. If this is your inclination, you can produce truly beautiful and individual output.

2.7 Putting it together

Let me give you just a brief idea of how this all works. If you have a good TeXsetup it's like this: You type your text into a TeX-aware editor. You can type the required annotations directly or have the editor insert them via menus or buttons. When you reach a point where you'd like to take a look at the typeset version you make a menu choice or click a button in the editor to invoke the typesetter. Another menu item or button will open a previewer in which you see the text as it will appear at the printer. And generally this is true "WYSIWYG"-the previewer will show a highly accurate representation of the printed output. You can zoom in or out, page around, and so on. You send the output to the printer with another menu choice or button, or go back to editing.

At some later point in the process you want to preview the updated file. Click the typesetter button again. This time you don't have to invoke the previewer again: if you've left it running in the background it will now automatically display the updated typeset version. When you're done with an editing session you can delete the typeset version of the file to conserve disk space. You just need to save the ASCII source file; the typeset version can easily be recreated whenever you need it.

3 Digital dissemination

...