

The cool package*

nsetzer

October 7, 2006

This is the cool package: a COnent Oriented L^AT_EX package. That is, it is designed to give L^AT_EX commands the ability to contain the mathematical meaning while retaining the typesetting versatility.

Please note that there are examples of use of each of the defined commands at the location where they are defined.

This package requires the following, non-standard L^AT_EX packages (all of which are available on www.ctan.org): coolstr, coollist, forloop

1 Implementation

```
1 \newcounter{COOL@ct} %just a general counter
2 \newcounter{COOL@ct@}%just a general counter
```

1.1 Parenthesis

```
3 \newcommand{\inp}[2][0cm]{\mathopen{}\left(#2\parbox[h]{#1}{0cm}{\}\right)}
4           % in parentheses ()
5 \newcommand{\inb}[2][0cm]{\mathopen{}\left[#2\parbox[h]{#1}{0cm}{\}\right]}
6           % in brackets []
7 \newcommand{\inbr}[2][0cm]{\mathopen{}\left\{#2\parbox[h]{#1}{0cm}{\}\right\}}
8 % in braces {}
9 \newcommand{\inap}[2][0cm]{\mathopen{}\left<#2\parbox[h]{#1}{0cm}{\}\right>}
10 % in angular parentheses <>
11 \newcommand{\nop}[1]{\mathopen{}\left.{#1}\right.}
12 % no parentheses
```

`\COOL@decide@paren` `\COOL@decide@paren[<parenthesis type>]{<function name>}{<contained text>}`.
Since the handling of parentheses is something that will be common to many elements this function will take care of it.

If the optional argument is given, `\COOL@notation@<function name>Paren` is ignored and *<parenthesis type>* is used

<parenthesis type> and `\COOL@notation@<function name>Paren` must be one of none, p for (), b for [], br for {}, ap for ⟨⟩, inv for \left.\right.

```
13 \let\COOL@decide@paren@no@type=\relax
14 \newcommand{\COOL@decide@paren}[3][\COOL@decide@paren@no@type]{%
```

*This document corresponds to cool v1.3, dated 2006/10/07.

```

15 \ifthenelse{ \equal{#1}{\COOL@decide@paren@no@type} }%
16 {%
17 \def\COOL@decide@paren@type{\csname COOL@notation@#2Paren\endcsname}%
18 }%
19 % Else
20 {%
21 \def\COOL@decide@paren@type{#1}%
22 }%
23 \ifthenelse{ \equal{\COOL@decide@paren@type}{none} }%
24 {%
25 #3%
26 }%
27 % Else
28 {%
29 \ifthenelse{ \equal{\COOL@decide@paren@type}{p} }%
30 {%
31 \inp{#3}%
32 }%
33 % Else
34 {%
35 \ifthenelse{ \equal{\COOL@decide@paren@type}{b} }%
36 {%
37 \inb{#3}%
38 }%
39 % Else
40 {%
41 \ifthenelse{ \equal{\COOL@decide@paren@type}{br} }%
42 {%
43 \inbr{#3}%
44 }%
45 % Else
46 {%
47 \ifthenelse{ \equal{\COOL@decide@paren@type}{ap} }%
48 {%
49 \inap{#3}%
50 }%
51 % Else
52 {%
53 \ifthenelse{ \equal{\COOL@decide@paren@type}{inv} }%
54 {%
55 \nop{#3}%
56 }%
57 % Else
58 {%
59 \PackageError{cool}{Invalid Parenthesis Option}%
60 {*Paren can only be 'none', 'p', 'b', 'br', 'ap', 'inv'}%
61 }%
62 }%
63 }%
64 }%

```

```

65 }%
66 }%
67 }

```

1.2 Indices

```

\COOL@decide@indices \COOL@decide@indices{\function name}{\local indication}{\indices}
Since up or down indices can be as common as the parenthesis decision, this
macro is the solution.
    \local indication must be either u or d
    \indices is very likely to be required to be a comma separated list in the near
future
the options for indices are
    local allow the indices to be decided by an optional argument to
           the function (such as \LeviCivita[u]{i j})
    up    force the indices to appear as superscript
    down  force the indices to appear as subscript

68 \newcommand{\COOL@decide@indices}[3]{%
69 \def\COOL@decide@indices@placement%
70 {\csname COOL@notation@#1Indices\endcsname}%
71 \ifthenelse{\equal{\COOL@decide@indices@placement}{local}}%
72 {%
73 \ifthenelse{\equal{#2}{u}}%
74 {\^{#3}}%
75 {\_{#3}}%
76 }%
77 % Else
78 {%
79 \ifthenelse{\equal{\COOL@decide@indices@placement}{up}}%
80 {%
81 {\^{#3}}%
82 }%
83 % Else
84 {%
85 \ifthenelse{\equal{\COOL@decide@indices@placement}{down}}%
86 {%
87 {\_{#3}}%
88 }%
89 % else
90 {%
91 \PackageError{cool}{Invalid Option Sent}%
92 {#1Indices can only be 'up', 'down', or 'local'}%
93 }%
94 }%
95 }%
96 }

```

1.3 COntent Oriented LaTeX (COOL)

`\Style` `\Style{⟨options⟩}` sets the style of the output (how to notate particular functions). `⟨options⟩` is a comma delimited list of the form `⟨key⟩=⟨value⟩`, where `⟨key⟩` is the *long* form of the command name without the preceeding backslash (i.e. `Integrate` and not `Int` or `\Int`). The list can be in any order and need only contain the styles that the user desires to set.

There can be multiple `\Style` commands within any document—the styled output of the command depends on the last `\Style` command to have specified its style.

For a list of styling options for a command, see the code where the command is defined

```

97 \newcommand{\Style}[1]{%
98 \COOL@keyeater#1,\COOL@keystop\COOL@keyend%
99 }
100 \newcommand{\COOL@keystop}{@@@}%
101 \def\COOL@keyeater#1=#2,#3\COOL@keyend{%
102 \ifx#3\COOL@keystop%
103 \expandafter\gdef\csname COOL@notation@#1\endcsname{#2}%
104 \else%
105 \expandafter\gdef\csname COOL@notation@#1\endcsname{#2}%
106 \COOL@keyeater#3\COOL@keyend%
107 \fi%
108 }
```

`\UseStyleFile` Since notational style should be kept consistent and will likely need to span several documents, use this command to input a notation style file that has previously been prepared. (to be implemented in a future release)

```

109 \newcommand{\UseStyleFile}[1]{}
```

1.3.1 Fundamental Constants

see <http://functions.wolfram.com/> for the definitions

`\I` The square root of minus 1, $i = \sqrt{-1}$.

`\Style{ISymb=\mathbbm{i}}`, `\I` gives \mathbb{i} .

```

110 \newcommand{\COOL@notation@ISymb}{i}
111 \newcommand{\I}{\COOL@notation@ISymb}
```

`\E` Euler’s constant and the base of the natural logarithm, e .

`\Style{ESymb=\mathbbm{e}}`, `\E` gives \mathbb{e} .

```

112 \newcommand{\COOL@notation@ESymb}{e}
113 \newcommand{\E}{\COOL@notation@ESymb}
```

`\PI` Pi—the ratio of the circumference of a circle to its diameter, π .

`\Style{PISymb=\bbpi}`¹, `\PI` gives $\mathbb{\pi}$.

¹to get the ‘bbpi’ symbol, you will need to use the package `mathbbol` and pass the `bbgreekl` option

```

114 \newcommand{\COOL@notation@PISymb}{\pi}
115 \newcommand{\PI}{\COOL@notation@PISymb}

\GoldenRatio The Golden Ratio,  $\varphi$ 
116 \newcommand{\GoldenRatio}{\varphi}

\EulerGamma Euler's Gamma constant,  $\gamma$ .
\Style{EulerGammaSymb=\gamma_E} , \EulerGamma gives  $\gamma_E$ 
117 \newcommand{\COOL@notation@EulerGammaSymb}{\gamma}
118 \newcommand{\EulerGamma}{\COOL@notation@EulerGammaSymb}

\Catalan Catalan constant,  $C$ 
119 \newcommand{\Catalan}{C}

\Glaisher Glaisher constant, Glaisher
120 \newcommand{\Glaisher}{\mathord{\operatorname{Glaisher}}}

\Khinchin Khinchin constant, Khinchin
121 \newcommand{\Khinchin}{\mathord{\operatorname{Khinchin}}}

1.3.2 Symbols

\Infinity Infinity,  $\infty$ 
122 \newcommand{\Infinity}{\infty}

\Indeterminant An indeterminant quantity
123 \newcommand{\Indeterminant}{%
124 \mathchoice%
125 {\mbox{\textrm>}}%
126 {\mbox{\small>}}%
127 {\mbox{\scriptsize>}}%
128 {\mbox{\tiny>}}%
129 }

\DirectedInfinity Directed Infinity \DirectedInfinity{<complex number>} or \DirInfty{<complex
\DirInfty number>}
130 \newcommand{\DirectedInfinity}[1]{#1 \, \infty}
131 \newcommand{\DirInfty}[1]{\DirectedInfinity{#1}}

\ComplexInfinity Complex infinity,  $\tilde{\infty}$ 
\CInfty 132 \newcommand{\ComplexInfinity}{\tilde{\infty}}
133 \newcommand{\CInfty}{\ComplexInfinity}

```

1.3.3 Exponential Functions

`\Exp` Exponential—for use when e^x won't suffice, $\exp(x)$

```
134 \newcommand{\COOL@notation@ExpParen}{p}
135 \newcommand{\Exp}[1]
136 {%
137 \exp\COOL@decide@paren{Exp}{#1}%
138 }
```

`\Log` Logarithm, `\Log{x}`. This function has several options to be set. The usual parentheses, then some about the notation to be used for displaying the symbol.

```
139 \newcommand{\COOL@notation@LogParen}{none}
```

The following set the symbols:

`LogBaseESymb` can be `ln` or `log`, indicating what symbol should be used for the natural logarithm. If set to `log` then logarithms of base 10 are displayed as \log_{10} .

