PM-ISOmath
The Poor Man ISO math bundle

Claudio Beccari*

v.1.0.06 — 2020/06/30

Contents

Abstract

The ISO regulations for typesetting math in the field of physics and technology are pretty stringent and imply legal questions that we do not treat here; it suffices to say that in certain countries an "Expertise" for a Legal Court that does not fulfil such regulations may be rejected by the Court, independently from its expert contents.

Authors may not like them, but in the filed of *applied sciences*, or better, of *experimental sciences* that use measured quantities and units of measure, they are compulsory.

With LuaLaTeX and XeLaTeX, while using OpenType math fonts there should not be any difficulty in fulfilling the regulations, but with pdfLaTeX; things are not so simple. There exist some facilities, but sometimes they do not work.

This package provides some robust work-arounds to bypass the difficulties experienced by some pdfLaTeX users.

1 Introduction

The ISO regulations (formerly ISO 31/XI, now ISO 80000) are stringent rules to typeset mathematics in the domains of physics and applied sciences; their title explicitly mentions "physics and technology", but their careful reading lets understand that they apply to all sciences that use the "mathematics of quantities".

*E-mail: claudio dot beccari at gmail dot com

Such entities form a special group or space, where the elements are couples of two ordered entities (x, y), where y represents the unit of measure and x represents the ratio of the quantity to the unit of measure. Such paired entities may not be separated, therefore some special mathematical rules are established in order to operate on quantities.

Add to these special mathematical bases the fact that the measure component of the quantity is pretty fuzzy and it is always accompanied by a certain degree of uncertainty; metrologists are the masters in measuring quantities and handling their measures and uncertainties, but although for simplicity measures are handled by lay people as rational numbers (after all aren't they the ratio of something to be measured and the unit of measure?) we are facing the domain of fuzzy sets.

Furthermore quantities are so many that any work in applied sciences should contain a nomenclature list in order to explain which symbol is used for which quantity. This is where the ISO regulations set some order and establish a long list of named quantities with their preferred symbols and their "normal" units according to the prescriptions of the Comité International des Pois et Mésures, that established the International System of measures (Systhème International, SI).

This ISO nomenclature is used uniformly by most, if not all, people involved in applied sciences; therefore among the few letters of the Latin and Greek alphabets almost none is free to be used for other purposes.

This means that the usual math italic alphabet can be used for very few symbols if confusion is to be avoided. Other series and shapes must be judiciously used and the ISO regulations say how.

There are no problems when typesetting applied science documents with LuaLaTeX or XeLaTeX, at least if the proper OpenType math fonts are used; such fonts have available so many slots (code points) that may contain any variation of any glyph; it suffices to specify the option `math style = ISO` to the math handling package `unicode-math` and to select an OpenType math font. The only problem, if any, is to know which font series and shapes should be used according to the ISO regulations.

These regulations can be purchased from the ISO site in Switzerland; they are quite expensive and the cost is affordable by associated professional studios or large academic and/or research institutions.

For private users I'd suggest to download the PDF document `https://www.nist.gov/pml/special-publication-811-extended-%contents`. This document has been produced by the National Institute for Science and Technology, the Institution that several years ago was appointed to replace the United States National Bureau of Standards. Their staff is made essentially by metrologists and this guide is written to give precise instructions for handling the applied science mathematics according to the ISO regulations; it establishes also several rules for writing text about mathematics and metrology. It is extremely valuable for anglophones, but, with the due differences concerning the mother languages, it is extremely useful also for people using languages different from English.

2 The pdfLaTeX handicaps

Users of pdfLaTeX, on the opposite are in trouble. In facts this typesetting program suffers from an inherited limitation: math fonts are encoded with the old 128 glyph encodings; this is not a limitation set forth by the underlying interpreter pdfTeX. Matter of fact there exists the quite recent LibertinusT1math fonts for pdfLaTeX, created by Michael Sharpe, that, to my best knowledge, are the only math fonts with 256 glyph encodings. Package `libertinust1math` accepts the `ISO` option that allows to fulfil the ISO regulations; it accepts other options; depending on which options are specified the number of math groups, beyond the essential first four ones, increases by three to six units, reaching a maximum of ten; there remains enough free math groups to satisfy most user requirements.

But even while using such LibertinusT1math fonts, pdfTeX suffers from another handicap derived from the knuthian original TeX-the-program and by the NFSS (New Font Selection Scheme, which is not new any more, because it dates back to 1994, when LaTeX 209 became obsolete and was substituted by LaTeX 2_ϵ.). I am not complaining about these pdfTeX and LaTeX limitations; for decades people have been happily typesetting math with results that are much superior to any other typesetting program at least when the latter does not use some TeX software.

The handicap I am talking about is the way math alphabets, at the moment, are handled by pdfTeX and LaTeX; such alphabets are loaded in the form of math groups, the number of which cannot exceed 16 (numbered from 0 to 15); each group loads three sizes for normal math style, for script style and for script-script style. Taking into account the bold version, the number of math groups would risk to exceed its capabilities; in order to avoid exceeding the number limit on the math groups usable in any math environment, only the medium *or* the bold series is loaded and the math version (bold or unbold) must be chosen before entering the math environment. Packages that allow mixed series math formulas load both versions and become very critical for using other math alphabets.

