System of compact notations for numerical codes

Leonid E. Zakharov, Alex Pletzer,
Princeton University, Plasma Physics Laboratory,
Sergei Galgin,
University of California, San Diego
Andrei S. Kukushkin,
ITER, Max Plank IPP, Garshing, Germany

June 25, 2000

For easy navigation the enumeration in the Table of Contents, and the "(to ToC)" right after the section names are the forward and backward hyperlinks between Table of Contents and the beginning of sections.

Contents

1 Introduction *(to ToC)*

The system of compact notations for mathematical symbols in numerical codes, being used for a number of years (at some lower level of development) by one of the authors (LEZ), provides unambiguous names for variables either in FORTRAN- (case insensitive and referred later as F-) or C- (case sensitive) codes.

Adoption of any formalized system is useful in several aspects. First, it makes name generating easy. Then, it allows to read arbitrary section of the source code without referring to definition of variables or to the lists of notations. It makes documentation of the code more transparent and can be used for converting mathematical documentation into a computer language source of the code in a semi-automatic manner. It is especially useful for compact and efficient GUI, when the mathematical symbols or formulas should be generated inside the code and then appear on the plots or in the I/O Dialog boxes.

The formalism of the *n*ame *g*enerating *s*ystem (being referred as NGS) is consistent with the practice of using compact name substitutions for trivial mathematical symbols, e.g., `a` for a or `a1` for a_1. In other cases, it produces typically more compact names, like `ga` (rather than `alpha`) for α, or `p2ga` for α^2 or can generate names for much more complicated mathematical variables. Note, in this system the name `pi` stands for p_i not for π. Instead, a mathematical variable π is linked with the name `gp`, while the constant $\pi = 3.14159265358\ldots$ has the name `cgp`.

Depending on their position in the name, the system considers some characters as the service symbols, attached to the name of the variable in order to reflect the mathematical meaning of the variable. The formalism allows also to drop from the name some of the "ugly" system service constructions if this keeps interpretation of the simplified names unambiguous.

The described system does not interfere with the definitions of variables or structures in the source code which determine the computer representation of the variables. This system only simplifies the name making process for objects which are related specifically to mathematics of the code algorithm. In this sense, it stays comfortably with use of the English based descriptive names for strings, structures, functions, routines and other non-mathematical objects in the codes.

In this paper, Sects. 2,3 describe the approach in general and introduces special TeX/LaTeX macros, which reflect interpretation of the names and help to convert them into the math typesetting. Sects. 4-6 discuss the syntax and elements of the system more specifically and describe the design of names for simple and composit variables. Sect. 7 specifies the rigorous rules of name interpretation (as they are implemented in the name-interpretor code `ngs`). Sect. 8 gives examples of the names taken from ESC (Equilibrium and Stability Code) for specification of equilibrium plasma magnetic configurations. Sect. 9 gives the summary.

2 General idea of the name design *(to ToC)*

Compactness and ambiguity in names come together. Without a context a simple example of legitimate name

$$\texttt{pa} \tag{2.1}$$

makes a real puzzle in interpretation

$$\texttt{pa}\overset{?}{\rightarrow}pa,\quad \texttt{pa}\overset{?}{\rightarrow}p+a,\quad \texttt{pa}\overset{?}{\rightarrow}p-a,\quad \texttt{pa}\overset{?}{\rightarrow}(p)^a,\quad \texttt{pa}\overset{?}{\rightarrow}(p\rightarrow a),\quad \texttt{pa}\overset{?}{\rightarrow}p_a,\quad \texttt{pa}\overset{?}{\rightarrow}p^a. \tag{2.2}$$

Does this name represent an algebraic or other mathematical combination of 2 variables, or it is a pointer to "`a`", or it is "`p`" with a sub- or super- script "`a`" is impossible to determine. Or may be "`p`" in front of the variable name has a special meaning. Even the author of the code in a while will not answer this question. FORTRAN adds even more ambiguity and confusion by not distinguishing the small and capital characters in the names.

It is a problem for the name design to save informativeness and compactness, which are intrinsic for the math, while removing ambiguity. Note, that some level of ambiguity is reasonable and admissible in favor of compactness. Thus, nobody in the math uses $(p)^a$ for the power of p in order to make it look different from p^a (which could be, e.g., p with a superscript a). But, the total mess in names of math variables in the existing codes lacks them all, compactness, informativeness as well as uniqueness in the name interpretation. Instead of leaving the naming math objects without a guidance, it should be a reasonable reference system of conventions which makes the name design rigorous.

In fact, it is possible to keep names compact and still unambiguous. E.g., in the above example, different possible meanings of the combination of p and a are reflected by the names

$$\begin{aligned} &pa \to \texttt{spa}, \quad p+a \to \texttt{popa}, \quad p-a \to \texttt{poma}, \quad (p)^a \to \texttt{poha}, \\ &(p \to a) \to \texttt{la}, \quad p_a \to \texttt{s1pa}, \quad p^a \to \texttt{s1\_pa}, \end{aligned} \tag{2.3}$$

as explained later in the paper.

The general idea is to consider the name of the math variable as a name-object, whose structure can be presented as

$$< name - object > \equiv \underbrace{[< \texttt{PREFIXES} >] < \texttt{CORE} > [\texttt{SUFFIX}]}_{token-name\ object} \underbrace{[< \texttt{o} - \texttt{OPERATOR} >]}_{\texttt{o}-operator} \underbrace{[< name - object >]}_{\substack{next \\ name-object}}, \tag{2.4}$$

where brackets "[. . .]" specify optional elements.

`<CORE>` is the only obligatory element of the name. It specifies the name of the math variable and, optionally, the lengths of the super- and sub- scripts of the variable.

The optional `<SUFFIX>` object contains both super- and sub- scripts of the variable and has the general form

$$< \texttt{SUFFIX} > \equiv [< \texttt{superscript name} - \texttt{obj} >][< \texttt{underscore} >][< \texttt{subscript name} - \texttt{obj} >]. \tag{2.5}$$

Suffix is attached to the core of the token name.

Optional `<PREFIXES>` can be a series of prefixes

$$< \texttt{PREFIXES} > \equiv [< \texttt{PREFIX} >_1][< \texttt{PREFIX} >_2][< \texttt{PREFIX} >_3] \ldots. \tag{2.6}$$

These three together make a token-name in (2.4), standing, basically, for one math variable (or the math symbol). In the name for combination of variables, other names of math variables can be attached either directly or through the `o`-operators which interpret left and right name-objects together (e.g., like a power or a fraction).

`<CORE>` together with the `<SUFFIX>` and any number of preceding prefixes constitutes the token-name objects (nested into each other). Each of the prefixes acts like an interpretation **operator** applied to the nested token-name and specifies the mathematical meaning of this token-name.

What is called here the `o`-operator (`<o-OPERATOR>`) helps to combine two name-objects together. In interpretation, it acts, like a mathematical symbol in a FORTRAN expression, on both left and right name-objects surrounding it. Unlike the mathematics and FORTRAN, `o`-operator considers everything up to beginning of the name-object (or to the previous `o`-operator) as its left operand. Everything that follows it up to the end of the name-object is considered as the right operand of the `o`-operator. This is illustrated by

$$\underbrace{< name - obj. >}_{\substack{left\ operand \\ for\ o-operator\ 1}} < \texttt{o} - \texttt{OPERATOR 1} > \underbrace{\underbrace{< nm - obj. >}_{\substack{left\ operand \\ for\ o-operator\ 2}} < \texttt{o} - \texttt{OPERATOR 2} > \underbrace{< name - obj. > \ldots}_{\substack{right\ operand \\ for\ o-operator\ 2}}}_{\substack{right\ operand \\ for\ o-operator\ 1}}. \tag{2.7}$$

Thus, unlike the math symbol, the scope of action of the `o`-operator is global.

The format of the name-object is not fixed. Service elements can be dropped if this still keeps interpretation the same or within an admissible level of ambiguity.

The above construction with a sufficient number of prefixes, o-operators and special symbols for the name-objects gives a comprehensive way of name generating which could reflect the mathematical sense of the variables or their combination.

NGS of this paper implements this idea in the name design within the restrictions imposed on the names in the computer codes. The system relies on a set of English characters allowable for the C-code names. In order to distinguish `<PREFIXES>`, `<CORE>` with `<SUFFIX>`, and o-operators in the name, it classifies all C-name characters in the following categories

1. 14 prefix recognition characters

$$\mathtt{b, c, d, e, f, h, p, q, r, t, u, v, w, x} \tag{2.8}$$

2. 5 core recognition characters

$$\mathtt{a, g, s, y, z} \tag{2.9}$$

3. 1 o-operator recognition character

$$\mathtt{o} \tag{2.10}$$

4. 6 pseudo digit characters

$$\mathtt{i, j, k, l, m, n} \tag{2.11}$$

5. 10 digits

$$\mathtt{0} \text{ — } \mathtt{9} \tag{2.12}$$

6. underscore

$$\_ \tag{2.13}$$

7. upper case alphabet characters

$$\mathtt{A} \text{ — } \mathtt{Z.} \tag{2.14}$$

Digits and pseudo digits together will be referred as ExtDigits. By default, ExtDigits serve as the arguments for 3 first categories of characters. Underscore is used, primarily, in arguments or in order to distinguish between sub- and super- scripts in default suffixes. In other cases, underscore represents a void token-name or serves as a decimal point in digital token-names. Upper case letters are considered as the symbols of math variables in C-names. In F-names, they are interpreted in the same manner as the low case letters.

There are no suffix recognition characters in the system. Instead certain conventions are introduced for the length of the suffix. Thus, by default, any ExtDigit sequence following the core is considered as the suffix.

As the result, this approach allows to reflect the structure of the name-object without use of special symbols, while still retaining compactness of the name.

In the process of name interpretations, each character (with its ExtDigital optional argument), first, is interpreted as the prefix. If this fails, it is interpreted as the recognition character for the core. If this fails, the character is interpreted as the symbol of the math variable in the core. Suffix is then recognized with use of numerical argument in the core or by default conventions in the system. After the token-name has been interpreted, the remaining characters are interpreted either as the o-operators or as the next name-object.

The following example of the name may serves as an illustration (without the full explanation of the system)

$$\mathtt{q3h2d2a1b} \equiv \underbrace{\mathtt{q3h2d2a1}}_{\substack{token\\name}}\underbrace{\mathtt{b}}_{\substack{token\\name}}, \tag{2.15}$$

where underbraces show the token name objects. The first of them has the following structure

$$\texttt{q3h2d2a1} \equiv \underbrace{\texttt{q3h2d2}}_{prefix}\underbrace{\texttt{a}}_{core}\underbrace{\texttt{1}}_{suffix}. \tag{2.16}$$

Now, it is possible to explain, that all `q`, `h`, `d` (by convention in the system) are the prefix characters, while `a` is not. This uniquely determines the beginning of the first core, i.e., the character `a`. First, `a` is interpreted as the core recognition character. This interpretation fails: the number after "`a`" is the length of the subscript which should follow the symbol of the math variable "`b`". This is inconsistent with the absence of characters after "`b`". Thus, "`a`" is interpreted as the symbol of the variable.

Again by convention, a number after the core is a suffix (without the underscore, a subscript). The next core has no prefixes. Although "`b`" is the prefix character, it cannot be interpreted this way, and in this example represents the second core.