`LogShowBase` can be either `at will` or `always` and decides whether or not one should show the base, as in $\log_b x$. If this option is set to `always` then `LogBaseESymb` is ignored.

<code>\Log{5}</code>	$\ln 5$	$\ln 5$
<code>\Log[10]{5}</code>	$\log 5$	$\log 5$
<code>\Log[4]{5}</code>	$\log_4 5$	$\log_4 5$
<code>\Style{LogBaseESymb=log}</code>		
<code>\Log{5}</code>	$\log 5$	$\log 5$
<code>\Log[10]{5}</code>	$\log_{10} 5$	$\log_{10} 5$
<code>\Log[4]{5}</code>	$\log_4 5$	$\log_4 5$
<code>\Style{LogShowBase=always}</code>		
<code>\Log{5}</code>	$\log_e 5$	$\log_e 5$
<code>\Log[10]{5}</code>	$\log_{10} 5$	$\log_{10} 5$
<code>\Log[4]{5}</code>	$\log_4 5$	$\log_4 5$
<code>\Style{LogShowBase=at will}</code>		
<code>\Log{5}</code>	$\ln 5$	$\ln 5$
<code>\Log[10]{5}</code>	$\log 5$	$\log 5$
<code>\Log[4]{5}</code>	$\log_4 5$	$\log_4 5$
<code>\Style{LogParen=p}</code>		
<code>\Log[4]{5}</code>	$\log_4(5)$	$\log_4(5)$

```
140 \newcommand{\COOL@notation@LogBaseESymb}{ln}% 'ln', 'log'
141 \newcommand{\COOL@notation@LogShowBase}{at will}% 'at will', 'always'
142 \newcommand{\Log}[2][E]
143 {%
144 \ifthenelse{ \equal{\COOL@notation@LogShowBase}{at will} }{%
145 {%
146 \ifthenelse{ \equal{#1}{\E} }{%
147 {%
148 \ifthenelse{ \equal{\COOL@notation@LogBaseESymb}{ln} }{%
149 {%
150 \ln \COOL@decide@paren{Log}{#2}%
```

```

151 }%
152 % Else
153 {%
154 \ifthenelse{ \equal{\COOL@notation@LogBaseESymb}{log} }%
155 {%
156 \log \COOL@decide@paren{Log}{#2}%
157 }%
158 % Else
159 {%
160 \PackageError{cool}{Invalid Option Sent}%
161 {LogBaseESymb can only be 'ln' or 'log'}%
162 }%
163 }%
164 }%
165 % Else
166 {%
167 \ifthenelse{ \equal{#1}{10} \AND
168 \NOT \equal{\COOL@notation@LogBaseESymb}{log} }%
169 {%
170 \log \COOL@decide@paren{Log}{#2}%
171 }%
172 % Else
173 {%
174 \log_{#1} \COOL@decide@paren{Log}{#2}%
175 }%
176 }%
177 }%
178 % Else
179 {%
180 \ifthenelse{ \equal{\COOL@notation@LogShowBase}{always} }%
181 {%
182 \log_{#1}\COOL@decide@paren{Log}{#2}%
183 }%
184 % Else
185 {%
186 \PackageError{cool}{Invalid Option Sent}%
187 {LogShowBase can only be 'at will' or 'always'}%
188 }%
189 }%
190 }

```

1.3.4 Trigonometric Functions

\Sin The sine function, $\text{\Sin{x}}$, $\sin(x)$

```

191 \newcommand{\COOL@notation@SinParen}{p}
192 \newcommand{\Sin}[1]{\sin\COOL@decide@paren{Sin}{#1}}

```

\Cos The cosine function, $\text{\Cos{x}}$, $\cos(x)$

```

193 \newcommand{\COOL@notation@CosParen}{p}
194 \newcommand{\Cos}[1]{\cos\COOL@decide@paren{Cos}{#1}}

```

`\Tan` The tangent function, `\Tan{x}`, $\tan(x)$

```

195 \newcommand{\COOL@notation@TanParen}{p}
196 \newcommand{\Tan}[1]{\tan\COOL@decide@paren{Tan}{#1}}

\Csc The cosecant function, \Csc{x},  $\csc(x)$ 
197 \newcommand{\COOL@notation@CscParen}{p}
198 \newcommand{\Csc}[1]{\csc\COOL@decide@paren{Csc}{#1}}

\Sec The secant function, \Sec{x},  $\sec(x)$ 
199 \newcommand{\COOL@notation@SecParen}{p}
200 \newcommand{\Sec}[1]{\sec\COOL@decide@paren{Sec}{#1}}

\Cot The cotangent function, \Cot{x},  $\cot(x)$ 
201 \newcommand{\COOL@notation@CotParen}{p}
202 \newcommand{\Cot}[1]{\cot\COOL@decide@paren{Cot}{#1}}
```

1.3.5 Inverse Trigonometric Functions

`\COOL@notation@ArcTrig` The inverse trigonometric functions style is governed by this global key. Its options are

`inverse` (default), this displays as \sin^{-1}
`arc`, this displays as `arcsin`

```

203 \def\COOL@notation@ArcTrig{inverse}
```

`\ArcSin` The inverse of the sine function, `\ArcSin{x}`, $\sin^{-1}(x)$

```

204 \newcommand{\COOL@notation@ArcSinParen}{p}
205 \newcommand{\ArcSin}[1]{%
206 \ifthenelse{ \equal{\COOL@notation@ArcTrig}{inverse} }{%
207 {%
208 \sin^{-1}\COOL@decide@paren{ArcSin}{#1}%
209 }
210 % else
211 {
212 \ifthenelse{ \equal{\COOL@notation@ArcTrig}{arc} }{%
213 {%
214 \arcsin\COOL@decide@paren{ArcSin}{#1}%
215 }%
216 % else
217 {%
218 \PackageError{cool}{Invalid option sent}{}%
219 }%
220 }%
221 }
```

`\ArcCos` the inverse of the cosine function, `\ArcCos{x}`, $\cos^{-1}(x)$

```

222 \newcommand{\COOL@notation@ArcCosParen}{p}
223 \newcommand{\ArcCos}[1]{%
224 \ifthenelse{ \equal{\COOL@notation@ArcTrig}{inverse} }{%
```



```

225 {%
226 \cos^{-1}\COOL@decide@paren{ArcCos}{#1}%
227 }%
228 % else
229 {%
230 \ifthenelse{\equal{\COOL@notation@ArcTrig}{arc}}%
231 {%
232 \arccos\COOL@decide@paren{ArcCos}{#1}%
233 }%
234 % else
235 {%
236 \PackageError{cool}{Invalid option sent}{}%
237 }%
238 }%
239 }

\ArcTan The inverse of the tangent function, \ArcTan{x},  $\tan^{-1}(x)$ 
240 \newcommand{\COOL@notation@ArcTanParen}{p}
241 \newcommand{\ArcTan}[1]{%
242 \ifthenelse{\equal{\COOL@notation@ArcTrig}{inverse}}%
243 {%
244 \tan^{-1}\COOL@decide@paren{ArcTan}{#1}%
245 }%
246 % else
247 {%
248 \ifthenelse{\equal{\COOL@notation@ArcTrig}{arc}}%
249 {%
250 \arctan\COOL@decide@paren{ArcTan}{#1}%
251 }%
252 % else
253 {%
254 \PackageError{cool}{Invalid option sent}{}%
255 }%
256 }%
257 }

\ArcCsc The Inverse Cosecant function, \ArcCsc{x},  $\csc^{-1}(x)$ 
258 \newcommand{\COOL@notation@ArcCscParen}{p}
259 \newcommand{\ArcCsc}[1]{\csc^{-1}\COOL@decide@paren{ArcCsc}{#1}}

\ArcSec The inverse secant function, \ArcSec{x},  $\sec^{-1}(x)$ 
260 \newcommand{\COOL@notation@ArcSecParen}{p}
261 \newcommand{\ArcSec}[1]{\sec^{-1}\COOL@decide@paren{ArcSec}{#1}}

\ArcCot The inverse cotangent function, \ArcCot{x},  $\cot^{-1}(x)$ 
262 \newcommand{\COOL@notation@ArcCotParen}{p}
263 \newcommand{\ArcCot}[1]{\cot^{-1}\COOL@decide@paren{ArcCot}{#1}}

```

1.3.6 Hyperbolic Functions

`\Sinh` Hyperbolic sine, `\Sinh{x}`, $\sinh(x)$
264 `\newcommand{\COOL@notation@SinhParen}{p}`
265 `\newcommand{\Sinh}[1]{\sinh\COOL@decide@paren{Sinh}{#1}}`

`\Cosh` Hyperbolic cosine, `\Cosh{x}`, $\cosh(x)$
266 `\newcommand{\COOL@notation@CoshParen}{p}`
267 `\newcommand{\Cosh}[1]{\cosh\COOL@decide@paren{Cosh}{#1}}`

`\Tanh` Hyperbolic Tangent, `\Tanh{x}`, $\tanh(x)$
268 `\newcommand{\COOL@notation@TanhParen}{p}`
269 `\newcommand{\Tanh}[1]{\tanh\COOL@decide@paren{Tanh}{#1}}`