By default four groups are always loaded; group 0 for operators; group 1 for letters; group 2 for symbols; group 3 for large operators and delimiters; users very often load also the two symbol-fonts provided by the American Mathematical Society, and this means two more math groups. Apparently there are ten more groups for other math groups. It seems that this number is abundantly sufficient to handle any situation, but it is not so. The author of this extension already reached the limit of 16 without doing anything special – at least he thought so, but he was evidently wrong. Users who use packages such as the Fourier or the KPfonts have to pay special attention to the package documentation because they might specify options that imply loading up to 14 math groups.

Several authors provided packages to help users produce perfect documents that fulfil the ISO regulations; I would like to cite the excellent Package `ISOmath` by Günter Milde, entitled "Mathematical style for science and technology". It should be the perfect package to use in order to fulfil the ISO regulations; sometimes it succeeds with excellent results, but more often than not the results are just partial, because of the limitations of the math fonts available to pdfLaTeX that

are in contrast with the requirements of the ISO regulations.

Therefore, dear reader, before using the poor man solutions of this package, try `ISOmath`, if your font set passes all the requirements described in that package documentation, you don't need the poor man patches offered by this package.

3 ISO rules summary

This summary does not replace the original ISO document nor what is written in the instructions published by NIST. It simply recalls those rules that this package tries to implement. In what follows, the word "quantity" is used to represent any physical entity that may be measured according to the metrological practice; the word "variable" is used to represent a mathematical entity that represents variable data.

1. All quantity and variable symbols are represented by one letter (with as many appositions are needed); but no acronyms are allowed; in computer science programming many strings are called variables, in the sense that they may represent variable data; but computer programs are not mathematics, rather they are a special language that tells the computer what to do, even mathematics, but the language is not the one that represents math. Unfortunately some acronyms have gained strong popularity and wide usage, but they are forbidden by the ISO rules; example: *CMRR* is often used to mean "common mode rejection ratio", but this is an improper usage of mathematics. Obviously this rule must be applied by the user, because LaTeX is a typesetting language and does not understand the real meaning of what it typesets.
2. All quantity symbols must be set in italics, slanted type is allowed, but serifed italics should be preferred unless the ISO rules prescribe a sans serif font. This implies that the differential symbol be in upright font to avoid confusion with the physical quantity d; the Napier number 'e' must be set in upright font to avoid confusion with the elementary electric charge e; the imaginary unit j in electrical engineering, (i in other applied sciences) must be set in upright font in order to avoid confusion with the electric current density j or the electric current i; more difficult: the transcendental number $\pi = 3.14159\ldots$ should be distinguished from the plane angle π; and similar other numerical constants represented by Latin or Greek letters.
3. All symbols that do not represent quantities should be typeset in an upright font, preferably a serifed font, except when the ISO rules require a sans serif font. This rule includes numbers and their digits, symbols that represent constant numeric values, all appositions both in subscript and superscript position. Appositions are not quantities or variables: for example in V_i, the subscript i is a variable because it represents the i-th element in a sequence, such as $V_0, V_1, V_2, \ldots$; on the opposite V_{i} the subscript is an apposition because 'i' means, say, 'input'.
4. Upright bold roman or black board bold symbols represent sets.

5. Italic bold symbols represent matrices; one column matrices may represent vectors in an algebraic way and should be treated as any other matrix; in general multirow and multicolumn matrices are typeset with uppercase letters, while lowercase ones are reserved for vectors; but in some sciences also vectors may have uppercase letters. Geometrical vectors that might be typeset with a medium series upper o lowercase italic letter with an arrow on top of it are not treated by the ISO regulations that speak of vectors irrespective if they are considered as one column matrices or oriented segments; apparently oriented segments should be treated the same as one column matrices.
6. Labels to geometrical entities, such as points, segments, angles (not their measures) should be set in upright medium series sans serif fonts; the same rule applies to labels used in sketches and drawings representing machinery, electric circuitry, and the like when the label refers to an object and not to its measure: "the lens L_1", "the switch S_2", "the planetary gear G_3", and so on.
7. Tensors should be set in slanted bold sans serif font.
8. The above rules apply to both Latin and Greek letters.

Such font rules for families, series and shapes are difficult to implement with pdfLaTeX for many reasons that do not imply only the limited number of math groups, but also the categories of symbols; Greek letters in particular are troublesome for a couple of reasons connected with rule 8:

- Uppercase greek letters are taken from the "operators" alphabet and are letter symbols; they are upright; but if they represent quantities they must be in italics, or at least slanted;
- Lowercase Greek letters are taken from the "letters" math alphabet and are ordinary symbols; they are oblique, and this is fine, but as ordinary math symbols they cannot be modified by commands such as `\mathrm`; furthermore upright lowercase Greek letters are not available, at least not directly.

4 Some existing solutions

Several packages to be used with pdfLaTeX allow for upright Greek letters, especially those packages for French typography, where the national rules (in contrast with the ISO regulations) require that all math entities typeset with Greek letters be upright. Among such packages therenare `fourier` and `kpfonts`. Other packages such as `newpxmath` and `newtxmath` are intended for general use, but with suitable options and extra math groups let the user employ upright lowercase Greek letters as well as oblique uppercase ones.