The prefixes to the first core are interpreted in the following way, starting from the last one

$$\texttt{d2a1} \equiv \texttt{d2}\underbrace{\texttt{a1}}_{\substack{token\\name}} \to a_1'', \quad \texttt{h2}\underbrace{\texttt{d2a1}}_{\substack{token\\name}} \to \frac{a_1''}{2}, \quad \texttt{q3}\underbrace{\texttt{h2d2a1}}_{\substack{token\\name}} \to \left(\frac{a_1''}{2}\right)^{\frac{1}{3}} \tag{2.17}$$

and the whole name as

$$\texttt{q3h2d2a1b} \to \left(\frac{a_1''}{2}\right)^{\frac{1}{3}} b. \tag{2.18}$$

Although the entire name looks quite ugly, every symbol in these name is meaningful.

3 NGS and compact TeX/LaTeX macros *(to ToC)*

TeX/LaTeX provides comprehensive capabilities for expressing the meaning of the math variables when the descriptive commands of the language are converted in the typesetting. NGS has some common ideas with TeX/LaTeX. Although, being much simpler, NGS does not require the knowledge of TeX/LaTeX, it is useful to demonstrate the similarity.

3.1 Prefixes and TeX/LaTeX macros *(to ToC)*

It is possible to notice that the interpretation action of prefixes is analogous to the TeX/LaTeX macros, where the syntax is such that the name of macros is followed by the argument the macro is acting on. In order to make the analogy more obvious, we introduce some compact abbreviations for typical math constructions in TeX/LaTeX which help to convert the prefix interpretation into a typesetting (some exceptions will be explained in the following sections). Note, that vice versa these TeX/LaTeX expressions for typesetting can be easily converted into names of NGS.

Compact TeX/LaTeX macros					**Table 1**
Use	Math	TeX/Latex definition	Use	Math	TeX/Latex definition
`\b{a}`	$\bar{a}$	`\def\b#1{\bar{#1}}`			undefined
`\c{a}`	$\texttt{a}$	`\def\c#1{{\tt #1}}`	`\C{3}{a}`	$3a$	`\def\C#1#2{{\tt #1#2}}`
`\d{a}`	$\dot{a}$	`\def\d#1{{\dot{#1}}}`	`\D{3}{a}`	$a^{(3)}$	`\def\D#1#2{{{#2}^{(#1)}}}`
`\e{a}`	e^a	`\def\e#1{e^{#1}}}`	`\E{2}{a}`	2^a	`\def\E#1#2{{#2}^{#1}}`
`\h{a}`	$\widehat{a}$	`\def\h#1{\hat{#1}}`	`\H{3}{a}`	$\frac{a}{3}$	`\def\H#1#2{{#2\over#1}}`
`\p{a}`	a^2	`\def\p#1{{#1}^2}`	`\P{3}{a}`	a^3	`\def\P#1#2{{{#2}^{#1}}}`
`\q{a}`	$\sqrt{a}$	`\def\q#1{\sqrt{#1}}`	`\Q{3}{a}`	$a^{\frac{1}{3}}$	`\def\Q#1#2{{#2^{1\over#1}}}`
`\r{a}`	$\frac{1}{a}$	`\def\r#1{{1\over #1}}`	`\R{3}{a}`	$\frac{3}{a}$	`\def\R#1#2{{#1\over #2}}`
`\t{a}`	$\widetilde{a}$	`\def\t#1{\tilde{#1}}`			undefined
`\u{a}`	$\underline{a}$	`\def\u#1{\underline{#1}}`	`\U{3}{A}`	$\mathsf{A}_{[3]}$	`\def\U#1#2{{\mfnt{#2}_{[#1]}}}`
`\v{a}`	$\vec{a}$	`\def\v#1{\vec{#1}}`	`\V{3}{A}`	$\mathsf{A}^{[3]}$	`\def\V#1#2{{\mfnt{#2}^{[#1]}}}`
`\w{a}`	$\mathbf{a}$	`\def\w#1{{\bf #1}}`	`\W{3}{a}`	$\mathbf{a}^{[3]}$	`\def\W#1#2{{\bf #2}^{[#1]}}`
`\x{a}`	a^*	`\def\x#1{{#1}^*}`	`\X{3}{a}`	$3a$	`\def\X#1#2{{#1#2}}`
`\cm{a}`	$-\texttt{a}$	`\def\cm#1{{-#1}}`	`\Cm{3}{a}`	$-\texttt{3a}$	`\def\Cm#1#2{{\tt -#1#2}}`

The left column of macros uses the full set of prefix characters (except those starting with "`f`", which are described separately) for naming the macros. In the present name generating system NGS, these macros specify the meaning of the corresponding prefix characters, when they are not followed by an argument, and can be uses for converting the names into the proper TeX/LaTeX typesetting, e.g.,

$$\underbrace{\texttt{qhxa}}_{token\ name} \;\to\; \underbrace{\texttt{\textbackslash q\{\textbackslash h\{\textbackslash x\{a\}\}\}}}_{LaTeX} \to \underbrace{\sqrt{\widehat{a^*}}}_{\substack{type-\\setting}}\,. \tag{3.1}$$

The right column represents a set of macros with their names based on the capital version of the prefix characters. These macros have 2 arguments and their meaning is illustrated in the **Table 1**. The NGS does not use the capital characters for the prefixes. Instead, the same small character prefixes *with* an (ExtDigital) argument are interpreted in the sense of the right column of the **Table 1**, e.g.,

$$\underbrace{\texttt{q3hmx3a}}_{token\ name} \;\to\; \underbrace{\texttt{\textbackslash Q\{3\}\{\textbackslash left(\textbackslash H\{m\}\{\textbackslash X\{3\}\{a\}\}\}\textbackslash right)\}}}_{LaTeX} \to \underbrace{\left(\frac{3a}{m}\right)^{\frac{1}{3}}}_{\substack{type-\\setting}}\,. \tag{3.2}$$

Two prefix characters "`b`" and "`t`" have no corresponding TeX/LaTeX capital name macros. In the NGS, the prefix "`b`" with a simple numerical argument, e.g. `b3`, is interpreted in a special way when the next 3 characters are considered verbatim as the core. Similarly, the prefix "`t`" with the simple numerical argument, e.g. `t3`, specifies the number of following characters which are interpreted as the (nested) name-object

$$\underbrace{\texttt{qb9factorial}}_{tokenname} \to \underbrace{\sqrt{\texttt{factorial}}}_{\substack{type-\\setting}}, \quad \underbrace{\texttt{pd1t3asb}}_{tokenname} \to \underbrace{\left[(ab)'\right]^2}_{\substack{type-\\setting}}\,. \tag{3.3}$$

We leave detailed explanation for the following sections.

In **Table 1** there are 2 two-character macro names "`\cm`" and "`\Cm`". In NGS, the prefix "`c`" stands for constant variables. As an exception, in combination with "`c`" the ExtDigit character "`m`" is considered as the "minus" sign, not as the argument to the prefix `c`.

Prefixes and corresponding macros, starting with "`f`" (standing for "function"), contain 2 characters. At present only a few of them are specified. In the **Table 1a** unspecified functions are replaced temporarily by the `#` character.

2 character TeX/LaTeX macros							Table 1a
Use	Math	Use	Math	Use	Math	Use	Math
`\fa{a}`	$\vert a\vert$	`\Fa{3}{a}`	$\vert 3a\vert$	`\fn{a}`	#	`\Fn{3}{a}`	#
`\fb{a}`	#	`\Fb{3}{a}`	#	`\fo{a}`	#	`\Fo{3}{a}`	#
`\fc{a}`	$\cos a$	`\Fc{3}{a}`	$\cos 3a$	`\fp{a}`	#	`\Fp{3}{a}`	#
`\fd{a}`	#	`\Fb{3}{a}`	#	`\fq{a}`	#	`\Fq{3}{a}`	#
`\fe{a}`	#	`\Fe{3}{a}`	#	`\fr{a}`	#	`\Fr{3}{a}`	#
`\ff{a}`	#	`\Ff{3}{a}`	#	`\fs{a}`	$\sin a$	`\Fs{3}{a}`	$\sin 3a$
`\fg{a}`	#	`\Fg{3}{a}`	#	`\ft{a}`	#	`\Ft{3}{a}`	#
`\fh{a}`	#	`\Fh{3}{a}`	#	`\fu{a}`	#	`\Fu{3}{a}`	#
`\fI{a}`	#	`\Fi{3}{a}`	#	`\fv{a}`	#	`\Fv{3}{a}`	#
`\fj{a}`	#	`\Fj{3}{a}`	#	`\fw{a}`	#	`\Fw{3}{a}`	#
`\fk{a}`	#	`\Fk{3}{a}`	#	`\fx{a}`	#	`\Fx{3}{a}`	#
`\fl{a}`	$\ln a$	`\Fl{2}{a}`	$\log_2 a$	`\fy{a}`	#	`\Fy{3}{a}`	#
`\fm{a}`	#	`\Fm{3}{a}`	#	`\fz{a}`	#	`\Fz{3}{a}`	#

Note, that the macro "`\fI`" has the capital letter in its name because TeX contains a token `\fi`, which cannot be overridden. **Table 1** and **Table 1a** cover all prefix characters.

3.2 TeX/LaTeX macros for core recognition characters *(to ToC)*

In NGS the characters "`g`" and "`z`" serve for recognition of Greek and Russian math symbols. **Table 2** specifies TeX/LaTeX macros associated with these characters

TeX/LaTeX macros for Greek and Russian symbols													Table 2		
`\ga`	α	`\gA`	A	`\gn`	ν	`\gN`	N	`\za`	а	`\zA`	**А**	`\zn`	н	`\zN`	**Н**
`\gb`	β	`\gB`	B	`\go`	o	`\gO`	O	`\zb`	б	`\zB`	**Б**	`\zo`	о	`\zO`	**О**
`\gc`	χ	`\gC`	X	`\gp`	π	`\gP`	Π	`\zc`	ц	`\zC`	**Ц**	`\zp`	п	`\zP`	**П**
`\gd`	δ	`\gD`	Δ	`\gq`	θ	`\gQ`	Θ	`\zd`	д	`\zD`	**Д**	`\zq`	ч	`\zQ`	**Ч**
`\ge`	ϵ	`\gE`	E	`\gr`	ρ	`\gR`	R	`\ze`	е	`\zE`	**Е**	`\zr`	р	`\zR`	**Р**
`\gf`	φ	`\gF`	Φ	`\gs`	σ	`\gS`	Σ	`\zf`	ф	`\zF`	**Ф**	`\zs`	с	`\zS`	**С**
`\gg`	γ	`\gG`	Γ	`\gt`	τ	`\gT`	T	`\zg`	г	`\zG`	**Г**	`\zt`	т	`\zT`	**Т**
`\gh`	η	`\gH`	H	`\gu`	υ	`\gU`	Υ	`\zh`	х	`\zH`	**Х**	`\zu`	у	`\zU`	**У**
`\gi`	ι	`\gI`	I	`\gv`	ϖ	`\gV`	ς	`\zi`	и	`\zI`	**И**	`\zv`	в	`\zV`	**В**
`\gj`	ϕ	`\gJ`	ϑ	`\gw`	ω	`\gW`	Ω	`\zj`	j	`\zJ`	**J**	`\zw`	щ	`\zW`	**Щ**
`\gk`	κ	`\gK`	K	`\gx`	ξ	`\gX`	Ξ	`\zk`	к	`\zK`	**К**	`\zx`	ш	`\zX`	**Ш**
`\gl`	λ	`\gL`	Λ	`\gy`	ψ	`\gY`	Ψ	`\zl`	л	`\zL`	**Л**	`\zy`	ы	`\zY`	**Ы**
`\gm`	μ	`\gM`	M	`\gz`	ζ	`\gZ`	Z	`\zm`	м	`\zM`	**М**	`\zz`	з	`\zZ`	**З**

Because in new definitions the macro `\gg` stands for γ and, thus, overrides the original TeX/LaTeX definition for the symbol "$\gg$" ("much greater"), the macros of **Table 2**, if used in the TeX/LaTex style-file, should be complemented by a new command `\muchg`, introduced for this symbol

$$\texttt{\textbackslash mathchardef\textbackslash muchg} = '' \texttt{321D}. \tag{3.4}$$

The character "`s`" and "`a`" serve for recognition of small and capital English math variables and, thus, have no special macros associated with them.