`\Csch` Hyperbolic cosecant `\Csch{x}`, $\operatorname{csch}(x)$
270 `\newcommand{\COOL@notation@CschParen}{p}`
271 `\DeclareMathOperator{\csch}{csch}`
272 `\newcommand{\Csch}[1]{\csch\COOL@decide@paren{Csch}{#1}}`

`\Sech` Hyperbolic secant, `\Sech{x}`, $\operatorname{sech}(x)$
273 `\newcommand{\COOL@notation@SechParen}{p}`
274 `\DeclareMathOperator{\sech}{sech}`
275 `\newcommand{\Sech}[1]{\sech\COOL@decide@paren{Sech}{#1}}`

`\Coth` Hyperbolic Cotangent, `\Coth{x}`, $\operatorname{coth}(x)$
276 `\newcommand{\COOL@notation@CothParen}{p}`
277 `\newcommand{\Coth}[1]{\coth\COOL@decide@paren{Coth}{#1}}`

1.3.7 Inverse Hyperbolic Functions

`\ArcSinh` Inverse hyperbolic sine, `\ArcSinh{x}`, $\sinh^{-1}(x)$
278 `\newcommand{\COOL@notation@ArcSinhParen}{p}`
279 `\newcommand{\ArcSinh}[1]{\sinh^{-1}\COOL@decide@paren{ArcSinh}{#1}}`

`\ArcCosh` Inverse hyperbolic cosine, `\ArcCosh{x}`, $\cosh^{-1}(x)$
280 `\newcommand{\COOL@notation@ArcCoshParen}{p}`
281 `\newcommand{\ArcCosh}[1]{\cosh^{-1}\COOL@decide@paren{ArcCosh}{#1}}`

`\ArcTanh` Inverse hyperbolic tangent, `\ArcTanh{x}`, $\tanh^{-1}(x)$
282 `\newcommand{\COOL@notation@ArcTanhParen}{p}`
283 `\newcommand{\ArcTanh}[1]{\tanh^{-1}\COOL@decide@paren{ArcTanh}{#1}}`

`\ArcCsch` Inverse hyperbolic cosecant, `\ArcCsch{x}`, $\operatorname{csch}^{-1}(x)$
284 `\newcommand{\COOL@notation@ArcCschParen}{p}`
285 `\newcommand{\ArcCsch}[1]{\csch^{-1}\COOL@decide@paren{ArcCsch}{#1}}`

`\ArcSech` Inverse hyperbolic secant, `\ArcSech{x}`, $\operatorname{sech}^{-1}(x)$
286 `\newcommand{\COOL@notation@ArcSechParen}{p}`
287 `\newcommand{\ArcSech}[1]{\sech^{-1}\COOL@decide@paren{ArcSech}{#1}}`

`\ArcCoth` Inverse hyperbolic cotangent, `\ArcCoth{x}`, $\coth^{-1}(x)$
288 `\newcommand{\COOL@notation@ArcCothParen}{p}`
289 `\newcommand{\ArcCoth}[1]{\coth^{-1}\COOL@decide@paren{ArcCoth}{#1}}`

1.3.8 Product Logarithms

`\LambertW` Lambert Function. `\LambertW` is an alias for `\ProductLog` and its properties are therefore set using that function

290 `\newcommand{\LambertW}[1]{\ProductLog{#1}}`

`\ProductLog` Generalized Lambert Function `\ProductLog{[<index>],<variable>}`.

Lambert Function `\ProductLog{x}` $W(x)$

Generalized Lambert Function `\ProductLog{k,x}` $W_k(x)$

291 `\newcommand{\COOL@notation@ProductLogParen}{p}`
292 `\newcommand{\ProductLog}[1]{%`
293 `\listval{#1}{0}%`
294 `\ifthenelse{\value{COOL@listpointer}=1}%`
295 `{%`
296 `W\COOL@decide@paren{ProductLog}{#1}%`
297 `}%`
298 `% else`
299 `{%`
300 `\ifthenelse{\value{COOL@listpointer}=2}%`
301 `{%`
302 `W_{\listval{#1}{1}}\COOL@decide@paren{ProductLog}{\listval{#1}{2}}%`
303 `}%`
304 `% else`
305 `{%`
306 `\PackageError{cool}{‘ProductLog’ Invaoid Argument}%`
307 `{Must have a comma separated list of length 1 or 2}`
308 `}%`
309 `}%`
310 `}`

1.3.9 Max and Min

`\Max` the maximum function, `\Max{x,y,z}`, $\max(x,y,z)$

311 `\newcommand{\COOL@notation@MaxParen}{p}`
312 `\newcommand{\Max}[1]{\max\COOL@decide@paren{Max}{#1}}`

`\Min` the minimum function, `\Min{x,y,z}`, $\min(x,y,z)$

313 `\newcommand{\COOL@notation@MinParen}{p}`
314 `\newcommand{\Min}[1]{\min\COOL@decide@paren{Min}{#1}}`

1.3.10 Bessel Functions

`\BesselJ` Bessel Function of the first kind, `\BesselJ{<nu>}{x}`, $J_\nu(x)$

315 `\newcommand{\COOL@notation@BesselJSymb}{J}`

```

316 \newcommand{\COOL@notation@BesselJParen}{p}
317 \newcommand{\BesselJ}[2]%
318 {\COOL@notation@BesselJSymb_{#1}\COOL@decide@paren{BesselJ}{#2}}

\BesselY Bessel Function of the second kind, \BesselY{\nu}{x},  $Y_\nu(x)$ 
319 \newcommand{\COOL@notation@BesselYSymb}{Y}
320 \newcommand{\COOL@notation@BesselYParen}{p}
321 \newcommand{\BesselY}[2]%
322 {\COOL@notation@BesselYSymb_{#1}\COOL@decide@paren{BesselY}{#2}}

\BesselI Modified Bessel Function of the first kind, \BesselI{\nu}{x},  $I_\nu(x)$ 
323 \newcommand{\COOL@notation@BesselISymb}{I}
324 \newcommand{\COOL@notation@BesselIParen}{p}
325 \newcommand{\BesselI}[2]%
326 {\COOL@notation@BesselISymb_{#1}\COOL@decide@paren{BesselI}{#2}}

\BesselK Modified Bessel Function of the second kind, \BesselK{\nu}{x},  $K_\nu(x)$ 
327 \newcommand{\COOL@notation@BesselKSymb}{K}
328 \newcommand{\COOL@notation@BesselKParen}{p}
329 \newcommand{\BesselK}[2]%
330 {\COOL@notation@BesselKSymb_{#1}\COOL@decide@paren{BesselK}{#2}}

1.3.11 Airy Functions

\AiryAi Airy Ai Function, \AiryAi{x},  $\text{Ai}(x)$ 
331 \newcommand{\COOL@notation@AiryAiParen}{p}
332 \DeclareMathOperator{\AiryAiSymb}{Ai}
333 \newcommand{\AiryAi}[1]{\AiryAiSymb\COOL@decide@paren{AiryAi}{#1}}

\AiryBi Airy Bi Function, \AiryBi{x},  $\text{Bi}(x)$ 
334 \newcommand{\COOL@notation@AiryBiParen}{p}
335 \DeclareMathOperator{\AiryBiSymb}{Bi}
336 \newcommand{\AiryBi}[1]{\AiryBiSymb\COOL@decide@paren{AiryBi}{#1}}

1.3.12 Struve Functions

\StruveH Struve H function, \StruveH{\nu}{z},  $\text{H}_\nu(z)$ 
337 \newcommand{\COOL@notation@StruveHParen}{p}
338 \newcommand{\StruveH}[2]{\bf H_{#1}\COOL@decide@paren{StruveH}{#2}}

\StruveL Struve L function, \StruveL{\nu}{z},  $\text{L}_\nu(z)$ 
339 \newcommand{\COOL@notation@StruveLParen}{p}
340 \newcommand{\StruveL}[2]{\bf L_{#1}\COOL@decide@paren{StruveL}{#2}}

```

1.3.13 Integer Functions

```

\Floor floor, \Floor{x},  $\lfloor x \rfloor$ 
341 \newcommand{\Floor}[1]{\lffloor #1 \rffloor}

\Ceiling ceiling, \Ceiling{x},  $\lceil x \rceil$ 
342 \newcommand{\Ceiling}[1]{\lceil #1 \rceil}

\Round round, \Round{x},  $\lfloor x \rfloor$ 
343 \newcommand{\Round}[1]{\lffloor #1 \rceil}

\iPart The integer part of a real number, \iPart{x}, \IntegerPart{x},  $\text{int}(x)$ 
\IntegerPart 344 \newcommand{\COOL@notation@IntegerPartParen}{p}
345 \DeclareMathOperator{\iPartSymb}{int}
346 \newcommand{\iPart}[1]{\iPartSymb\COOL@decide@paren{IntegerPart}{#1}}
347 \newcommand{\IntegerPart}[1]{\iPart{#1}}

\fPart the fractional part of a real number, \fPart{x}, \FractionalPart{x},  $\text{frac}(x)$ 
\FractionalPart 348 \newcommand{\COOL@notation@FractionalPartParen}{p}
349 \DeclareMathOperator{\fPartSymb}{frac}
350 \newcommand{\fPart}[1]{\fPartSymb\COOL@decide@paren{FractionalPart}{#1}}
351 \newcommand{\FractionalPart}[1]{\fPart{#1}}

\Mod Modulo, \Mod{n}{m},  $n \bmod m$ 
352 \newcommand{\COOL@notation@ModDisplay}{mod}
353 \newcommand{\Mod}[2]{%
354 \ifthenelse{\equal{\COOL@notation@ModDisplay}{mod}}{%
355 {%
356 #1 \mod #2%
357 }%
358 % ElseIf
359 { \ifthenelse{\equal{\COOL@notation@ModDisplay}{bmod}}{%
360 {%
361 #1 \bmod #2%
362 }%
363 % ElseIf
364 { \ifthenelse{\equal{\COOL@notation@ModDisplay}{pmod}}{%
365 {%
366 #1 \pmod #2%
367 }%
368 % ElseIf
369 { \ifthenelse{\equal{\COOL@notation@ModDisplay}{pod}}{%
370 {%
371 #1 \pod #2%
372 }%
373 % Else
374 {%
375 \PackageError{cool}{Invalid Option Sent}%
376 {ModDisplay can only be 'mod', 'bmod', 'pmod', or 'pod'}}%