Package `libertinust1math` allows the use of the 256 glyph encoded fonts and are less sensitive to the limit of 16 math groups. Its many options allow to use the various font styles without requiring the `\boldmath` declaration; this implies that medium and bold series are both preloaded without actually using extra math alphabets beyond the small number it uses for its full functionality. An option

`ISO` is already available to fulfil the ISO regulations. Maybe the only drawback of the fonts used is that they are intended to match the Libertinus text fonts that are blacker than the standard Computer or Latin Modern ones. The auxiliary font selection commands used in the `ISOmath` package are already implemented with this `libertinust1math` one.

Of course the `ISOmath` package might solve all problems if the user math environment has the necessary functionalities, in particular all the math alphabets needed for the task and if there are no difficulties with the number of math groups.

5 The poor man solution

The poor man solution is very simple in theory; it handles text fonts in math expressions through the `\text` command provided by the `amsmath` package; of course there are functionalities to chose families series and shapes in a comfortable way by means of the powerful command definition commands provided by the `xparse` package; for Greek letters it uses the functionalities of the `alphabeta` package, that allows to use in text mode the same control sequences that are used in math mode. The default families for Latin and Greek letters are the Latin Modern ones that allow a piecewise continuous scaling of any available particular font of the collection.

It goes without saying that this poor man solution has advantages and disadvantages over the other indicated solutions.

The main advantage is that no math groups are involved, therefore the user may couple any font family with any series and shape that family allows; the default family for Greek fonts are the Latin Modern compliant LGR encoded collection of CBfonts; they are always available with any up to date and complete TEX system installation.

This is also a disadvantage, in the sense that Latin Modern fonts might not be the best ones to use with any specific text font; for example they have an x-height smaller than the Palatino ones, and a larger 'em' unit compared to the Times ones; they have also a slightly lighter 'color' than most other fonts.

Nevertheless they work very well with the ISO regulations and in spite of the disadvantages listed above, they are usable without problems. This very documentation is typeset with Latin Modern fonts. The examples shown in a following section show the ease with which the ISO regulations may be fulfilled.

There are other Greek fonts that may be used in place of those of the CBfonts collection, especially those distributed by the Greek Font Society (GFS) that are already part of the TEX system distribution; they are LGR encoded and have their specific `.fd` files. They are not so rich in series and shapes (some of them may lack the bold extended series and sometimes lack also the bold series); I would say the the GFS Bodoni family is sufficiently rich of series and shapes, and may be used also for the Latin fonts.

There are also the Greek fonts that are "companions" to the Times fonts. So the user is not limited to the Latin Modern fonts, but admittedly there is not a great choice and, if it is necessary, the disadvantage of an unmatching Greek font

is the price to pay if the user has to use pdfLaTeX.

6 Usage

The usage of the package is very simple; in your preamble add the line

```
\usepackage[engineer]{pm-isomath}
```

The `engineer` option is used to set the imaginary unit the way electrical engineers usually do.

The main macros that support the whole package, and that may be used also by the end user, are:

`\MathLatin{`⟨*letter*⟩`}{`⟨*family*⟩`}[`⟨*series*⟩`](`⟨*shape*⟩`)`

and

`\MathGreek{`⟨*letter*⟩`}{`⟨*family*⟩`}[`⟨*series*⟩`](`⟨*shape*⟩`)`

where `\MathLatin` sets the text encoding to `T1`, while `\MathGreek` sets it to `LGR`. In both commands the arguments specified with `{`⟨*family*⟩`}[`⟨*series*⟩`](`⟨*shape*⟩`)` are all optional, including the first one in spite of being surrounded by curly braces. In both cases the default values for each argument are respectively `lmr`, `m`, and `n` (normal, upright). Notice the codes: `lmr`, `m`, and`n` are the codes that appear in the `.fd` file to declare the possible family, series, and shape combinations available for a given family. Se bellow further information in table 2 on page 13, and how to discover these codes.

Such default values, after loading `pm-isomath`, may be globally redefined by using in the preamble:

```
\renewcommand{ISOfam}{⟨family⟩}
\renewcommand{ISOser}{⟨series⟩}
\renewcommand{ISOsha}{⟨shape⟩}
```

We discourage this global redefinitions unless the user really knows what s/he is doing; in practice it must be checked that T1 encoded families and LGR encoded ones have the same family names. If the encoding+family `.fd` files are not available either they have to be created, or such default values should not be redefined.

For Latin letters to use in the `\MathLatin` command mandatory argument there are no problems.