The character "`y`" is considered as the core recognition character for the special mathematical symbols. The set of these symbols is not completely specified yet. In **Table 3** undefined `y`-simbols are replaced temporarily by `@`.

				TeX/LaTeX macros for special symbols				**Table 3**
Use	Math	TeX/Latex	Use	Math	TeX/Latex	Use	Math	TeX/Latex
`\ya`	&	`\def\ya{&}`	`\yj`	@	`\def\yj{@}`	`\ys`	$\sum$	`\def\ys{\sum}`
`\yb`	@	`\def\yb{@}`	`\yk`	@	`\def\yk{@}`	`\yt`	Δ	`\def\yt{\gD}`
`\yc`	$\cdot$	`\def\yc{\cdot}`	`\yl`	@	`\def\yl{@}`	`\yu`	@	`\def\yu{@}`
`\yd`	.	`\def\yd{.}`	`\ym`	$-$	`\def\ym{-}`	`\yv`	@	`\def\yv{@}`
`\ye`	$=$	`\def\ye{=}`	`\yn`	$\prod$	`\def\yn{\prod}`	`\yw`	@	`\def\yw{@}`
`\yf`	/	`\def\yf{/}`	`\yo`	$\oint$	`\def\yo{\oint}`	`\yx`	$\times$	`\def\yx{\times}`
`\yg`	∇	`\def\yg{\nabla}`	`\yp`	$+$	`\def\yp{+}`	`\yy`	@	`\def\yy{@}`
`\yh`	$\hat{}$	`\def\yh{\h{}}`	`\yq`	@	`\def\yq{@}`	`\yz`	@	`\def\yz{@}`
`\yi`	$\int$	`\def\yi{\int}`	`\yr`	$\parallel$	`\def\yr{$\parallel$}`			

Here, the macro `\yt`, corresponding to combination `yt` in names, (mnemonics "triangle") for Laplacian Δ was introduced specially for F-names, which otherwise are incapable of reproducing the capital Greek letters. For the sake of uniformity the same macro is preferable for Laplacian in C-names, while `gD` for the Greek symbol Δ may be used for names of increments or finite differences.

3.3 `o`-operators and TeX/LaTeX macros *(to ToC)*

The remaining character "`o`" is reserved for recognition of `o`-operators acting on their left and right arguments, which are the name-objects (see, (2.7))

$$\underbrace{< \ldots >}_{argument\ 1} < \mathtt{o} - operator > \underbrace{< \ldots >}_{argument\ 2}\ . \tag{3.5}$$

The effect of `o`-operators cannot be simulated by the TeX/LaTeX macros (which have all arguments following the macro name). While such a construct as `o`-operator are useless for TeX/LaTeX, the name generating system can benefit from its use.

At present, only a few `o`-symbols are specified. One of them, `\oh`, stays for for the power symbol "hat", and another, `\of`, stays for the fraction line, `\oR` stays for logical "or" (we use the capital `R` in this macro because TeX has a token `\or`). In **Table4** unspecified `o`-operators are replaced by the `$` character.

				Reserved TeX/LaTeX definitions				**Table 4**
Use	Math	TeX/Latex	Use	Math	TeX/Latex	Use	Math	TeX/Latex
`\oa`	&	`\def\oa{&}`	`\oj`	\$	`\def\oj{$}`	`\os`	\$	`\def\os{$}`
`\ob`	\$	`\def\ob{$}`	`\ok`	\$	`\def\ok{$}`	`\ot`	\$	`\def\ot{$}`
`\oc`	$\cdot$	`\def\oc{\cdot}`	`\ol`	\$	`\def\ol{$}`	`\ou`	\$	`\def\ou{$}`
`\od`	.	`\def\od{.}`	`\om`	$-$	`\def\om{-}`	`\ov`	\$	`\def\ov{$}`
`\oe`	$=$	`\def\oe{=}`	`\on`	\$	`\def\on{$}`	`\ow`	\$	`\def\ow{$}`
`\of`	/	`\def\of{/}`	`\oo`	\$	`\def\oo{$}`	`\ox`	$\times$	`\def\ox{\times}`
`\og`	\$	`\def\og{$}`	`\op`	$+$	`\def\op{+}`	`\oy`	\$	`\def\oy{$}`
`\oh`	$\hat{}$	`\def\oh{\h{}}`	`\oq`	\$	`\def\oq{$}`	`\oz`	\$	`\def\oz{$}`
`\oi`	\$	`\def\oi{$}`	`\oR`	$\parallel$	`\def\oR{\parallel}`			

Here, the `o`-operators are represented by the same symbols as `y`-symbols in **Table 3**. In NGS the meaning for them is different, with `y`-symbols having local affect (as in math formulas) and `o`-operators having global effect in interpretation.

The `o`-operators with the argument are not introduced yet in the NGS.

4 Token names for simple variables *(to ToC)*

The TeX/LaTeX macros, introduced in the previous section, allows to express the mathematical objects in a compact and practically comprehensive form. Unlike TeX/LaTeX constructions, names in numerical codes have important restrictions

- "\", which in TeX/LaTeX distinguishes service symbols from significant symbols, is not allowed;
- "^", which specifies the superscripts, is not allowed;
- "{", "}", which specify the scope of action of a preceding command or macros, are not allowed;
- the mathematical symbols, binding different variables, are not allowed;
- only alphabetic, "_", and digits are allowed in names;
- in addition, F-names are case insensitive.

These restrictions create difficulties for the name generation and are essentially the reason of the mess in notations in existing numerical codes. So, one of the problem for the syntax of the name generating system is to mimic, as much as possible, the TeX/LaTeX capabilities within the restrictions on the name design.

Another problem is, that unlike the math, numerical codes frequently require names for composit (typically, temporary) variables, like

$$\mathtt{aoab} \rightarrow a + b, \quad \mathtt{vaoxvb} \rightarrow \overrightarrow{b} \times \overrightarrow{b}, \tag{4.1}$$

which may include names of several mathematical variables binded by mathematical symbols not permitted in the names for numerical codes.

For this reason, in the given system (2.4-2.6), the names of the code variables are considered, in general, as a combination of "simple", or token, names, each corresponding either to the math variable or to the special symbols (encoded as "`y`" and a letter) and `o`-operators specifying the bonds between them.

In this section, we consider the name construction for standing alone variables, or the token-names. The token name consists from

1. optional prefix (-es)
2. core
3. optional suffix (-es)

as shown earlier in (2.4). Typically, prefixes and the core are recognized using their recognition characters. Also, the core may contain information about the length of the suffix. Otherwise, any sequence of ExtDigits after the core is considered as the (default) suffix. Each of elements id discussed in the following subsections.

4.1 Core of the token name *(to ToC)*

The core specifies the name of the math variable and optionally the length of the super- and sub- scripts. The core has the following structure

$$< \mathtt{CORE} > \equiv \underbrace{[< \mathtt{low\ case\ CRC} >]}_{\substack{core\ recognition\\ character}} \underbrace{[< \mathtt{N_1\_N_2} >]}_{argument} \underbrace{< \mathtt{low/upper\ case\ letter} >}_{core\ symbol}. \tag{4.2}$$

Here, "`CRC`" stands for the core recognition character. In the argument, $\mathtt{N}_1$ and $\mathtt{N}_2$ are the numbers specifying the lengths of the super- and sub- scripts. Each of them as well as the underscore are optional. Mixing low and upper case letters for the core symbol in FORTRAN-names does not affect name interpretation and should be used with a caution.

If its is necessary to distinguish the core from the prefixes or to have an argument in the core, the beginning of the core should start with one of 5 core recognition characters: `s,a,g,z,y`. The `o`-operator starts with "`o`". Simplest examples, explaining the meaning of the core recognition characters of the names are given in **Table 5**.

Core recognition characters of the token name				**Table 5**
C-name	F-name	In Math	LaTeX	Meaning
`a,(sa)`	`a,(sa)`	a	a	small "a"
`A`	`aa`	A	A	capital "a"
`ga`	`ga`	α	\ga	Greek "a"
`gA`	N/A	A	\gA	"Capital" Greek "a"
`za`	`za`	a	\za	Russian "a"
`zA`	N/A	a	\zA	Russian "A"
`ys`	`ys`	$\sum$	\ys	special symbols
`oh`	`oh`	$\frown$	\oh	interpretation operator with left and right arguments

The core recognition character can be dropped if this does not create ambiguity in the name interpretation, e.g.,

$$\texttt{psb} \to b^2, \quad \texttt{pb} \to b^2. \tag{4.3}$$

(Prefix `p` is interpreted as the power of 2). In this example, even without "`s`" the prefix character "`b`" (see **Table 1**) cannot be interpreted as a prefix. Thus, the core recognition character "`s`" can be dropped.

The apparent duality (or in other words, flexibility) in the name for the same object (allowed by the present system) may, in fact, be useful. Thus, it could be desirable to distinguish a simple math variable, e.g., b from the temporary code service variable `b`. NGS allows to use the name "`sb`" for the math variable, while "`b`" for the service variable, or vice versa. The level of ambiguity would be acceptable. It will be much more bug prone to rely on the scope of the variable with the same local and global names in the code.

In C-codes, typically, there is no necessity to use the core recognition character "`a`" for the capital English math variables. Any capital letter in the C-name of variable is considered as the core symbol. E.g., the C-name, `ab` is considered as the name for a combined variable ab rather than the name for B. The necessity of using `a` may appear when it is necessary to have a numerical argument in the core for defining the suffix length (as, e.g., `a4Agagbb` $\to A_{\alpha\beta}b$). Anoother reason for use of the core recognition character `a` in C-names is that without it, NGS uses special rules for recognizing suffix, while use of `a` turn on the standard suffix interpretation

$$\texttt{RextBtor} \to R_{ext}B_{tor}, \quad \texttt{aAorB} \to A \parallel B. \tag{4.4}$$

F-names, which are case insensitive, require `ab` for variable B (by default, the name `b` is interpreted as the name for b). In the present NGS, F-names do not reproduce the capital Greek and Russian characters.