```

```

377 }}}} %
378 }

\Quotient quotient, \Quotient{m}{n}, quotient(m,n)
379 \newcommand{\COOL@notation@QuotientParen}{p}
380 \DeclareMathOperator{\QuotientSymb}{quotient}
381 \newcommand{\Quotient}[2] %
382 {\QuotientSymb\COOL@decide@paren{Quotient}{#1,#2}}

\GCD greatest common divisor, \GCD{n_1,n_2,\ldots,n_m}, gcd(n_1,n_2,\ldots,n_m)
383 \newcommand{\COOL@notation@GCDParen}{p}
384 \newcommand{\GCD}[1]{\gcd\COOL@decide@paren{GCD}{#1}}

\ExtendedGCD Extended Greatest Common Divisor,
\EGCD \EGCD{n}{m}, \ExtendedGCD{n}{m}, egcd(n,m)
385 \newcommand{\COOL@notation@ExtendedGCDParen}{p}
386 \DeclareMathOperator{\ExtendedGCDsymb}{egcd}
387 \newcommand{\ExtendedGCD}[2] %
388 {\ExtendedGCDsymb\COOL@decide@paren{ExtendedGCD}{#1,#2}}
389 \newcommand{\EGCD}[2]{\ExtendedGCD{#1}{#2}}

\LCM Least Common Multiple, \LCM{n_1,n_2,\ldots,n_m}, lcm(n_1,n_2,\ldots,n_m)
390 \newcommand{\COOL@notation@LCMParen}{p}
391 \DeclareMathOperator{\LCMSymb}{lcm}
392 \newcommand{\LCM}[1]{\LCMSymb\COOL@decide@paren{LCM}{#1}}

\Fibonacci Fibonacci number, \Fibonacci{n},  $F_n$ , and
Fibonacci Polynomial, \Fibonacci{n,x},  $F_n(x)$ 
393 \newcommand{\COOL@notation@FibonacciParen}{p}
394 \newcommand{\Fibonacci}[1] %
395 \liststore{#1}{\COOL@Fibonacci@arg@}%
396 \listval{#1}{0}%
397 \ifthenelse{\value{\COOL@listpointer} = 1} %
398 {%
399 F_{#1}%
400 }%
401 % ElseIf
402 { \ifthenelse{\value{\COOL@listpointer} = 2} %
403 {%
404 F_{\COOL@Fibonacci@arg@i}%
405 \COOL@decide@paren{Fibonacci}{\COOL@Fibonacci@arg@ii}%
406 }%
407 % Else
408 {%
409 \PackageError{cool}{Invalid Argument}%
410 {'Fibonacci' can only accept a
411 comma separate list of length 1 or 2}%
412 }%
413 }

```

```

\ Euler Euler number, \ Euler{n},  $E_n$ , and Euler Polynomial, \ Euler{n,x},  $E_n(x)$ 
414 \newcommand{\COOL@notation@EulerParen}{p}
415 \newcommand{\ Euler}[1]{%
416 \liststore{#1}{\COOL@Euler@arg}%
417 \listval{#1}{0}%
418 \ifthenelse{\value{\COOL@listpointer} = 1}%
419 {%
420 E_{#1}%
421 }%
422 % ElseIf
423 { \ifthenelse{\value{\COOL@listpointer} = 2}%
424 {%
425 E_{\COOL@Euler@arg%i}%
426 \COOL@decide@paren{ Euler}{\COOL@Euler@arg@ii}%
427 }%
428 % Else
429 {%
430 \PackageError{cool}{Invalid Argument}%
431 {' Euler' can only accept a
432 comma separate list of length 1 or 2}%
433 }%
434 }

\ Bernoulli Bernoulli number, \ Bernoulli{n},  $B_n$  and
Bernoulli Polynomial \ Bernoulli{n,x},  $B_n(x)$ 
435 \newcommand{\COOL@notation@BernoulliParen}{p}
436 \newcommand{\ Bernoulli}[1]{%
437 \liststore{#1}{\COOL@Bernoulli@arg}%
438 \listval{#1}{0}%
439 \ifthenelse{\value{\COOL@listpointer} = 1}%
440 {%
441 B_{#1}%
442 }%
443 % ElseIf
444 { \ifthenelse{\value{\COOL@listpointer} = 2}%
445 {%
446 B_{\COOL@Bernoulli@arg%i}%
447 \COOL@decide@paren{ Bernoulli}{\COOL@Bernoulli@arg@ii}%
448 }%
449 % Else
450 {%
451 \PackageError{cool}{Invalid Argument}%
452 {' Bernoulli' can only accept a
453 comma separate list of length 1 or 2}%
454 }%
455 }

\ StirlingSOne Stirling number of the first kind \ StirlingSOne{n}{m},  $S_n^{(m)}$ 
456 \newcommand{\ StirlingSOne}[2]{S_{#1}^{\in p{#2}}}
```

`\StirlingSTwo` Stirling number of the second kind, $\text{\StirlingSTwo}\{n\}\{m\}$, $\mathcal{S}_n^{(m)}$

```

457 \newcommand{\StirlingSTwo}[2]{\cal S}_{\#1}^{\inp{\#2}}

```

`\PartitionsP` Number of unrestricted partitions of an integer, $\text{\PartitionsP}\{x\}$, $p(x)$

```

458 \newcommand{\COOL@notation@PartitionsPParen}\{p\}
459 \newcommand{\PartitionsP}[1]{p\COOL@decide@paren{PartitionsP}\{\#1\}}

```

`\PartitionsQ` number of partitions of an integer into distinct parts, $\text{\PartitionsQ}\{x\}$, $q(x)$

```

460 \newcommand{\COOL@notation@PartitionsQParen}\{p\}
461 \newcommand{\PartitionsQ}[1]{q\COOL@decide@paren{PartitionsQ}\{\#1\}}

```

`\DiscreteDelta` Discrete delta function,

```

\DiscreteDelta{n_1,n_2,\ldots,n_m}, \delta(n_1,n_2,\ldots,n_m)
462 \newcommand{\COOL@notation@DiscreteDeltaParen}\{p\}
463 \newcommand{\DiscreteDelta}[1]{%
464 {\delta\COOL@decide@paren{DiscreteDelta}\{\#1\}}

```

`\KroneckerDelta` Kronecker Delta, $\text{\KroneckerDelta}\{n_1,n_2,\ldots,n_m\}$, $\delta^{n_1 n_2 \dots n_m}$

```

465 \newcommand{\COOL@notation@KroneckerDeltaUseComma}\{false\}%
466 \newcommand{\COOL@notation@KroneckerDeltaIndicies}\{local\}
467 \newcommand{\KroneckerDelta}[2][u]{%
468 \liststore{\#2}\COOL@arg@}%
469 \listval{\#2}\{0\}%
470 \def\COOL@arg@temp{%
471 \forLoop{1}\{value\COOL@listpointer\}\COOL@ct}%
472 {%
473 \ifthenelse{\equal{\COOL@notation@KroneckerDeltaUseComma}\{true\}}%
474 {%
475 \ifthenelse{\NOT \value\COOL@ct} = 1}
476 {%
477 \edef\COOL@arg@temp%
478 {\COOL@arg@temp, \csname COOL@arg@\roman\COOL@ct\endcsname}%
479 }%
480 % Else
481 {%
482 \edef\COOL@arg@temp%
483 {\COOL@arg@temp \csname COOL@arg@\roman\COOL@ct\endcsname}%
484 }%
485 }%
486 % Else
487 {%
488 \edef\COOL@arg@temp%
489 {\COOL@arg@temp \csname COOL@arg@\roman\COOL@ct\endcsname}%
490 }%
491 }%
492 \delta\COOL@decide@indicies{KroneckerDelta}\{\#1\}\COOL@arg@temp}%
493 }

```


`\LeviCivita` Levi-Civita totally anti-symmetric Tensor density,
`\Signature` `\LeviCivita{n_1,n_2,\ldots,n_m}, \epsilon^{n_1 n_2 \dots n_m}`

```

494 \newcommand{\COOL@notation@LeviCivitaUseComma}{false}
495 \newcommand{\COOL@notation@LeviCivitaIndicies}{local}
496 \newcommand{\LeviCivita}[2][u]{%
497 \liststore{#2}{COOL@arg@}%
498 \listval{#2}{0}%
499 \def\COOL@arg@temp{%
500 \forLoop{1}{\value{COOL@listpointer}}{\COOL@ct}%
501 {%
502 \ifthenelse{\equal{\COOL@notation@LeviCivitaUseComma}{true}}%
503 {%
504 \ifthenelse{\NOT \value{COOL@ct} = 1}%
505 {%
506 \edef\COOL@arg@temp%
507 {\COOL@arg@temp, \csname COOL@arg@\roman{COOL@ct}\endcsname}%
508 }%
509 % Else
510 {%
511 \edef\COOL@arg@temp%
512 {\COOL@arg@temp \csname COOL@arg@\roman{COOL@ct}\endcsname}%
513 }%
514 }%
515 % Else
516 {%
517 \edef\COOL@arg@temp%
518 {\COOL@arg@temp \csname COOL@arg@\roman{COOL@ct}\endcsname}%
519 }%
520 }%
521 \epsilon\COOL@decide@indicies{LeviCivita}{#1}{\COOL@arg@temp}%
522 }%
523 \newcommand{\Signature}[2][u]{\LeviCivita[#1]{#2}}
```