For Greek letters you might use `\MathGreek` and the math letter commands `\alpha`, `\beta`,..., `\Omega` commands, but it is much simpler to avoid`\MathGreek` and use the package commands `\ISOalpha`, `\ISObeta`,..., `\ISOOmega`. All these commands follow the syntax:

`\ISO`⟨*lettername*⟩`{`⟨*family*⟩`}[`⟨*series*⟩`](`⟨*shape*⟩`)`

where all the arguments are optional, including the first in spite of being surrounded by curly braces. Such optional arguments must follow that order but all possible combinations are usable, for example:

\ISOalpha	$\boldsymbol{\alpha}$	\ISObeta	$\boldsymbol{\beta}$	\ISOgamma	$\boldsymbol{\gamma}$	\ISOdelta	$\boldsymbol{\delta}$
\ISOepsilon	$\boldsymbol{\varepsilon}$	\ISOzeta	$\boldsymbol{\zeta}$	\ISOeta	$\boldsymbol{\eta}$	\ISOtheta	$\boldsymbol{\vartheta}$
\ISOiota	$\boldsymbol{\iota}$	\ISOkappa	$\boldsymbol{\varkappa}$	\ISOlambda	$\boldsymbol{\lambda}$	\ISOmu	$\boldsymbol{\mu}$
\ISOnu	$\boldsymbol{\nu}$	\ISOxi	$\boldsymbol{\xi}$	\ISOomicron	$\boldsymbol{o}$	\ISOpi	$\boldsymbol{\pi}$
\ISOrho	$\boldsymbol{\varrho}$	\ISOsigma	$\boldsymbol{\sigma}$	\ISOtau	$\boldsymbol{\tau}$	\ISOupsilon	$\boldsymbol{\upsilon}$
\ISOphi	$\boldsymbol{\varphi}$	\ISOchi	$\boldsymbol{\chi}$	\ISOpsi	$\boldsymbol{\psi}$	\ISOomega	$\boldsymbol{\omega}$
\ISOGamma	$\boldsymbol{\Gamma}$	\ISODelta	$\boldsymbol{\Delta}$	\ISOEta	$\boldsymbol{\mathrm{H}}$	\ISOTheta	$\boldsymbol{\Theta}$
\ISOLambda	$\boldsymbol{\Lambda}$	\ISOXi	$\boldsymbol{\Xi}$	\ISOPi	$\boldsymbol{\Pi}$	\ISORho	$\boldsymbol{\mathrm{P}}$
\ISOSigma	$\boldsymbol{\Sigma}$	\ISOUpsilon	$\boldsymbol{\Upsilon}$	\ISOPhi	$\boldsymbol{\Phi}$	\ISOChi	$\boldsymbol{\mathrm{X}}$
		\ISOPsi	$\boldsymbol{\Psi}$	\ISOOmega	$\boldsymbol{\Omega}$		

Table 1: The \ISO⟨*letter*⟩ macros and their rendering in bold style

\ISOalpha
\ISOalpha{⟨*family*⟩}
\ISOalpha[⟨*series*⟩]
\ISOalpha(⟨*shape*⟩)
\ISOalpha{⟨*family*⟩}[⟨*series*⟩]
\ISOalpha{⟨*family*⟩}(⟨*shape*⟩)
\ISOalpha[⟨*series*⟩](⟨*shape*⟩)
\ISOalpha{⟨*family*⟩}[⟨*series*⟩](⟨*shape*⟩)

This offers the maximum flexibility in using the necessary commands.

The package defines other macros for fulfilling the rules relative to the differential symbol and the numerical constants represented with letters; furthermore it defines the commands for the \ohm unit of measure and the \micro SI prefix; this latter macro uses a special shape of the CBfonts where an upright shape with serifed lowercase Greek letters is available; if another family lacking this shape is being used, then the normal upright shape is used. In typesetting this documentation, evidently there are no problems, but with other font selections, especially with Greek fonts, there might be some mismatching shapes.

Commands similar to those defined by the ISOmath package are also defined so as to simplify the font selection for vectors, matrices and tensors.

7 Examples

ISO Greek letters In example represented in table 1 we typeset an array in math mode, where we show all the Greek letters that can be typeset with the \ISO⟨*lettername*⟩ macros; the array is typeset in normal math style, but the ISO letters are in bold style so that there is no confusion with a normal bold math setting; some letters, equal to the Latin ones, are also defined because some users have experienced difficulties in remembering the correct signs especially while labelling diagrams. The array in table 1 may be a useful reference.

A small matrix equation

$$\boldsymbol{b} = \boldsymbol{M}\boldsymbol{a}$$

is typeset with the following code

```
\[ \vectorsymbol{b} = \matrixsymbol{M}\vectorsymbol{a} \]
```

A resistivity value The resistivity of copper is $1.68\,\mu\Omega\,\mathrm{cm}$ (in text mode: $1.68\,\mu\Omega\,\mathrm{cm}$) is typeset with the following code

```
$1.68\,\micro\ohm\,\mathrm{cm}$ (in text mode: 1.68\unit{\micro\ohm\,cm})
```

A tensor

$$\boldsymbol{D} = \varepsilon_0 \boldsymbol{\epsilon}_\mathrm{r} \boldsymbol{E}$$

is typeset with the following code

```
\[
\vectorsymbol{D} = \epsilon_0\tensorsymbol{\epsilon}\ped{r}\vectorsymbol{E}
\]
```

Solid angle An energy flux of light form a isomorphic source that irradiates the power P through the solid angle Ω generates a flux

$$\Phi = \frac{P}{\Omega}$$

is typeset with the following code

```
\[\Phi = \frac{P}{\Omega}\]
```

where, as you see the uppercase Greek letters are slanted, as the ISO rules require, instead of upright, as LaTeX sets them by default.