The core recognition character can be followed by a numerical *argument*, which determines the length of the suffix (see more in subsection (4.2) about the suffixes). Thus, the following correspondence is valid

$$\texttt{g2aidb} \to \alpha_{id}b. \tag{4.5}$$

In the absence of the numerical argument in the core, only ExtDigits after "`a`" will be considered as the suffix (subscript) of the name (see, Sect. 7) and, e.g., `gaidb` would stand for the variable $\alpha_i\dot{b}$.

In general, the digital argument (4.2) has the form $\texttt{N}_1\texttt{\_N}_2$, where $\texttt{N}_1$ and $\texttt{N}_2$ are the numbers *not starting* with `0`. There 4 possibilities for numerical arguments

$$\texttt{N}_2, \quad \texttt{\_N}_2, \quad \texttt{N}_1\texttt{\_}, \quad \texttt{N}_1\texttt{\_N}_2. \tag{4.6}$$

In the first and second cases, $\mathtt{N}_2$ specifies how much characters after the core belongs to the subscript. In the third case, $\mathtt{N}_1$ specifies the length of the superscript. In the forth case, $\mathtt{N}_1$ is the length of the superscript, while $\mathtt{N}_2$ is that of the subscript. In all cases, when the *length is specified*, sub- and super-scripts are interpreted as the *name-objects*.

Illustrative examples are

$$\mathtt{a3Agaib} \to A_{\alpha_i} b, \quad \mathtt{a\_3Agaib} \to A_{\alpha_i} b, \quad \mathtt{a2\_Ajb} \to A^{jb}, \quad \mathtt{a2\_3Ajbgai} \to A^{jb}_{\alpha_i}. \tag{4.7}$$

If the length of the suffix is inconsistent with the number of characters behind the core symbol, then the CRC is considered as the core symbol, e.g.,

$$\mathtt{g22rqg} \to \frac{g_{22}}{\sqrt{g}}. \tag{4.8}$$

Because `0` is not allowed as the leading character in the argument, the core recognition character with a leading `0` in its argument is considered as the core symbol, as in the following examples

$$\mathtt{g01ab} \to g_{01} ab, \quad \mathtt{g1\_01abk} \to g^1_{01} ab_k. \tag{4.9}$$

In C-names, if "`a`" is followed by a digital argument, it is, first, interpreted as the core recognition character. In accordance with this, the following is valid

$$\mathtt{ap} \to ap, \quad \mathtt{a2A} \to a_2 A, \quad \mathtt{a2A1} \to a_2 A_1, \quad \mathtt{a2A22} \to A_{22}. \tag{4.10}$$

Here, in the first example, "`a`" as the core recognition character is inconsistent with low case next character. Thus, the name stands for a composit variable. Two next examples does not allow to interpret "`a`" as the core recognition symbol (suffix is too short) and the name stands for a composit variable. In the last example, this interpretation went successful (it will be seen later that the simpler C-name "`A22`" has the same interpretation A_{22}).

Note, that unlike prefixes (see, subsection 4.3), the core recognition characters `a,s,g,z` does not allow the ExtDigital extension in their arguments, e.g.,

$$\mathtt{g3mab} \to g_{3m} ab, \quad \mathtt{g2mab} \to \mu_{ab}. \tag{4.11}$$

Special `y`-symbols can be used similarly to the regular core, except the form and interpretation of the argument to `y`-symbols depends on the symbol itself. Explanation of `y`-symbols without and with an argument is given in **Table 6**, where undefined `y`-simbols are substituted by `@` character.

Special symbols in names					**Table 6**
y-symbol	Math	Meaning	y-symbol	Math	Meaning
`ya`, $\mathtt{yN_1\_N_2a}$	$\&, \&^{sup}_{sub}$	logical "and"	`yn`, $\mathtt{yN_1\_N_2n}$	$\prod, \prod^{sup}_{sub}$	multiple product
`yb`, $\mathtt{yN_1\_N_2b}$	$@, @^{sup}_{sub}$	*undefined*	`yo`, $\mathtt{yN_1\_N_2o}$	$\oint, \oint^{sup}_{sub}$	integral
`yc`, $\mathtt{yN_1\_N_2c}$	$\cdot, \cdot\frac{N_1}{N_2}$	scalar product	`yp`, $\mathtt{yN_1\_N_2p}$	$+, +\frac{N_1}{N_2}$	plus
`yd`, .	.	decimal point	`yq`, $\mathtt{yN_1\_N_2q}$	$@, @^{sup}_{sub}$	*undefined*
`ye`, $\mathtt{yN_1\_N_2e}$	$=, = \frac{N_1}{N_2}$	equal	`yr`, $\mathtt{yN_1\_N_2r}$	$\Vert, \Vert^{sup}_{sub}$	logical "or"
`yf`, $\mathtt{yN_1\_N_2f}$	$/, \frac{sup}{sub}$	fraction	`ys`, $\mathtt{yN_1\_N_2s}$	$\sum, \sum^{sup}_{sub}$	sum
`yg`, $\mathtt{yN_1\_N_2g}$	$\nabla, \nabla^{sup}_{sub}$	gradient (nabla)	`yt`, $\mathtt{yN_1\_N_2t}$	N, N^{sup}_{sub}	triangle (Laplacian)
`yh`, $\mathtt{yN_1\_N_2h}$	$\hat{\ }, \hat{\ }(\frac{sup}{sup})$	power	`yu`, $\mathtt{yN_1\_N_2u}$	$@, @^{sup}_{sub}$	*undefined*
`yi`, $\mathtt{yN_1\_N_2i}$	$\int, \int^{sup}_{sub}$	integral	`yv`, $\mathtt{yN_1\_N_2v}$	$@, @^{sup}_{sub}$	*undefined*
`yj`, $\mathtt{yN_1\_N_2v}$	$@, @^{sup}_{sub}$	*undefined*	`yw`, $\mathtt{yN_1\_N_2v}$	$@, @^{sup}_{sub}$	*undefined*
`yk`, $\mathtt{yN_1\_N_2v}$	$@, @^{sup}_{sub}$	*undefined*	`yx`, $\mathtt{yN_1\_N_2x}$	$\times, \times\frac{N_1}{N_2}$	vector product
`yl`, $\mathtt{yN_1\_N_2v}$	$@, @^{sup}_{sub}$	*undefined*	`yy`, $\mathtt{yN_1\_N_2v}$	$@, @^{sup}_{sub}$	*undefined*
`ym`, $\mathtt{yN_1\_N_2m}$	$-, -\frac{N_1}{N_2}$	minus	`yz`, $\mathtt{yN_1\_N_2v}$	$@, @^{sup}_{sub}$	*undefined*

Numerical argument in special symbols yc,ye,ym,yp,yx interpreted as a corresponding fraction

$$\texttt{ay3i\_2ab} \to a + \frac{3i}{2}b. \tag{4.12}$$

In y-symbols yg,yh,yi,yp,yr,ys,yt the numbers in the argument specify the lengths of super- and sub-scripts (which, in their turn, are considered as the name-objects), e.g.,

$$\texttt{y1\_4sKkye0ak} \to \sum_{k=0}^{K} a_k. \tag{4.13}$$

Interpretation of super- and sub-scripts is given in **Table 6**.

As for ordinary core by convention in NGS any sequence of ExtDigits after the y-symbols with no argument is considered as a suffix, e.g.,

$$\texttt{yskak} \to \sum_k a_k, \quad \texttt{yskmakm} \to \sum_{km} a_{km}. \tag{4.14}$$

Symbols "yh" and "yf" without an argument act (like math symbols in FORTRAN expressions) on the left and right adjacent token-names

$$\begin{aligned} &\texttt{sxyhy} \to x^y, \quad \texttt{r2sxyhpab} \to \left(\frac{2}{x}\right)^{a^2} b, \quad \texttt{ar2sxyhpab} \to a\left(\frac{2}{x}\right)^{a^2} b, \\ &\texttt{sxyfsyi} \to \frac{x}{y_i}, \quad \texttt{sbpsxyfpsyi} \to b\frac{x^2}{y_i^2}. \end{aligned} \tag{4.15}$$

With a numerical argument, they use super- and sub- scripts as their operands and interpret them as the name-objects

$$\begin{aligned} &\texttt{xy4hsigw} \to x^{i\omega}, \quad \texttt{xy3\_hggt} \to x^{\gamma t}, \quad \texttt{xy\_3hd1q} \to x^{\frac{1}{q^\gamma}}, \quad \texttt{xy3\_3hggtd1q} \to x^{\frac{\gamma t}{q^\gamma}}, \\ &\texttt{xy2rpe} \to \frac{x}{e^2}, \quad \texttt{xy2\_rpbpe} \to xb^2e^2, \quad \texttt{xy\_2rpbpe} \to \frac{x}{b^2}e^2, \quad \texttt{xy2\_2rpbpe} \to x\frac{b^2}{e^2}. \end{aligned} \tag{4.16}$$

In NGS there are two additional forms of the core not reflected in (4.2). In sophisticated cases, the core may be specified with use of special prefixes "b" and "t" with a numeric argument "N". So, in fact, the full definition of the core is

$$\begin{aligned} <\texttt{CORE}> \equiv &\underbrace{[<\texttt{low case CRC}>]}_{\substack{\textit{core recognition}\\ \textit{character}}}\underbrace{[<\texttt{N}_1\texttt{\_N}_2>]}_{\textit{argument}}\underbrace{<\texttt{low/upper case letter}>}_{\textit{core symbol}} \\ &||\underbrace{<\texttt{bN}>}_{\substack{\textit{prefix}\\ b}}\underbrace{<\ldots>}_{\substack{N\ \textit{core}\\ \textit{symbols}}} \\ &||\underbrace{<\texttt{tN}>}_{\substack{\textit{prefix}\\ t}}\underbrace{<\ldots>}_{\substack{N\ \textit{core}\\ \textit{symbols}}}, \end{aligned} \tag{4.17}$$

where "||" means "or".

In the case of prefix "b", N following characters are taken verbatim. In the case of prefix "t", N following characters are interpreted as the name-object, e.g.,

$$\texttt{pb6matrix} \to \texttt{matrix}^2, \quad \texttt{pt4ayab} \to (a+b)^2. \tag{4.18}$$

With an underscore in the argument "b" and "t" are interpreted as prefixes and will be discussed in more detail later on.

Symbol "o" is interpreted as the o-operator recognition character and will be discussed in Section 6.

4.2 Sub- and Super-scripts (suffixes) of token names *(to ToC)*

The suffix of the token name reflects either the sub- or super-script of the math variable. By default, any sequence of ExtDigits after the core is considered as the suffix. In this case, the character "_" is designated for distinguishing the sub- and super- scripts. In default suffixes, the conventions are

1. If "_" is absent, then the suffix is considered as a subscript, e.g.,

$$\texttt{gaij2} \to \alpha_{ij2}. \tag{4.19}$$

2. If "_" is present, ExtDigits between the core and "_" are considered as the superscript, and remaining ExtDigits after "_" as the subscript, e.g.,

$$\texttt{gaij\_} \to \alpha^{ij}, \quad \texttt{gai\_j} \to \alpha^{i}_{j}. \tag{4.20}$$

In other cases, the length of the suffix has to be specified by a numerical argument in the core, which may have one of the forms

$$\texttt{N}_2, \quad \texttt{\_N}_2, \quad \texttt{N}_1\texttt{\_}, \quad \texttt{N}_1\texttt{\_N}_2. \tag{4.21}$$

As it was explained already, cases $\texttt{N}_2$ or $\texttt{\_N}_2$ specify the length of the subscript, $\texttt{N}_1\texttt{\_}$ specifies the length of the superscript, while in $\texttt{N}_1\texttt{\_N}_2$, $\texttt{N}_1$ is the length of the superscript and $\texttt{N}_2$ is the length of the subscript. We remind that the leading `0` is not allowed in numerical arguments.