1.3.14 Classical Orthogonal Polynomials

`\HermiteH` Hermite Polynomial, `\HermiteH{n}{x}, H_n(x)`

```

524 \newcommand{\COOL@notation@HermiteHParen}{p}
525 \newcommand{\COOL@notation@HermiteHSymb}{H}
526 \newcommand{\HermiteH}[2]%
527 {\COOL@notation@HermiteHSymb_{#1}\COOL@decide@paren{HermiteH}{#2}}
```

`\LagerreL` Lagerre Polynomial, `\LagerreL{\nu,x}, L_\nu(x)` and
Generalized Lagerre Polynomial `\LagerreL{\nu,\lambda,x}, L_\nu^\lambda(x)`

```

528 \newcommand{\COOL@notation@LagerreLParen}{p}
529 \newcommand{\COOL@notation@LagerreLSymb}{L}
530 \newcommand{\LagerreL}[1]{%
531 \liststore{#1}{COOL@list@temp@}%
532 \listval{#1}{0}%
533 \ifthenelse{\value{COOL@listpointer}=2}%

```

```

534 {%
535 \COOL@notation@LaugerreLSymb_{\COOL@list@temp@i}%
536 \COOL@decide@paren{LaugerreL}{\COOL@list@temp@ii}%
537 }%
538 % Else If
539 { \ifthenelse{\value{COOL@listpointer}=3}%
540 {%
541 \COOL@notation@LaugerreLSymb_{\COOL@list@temp@i}~{\COOL@list@temp@ii}%
542 \COOL@decide@paren{LaugerreL}{\COOL@list@temp@iii}%
543 }%
544 % Else
545 {%
546 \PackageError{cool}{Invalid Argument}%
547 {'LaugerreL' only accepts a comma separated list of length 2 or 3}%
548 }%
549 }

```

\LegendreP Legendre Polynomials

Legendre Polynomial	$\backslash\text{LegendreP}\{n,x\}$	$P_n(x)$
Associated Legendre Polynomial		
of the first kind of type 2	$\backslash\text{LegendreP}\{\ell,m,x\}$	$P_\ell^m(x)$
	$\backslash\text{LegendreP}\{\ell,m,2,x\}$	$P_\ell^m(x)$
Associated Legendre Function		
of the first kind of type 3	$\backslash\text{LegendreP}\{\ell,m,3,x\}$	$\mathcal{P}_\ell^m(x)$

```

550 \newcommand{\COOL@notation@LegendrePParen}{p}
551 \newcommand{\COOL@notation@LegendrePSymb}{P}
552 \newcommand{\LegendreP}[1]{%
553 \liststore{#1}{COOL@LegendreP@arg@}%
554 \listval{#1}{0}%
555 \ifthenelse{\value{COOL@listpointer} = 2}%
556 {%
557 \COOL@notation@LegendrePSymb_{\COOL@LegendreP@arg@i}%
558 \COOL@decide@paren{LegendreP}{\COOL@LegendreP@arg@ii}%
559 }%
560 % ElseIf
561 { \ifthenelse{\value{COOL@listpointer} = 3}%
562 {%
563 \COOL@notation@LegendrePSymb_{\COOL@LegendreP@arg@i}%
564 ~{\COOL@LegendreP@arg@ii}%
565 \COOL@decide@paren{LegendreP}{\COOL@LegendreP@arg@iii}%
566 }%
567 % ElseIf
568 { \ifthenelse{\value{COOL@listpointer} = 4}%
569 {%
570 \isint{\COOL@LegendreP@arg@iii}%
571 \ifthenelse{\boolean{COOL@isint}}{%
572 {%
573 \ifcase\COOL@LegendreP@arg@iii\relax%
574 \PackageError{cool}{Invalid Argument}%

```

```

575 {'LegendreP' third argument must be > $ 1}%
576 \or%
577 \PackageError{cool}{Invalid Argument}%
578 {'LegendreP' third argument must be > $ 1}%
579 \or%
580 \COOL@notation@LegendrePSymb_{\COOL@LegendreP@arg@i}%
581 ^{\COOL@LegendreP@arg@ii}%
582 \COOL@decide@paren{LegendreP}{\COOL@LegendreP@arg@iv}%
583 \or%
584 {\cal P}_{\COOL@LegendreP@arg@i}%
585 ^{\COOL@LegendreP@arg@ii}%
586 \COOL@decide@paren{LegendreP}{\COOL@LegendreP@arg@iv}%
587 \else%
588 \PackageError{cool}{Invalid Argument}{unsupported}%
589 \fi%
590 }
591 % Else
592 {%
593 \PackageError{cool}{Invalid Argument}{third arg must be int}%
594 }%
595 }%
596 % Else
597 {%
598 \PackageError{cool}{Invalid Argument}%
599 {'LegendreP' can only accept a%
600 comma separated list of length 2-4}%
601 }}%
602 }

```

```

\LegendreQ Legendre Polynomials of the second kind
Legendre Polynomial \LegendreQ{n,x}  $Q_n(x)$ 
Associated Legendre Polynomial
of the second kind of type 2 \LegendreQ{\ell,m,x}  $Q_\ell^m(x)$ 
\LegendreQ{\ell,m,2,x}  $Q_\ell^m(x)$ 
Associated Legendre Function
of the second kind of type 3 \LegendreQ{\ell,m,3,x}  $Q_\ell^m(x)$ 
603 \newcommand{\COOL@notation@LegendreQParen}{p}
604 \newcommand{\COOL@notation@LegendreQSymb}{Q}
605 \newcommand{\LegendreQ}[1]{%
606 \liststore{#1}{\COOL@LegendreQ@arg@}%
607 \listval{#1}{0}%
608 \ifthenelse{\value{\COOL@listpointer} = 2}%
609 {%
610 \COOL@notation@LegendreQSymb_{\COOL@LegendreQ@arg@i}%
611 \COOL@decide@paren{LegendreQ}{\COOL@LegendreQ@arg@ii}%
612 }%
613 % ElseIf
614 { \ifthenelse{\value{\COOL@listpointer} = 3}%
615 {%

```

```

616 \COOL@notation@LegendreQSymb_{\COOL@LegendreQ@arg@i}%
617 ^{\COOL@LegendreQ@arg@ii}%
618 \COOL@decide@paren{LegendreQ}{\COOL@LegendreQ@arg@iii}%
619 }%
620 % ElseIf
621 { \ifthenelse{\value{COOL@listpointer} = 4}%
622 {%
623 \isint{\COOL@LegendreQ@arg@iii}%
624 \ifthenelse{\boolean{COOL@isint}}%
625 {%
626 \ifcase\COOL@LegendreQ@arg@iii\relax%
627 \PackageError{cool}{Invalid Argument}%
628 {'LegendreQ' third argument must be >= 1}%
629 \or%
630 \PackageError{cool}{Invalid Argument}%
631 {'LegendreQ' third argument must be >= 1}%
632 \or%
633 \COOL@notation@LegendreQSymb_{\COOL@LegendreQ@arg@i}%
634 ^{\COOL@LegendreQ@arg@ii}%
635 \COOL@decide@paren{LegendreQ}{\COOL@LegendreQ@arg@iv}%
636 \or%
637 {\cal Q}_{\COOL@LegendreQ@arg@i}%
638 ^{\COOL@LegendreQ@arg@ii}%
639 \COOL@decide@paren{LegendreQ}{\COOL@LegendreQ@arg@iv}%
640 \else%
641 \PackageError{cool}{Invalid Argument}{unsupported}%
642 \fi%
643 }
644 % Else
645 {%
646 \PackageError{cool}{Invalid Argument}{third arg must be int}%
647 }%
648 }%
649 % Else
650 {%
651 \PackageError{cool}{Invalid Argument}%
652 {'LegendreQ' can only accept a%
653 comma separated list of length 2-4}%
654 }%
655 }

\ChebyshevT Chebyshev Polynomial of the first kind, ChebyshevT{n}{x}, ChebyshevTnx
656 \newcommand{\COOL@notation@ChebyshevTParen}{p}
657 \newcommand{\COOL@notation@ChebyshevTSymb}{T}
658 \newcommand{\ChebyshevT}[2]%
659 {\COOL@notation@ChebyshevTSymb_{#1}\COOL@decide@paren{ChebyshevT}{#2}}

\ChebyshevU , \ChebyshevU{n}{z},  $U_n(z)$  Chebyshev Polynomial of the second kind
660 \newcommand{\COOL@notation@ChebyshevUParen}{p}
661 \newcommand{\COOL@notation@ChebyshevUSymb}{U}

```

```

662 \newcommand{\ChebyshevU}[2]%
663 {\COOL@notation@ChebyshevUSymb_{#1}\COOL@decide@paren{ChebyshevU}{#2}}