A bold formula This is the very important inverse Laplace transform[1]

$$\boldsymbol{f(t) = \frac{1}{2\pi i} \int_{\sigma-i\infty}^{\sigma+i\infty} e^{pt}\,dp \qquad \text{for } \sigma > \sigma_c}$$

typeset with the following code

```
{\boldmath\[
f(t) = \frac{1}{2\ISOpi\iu} -\mkern-19mu
   \int_{\sigma-\iu\infty}^{\sigma+\iu\infty}\eu^{pt}\diff p
     \qquad \text{for } \sigma > \sigma\ped{c}
\]}
```

Notice the the use of `\boldmath` does not imply the use of new math groups; but the bold upright π is rendered without any problem.

[1] Some packages may have a control sequence to insert a Cauchy principal value integral sign into a math expression; here we fake it by means of the superposition of a normal integral sign to a minus sign.

Various styles of Greek fonts Here are some examples of Greek fonts in various styles; within the same table bold and medium series fonts stay side by side, as well as glyphs coming from different families.

`\ISOgamma`, `\ISOzeta`, `\ISOeta`, `\ISOOmega`	$\gamma, \zeta, \eta, \Omega$
`\ISOgamma`[bx], `\ISOzeta`[bx], `\ISOeta`[bx], `\ISOOmega`[bx]	$\boldsymbol{\gamma}, \boldsymbol{\zeta}, \boldsymbol{\eta}, \boldsymbol{\Omega}$
`\ISOgamma`{lmss}, `\ISOzeta`{lmss}, `\ISOeta`{lmss}, `\ISOOmega`{lmss}	$\mathsf{\gamma}, \mathsf{\zeta}, \mathsf{\eta}, \mathsf{\Omega}$
`\ISOgamma`{artemisia}, `\ISOzeta`{artemisia}, `\ISOeta`{artemisia}, `\ISOOmega`{artemisia}	$\gamma, \zeta, \mathrm{n}, \Omega$
`\ISOgamma`(rs), `\ISOzeta`(rs), `\ISOeta`(rs), `\ISOOmega`(rs)	$\gamma, \zeta, \eta, \Omega$

8 Final remarks

This package `pm-isomath` is far from perfect, and its results are questionable; of course poor man solutions are just patches; incomplete solutions; but the results are not so bad. It has the indubitable advantage that is does not use any math groups, therefore there is no risk to exceed the limit of 16 math groups.

9 The code

This package was loosely inspired by the `ISOmath` package by Günter Milde, but tackles the problem of insufficient maximum number of math font groups so as to avoid any problem with such group limitation, and therefore all the caveats in Milde's package. That package is much more comfortable to use than this one; but it is subject to a number of conditions that, depending on the user environment, may even result in a complete failure. This package avoids problems with math font groups because it does not use any, but it is not so comfortable to use because often the user has to specify optional settings.

The preliminary lines have been already defined; therefore we start with real code.

The trick of this package is that all fonts different from the four or six ones (including the AMS symbol fonts) are textual fonts used in math typesetting through the intermediate action of the `\text` command defined by the `amsmath` package. Therefore we start by verifying if packages `amsmath`, `alphabeta` and `xparse` have already been loaded in the document preamble; this implies a weak loading order, that is this package must be loaded after all the above packages are loaded; in facts if such packages are not loaded, they get loaded by this one, but without any option. The package loading mechanism assures avoiding conflicts if packages are loaded without options; this is why if one of the three packages is loaded after this one but with some option specified, an "Option clash" error flag is raised; this is where the "weak" loading error becomes a very "strong" one.

```
% Then we verify if the document is being typeset with \pdfLaTeX;
% if it is not, an error flag is raised and reading of this
% package is immediately interrupted. For this purpose we need
% an engine-detecting package, and we generally use the |iftex| one.
\@ifpackageloaded{iftex}{}{\RequirePackage{iftex}}
\unless\ifPDFTeX
    \PackageError{pm-isomath}{%
```

```
      ****************************************\MessageBreak
      This package should be used only when    \MessageBreak
      typesetting with pdfLaTeX.               \MessageBreak
      Skipping loading the package             \MessageBreak
      ****************************************\MessageBreak
   }{%
      ****************************************\MessageBreak
      Press the X key and restart typesetting  \MessageBreak
      while using pdfLaTeX\MessageBreak
      ****************************************\MessageBreak
   }
\expandafeter\@firstoftwo
\else
   \PackageInfo{pm-isomath}{%
      ****************************************\MessageBreak
      Typesetting this document with pdfLaTeX! \MessageBreak
      ****************************************\MessageBreak
   }
\expandafter\@secondoftwo
\fi
{\endinput}{\relax}
```

Actually this package accepts an option: `engineer`. This option is for deciding if the imaginary unit should be defined as 'i' or 'j'. As we have remarked in the previous documentation, engineers, especially those who deal with electricity and electrical quantities, but also electronics, control and telecommunications engineers, use 'j'; all these varieties of engineers could not do anything in their profession if they don't use complex numbers and quantities (the latter called phasors). Possibly they are the applied scientists who use complex numbers more than any other scientist. Note: this option has *not* been used to prepare this very document.

```
\newif\ifengineer \engineerfalse
\DeclareOption{engineer}{\engineertrue}
\ProcessOptions*\relax
```