If the length is given, sub- and super-scripts are interpreted as the name-objects. The examples are

$$\texttt{s2bga} \to b^{\alpha}, \quad \texttt{s2\_1bga3} \to b^{\alpha}_{3}, \quad \texttt{s3bgai} \to b^{\alpha_i}, \quad \texttt{s4bgai\_} \to b_{\alpha^i}, \quad \texttt{s4\_bgai\_} \to b^{\alpha^i}. \tag{4.22}$$

As an *exception* to these rules, in C-names, the sequence of small alphabetic characters after the capital core symbol with *no* argument is considered as the suffix. In this case, the underscore separates super- and sub-scripts which are interpreted verbatim, e.g.,

$$\texttt{pAtor} \to A^2_{tor}, \quad \texttt{qAtor\_R} \to \sqrt{A^{tor}}R, \quad \texttt{qAtop\_bottom} \to \sqrt{A^{tor}_{bottom}}. \tag{4.23}$$

4.3 Prefixes to the simple names *(to ToC)*

Prefix (-es) precedes the core of the name and intends to specify the meaning of the math variable. Simplest examples of using names with a prefix are listed in **Tables 7,8**.

Simple variables		Table 7
English→F-, C-name	Greek→F-, C-name	Russian→F-, C-name
a $\vec{a}$ a → a (sa) va (vsa) wa (wsa)	α $\vec{\alpha}$ → ga vga	а $\vec{\text{а}}$ **а** → za vza wza
b $\vec{b}$ b → b (sb) vb (vsb) wb (wsb)	β $\vec{\beta}$ → gb vgb	б $\vec{\text{б}}$ **б** → zb vzb wzb
c $\vec{c}$ c → c (sc) vc (vsc) wc (wsc)	χ $\vec{\chi}$ → gc vgc	ц $\vec{\text{ц}}$ **ц** → zc vzc wzc
d $\vec{d}$ d → d (sd) vd (vsd) wd (wsd)	δ $\vec{\delta}$ → gd vgd	д $\vec{\text{д}}$ **д** → zd vzd wzd
e $\vec{e}$ e → e (se) ve (vse) we (wse)	ϵ $\vec{\epsilon}$ → ge vge	е $\vec{\text{е}}$ **е** → ze vze wze
f $\vec{f}$ f → f (sf) vf (vsf) wf (wsf)	φ $\vec{\varphi}$ → gf vgf	ф $\vec{\text{ф}}$ **ф** → zf vzf wzf
g $\vec{g}$ g → g (sg) vg (vsg) wg (wsg)	γ $\vec{\gamma}$ → gg vgg	г $\vec{\text{г}}$ **г** → zg vzg wzg
h $\vec{h}$ h → h (sh) vh (vsh) wh (wsh)	η $\vec{\eta}$ → gh vgh	х $\vec{\text{х}}$ **х** → zh vzh wzh
i $\vec{\imath}$ i → si ( ) vi (vsi) wi (wsi)	ι $\vec{\iota}$ → gi vgi	и $\vec{\text{и}}$ **и** → zi vzi wzi
j $\vec{\jmath}$ j → sj ( ) vj (vsj) wj (wsj)	ϕ $\vec{\phi}$ → gj vgj	j $\vec{\text{j}}$ **j** → zj vzj wzj
k $\vec{k}$ k → sk ( ) vk (vsk) wk (wsk)	κ $\vec{\kappa}$ → gk vgk	к $\vec{\text{к}}$ **к** → zk vzk wzk
l $\vec{l}$ l → sl ( ) vl (vsl) wl (wsl)	λ $\vec{\lambda}$ → gl vgl	л $\vec{\text{л}}$ **л** → zl vzl wzl
m $\vec{m}$ m → sm ( ) vm (vsm) wm (wsm)	μ $\vec{\mu}$ → gm vgm	м $\vec{\text{м}}$ **м** → zm vzm wzm
n $\vec{n}$ n → sn ( ) vn (vsn) wn (wsn)	ν $\vec{\nu}$ → gn vgn	н $\vec{\text{н}}$ **н** → zn vzn wzn
o $\vec{o}$ o → o (so) vo (vso) wo (wso)	o $\vec{o}$ → go vgo	о $\vec{\text{о}}$ **о** → zo vzo wzo
p $\vec{p}$ p → p (sp) vp (vsp) wp (wsp)	π $\vec{\pi}$ → gp vgp	п $\vec{\text{п}}$ **п** → zp vzp wzp
q $\vec{q}$ q → q (sq) vq (vsq) wq (wsq)	θ $\vec{\theta}$ → gq vgq	ч $\vec{\text{ч}}$ **ч** → zq vzq wzq
r $\vec{r}$ r → r (sr) vr (vsr) wr (wsr)	ρ $\vec{\rho}$ → gr vgr	р $\vec{\text{р}}$ **р** → zr vzr wzr
s $\vec{s}$ s → s (ss) vs (vss) ws (wss)	σ $\vec{\sigma}$ → gs vgs	с $\vec{\text{с}}$ **с** → zs vzs wzs
t $\vec{t}$ t → t (st) vt (vst) wt (wst)	τ $\vec{\tau}$ → gt vgt	т $\vec{\text{т}}$ **т** → zt vzt wzt
u $\vec{u}$ u → u (su) vu (vsu) wu (wsu)	υ $\vec{\upsilon}$ → gu vgu	у $\vec{\text{у}}$ **у** → zu vzu wzu
v $\vec{v}$ v → v (sv) vv (vsv) wv (wsv)	ϖ $\vec{\varpi}$ → gv vgv	в $\vec{\text{в}}$ **в** → zv vzv wzv
w $\vec{w}$ w → w (sw) vw (vsw) ww (wsw)	ω $\vec{\omega}$ → gw vgw	щ $\vec{\text{щ}}$ **щ** → zw vzw wzw
x $\vec{x}$ x → x (sx) vx (vsx) wx (wsx)	ξ $\vec{\xi}$ → gx vgx	ш $\vec{\text{ш}}$ **ш** → zx vzx wzx
y $\vec{y}$ y → y (sy) vy (vsy) wy (wsy)	ψ $\vec{\psi}$ → gy vgy	ы $\vec{\text{ы}}$ **ы** → zy vzy wzy
z $\vec{z}$ z → z (sz) vz (vsz) wz (wsz)	ζ $\vec{\zeta}$ → gz vgz	з $\vec{\text{з}}$ **з** → zz vzz wzz

Small letters coded similarly in F- and C-names.

Simple capital variables			Table 8
English→C-name	English→F-name	Greek→C-name	Russian→C-name
A $\vec{A}$ A → A vA wA	A $\vec{A}$ A → aa vaa waa	A $\vec{A}$ → gA vgA	А $\vec{\text{А}}$ **А** → zA vzA wzA
B $\vec{B}$ B → B vB wB	B $\vec{B}$ B → ab vab wab	B $\vec{B}$ → gB vgB	Б $\vec{\text{Б}}$ **Б** → zB vzB wzB
C $\vec{C}$ C → C vC wC	C $\vec{C}$ C → ac vac wac	X $\vec{X}$ → gC vgC	Ц $\vec{\text{Ц}}$ **Ц** → zC vzC wzC
D $\vec{D}$ D → D vD wD	D $\vec{D}$ D → ad vad wad	Δ $\vec{\Delta}$ → gD vgD	Д $\vec{\text{Д}}$ **Д** → zD vzD wzD
E $\vec{E}$ E → E vE wE	E $\vec{E}$ E → ae vae wae	E $\vec{E}$ → gE vgE	Е $\vec{\text{Е}}$ **Е** → zE vzE wzE
F $\vec{F}$ F → F vF wF	F $\vec{F}$ F → af vaf waf	Φ $\vec{\Phi}$ → gF vgF	Ф $\vec{\text{Ф}}$ **Ф** → zF vzF wzF
G $\vec{G}$ G → G vG wG	G $\vec{G}$ G → ag vag wag	Γ $\vec{\Gamma}$ → gG vgG	Г $\vec{\text{Г}}$ **Г** → zG vzG wzG
H $\vec{H}$ H → H vH wH	H $\vec{H}$ H → ah vah wah	H $\vec{H}$ → gH vgH	Х $\vec{\text{Х}}$ **Х** → zH vzH wzH
I $\vec{I}$ I → I vI wI	I $\vec{I}$ I → ai vai wai	I $\vec{I}$ → gI vgI	И $\vec{\text{И}}$ **И** → zI vzI wzI
J $\vec{J}$ J → J vJ wJ	J $\vec{J}$ J → aj vaj waj	ϑ $\vec{\vartheta}$ → gJ vgJ	J $\vec{\text{J}}$ **J** → zJ vzJ wzJ
K $\vec{K}$ K → K vK wK	K $\vec{K}$ K → ak vak wak	K $\vec{K}$ → gK vgK	К $\vec{\text{К}}$ **К** → zK vzK wzK
L $\vec{L}$ L → L vL wL	L $\vec{L}$ L → al val wal	Λ $\vec{\Lambda}$ → gL vgL	Л $\vec{\text{Л}}$ **Л** → zL vzL wzL
M $\vec{M}$ M → M vM wM	M $\vec{M}$ M → am vam wam	M $\vec{M}$ → gM vgM	М $\vec{\text{М}}$ **М** → zM vzM wzM
N $\vec{N}$ N → N vN wN	N $\vec{N}$ N → an van wan	N $\vec{N}$ → gN vgN	Н $\vec{\text{Н}}$ **Н** → zN vzN wzN
O $\vec{O}$ O → O vO wO	O $\vec{O}$ O → ao vao wao	O $\vec{O}$ → gO vgO	О $\vec{\text{О}}$ **О** → zO vzO wzO
P $\vec{P}$ P → P vP wP	P $\vec{P}$ P → ap vap wap	Π $\vec{\Pi}$ → gP vgP	П $\vec{\text{П}}$ **П** → zP vzP wzP
Q $\vec{Q}$ Q → Q vQ wQ	Q $\vec{Q}$ Q → aq vaq waq	Θ $\vec{\Theta}$ → gQ vgQ	Ч $\vec{\text{Ч}}$ **Ч** → zQ vzQ wzQ
R $\vec{R}$ R → R vR wR	R $\vec{R}$ R → ar var war	R $\vec{R}$ → gR vgR	Р $\vec{\text{Р}}$ **Р** → zR vzR wzR
S $\vec{S}$ S → S vS wS	S $\vec{S}$ S → as vas was	Σ $\vec{\Sigma}$ → gS vgS	С $\vec{\text{С}}$ **С** → zS vzS wzS
T $\vec{T}$ T → T vT wT	T $\vec{T}$ T → at vat wat	T $\vec{T}$ → gT vgT	Т $\vec{\text{Т}}$ **Т** → zT vzT wzT
U $\vec{U}$ U → U vU wU	U $\vec{U}$ U → au vau wau	Υ $\vec{\Upsilon}$ → gU vgU	У $\vec{\text{У}}$ **У** → zU vzU wzU
V $\vec{V}$ V → V vV wV	V $\vec{V}$ V → av vav wav	ς $\vec{\varsigma}$ → gV vgV	В $\vec{\text{В}}$ **В** → zV vzV wzV
W $\vec{W}$ W → W vW wW	W $\vec{W}$ W → aw vaw waw	Ω $\vec{\Omega}$ → gW vgW	Щ $\vec{\text{Щ}}$ **Щ** → zW vzW wzW
X $\vec{X}$ X → X vX wX	X $\vec{X}$ X → ax vax wax	Ξ $\vec{\Xi}$ → gX vgX	Ш $\vec{\text{Ш}}$ **Ш** → zX vzX wzX
Y $\vec{Y}$ Y → Y vY wY	Y $\vec{Y}$ Y → ay vay way	Ψ $\vec{\Psi}$ → gY vgY	Ы $\vec{\text{Ы}}$ **Ы** → zY vzY wzY
Z $\vec{Z}$ Z → Z vZ wZ	Z $\vec{Z}$ Z → az vaz waz	Z $\vec{Z}$ → gZ vgZ	З $\vec{\text{З}}$ **З** → zZ vzZ wzZ

F-names in NGS cannot reproduce the capital Greek and Russian letters.