```

```

\JacobiP Jacobi Polynomial, \JacobiP{n}{a}{b}{x},  $P_n^{(a,b)}(x)$ 
664 \newcommand{\COOL@notation@JacobiPParen}{p}
665 \newcommand{\COOL@notation@JacobiPSymb}{P}
666 \newcommand{\JacobiP}[4]{%
667 \COOL@notation@JacobiPSymb_{#1}^{\inp{#2, #3}}%
668 \COOL@decide@paren{JacobiP}{#4}%
669 }

```

1.3.15 Associated Polynomials

```

\AssocLegendreP Associated Legendre Polynomial of the first kind of type 2
\AssocLegendreP{\ell}{m}{x},  $P_\ell^m(x)$ 
670 \newcommand{\AssocLegendreP}[3]{\LegendreP{#1,#2,#3}}

```

```

\AssocLegendreQ Associated Legendre Polynomial of the second kind of type 2
\AssocLegendreQ{\ell}{m}{x},  $Q_\ell^m(x)$ 
671 \newcommand{\AssocLegendreQ}[3]{\LegendreQ{#1,#2,#3}}

```

```

\GegenbauerC Gegenbauer Polynomial, \GegenbauerC{n}{\lambda}{x},  $C_n^\lambda(x)$ 
672 \newcommand{\COOL@notation@GegenbauerCParen}{p}
673 \newcommand{\COOL@notation@GegenbauerCSymb}{C}
674 \newcommand{\GegenbauerC}[3]{%
675 \COOL@notation@GegenbauerCSymb_{#1}^{#2}%
676 \COOL@decide@paren{GegenbauerC}{#3}%
677 }

```

```

\SphericalHarmonicY Spherical Harmonic, \SpHarmY{\ell}{m}{\theta}{\phi},
\SphericalHarmY \SphericalHarmY{\ell}{m}{\theta}{\phi},
\SpHarmY \SphericalHarmonicY{\ell}{m}{\theta}{\phi},  $Y_\ell^m(\theta, \phi)$ 
678 \newcommand{\COOL@notation@SphericalHarmonicYParen}{p}
679 \newcommand{\COOL@notation@SphericalHarmonicYSymb}{Y}
680 \newcommand{\SphericalHarmonicY}[4]{%
681 \COOL@notation@SphericalHarmonicYSymb_{#1}^{#2}%
682 \COOL@decide@paren{SphericalHarmonicY}{#3,#4}%
683 }
684 \newcommand{\SphericalHarmY}[4]{\SphericalHarmonicY{#1}{#2}{#3}{#4}}
685 \newcommand{\SpHarmY}[4]{\SphericalHarmonicY{#1}{#2}{#3}{#4}}

```

1.3.16 Other Polynomials

```

\CyclotomicC Cyclotomic Polynomial, \CyclotomicC{n}{z},  $C_n(z)$ 
686 \newcommand{\COOL@notation@CyclotomicCParen}{p}
687 \newcommand{\CyclotomicC}[2]%
688 {C_{#1}\COOL@decide@paren{CyclotomicC}{#2}}

```

`\FibonacciF` Fibonacci Polynomial, `\FibonacciF{n}{z}`, $F_n(z)$
689 `\newcommand{\FibonacciF}[2]{\Fibonacci{#1,#2}}`

`\EulerE` Euler Polynomial, `\EulerE{n}{z}`, $E_n(z)$
690 `\newcommand{\EulerE}[2]{\Euler{#1,#2}}`

`\BernoulliB` Bernoulli Polynomial, `\BernoulliB{n}{z}`, $B_n(z)$
691 `\newcommand{\BernoulliB}[2]{\Bernoulli{#1,#2}}`

1.3.17 Factorial Functions

`\Factorial` Factorial, `\Factorial{n}`, $n!$
692 `\newcommand{\Factorial}[1]{#1!}`

`\DblFactorial` Double Factorial, `\DblFactorial{n}`, $n!!$
693 `\newcommand{\DblFactorial}[1]{#1!!}`

`\Binomial` binomial, `\Binomial{n}{r}`, $\binom{n}{r}$
694 `\newcommand{\Binomial}[2]{\binom{#1}{#2}}`

`\Multinomial` Multinomial, `\Multinomial{n_1,\ldots,n_m}`, $(n_1 + \dots + n_m; n_1, \dots, n_m)$
695 `\newcommand{\Multinomial}[1]{%`
696 `{%`
697 `\listval{#1}{0}% get the length of the list`
698 `\setcounter{COOL@listlen}{\value{COOL@listpointer}}% record length`
699 `\liststore{#1}{COOL@list@temp@}%`
700 `\isint{\COOL@list@temp@i}% check that the entries are integers`
701 `\setcounter{COOL@ct}{2}%`
702 `\whiledo{ \boolean{COOL@isint} \AND`
703 `\NOT \value{COOL@ct}>\value{COOL@listlen} }%`
704 `{%`
705 `\def\COOL@Multinomial@tempa%`
706 `{\csname COOL@list@temp@\roman{COOL@ct}\endcsname}%`
707 `\isint{\COOL@Multinomial@tempa}%`
708 `\stepcounter{COOL@ct}%`
709 `}%`
710 `\ifthenelse{\boolean{COOL@isint}}%`
711 `{%`
712 `% all of them are integers`
713 `\setcounter{COOL@ct@}{\COOL@list@temp@i}% records the sum`
714 `\forLoop{2}{\value{COOL@listlen}}{\COOL@ct}%`
715 `{%`
716 `\addtocounter{COOL@ct@}%`
717 `{\csname COOL@list@temp@\roman{COOL@ct}\endcsname}%`
718 `}%`
719 `\left(\arabic{COOL@ct}%`
720 `}%`
721 `% Else`

```

722 {%
723 \left(
724 \listval{#1}{1}%
725 \forLoop{2}{\value{COOL@listlen}}{COOL@ct}%
726 {%
727 + \listval{#1}{\arabic{COOL@ct}}%
728 }%
729 }%
730 ;#1\right)%
731 }

```

1.3.18 Gamma Functions

<code>\GammaFunc</code>	Gamma Function	<code>\GammaFunc{z}</code>	$\Gamma(z)$
	Incomplete Gamma Function	<code>\GammaFunc{a,z}</code>	$\Gamma(a,z)$
	Generalized Incomplete Gamma Function	<code>\GammaFunc{a,x,y}</code>	$\Gamma(a,x,y)$

```

732 \newcommand{\COOL@notation@GammaFuncParen}{p}
733 \newcommand{\GammaFunc}[1]{%
734 \listval{#1}{0}%
735 \ifthenelse{\value{COOL@listpointer} = 1}%
736 {%
737 \Gamma\COOL@decide@paren{GammaFunc}{#1}%
738 }%
739 % ElseIf
740 { \ifthenelse{\value{COOL@listpointer} = 2}%
741 {%
742 \Gamma\COOL@decide@paren{GammaFunc}{#1}%
743 }%
744 % ElseIf
745 { \ifthenelse{\value{COOL@listpointer} = 3}%
746 {%
747 \Gamma\COOL@decide@paren{GammaFunc}{#1}%
748 }%
749 % Else
750 {%
751 \PackageError{cool}{Invalid Argument}%
752 {'GammaFunc' can only accept a comma separate list of length 1 to 3}%
753 }%
754 }%
755 }

```

`\IncGamma` incomplete Gamma function, `\IncGamma{a}{x}`, $\Gamma(a,x)$

```

756 \newcommand{\IncGamma}[2]{\GammaFunc{#1,#2}}

```

`\GenIncGamma` Generalized Incomplete Gamma, `\GenIncGamma{a}{x}{y}`, $\Gamma(a,x,y)$

```

757 \newcommand{\GenIncGamma}[3]{\GammaFunc{#1,#2,#3}}

```

`\GammaRegularized` Regularized Incomplete Gamma

`\RegIncGamma`

`\GammaReg`

```

\GammaRegularized{a,x}  Q(a,x)
\RegIncGamma{a}{x}      Q(a,x)
\GammaReg{a,x}          Q(a,x)
758 \newcommand{\COOL@notation@GammaRegularizedParen}{p}%
759 \newcommand{\GammaRegularized}[1]{%
760 \listval{#1}{0}%
761 \ifthenelse{\value{COOL@listpointer} = 2}%
762 {%
763 Q\COOL@decide@paren{\GammaRegularized}{#1}%
764 }%
765 % ElseIf
766 { \ifthenelse{\value{COOL@listpointer} = 3}%
767 {%
768 Q\COOL@decide@paren{\GammaRegularized}{#1}%
769 }%
770 % Else
771 {%
772 \PackageError{cool}{Invalid Argument}%
773 {'GammaRegularized' can only accept comma%
774 separated lists of length 2 or 3}%
775 }%
776 }%
777 }
778 \newcommand{\RegIncGamma}[2]{\GammaRegularized{#1, #2}}
779 \newcommand{\GammaReg}[1]{\GammaRegularized{#1}}

```

```

\RegIncGammaInv  Inverse of Regularized Incomplete Gamma,
\InverseGammaRegularized  \RegIncGammaInv{a}{x}  Q^{-1}(a,x)
\GammaRegInv      \InverseGammaRegularized{a,x}  Q^{-1}(a,x)
                  \GammaRegInv{a,x}             Q^{-1}(a,x)
780 \newcommand{\COOL@notation@InverseGammaRegularizedParen}{p}%
781 \newcommand{\InverseGammaRegularized}[1]{%
782 \listval{#1}{0}%
783 \ifthenelse{\value{COOL@listpointer} = 2}%
784 {%
785 Q^{-1}\COOL@decide@paren{\InverseGammaRegularized}{#1}%
786 }%
787 % ElseIf
788 { \ifthenelse{\value{COOL@listpointer} = 3}%
789 {%
790 Q^{-1}\COOL@decide@paren{\InverseGammaRegularized}{#1}%
791 }%
792 % Else
793 {%
794 \PackageError{cool}{Invalid Argument}%
795 {'InverseGammaRegularized' can only accept%
796 a comma separated list of length 2 or 3}%
797 }%
798 }%