Are the necessary packages already loaded?

```
\@ifpackageloaded{amsmath}{}{\RequirePackage{amsmath}}
\@ifpackageloaded{etoolbox}{}{\RequirePackage{etoolbox}}
\@ifpackageloaded{xparse}{}{\RequirePackage{xparse}}
```

Now we have almost all software instruments available. We define a macro to switch the definitions of certain math Greek symbols; some of these are defined in the LaTeX kernel: the lowercase Greek variant letters; some others are defined in the `amsmath` package: the uppercase slanted greek letters. All these variant letters have a name identical to the regular ones but prefixed with the string `var`; example `\epsilon` and `\varepsilon`, `\Omega` and `\varOmega`. We switch the control sequence definitions between the `var`-less ones and the `var`-prefixed ones. The first group of lowercase letters is switched because the glyphs give a better match with those produced with the textual Greek glyphs obtained when the ISO macros are used.

```
\newcommand\switchvarsymbols[1]{%
\letcs{\tempA}{#1}\csletcs{#1}{var#1}\cslet{var#1}{\tempA}}
%%%%
\switchvarsymbols{epsilon}
\switchvarsymbols{theta}
\switchvarsymbols{rho}
\switchvarsymbols{phi}
%%%%%
\switchvarsymbols{Gamma}
\switchvarsymbols{Delta}
\switchvarsymbols{Theta}
\switchvarsymbols{Lambda}
\switchvarsymbols{Xi}
\switchvarsymbols{Pi}
\switchvarsymbols{Sigma}
\switchvarsymbols{Upsilon}
\switchvarsymbols{Phi}
\switchvarsymbols{Psi}
\switchvarsymbols{Omega}
```

Eventually we load the `alphabeta` package; this package allows using the same macros used in math mode while typesetting in text mode. We find it very useful in this package.

```
\@ifpackageloaded{alphabeta}{}{\RequirePackage{alphabeta}}
```

The next line defines the default family, series and shape to be used in the macros that follow; as it can be seen the default family is the Latin Modern regular (or roman); the series is medium and the shape is normal (or upright). The codes used are the same used in the font description files with extension `.fd`. The name of these `.fd` files is obtained by merging the encoding name with the family name; therefore the default font description file for Latin characters is `t1lmr.fd` while the one for Greek characters is `lgrlmr.fd`; these files define the series they contain and that are identified with codes such as `m` (medium); `bx` (bold extended); `b` (bold). Other fonts, with different series may have also other codes. For each series the `.fd` defines the codes for shapes, and for every valid combination of series, shape and size it defines the specific font file to use.

We should not care for the font names, but in order to use different font families, series, and shapes the user should know their codes. this is generally a difficult task, but not impossible; it "suffices" to open the packages that allow to use the desired fonts, read the code and find out the names of the `.fd` files; then search these files on the trees of the TEX system, and eventually find out the codes for the available series and shapes.

For the Latin and Greek `.fd` files we have the series and shapes shown in table 2.

```
\def\ISOfam{lmr}\def\ISOser{m}\def\ISOsha{n}
```

As explained in the initial documentation all font changing commands are constructed in such a way as to have a default family, series and shape common to both Latin and Greek fonts; therefore with three optional arguments that the

		Latin (T1)		Greek (LGR)	
Series	Code	Shape	Code	Shape	Code
medium	m	normal	n	normal	n
		italics	it	italics	it
		slanted	sl	slanted	sl
				lipsian	li
				serif	rs
				serif oblique	ro
		upright italics	ui	upright italics	ui
		small caps	sc	small caps	sc
bold	b			lipsian	li
		normal	n		
		slanted	sl		
bold extended	bx	normal	n	normal	n
		italics	it	italics	it
		slanted	sl	slanted	sl
				lipsian	li
				serif	rs
				serif oblique	rs
				upright italics	ui
				small caps	sc

Table 2: Seres and shapes available with the Latin Modern regular family with Latin and Greek fonts

user can specify with different delimiters but respecting their order, the user can get eight different choice combinations that allow the selection of a large number of different looks.

We now define the main and service macros that allow such font selection; we have to create similar macros that mostly differ in the encoding choice for Latin or Greek letters.

The user macros are defined by means of the defining commands provided by the `xparse` package, while the service macros use normal LaTeX commands. The user commands follow this special syntax:

`\MathLatin{`⟨*Latin letter*⟩`}{`⟨*family*⟩`}[`⟨*series*⟩`](`⟨*shape*⟩`)`
`\MathGreek{`⟨*Greek letter*⟩`}{`⟨*family*⟩`}[`⟨*series*⟩`](`⟨*shape*⟩`)`