Prefixes in interpretation act on the part of the token-name following the prefix and, thus, can follow each other. The meaning of prefixes without and with argument is explained in the following **Tables 9,10**. (For typesetting, they have a straightforward analogy in TeX/LaTeX macros of the **Tables 1, 1a**).

Simple prefixes without argument			**Table 9**
C- or F- name	Math	LaTeX	Meaning
`ba`	$\overline{a}$	\b{a}	“a”-bar
`ca`	a	\c{a}	constant “a”
`da`	$\dot{a}$	\d{a}	“a”-dot
`ea`	e^{a}	\e{a}	exponent of “a”
`ha`	$\widehat{a}$	\h{a}	“a”-hat
`pa`	a^{2}	\p{a}	“a”-squared
`qa`	$\sqrt{a}$	\q{a}	sqrt of “a”
`ra`	$\frac{1}{a}$	\r{a}	“a”- reversed
`ta`	$\widetilde{a}$	\t{a}	“a”-tilde
`ua`	$\underline{a}$	\u{a}	underlined (covariant) vector “a”
`va`	$\overrightarrow{a}$	\v{a}	vector (contravariant) “a”
`wa`	$\boldsymbol{\mathsf{a}}$	\w{a}	bold “a”
`xa`	a^{*}	\x{a}	“a”-star

Most of prefixes with the argument have the meaning unrelated to same prefixes without argument as it is explained in **Table 10** with the simplest examples

		Simple prefixes with ExtDigital argument		**Table 10**
C- F- name	Argument	Math	LaTeX	Meaning
`b4dxdy`	$\mathtt{N}_1\_\mathtt{N}_2$	$dxdy$	dxdy	verbatim 4 characters `dxdy`
`c2a`	$\mathtt{D}_1\_\mathtt{D}_2$	$2a$	\C{2}{a}	constant “2a”
`d3a`	$\mathtt{N}_1\_\mathtt{N}_2$	$a^{(3)}$	\D{3}{a}	third derivative of “a”
`e2a`	$\mathtt{D}_1\_\mathtt{D}_2$	2^{a}	\E{2}{a}	double integral of “a”
`h2a`	$\mathtt{D}_1\_\mathtt{D}_2$	$\frac{a}{2}$	\H{2}{a}	“a” divided by 2
`p3a`	$\mathtt{D}_1\_\mathtt{D}_2$	a^{3}	\P{3}{a}	“a” in the 3rd power
`q3a`	$\mathtt{D}_1\_\mathtt{D}_2$	$a^{\frac{1}{3}}$	\Q{3}{a}	1/3-root of “a”
`r2a`	$\mathtt{D}_1\_\mathtt{D}_2$	$\frac{2}{a}$	\R{2}{a}	2 divided by “a”
`t2pa`	$\mathtt{N}_1\_\mathtt{N}_2$	a^{2}	a^{2}	2 character name-object `pa`
`u3a`	$\mathtt{D}_1\_\mathtt{D}_2$	$\mathsf{a}_{[3]}$	\U{3}{a}	3D covariant tensor “a”
`v3a`	$\mathtt{D}_1\_\mathtt{D}_2$	$\mathsf{a}^{[3]}$	\V{3}{a}	3D contravariant tensor “a”
`w3a`	$\mathtt{D}_1\_\mathtt{D}_2$	$\mathsf{a}^{[3]}$	\W{3}{a}	3D matrix “a”
`x2a`	$\mathtt{D}_1\_\mathtt{D}_2$	$2a$	\X{2}{a}	“2a”

Here, “$\mathtt{N}_1$” and “$\mathtt{N}_2$” denote a non-zero integer numbers, while “$\mathtt{D}_1$” and “$\mathtt{D}_2$” represent the sequences of ExtDigits. Except for prefixes `u,v,w`, zero character “`0`” is *not allowed* as the first character in “$\mathtt{N}_{1,2}$” or in “$\mathtt{D}_{1,2}$”, e.g.,

$$\mathtt{r2a} \to \frac{2}{a}, \quad \mathtt{r02a} \to r_{02}a \not\to \frac{02}{a}. \tag{4.24}$$

Prefix “c” has one double character extended form, and “f” prefix has only double character forms. Only a few of them are specified at the moment in NGS. All others are interpreted as *a fictitious* `#`*-function* prefix.

2 character TeX/LaTeX macros							Table 11
Name	Math	Name	Math	Name	Math	Name	Math
cma	$-a$	cm3a	$-3a$				
faa	$\lvert a\rvert$	fa3a	$\lvert 3a\rvert$	fna	#	fn3a	#
fba	#	fb3a	#	foa	#	fo3a	#
fca	$\cos a$	fc3a	$\cos 3a$	fpa	#	fp3a	#
fda	#	fb3a	#	fqa	#	fq3a	#
fea	#	fe3a	#	fra	#	fr3a	#
ffa	#	ff3a	#	fsa	$\sin a$	fs3a	$\sin 3a$
fga	#	fg3a	#	fta	#	ft3a	#
fha	#	fh3a	#	fua	#	fu3a	#
fia	#	fi3a	#	fva	#	fv3a	#
fja	#	fj3a	#	fwa	#	fw3a	#
fka	#	fk3a	#	fxa	#	fx3a	#
fla	$\ln a$	fl2a	$\log_2 a$	fya	#	fy3a	#
fma	#	fm3a	#	fza	#	fz3a	#

Interpretation of both numeric and ExtDigital arguments depends on the prefix and requires a special explanation.

4.3.1 Arguments to prefixes b and t *(to ToC)*

Prefixes "$\mathtt{bN_1\_N_2}$" and "$\mathtt{tN_1\_N_2}$" with an argument play a special role in the NGS. All other prefix operators are applied to one token name, which stands for only one math variable. The prefixes "$\mathtt{bN_1\ N_2}$" and "$\mathtt{tN_1\_N_2}$" allow to have an arbitrary name-object as the operand of the previous prefixes. (In some sense, prefixes b and t replace the pair of braces "{", "}" existing in TeX/LaTeX for the scope of commands.) Prefixes b and t may have only numerical argument $\mathtt{N_1\_N_2}$ (otherwise, they are interpreted as the core symbols).

4 possible forms of this argument are interpreted depending on presence of underscore.

1. With a number in the argument without underscore, the argument specifies the number N of the following characters, which are treated as the core. In the case of "b"-prefix, this string of characters is taken verbatim, while for the "t"-prefix, it is interpreted as the name-object
$$\mathtt{pb4time} \to (\mathtt{time})^2, \quad \mathtt{pt3gab} \to (\alpha b)^2, \quad \mathtt{qt4ayab} \to \sqrt{a+b}, \quad \mathtt{qt4wAwB22} \to \sqrt{(\mathsf{AB})_{22}}. \tag{4.25}$$
2. Form $\mathtt{N_1\_}$ means that the string of next $\mathtt{N_1}$ characters represent the superscript for the following token-name at the current level of nestness
$$\mathtt{qb5trans\_pwA} \to \sqrt{\left(\mathsf{A}^2\right)^{trans}}, \quad \mathtt{qt4gagbhx} \to \sqrt{(\widehat{x})^{\alpha\beta}}; \tag{4.26}$$
3. form $\mathtt{\_N_2}$ means that the next $\mathtt{N_2}$ characters represent the subscript for the following token-name
$$\mathtt{d2b2xyeF} \to \left(e^F\right)''_{xy}, \quad \mathtt{d2t4gagbeF} \to \left(e^F\right)''_{\alpha\beta}; \tag{4.27}$$
4. form $\mathtt{N_1\_N_2}$ specify the length $\mathtt{N_1}$ of superscript and $\mathtt{N_2}$ of subscript for the following token-name .

In the case of "b" prefix, the resulting strings are considered verbatim, while for "t", they are interpreted as a name-object.

Thus, prefixes "b" and "t" with an argument provide opportunity for names of nested math objects.

4.3.2 Arguments to prefixes `c`,`cm`,`e`,`p`,`r`,`x` *(to ToC)*

ExtDigital argument $\mathtt{D}_1\_\mathtt{D}_2$ of these prefixes is interpreted as a factor

$$\mathtt{D}_1 \to D_1, \quad \mathtt{D}_1\_ \to D_1 \tag{4.28}$$

or as a fraction

$$\_\mathtt{D}_2 \to \frac{1}{D_2}, \quad \mathtt{D}_1\_\mathtt{D}_2 \to \frac{D_1}{D_2}, \tag{4.29}$$

e.g.,

$$\mathtt{c2\_5gp} \to 0.4\pi, \quad \mathtt{e1\_2a} \to \left(\frac{1}{2}\right)^a, \quad \mathtt{p3i\_2a} \to a^{\frac{3i}{2}}, \quad \mathtt{r3i\_2a} \to \frac{3i}{2a}, \quad \mathtt{x3i\_2a} \to \frac{3i}{2}a. \tag{4.30}$$

Argument to 2-character prefix `cm` is interpreted in the same way as for prefix `c`.

4.3.3 Arguments to prefixes `h`,`q` *(to ToC)*

ExtDigital argument $\mathtt{D}_1\_\mathtt{D}_2$ of these prefixes is interpreted as an inverse factor

$$\mathtt{D}_1 \to \frac{1}{D_1}, \quad \mathtt{D}_1\_ \to \frac{1}{D_1} \tag{4.31}$$

or as an inverse fraction

$$\_\mathtt{D}_2 \to D_2, \quad \mathtt{D}_1\_\mathtt{D}_2 \to \frac{D_2}{D_1}, \tag{4.32}$$

e.g.,

$$\mathtt{h2\_5gp} \to \frac{5}{2}\pi, \quad \mathtt{q3i\_2a} \to a^{\frac{2}{3i}}. \tag{4.33}$$

This interpretation is consistent with interpretation of the simple argument for these prefixes in **Table 10**.