```



```

799 }
800 \newcommand{\RegIncGammaInv}[2]{\InverseGammaRegularized{#1, #2}}
801 \newcommand{\GammaRegInv}[1]{\InverseGammaRegularized{#1}}

\GenRegIncGamma Generalized Regularized Incomplete Gamma
      \GenRegIncGamma{a}{x}{y}    $Q(a, x, y)$ 
      \GammaRegularized{a, x, y}   $Q(a, x, y)$ 
802 \newcommand{\GenRegIncGamma}[3]{\GammaRegularized{#1, #2, #3}}

\GenRegIncGammaInv Inverse of Gen. Reg. Incomplete Gamma, \GenRegIncGammaInv{a}{x}{y},
       $Q^{-1}(a, x, y)$ 
803 \newcommand{\GenRegIncGammaInv}[3]{\InverseGammaRegularized{#1, #2, #3}}

\Pochhammer Pochhammer Symbol \Pochhammer{a}{n},  $(a)_n$ 
804 \newcommand{\Pochhammer}[2]{\inp{#1}_{#2}}

\LogGamma Log Gamma Function, \LogGamma{x},  $\log\Gamma(x)$ 
805 \newcommand{\COOL@notation@LogGammaParen}{p}
806 \DeclareMathOperator{\LogGammaSymb}{log\Gamma}
807 \newcommand{\LogGamma}[1]{\LogGammaSymb\COOL@decide@paren{LogGamma}{#1}}
```

1.3.19 Derivatives of Gamma Functions

```

\DiGamma Digamma function, \DiGamma{x},  $F(x)$ 
808 \newcommand{\COOL@notation@DiGammaParen}{p}
809 \newcommand{\DiGamma}[1]{\digamma\COOL@decide@paren{DiGamma}{#1}}

PolyGamma function, \PolyGamma{\nu}{x},  $\psi^{(\nu)}(x)$ 

\PolyGamma
810 \newcommand{\COOL@notation@PolyGammaParen}{p}
811 \newcommand{\PolyGamma}[2]{%
812 {\psi~{\inp{#1}}}\COOL@decide@paren{PolyGamma}{#2}}

\HarmNum Harmonic Number
      Harmonic Number      \HarmNum{x}       $H_x$ 
      General Harmonic Number \HarmNum{x, r}   $H_x^{(r)}$ 
813 \newcommand{\HarmNum}[1]{%
814 \listval{#1}{0}%
815 \ifthenelse{\value{COOL@listpointer}=1}%
816 {%
817 H_{#1}
818 }%
819 % Else If
820 { \ifthenelse{\value{COOL@listpointer}=2}%
821 {%
822 \liststore{#1}{COOL@list@temp}%
823 H^{\inp{\COOL@list@temp@ii}}_{\COOL@list@temp@i}%

```

```

824 }%
825 % Else
826 {%
827 \PackageError{cool}{Invalid Argument}%
828 {'Harm Num' can only accept a comma separated list of length 1 or 2}%
829 }}%
830 }

```

1.3.20 Beta Functions

	Beta Function	$\backslash\text{Beta}\{a,b\}$	$B(a,b)$
$\backslash\text{Beta}$	Incomplete Beta Function	$\backslash\text{Beta}\{z,a,b\}$	$B_z(a,b)$
	Generalized Incomplete Beta Function	$\backslash\text{Beta}\{z_1,z_2,a,b\}$	$B_{(z_1,z_2)}(a,b)$

```

831 \newcommand{\COOL@notation@BetaParen}{p}
832 \newcommand{\Beta}[1]{%
833 \liststore{#1}{\COOL@Beta@arg@}%
834 \listval{#1}{0}%
835 \ifthenelse{\value{\COOL@listpointer} = 2}%
836 {%
837 B\COOL@decide@paren{Beta}{\COOL@Beta@arg@i, \COOL@Beta@arg@ii}%
838 }%
839 % ElseIf
840 { \ifthenelse{\value{\COOL@listpointer} = 3}%
841 {%
842 B_{\COOL@Beta@arg@i}%
843 \COOL@decide@paren{Beta}{\COOL@Beta@arg@ii, \COOL@Beta@arg@iii}%
844 }%
845 % ElseIf
846 { \ifthenelse{\value{\COOL@listpointer} = 4}%
847 {%
848 B_{\inp{\COOL@Beta@arg@i,\COOL@Beta@arg@ii}}%
849 \COOL@decide@paren{Beta}{\COOL@Beta@arg@iii, \COOL@Beta@arg@iv}%
850 }%
851 % Else
852 {%
853 \PackageError{cool}{Invalid Argument}%
854 {'Beta' can only accept a comma separated list of length 2 to 4}%
855 }%
856 }}%
857 }

```

$\backslash\text{IncBeta}$ Incomplete Beta Function

$\backslash\text{IncBeta}\{z\}\{a\}\{b\}$	$B_z(a,b)$
$\backslash\text{Beta}\{z,a,b\}$	$B_z(a,b)$

```

858 \newcommand{\IncBeta}[3]{\Beta{#1,#2, #3}}

```

$\backslash\text{GenIncBeta}$ Generalized Incomplete Beta Function

$\backslash\text{GenIncBeta}\{x\}\{y\}\{a\}\{b\}$	$B_{(x,y)}(a,b)$
$\backslash\text{Beta}\{x,y,a,b\}$	$B_{(x,y)}(a,b)$

```

859 \newcommand{\GenIncBeta}[4]{\Beta{#1,#2,#3,#4}}

\BetaRegularized Regularized Incomplete Beta Function
\BetaReg \BetaRegularized{z,a,b}  $I_z(a,b)$ 
\RegIncBeta \BetaReg{z,a,b}  $I_z(a,b)$ 
\RegIncBeta{z}{a}{b}  $I_z(a,b)$ 

860 \newcommand{\COOL@notation@BetaRegularizedParen}{p}
861 \newcommand{\BetaRegularized}[1]{%
862 \liststore{#1}{\COOL@BetaRegularized@arg@}%
863 \listval{#1}{0}%
864 \ifthenelse{\value{\COOL@listpointer} = 3}%
865 {%
866 I_{\COOL@BetaRegularized@arg@i}%
867 \COOL@decide@paren{\BetaRegularized}%
868 {\COOL@BetaRegularized@arg@ii, \COOL@BetaRegularized@arg@iii}%
869 }%
870 % ElseIf
871 { \ifthenelse{\value{\COOL@listpointer} = 4}%
872 {%
873 I_{\inp{\COOL@BetaRegularized@arg@i, \COOL@BetaRegularized@arg@ii}}%
874 \COOL@decide@paren{\BetaRegularized}%
875 {\COOL@BetaRegularized@arg@iii, \COOL@BetaRegularized@arg@iv}%
876 }%
877 % Else
878 {%
879 \PackageError{cool}{Invalid Argument}%
880 {'BetaRegularized' can only accept%
881 a comma separated list of length 3 or 4}%
882 }%
883 }%
884 }
885 \newcommand{\RegIncBeta}[3]{\BetaRegularized{#1,#2,#3}}
886 \newcommand{\BetaReg}[1]{\BetaRegularized{#1}}

\InverseBetaRegularized Inverse of Regularized Incomplete Beta Function
\BetaRegInv \InverseBetaRegularized{z,a,b}  $I_z^{-1}(a,b)$ 
\RegIncBetaInv \BetaRegInv{z,a,b}  $I_z^{-1}(a,b)$ 
\RegIncBetaInv{z}{a}{b}  $I_z^{-1}(a,b)$ 

887 \newcommand{\COOL@notation@InverseBetaRegularizedParen}{p}
888 \newcommand{\InverseBetaRegularized}[1]{%
889 \liststore{#1}{\COOL@InverseBetaRegularized@arg@}%
890 \listval{#1}{0}%
891 \ifthenelse{\value{\COOL@listpointer} = 3}%
892 {%
893 I^{-1}_{\COOL@InverseBetaRegularized@arg@i}%
894 \COOL@decide@paren{\InverseBetaRegularized}%
895 {\COOL@InverseBetaRegularized@arg@ii,%
896 \COOL@InverseBetaRegularized@arg@iii}%
897 }%

```

```

898 % ElseIf
899 { \ifthenelse{\value{COOL@listpointer} = 4}%
900 {%
901 I^{-1}_{\inp{ \COOL@InverseBetaRegularized@arg@i,%
902 \COOL@InverseBetaRegularized@arg@ii,%
903 }%
904 }%
905 \COOL@decide@paren{InverseBetaRegularized}%
906 {\COOL@InverseBetaRegularized@arg@iii,%
907 \COOL@InverseBetaRegularized@arg@iv}%
908 }%
909 % Else
910 {%
911 \PackageError{cool}{Invalid Argument}%
912 {'InverseBetaRegularized' can only accept%
913 a comma separated list of length 3 or 4}%
914 }%
915 }%
916 }
917 \newcommand{\RegIncBetaInv}[3]{\InverseBetaRegularized{#1,#2,#3}}
918 \newcommand{\BetaRegInv}[1]{\InverseBetaRegularized{#1}}

\GenRegIncBeta Generalized Regularized Incomplete Beta Func
\GenRegIncBeta{x}{y}{a}{b} B_{(x,y)}(a,b)
\Beta{x,y,a,b} B_{(x,y)}(a,b)
919 \newcommand{\GenRegIncBeta}[4]{\Beta{#1,#2,#3,#4}}

\GenRegIncBetaInv Inverse of Generalized Regularized Incomplete Beta Function
\GenRegIncBetaInv{x}{y}{z}{b} I^{-1}_{(x,y)}(z,b)
\InverseBetaRegularized{x,y,z,b} I^{-1}_{(x,y)}(z,b)
920 \newcommand{\GenRegIncBetaInv}[4]{\InverseBetaRegularized{#1,#2,#3,#4}}