where in both cases the last three arguments are differently delimited optional values, even the first one of the three, in spite of being delimited by curly braces. In both cases the only mandatory argument is the Latin or Greek letter; the latter one may be specified by the macros `\alpha`, `\beta`,..., `\Omega`, the same ones that are normally used in math (although they are going to be used in textmode).

```
56 \NewDocumentCommand\MathLatin{m gO{m}D(){it}}{\bgroup
```

```
\edef\y{\IfNoValueTF{#2}{\ISOfam}{#2}}%
\edef\x{\noexpand\egroup\noexpand\MLatin{\noexpand#1}{\y}}\x{#3}{#4}}

\providecommand\MLatin[4]{\text{%
\def\ISOfam{#2}\def\ISOser{#3}\def\ISOsha{#4}%
\ifcsstring{math@version}{bold}{\def\ISOser{bx}}{}%
\usefont{T1}{\ISOfam}{\ISOser}{\ISOsha}#1}}

\NewDocumentCommand\MathGreek{ m g O{m} d()}{%
\edef\y{\IfNoValueTF{#2}{\ISOfam}{#2}}%
\edef\x{\IfNoValueTF{#4}{\ISOsha}{#4}}%
\MGreek{#1}{\y}{#3}{\x}}

\newcommand{\MGreek}[4]{\text{%
\def\ISOfam{#2}\def\ISOser{#3}\def\ISOsha{#4}%
\ifcsstring{math@version}{bold}{\def\ISOser{bx}}{}%
{\usefont{LGR}{\ISOfam}{\ISOser}{\ISOsha}#1}}}

```

We now define the macros for all lowercase Greek letters and several uppercase ones (even some that are identical to some Latin letters) that should save several keystrokes when entering such letters in the source file. The shorter the code to type in, the smaller the the number of potential typos.

```
\newcommand\ISOalpha{\MathGreek{\alpha}}
\newcommand\ISObeta{\MathGreek{\beta}}
\newcommand\ISOgamma{\MathGreek{\gamma}}
\newcommand\ISOdelta{\MathGreek{\delta}}
\newcommand\ISOepsilon{\MathGreek{\epsilon}}
\newcommand\ISOzeta{\MathGreek{\zeta}}
\newcommand\ISOeta{\MathGreek{\eta}}
\newcommand\ISOtheta{\MathGreek{\theta}}
\newcommand\ISOiota{\MathGreek{\iota}}
\newcommand\ISOkappa{\MathGreek{\kappa}}
\newcommand\ISOlambda{\MathGreek{\lambda}}
\newcommand\ISOmu{\MathGreek{\mu}}
\newcommand\ISOnu{\MathGreek{\nu}}
\newcommand\ISOxi{\MathGreek{\xi}}
\newcommand\ISOomicron{\MathGreek{\omicron}}
\newcommand\ISOpi{\MathGreek{\pi}}
\newcommand\ISOrho{\MathGreek{\rho}}
\newcommand\ISOsigma{\MathGreek{\sigma}}
\newcommand\ISOtau{\MathGreek{\tau}}
\newcommand\ISOupsilon{\MathGreek{\upsilon}}
\newcommand\ISOphi{\MathGreek{\phi}}
\newcommand\ISOchi{\MathGreek{\chi}}
\newcommand\ISOpsi{\MathGreek{\psi}}
\newcommand\ISOomega{\MathGreek{\omega}}
\newcommand\ISOGamma{\MathGreek{\Gamma}}
\newcommand\ISODelta{\MathGreek{\Delta}}
\newcommand\ISOEta{\MathGreek{\Eta}}
```

```
\newcommand\ISOTheta{\MathGreek{\Theta}}
\newcommand\ISOLambda{\MathGreek{\Lambda}}
\newcommand\ISOXi{\MathGreek{\Xi}}
\newcommand\ISOPi{\MathGreek{\Pi}}
\newcommand\ISORho{\MathGreek{\Rho}}
\newcommand\ISOSigma{\MathGreek{\Sigma}}
\newcommand\ISOUpsilon{\MathGreek{\Upsilon}}
\newcommand\ISOPhi{\MathGreek{\Phi}}
\newcommand\ISOChi{\MathGreek{\Chi}}
\newcommand\ISOPsi{\MathGreek{\Psi}}
\newcommand\ISOOmega{\MathGreek{\Omega}}
```

We redefine also the `\mathrm` and `\mathit` so that if they have to contain Greek letters the default shape is set equal to that of the enclosing command. Such macros are just those defined in the LaTeX kernel where the `\ISOsha` redefinition is added.

```
\DeclareRobustCommand{\mathrm}%
{\relax\ifmmode\else\expandafter\non@alpherr
\csname mathrm \endcsname\fi
\def\ISOsha{n}\expandafter\use@mathgroup
\csname M@OT1\endcsname\symoperators}

\DeclareRobustCommand{\mathit}%
{\relax\ifmmode\expandafter\non@alpherr
\csname mathit \endcsname\fi
\def\ISOsha{it}\expandafter\use@mathgroup
\csname M@OT1\endcsname{9}}
```

Imitating the `ISOmath` package we define also the macros for selecting the bold italics (produces results similar to those obtained with package `bm`, but it does not require any math group), and sans serif in both normal and bold slanted shape.

```
\AtBeginDocument{%
\providecommand\mathbfit[1]{\MathLatin{#1}{lmr}[bx](it)}
\providecommand\mathsfit[1]{\MathLatin{#1}{lmss}[m](sl)}
\providecommand\mathsfbfit[1]{\MathLatin{#1}{lmss}[bx](sl)}}
```

Package `ISOmath` defines macros for typesetting vectors, matrices and tensors; we do the same, but avoid the abbreviation `sym` in place of `symbol`.

```
\AtBeginDocument{%
\let\vectorsymbol\mathbfit
\let\matrixsymbol\mathbfit
\let\tensorsymbol\mathsfbfit}
```

We now define some macros for setting some elements in the proper fonts; the idea is the same as that for vectors, matrices and tensors, except that these macros produce directly the desired symbol without using arguments, if possible.