4.3.4 Arguments to prefixes `u`,`v`,`w` *(to ToC)*

ExtDigital argument $\mathtt{D}_1\_\mathtt{D}_2$ of these prefixes is interpreted as the superscript $\mathtt{D}_1\_$ and the subscript $\mathtt{D}_2$, e.g.,

$$\mathtt{u2A} \to \mathsf{A}_2, \quad \mathtt{u2\_A} \to \mathsf{A}^2, \quad \mathtt{u2\_1A} \to \mathsf{A}_1^2, \quad \mathtt{v2A} \to \mathsf{A}^2, \quad \mathtt{v2\_A} \to \mathsf{A}^2, \quad \mathtt{v2\_1A} \to \mathsf{A}_1^2, \quad \mathtt{w2\_1A} \to \mathsf{A}_1^2. \tag{4.34}$$

4.3.5 Arguments to prefix `d` *(to ToC)*

Note, that in NGS there is no special prefix for the TeX character " ′ " meaning the derivative in math. Prefix `d` with the argument is introduced for these purposes, e.g.,

$$\mathtt{d1f} \to f', \quad \mathtt{d2f} \to f'', \quad \mathtt{d4f} \to f^{(4)}. \tag{4.35}$$

Thus, this prefix may have only numerical argument of the form $\mathtt{N}_1\_\mathtt{N}_2$ (rather than ExtDigital). Its simplest form is explained in the above examples. Other forms, containing the underscore, give more flexibility to the meaning of this prefix.

Thus, one number with an underscore (either $\mathtt{N}_1\_$ or $\_\mathtt{N}_2$) specifies both the rank of derivative as well as the length of the following name-object, which is treated as the superscript to the operand of the prefix, e.g.,

$$\mathtt{d2\_xypf} \to \left(f^2\right)''_{xy}, \quad \mathtt{d\_2xypf} \to \left(f^2\right)''_{xy}. \tag{4.36}$$

In the form $\mathtt{N}_1\_\mathtt{N}_2$ with both numbers present, $\mathtt{N}_1$ specifies the rank of derivative, while $\mathtt{N}_2$ the length of the subscript name-object

$$\mathtt{d2\_4gqgqpf} \to \left(f^2\right)''_{\theta\theta}. \tag{4.37}$$

4.3.6 Arguments to prefixes starting with `f` *(to ToC)*

ExtDigital arguments are allowed in double character prefixes starting with `f`. For prefixes defined in **Table 11** these arguments with an underscore are interpreted as a fraction, e.g.,

$$\texttt{fc3i\_mgq} \to \cos\frac{3i}{m}\theta, \quad \texttt{fl\_2a} \to \log_{\frac{1}{2}} a. \tag{4.38}$$

4.3.7 Repeatable and non-repeatable prefixes *(to ToC)*

The token name can have several prefixes and each of them acts on the following shortened token-name. The following prefixes

$$\texttt{b}, \texttt{t}, \texttt{h}, \texttt{x}, \texttt{d}, \texttt{e}, \texttt{p}, \texttt{q}, \texttt{r} \tag{4.39}$$

as well as `f`-based double character prefixes are repeatable. In most cases (important exceptions follow), they are interpreted in a straightforward manner (as the corresponding TeX/LaTeX commands of **Table 1**) applied up to the end of the token-name,

$$\texttt{erga} \equiv \texttt{\textbackslash e\{\textbackslash r\{\textbackslash ga\}\}} \to e^{\frac{1}{\alpha}}, \quad \texttt{ere2gai} \equiv \texttt{\textbackslash e\{\textbackslash r\{\textbackslash E\{2\}\{\textbackslash ga\_i\}\}\}} \to e^{\frac{1}{2^{\alpha_i}}}. \tag{4.40}$$

As a deviation from the general rule, in the present system, the sequence of repetitive prefixes `p,r,q` following each other is considered in a special "multiplicative" way:

- presents of `r` is treated as a negative power for accumulated (from the right to the left) power;
- powers corresponding to each `p` and `q` are added.

This exception from the analogy with TeX/LaTeX, in fact, makes some name generating easier. Illustrative examples are

$$\texttt{p3ipa} \to a^{3i+2} \not\to \left(a^2\right)^{3i}, \quad \texttt{p3irp3a} \to a^{3i-3} \not\to \left(\frac{1}{a^3}\right)^{3i}, \quad \texttt{p3irq3ea} \to \left(e^a\right)^{3i-1/3} \not\to \left(\frac{1}{e^{a^{1/3}}}\right)^{3i}. \tag{4.41}$$

Instead,

$$\left(a^2\right)^{3i} = a^{6i} \to \texttt{p6ia}, \quad \left(\frac{1}{a^3}\right)^{3i} = \frac{1}{a^{9i}} \to \texttt{p9ira}, \quad \left(\frac{1}{e^{a^{1/3}}}\right)^{3i} = \frac{1}{e^{3ia^{1/3}}} \to \texttt{rex3iq3a}. \tag{4.42}$$

All other prefixes

$$\texttt{c}, \texttt{w}, \texttt{u}, \texttt{v} \tag{4.43}$$

are not repeatable in the prefix to the same token name. Also

$$\texttt{u}, \texttt{v}, \texttt{w} \tag{4.44}$$

are not compatible with each other. Their repetition in the name is interpreted as the beginning of the core, e.g.,

$$\texttt{cpc3r} \to \texttt{p3r}, \quad \texttt{pwuvx} \to \mathsf{u}^2\overrightarrow{x}. \tag{4.45}$$

The preference is to use "`c`" in front of the names for the constants.

4.3.8 Prefixes for names of integer variables *(to ToC)*

So far, prefixes with or without argument specified the meaning of math variables in the code names irrespective to their integer or non-integer character. If it is necessary to emphasize the integer character of the variable one of the pseudo digital character "`i`"— "`n`" should stay in front of the name. In interpretation, it is considered as a prefix for integer variables. All the rest is interpreted in the same way as explained already

$$\texttt{i} \to i, \quad \texttt{ii} \to i \quad \texttt{isi} \to i, \quad \texttt{iik} \to i_k, \quad \texttt{ii2\_k} \to i_k^2, \quad \texttt{ix2sk} \to 2k, \quad \texttt{ix2k} \to x_{2k}. \tag{4.46}$$

NGS suggests use of leading "`l`" in names for pointer variables (as the first character of the name, it cannot be confused with a digital character "`1`").

Present NGS does not restrict use of prefixes for integer variables despite some of them are irrelevant for integers.

5 Void and Digital token-names *(to ToC)*

So far, underscore character _ was used for interpretation of numerical or ExtDigital arguments either to the prefix or to core recognition characters or for interpretation of ExtDigital suffixes. As it is explained later in this section, underscore also plays a role of a decimal point in digital token names. In other situations, NGS considers the underscore simply as a void token-name, which is interpreted as a space " " character. This interpretation of underscore has the lowest priority with respect to other interpretations.

Void token-names can be used for separating token-names in the name-object, e.g.,

$$\texttt{c\_\_3} \to c\ 3, \quad \texttt{pg\_b} \to g^2\ b, \quad \texttt{pgsb} \to g^2 b, \quad \texttt{pi\_jk} \to p^i_{jk}, \quad \texttt{pi\_j\_k} \to p^i_j k, \quad \texttt{pi\_\_j} \to p^i j, \quad \texttt{p\_\_j} \to pj, \quad \texttt{\_pp} \to p^2. \tag{5.1}$$

When placed in front of the first token name, any number of underscores are ignored in interpretation as void token-names. Exceptional names, consisting exclusively from underscores, stand for a fictitious variable _

$$\texttt{\_} \to \_, \quad \texttt{\_\_\_} \to \_. \tag{5.2}$$

Void token-name has no suffix.

In NGS, the token-name may start with a sequence of digits. E.g., underscore in front of the name allows to have leading digits even in the first functional token-name without conflict with the syntax of computer language. The subsequent token-names may also have leading digits in the name.

NGS interprets the leading sequence of digits in the token name as *a separate* token name, e.g.,

$$\texttt{\_25b} \to 25b, \quad \texttt{p\_\_25b} \to p25b, \quad \texttt{p\_\_25ib} \to p25b. \tag{5.3}$$

(In the third example, `i` in front of the token name `ib` serves as a prefix specifying b as an integer math variable and, thus, does not appear in typesetting).

Leading sequence of digits with an underscore is interpreted as a decimal fraction (second underscore in the sequence of digits just starts a new token-name), e.g.,

$$\texttt{\_3} \to 3, \quad \texttt{\_3\_b} \to 3.b, \quad \texttt{\_3\_1415} \to 3.1415, \quad \texttt{\_3\_1415\_2} \to (3.1415)2. \tag{5.4}$$

6 Composit names. o-operators *(to ToC)*

Composit names represent either concatenation of the token-names (with or without `o`-operators) or different kind of nested name-objects.

Prefix `t`, as it was explained in subsection 4.3.1, provides an opportunity to have one name-object inside another name-object. Also, numerical argument after the core recognition character allows to have name-objects either in super- or sub-scripts.

In NGS, as it was already explained, the end of the token name is well-defined by conventions concerning the suffixes. What follows after the suffix is considered as the next name-object which could be further decomposit into the token names.

All of these opportunities allow to generate compact names for complicated math objects. Thus, using special symbols from **Table 6** one could create such names as

$$\texttt{yskvxk} \to \sum_k (\vec{x})_k, \quad \texttt{y2txyf} \to \Delta_{xy} f, \quad \texttt{yib5Fdxdy} \to \int F dxdy, \quad \texttt{vAyxvBij} \to (\vec{A} \times \vec{B})_{ij}. \tag{6.1}$$

Here, in the last example `yx`, in fact, acts similar to the `o`-operator on the token names on the left and right side from it.

The `o`-operators have been introduced into NGS for expressing mathematical relations which may involve several variables. The global nature of the scope of action of `o`-operators was explained in scheme (2.7). At present, a very few of them are defined. In **Table 13** undefined `o`-operators are replaced by `&` character.

`o`-operators for composit names								**Table 13**
`o`-symbol	Math	Meaning	`o`-symbol	Math	Meaning	`o`-symbol	Math	Meaning
`oa`	&	logical "and"	`oj`	\$	*undefined*	`os`	\$	*undefined*
`ob`	\$	*undefined*	`ok`	\$	*undefined*	`ot`	\$	*undefined*
`oc`	$\cdot$	product	`ol`	\$	*undefined*	`ou`	\$	*undefined*
`od`	.	decimal point	`om`	$-$	subtraction	`ov`	\$	*undefined*
`oe`	\$	*undefined*	`on`	\$	*undefined*	`ow`	\$	*undefined*
`of`	/	fraction	`oo`	\$	*undefined*	`ox`	$\times$	product
`og`	\$	*undefined*	`op`	$+$	addition	`oy`	\$	*undefined*
`oh`	$\hat{}$	power	`oq`	\$	*undefined*	`oz`	\$	*undefined*
`oi`	\$	*undefined*	`or`	$\Vert$	logical "or"			

(See also TeX/LaTeX macros in **Table 4** for symbols of `o`-operators).

Unlike math symbols and their NGS analogs in `y`-symbols, whose interpretation include only adjacent token-names, `o`-operators have global arguments (2.7). Using `o`-operators it is possible to make names like

$$\texttt{ga2ohaypb} \to \alpha^{a+b}, \quad \texttt{aymboraypb} \to \frac{a-b}{a+b}. \tag{6.2}$$

Same effect using `y`-simbols can be achieved only with more complicated names, i.e.,

$$\texttt{ga2yht4ayab} \to \alpha^{a+b}, \quad \texttt{t4aymbyrt4ayab} \to \frac{a-b}{a+b}. \tag{6.3}$$

7 Interpretation of computer code names *(to ToC)*

What was explained already is practically sufficient for the name generation using NGS. This section describes a formal algorithm of name interpretation as it is implemented into the computer code `gns`. NGS takes advantage of capital letters in C-names, while in F-names capital letters are interpreted as small letters. Because of this, both name generation and name interpretation are slightly different for C- and F-names.

7.1 Interpretation of C-names *(to ToC)*

Interpretation proceeds from left to right.

1. First, all prefixes with their optional ExtDigital arguments are recognized, until the interpretation of prefixes fails. There are 4 reasons for such a **failure**

(a) The current character is **not** from the prefix character set "`b`", "`c`", "`d`", "`e`", "`f`", "`h`", "`p`", "`q`", "`r`", "`u`", "`v`", "`w`", "`t`", "`x`" as well as not one of the pseudo digits "`i`", "`k`", "`l`", "`m`", "`n`" in front of the token-name.

(b) The current character is **incompatible** with the previous prefix characters, i.e.,

  i. It repeats one of **unrepeatable** prefix characters, i.e., `c, u, v, w`.

  ii. It is **incompatible** with the previous prefixes. Prefixes `u, v, w` are incompatible with each other.

  iii. It is either `b` or `t` with an argument in the form `N`.

(c) The argument $\mathtt{D}_1\_\mathtt{D}_2$ is **wrong** or incompatible with the prefix character, i.e,

  i. In the argument for prefixes other than `u, v, w`, either $\mathtt{D}_1$ or $\mathtt{D}_2$ starts with `0` character.

  ii. In the argument to prefixes `b, d, t`, either $\mathtt{D}_1$ or $\mathtt{D}_2$ is not digital.

  iii. The number of remaining characters in the name is smaller than specified by the argument to prefixes `b,d,t`.

(d) The prefix would have no core in the token-name, e.g.,

$$\mathtt{ppib2} \to b_2^{1+i}, \quad \mathtt{ppib2orayab} \to \frac{b_2^{1+i}}{a+b}, \quad \mathtt{ppib_a yab} \to b^{1+i}a+b. \tag{7.1}$$

The underscore, which is not a part of the argument or of the suffix, and `o`-operators terminate the token-name.

In the case of failure in prefix interpretation, the interpretation process is switched to the core.

2. Second, the core is interpreted.

(a) If it is specified by the prefix `b` with a numerical argument, then it is considered verbatim.

(b) If it is specified by the prefix `t` with a numerical argument, then it is interpreted as a (nested) name-object.

(c) If the first character is one of `s, a, g, y, z`, then it is interpreted as the core recognition character (**Table 5**). **This interpretation fails, if**

  i. It would be no core symbol in the token-name, e.g.

$$\mathtt{pg} \to g^2, \quad \mathtt{pg_a} \to g^2 a. \tag{7.2}$$

  Underscore which is not part of the argument or of the suffix terminates the token-name.

  ii. Lenght of the suffix, specified by the argument to the core recognition character, is inconsistent with the number of characters after the core, e.g.,

$$\mathtt{s2a} \to s_2 a. \tag{7.3}$$

If interpretation of the core recognition character fails, the character is considered as the core symbol.

3. Third, the super- and sub-scripts in the suffix are interpreted.

(a) If their lengths are specified by the argument of the core recognition character, then they are interpreted as the name-objects.

(b) Otherwise a sequence of ExtDigits with an optional underscore is interpreted as a suffix. Also, in the case when the core symbol is the capital letter, the suffix is determined by the special rules for C-names.

4. After the token-name, the `o`-operator, if present, is recognized as a double alphabet string starting with `o`.

5. The process of recognition of token names and `o`-operators is repeated for the next token name recognition starting with the interpretation of prefixes until the end of the name is encountered.

6. Finally, the name and its structure are interpreted based on interpretation of the token-names and `o`-operators.

7.2 Interpretation of F-names *(to ToC)*

The only difference between interpretations of F- and C-names is that during interpretation the capital letters in the F-names are converted into small letters, e.g.,

$$\texttt{gB} \to \beta, \quad \texttt{gb} \to \beta, \quad \texttt{ab} \to B, \quad \texttt{sB} \to b. \tag{7.4}$$

As a result, all C-name conventions, which rely explicitly on capital letters (like recognition of core symbols, convention about suffixes to capital variables) are not applicable for F-names.

Otherwise, the algorithm of F-name interpretation is the same as described in the previous subsection.

Because of case insensitivity, F-names cannot reproduce the capital Greek and Russian letters, while names for capital English math variables should use the core recognition character `a`.

8 Name design in Equilibrium and Stability Code (ESC) *(to ToC)*

In `ESC`, reference flux coordinates a, τ, φ are defined by Fourier representation

$$\begin{aligned}
&r = R_0 + R_0^c + 2\sum_{k=1}^{K_r-1}\left(R_k^c(a)\cos(m_k\tau - n_k\varphi) + R_k^s(a)\sin(m_k\tau - n_k\varphi)\right),\\
&z = Z_0 + Z_0^c + 2\sum_{k=1}^{K^z-1}\left(Z_k^c(a)\cos(m_k\tau - n_k\varphi) + Z_k^s(a)\sin(m_k\tau - n_k\varphi)\right),\\
&r'_p \equiv r'_\tau, \quad r'_t \equiv r'_\varphi, \quad z'_p \equiv z'_\tau, \quad z'_t \equiv z'_\varphi.
\end{aligned} \tag{8.1}$$

The corresponding data structure in the code is

```
Typedef struct EQCOORD{
  int Na1;
  int Np;
  int Np1;
  int Nt;
  int Nt1;
  int Kr;
  int Kz;
  int *mk;
  int *nk;
  double R0;
  double Z0;
  double *Rc;
  double *d1Rc;
  double *d2Rc;
  double *Rs;
  double *d1Rs;
```

```
  double *d2Rs;
  double *Zc;
  double *d1Zc;
  double *d2Zc;
  double *Zs;
  double *d1Zs;
  double *d2Zs;
  double *r;
  double *d1_ar;
  double *d1_pr;
  double *d1_tr;
  double *z;
  double *d1_az;
  double *d1_pz;
  double *d1_tz;
}EQCOORD;
```

Here, `Na` is the number of radial intervals, `Na1=Na+1`, `Na` is the number of poloidal intervals, `Np1=Np+1`, `Nt` is the number of toroidal intervals, `Nt1=Nt+1`, `Lt` is the number of toroidal periods.

Its metric tensor is represented in Fourier space as

$$
\begin{aligned}
&\frac{g_{ij}}{\sqrt{g}} = G^c_{ij} + 2\sum_{k=1}^{K_g-1}\left(G^c_{aa,k}(a)\cos(m_k\tau - n_k\varphi) + G^s_{aa,k}(a)\sin(m_k\tau - n_k\varphi)\right),\\
&J \equiv \sqrt{g} = \frac{D(x,y,z)}{D(a,\tau,\varphi)}, \quad i = a,\tau,\varphi, \quad j = a,\tau,\varphi.
\end{aligned}
\tag{8.2}
$$

The corresponding data structure in the code is

```
Typedef struct EQMETRIC{
  int Na1;
  int K;
  int *mk;
  int *nk;
  double *Jc;
  double *d1Jc;
  double *d2Jc;
  double *Js;
  double *d1Js;
  double *d2Js;
  double *G11c;
  double *d1G11c;
  double *d2G11c;
  double *G11s;
  double *d1G11s;
  double *d2G11s;
  double *G12c;
  double *d1G12c;
  double *d2G12c;
  double *G12s;
  double *d1G12s;
  double *d2G12s;
  double *G22c;
  double *d1G22c;
  double *d2G22c;
  double *G22s;
  double *d1G22s;
  double *d2G22s;
  double *G13c;
  double *d1G13c;
  double *d2G13c;
  double *G13s;
  double *d1G13s;
  double *d2G13s;
  double *G22c;
  double *d1G23c;
```

```
  double *d2G23c;
  double *G23s;
  double *d1G23s;
  double *d2G23s;
  double *G33c;
  double *d1G33c;
  double *d2G33c;
  double *G33s;
  double *d1G33s;
  double *d2G33s;
}EQMETRIC;
```

Plasma profiles are represented by one-dimentional functions

$$
\begin{aligned}
&P \equiv \frac{d\bar{p}}{d\bar{\Psi}}, \quad \bar{p} \equiv \mu_0 p, \quad \bar{\Psi} \equiv \frac{\Psi}{2\pi}, \\
&T \equiv \bar{F}\frac{d\bar{F}}{d\bar{\Psi}}, \quad \bar{F} \equiv rB_{tor}, \\
&j_p \equiv R_0 P, \quad j_s \equiv \frac{T}{R_0} + R_0 P, \quad q, \quad \mu \equiv \frac{1}{q}, \quad \ldots.
\end{aligned}
\tag{8.3}
$$

The corresponding data structure is

```
Typedef struct EQPROF{
  int Na1;
  double *aP;
  double *d1aP;
  double *d2aP;
  double *aT;
  double *d1aT;
  double *d2aT;
  double *s1jp;
  double *d1s1jp;
  double *d2s1jp;
  double *s1js;
  double *d1s1js;
  double *d2s1js;
  double *sq;
  double *d1sq;
  double *d2sq;
  double *gm;
  double *d1gm;
  double *d2gm;
  double .....;
}EQPROF;
```

The meaning of the names here is straightforward. Presence of first derivatives allows to reconstruct function at any point using bi-cubic interpolation, while the second derivatives allow to use cubic splines.

These structures together represent the structure of equilibrium configuration

```
Typedef struct EQCONF{
  int Na;
  int Na1;
  int Np;
  int Np1;
  int Nt;
  int Nt1;
  int Lt;
  double Bext;
  double Rext;
  EQCOORD *Coord;
  EQMETRIC *G;
  EQPROF *p;
}EQCONF;
```

Here, `Bext` represents the vacuum toroidal magnetic field at the reference point $r = R_{ext}$. This structure is the output of `ESC`.

9 Summary *(to ToC)*

The very recognition of the name design as a problem in the environment of development and sharing big numerical codes is already a step forward from the present state of absence of guidance in naming the math variables.

For the math variables, the `JAVA`-like English based descriptive approach (good in for its own purposes), lacks compactness and, thus, is incompatible with the spirit of the mathematics. On the other hand, a formalized system of the compact name design, based on some general principles, simplifies both the name generation and understanding the source code.

The name generation system NGS described in the paper gives practically comprehensive means for expressing the meaning of mathematical variables in compact computer language names. In future, it could allow an automatic maintenance of certain portions of documentation in a form consistent with the source code. The system also provides a room for extending its capabilities if it will be necessary.

Modest efforts in keeping entropy low in the growing ensemble of names by using a name design system simplify significantly the maintenance of the transparent documentation and the source code. There is no penalty for deviation from the system conventions. In fact, deviations may represent a reasonable compromise between simplicity in names and uniqueness of their interpretation with individually acceptable level of ambiguity. What is important is that presence of a rigorous reference system, such as NGS, in any case provides a guidance for the name design.