The imaginary unit is subject to the state of the `engineer` switch, set with the proper option on calling the package. If such option has been specified in calling this package, the `\iu` command is let to `\junit`, otherwise it's let to `\iunit`. In spite of this option driven aliases, both commands `\iunit` and `\junit` are still available to the user.

The Napier number ‘e’ is defined in roman type, but as an operator; this number is not an operator in the mathematical sense, but it is most often used as the base of an exponential; therefore such math “atom” must be treated as an operator as well as when the official operator macro `\exp` is used.

The transcendental number π, different from the plane angle π, is defined so as to be typeset in upright style; similar definitions may be used for other numerical constants; we define just `\uppi` because we think it is the most used symbol in any kind of mathematics. The $\mathsf{\pi}$ Greek letter in upright sans serif font may indicate the subatomic particle “pion” so that a similar macro, say, `\sspi` or, more expressively, `\pion` may be given a suitable definition.

The differential symbol is not an operator, but it requires a special treatment; a macro `\diff` for the differential symbol uses an empty ‘operator’ and a negative shift to typeset an upright letter ‘d’ with an operator spacing on its left, so that proper spacing is used in math typesetting; notice that the given definition does not perform as the direct use of a thin math space before the upright ‘d’, because spacing between math atoms depends on their category, while the thin space `\,` is absolute and does not change depending on the preceding math atom.

The command `\unit` for appending the units of measure to the numerical value of the measure is added if no packages have already defined it; package `siunitx` is a particularly recommended one. Similar considerations hold true for the `\ap` and `\ped` (apex and pedex, respectively; i.e superscript and subscript); therefore such command definitions are deferred to the start of the document so as to be sure to avoid damaging other package settings. All these commands may be used in both text and math modes; therefore a robust macro `\textormath` is (re)defined even if often such command is already available; the `\DeclareRobustCommand` unconditionally declares and redeclares robust commands even if they are already defined; possibly an information line is written in the `.log` file in case a redefinition takes palace. These commands typeset their arguments in upright math fonts, but with the current font in text mode.

The `\ohm` and `\micro` macros produce their symbols in upright style while in math mode, and with the current font and shape in text mode.

```
\newcommand\iunit{\MathLatin{i}(n)}
\newcommand\junit{\MathLatin{j}(n)}
\ifengineer
   \let\iu\junit
\else
   \let\iu\iunit
\fi
% i
\let\eu\undefined
\DeclareMathOperator\eu{\MathLatin{e}(n)}
\providecommand\uppi{}
\renewcommand\uppi{\ISOpi(n)}
%
\providecommand*\diff{}
\renewcommand*\diff{\ensuremath{\mathop{}\!\MathLatin{d}(n)}}
%
```

```
\providecommand*\micro{}
\renewcommand*\micro{%
\textormath{\ifcsdef{textmicro}{\textmicro}{\ISOmu(rs)}}{\ISOmu(rs)}}
%
\providecommand*\ohm{}
\AtBeginDocument{\@ifpackageloaded{textcomp}{%
\renewcommand*\ohm{\textormath{{\textohm}}{\ISOOmega(n)}}}%
{\renewcommand*\ohm{\textormath{\ISOOmega(\f@shape)}{\ISOOmega(n)}}}}
%
\global\csletcs{bbl@it@ped}{undefined}
\global\csletcs{bbl@it@ap}{undefined}
%
\DeclareRobustCommand\textormath{%
\unless\ifmmode\expandafter\@firstoftwo
\else\expandafter\@secondoftwo\fi}
%
\AfterEndPreamble{\let\ped\undefined\let\ap\undefined
\DeclareRobustCommand*\ped[1]{%
\textormath{\textsubscript{#1}}{_{\mathrm{#1}}}}%
%
\providecommand\ap{}
\DeclareRobustCommand\ap[1]{%
\textormath{\textsuperscript{#1}}{^{\mathrm{#1}}}}%
%
\unless\ifcsname unit\endcsname
\DeclareRobustCommand{\unit}[1]{\,\textormath{#1}{\mathrm{#1}}}
\fi}

\endinput
```

This is the end

HAPPY TEXing!

```
This bundle has the LPPL maintenance status "author-maintained".

The list of all files belonging to the PM-ISOmath bundle is
 pm-isomath.dtx, a README.txt file
with the derived files: pm-isomath.sty, pm-isomath.pdf.

The set of derived (unpacked) files belonging to the distribution
and covered by LPPL is created by the self unpacking file
pm-isomath.dtx which is the principal part of the distribution.

In the TDS composed of various branches they should be in the
following folders:
.../tex/latex/pm-isomath/     contains pm-isomath.sty
.../doc/latex/pm-isomath/     contains pm-isomath.pdf and README.txt
.../source/latex/pm-isomath   contians pm-isomath.dtx
```