```

1.3.21 Error Functions

```

\Erf Error Function \Erf{x} erf(x)
Generalized Error Function \Erf{x,y} erf(x,y)
921 \newcommand{\COOL@notation@ErfParen}{p}
922 \DeclareMathOperator{\ErfSymb}{erf}
923 \newcommand{\Erf}[1]{%
924 \liststore{#1}{\COOL@Erf@arg@}%
925 \listval{#1}{0}%
926 \ifthenelse{\value{COOL@listpointer} = 1}%
927 {%
928 \ErfSymb\COOL@decide@paren{Erf}{#1}
929 }%
930 % ElseIf
931 { \ifthenelse{\value{COOL@listpointer} = 2}%
932 {%

```

```

933 \ErfSymb\COOL@decide@paren{Erf}{#1}
934 }%
935 % Else
936 {%
937 \PackageError{cool}{Invalid Argument}%
938 {'Erf' can only accept a comma separated list of length 1 or 2}%
939 }%
940 }%
941 }

\ErfInv Inverse of Error Function
      \ErfInv{x}      erf-1(x)
      \ErfInv{x,y}    erf-1(x,y)
942 \newcommand{\COOL@notation@ErfInvParen}{p}
943 \newcommand{\ErfInv}[1]{%
944 \liststore{#1}{COOL@Erf@arg@}%
945 \listval{#1}{0}%
946 \ifthenelse{\value{COOL@listpointer} = 1}%
947 {%
948 \ErfSymb-1\COOL@decide@paren{ErfInv}{#1}
949 }%
950 % ElseIf
951 { \ifthenelse{\value{COOL@listpointer} = 2}%
952 {%
953 \ErfSymb-1\COOL@decide@paren{ErfInv}{#1}
954 }%
955 % Else
956 {%
957 \PackageError{cool}{Invalid Argument}%
958 {'Erf' can only accept a comma separated list of length 1 or 2}%
959 }%
960 }%
961 }

\GenErf Generalized Error Function and its inverse
\GenErfInv      \GenErf{z_1}{z_2}      erf(z1, z2)
                \GenErfInv{z_1}{z_2}    erf-1(z1, z2)
962 \newcommand{\GenErf}[2]{\Erf{#1,#2}}
963 \newcommand{\GenErfInv}[2]{\ErfInv{#1, #2}}

\Erfc Complimentary Error Function and its inverse
      \Erfc{z}      erfc(z)
      \ErfcInv{z}    erfc-1(z)
964 \newcommand{\COOL@notation@ErfcParen}{p}
965 \DeclareMathOperator{\ErfcSymb}{erfc}
966 \newcommand{\Erfc}[1]{\ErfcSymb\COOL@decide@paren{Erfc}{#1}}
967 \newcommand{\COOL@notation@ErfcInvParen}{p}
968 \newcommand{\ErfcInv}[1]{%
969 {\ErfcSymb-1\COOL@decide@paren{ErfcInv}{#1}}

```

`\Erfi` Imaginary Error Function, `\Erfi{z}`, $\operatorname{erfi}(z)$
970 `\newcommand{\COOL@notation@ErfiParen}{p}`
971 `\DeclareMathOperator{\ErfiSymb}{erfi}`
972 `\newcommand{\Erfi}[1]{\ErfiSymb\COOL@decide@paren{Erfi}{#1}}`

1.3.22 Fresnel Integrals

`\FresnelS` Fresnel Integral, `\FresnelS{z}`, $S(z)$
973 `\newcommand{\COOL@notation@FresnelSParen}{p}`
974 `\newcommand{\FresnelS}[1]{S\COOL@decide@paren{FresnelS}{#1}}`

`\FresnelC` Fresnel Integral, `\FresnelC{z}`, $C(z)$
975 `\newcommand{\COOL@notation@FresnelCParen}{p}`
976 `\newcommand{\FresnelC}[1]{C\COOL@decide@paren{FresnelC}{#1}}`

1.3.23 Exponential Integrals

`\ExpIntE` Exponential Integral, `\ExpIntE{\nu}{x}`, $E_\nu(x)$
977 `\newcommand{\COOL@notation@ExpIntEParen}{p}`
978 `\newcommand{\ExpIntE}[2]{E_{#1}\COOL@decide@paren{ExpIntE}{#2}}`

`\ExpIntEi` Exponential Integral, `\ExpIntEi{x}`, $Ei(x)$
979 `\newcommand{\COOL@notation@ExpIntEiParen}{p}`
980 `\DeclareMathOperator{\ExpIntEiSymb}{Ei}`
981 `\newcommand{\ExpIntEi}[1]{\%`
982 `{\ExpIntEiSymb\COOL@decide@paren{ExpIntEi}{#1}}`

`\LogInt` Logarithmic Integral, `\LogInt{x}`, $\operatorname{li}(x)$
983 `\newcommand{\COOL@notation@LogIntParen}{p}`
984 `\DeclareMathOperator{\LogIntSymb}{li}`
985 `\newcommand{\LogInt}[1]{\LogIntSymb\COOL@decide@paren{LogInt}{#1}}`

`\SinInt` Sine Integral, `\SinInt{x}`, $\operatorname{Si}(x)$
986 `\newcommand{\COOL@notation@SinIntParen}{p}`
987 `\DeclareMathOperator{\SinIntSymb}{Si}`
988 `\newcommand{\SinInt}[1]{\SinIntSymb\COOL@decide@paren{SinInt}{#1}}`

`\CosInt` Cosine Integral, `\CosInt{x}`, $\operatorname{Ci}(x)$
989 `\newcommand{\COOL@notation@CosIntParen}{p}`
990 `\DeclareMathOperator{\CosIntSymb}{Ci}`
991 `\newcommand{\CosInt}[1]{\CosIntSymb\COOL@decide@paren{CosInt}{#1}}`

`\SinhInt` Hyperbolic Sine Integral, `\SinhInt{x}`, $\operatorname{Shi}(x)$
992 `\newcommand{\COOL@notation@SinhIntParen}{p}`
993 `\DeclareMathOperator{\SinhIntSymb}{Shi}`
994 `\newcommand{\SinhInt}[1]{\SinhIntSymb\COOL@decide@paren{SinhInt}{#1}}`

```

\CoshInt   Hyberbolic Cosine Integral, \CoshInt{x}, Chi( $x$ )
995 \newcommand{\COOL@notation@CoshIntParen}{p}
996 \DeclareMathOperator{\CoshIntSymb}{Chi}
997 \newcommand{\CoshInt}[1]{\CoshIntSymb\Cool@decide@paren{CoshInt}{#1}}

```

1.3.24 Hypergeometric Functions

COOL@Hypergeometric@pq@ab@value This macro is a decision maker that decides what to return for the Hypergeometric function since its results vary based on the nature of the input. This macro is called as

```

\COOL@Hypergeometric@pq@ab@value {‘p’|‘q’} { $\langle p\_input | q\_input \rangle$ } {‘a’|‘b’}
{ $\langle a\_input | b\_input \rangle$ }
998 \newcommand{\COOL@Hypergeometric@pq@ab@value}[4]{%
999 \ifthenelse{\boolean{COOL@#1@isint} \AND \boolean{COOL@#3@islist}}{%
1000 {#1 is an INT and #3 is a LIST
1001 \ifthenelse{ #2 = 0 }{%
1002 {%
1003 \PackageWarning{cool}{‘#3’-arg ignored}%
1004 }%
1005 % Else
1006 {%
1007 \ifthenelse{ #2 = 1 }{%
1008 {%
1009 \PackageError{cool}{‘Hypergeometric’ ‘#1’-arg mismatch with ‘#3’-arg}{}%
1010 }%
1011 % Else
1012 {%
1013 #4%
1014 }%
1015 }%
1016 }%
1017 % Else
1018 {%
1019 \ifthenelse{ \boolean{COOL@#1@isint} \AND
1020 \NOT \boolean{COOL@#3@islist} }{%
1021 {%
1022 \ifthenelse{ #2 = 0 }{%
1023 {%
1024 % return nothing
1025 }%
1026 % Else
1027 {%
1028 \ifthenelse{ #2 = 1 }{%
1029 {%
1030 % return
1031 #4%
1032 }%
1033 % Else
1034 {%

```

```

1035 \forLoop{1}{#2}{COOL@ct}
1036 {%
1037 \ifthenelse{ \value{COOL@ct} = 1 }{,}{,}%
1038 #4_{\arabic{COOL@ct}}}%
1039 }% end for loop
1040 }%
1041 }%
1042 }%
1043 % else
1044 {%
1045 \ifthenelse{ \NOT \boolean{COOL@#1@isint} \AND
1046 \boolean{COOL@#3@islist} }%
1047 {%
1048 \PackageError{cool}{Invalid Argument}%
1049 {'Hypergeometric': '#1'-arg is not int but '#3'-arg is list}
1050 }%
1051 % else
1052 {%
1053 \ifthenelse{ \NOT \boolean{COOL@#1@isint} \AND
1054 \NOT \boolean{COOL@#3@islist} }%
1055 {%
1056 %return
1057 #4_1,\ldots,#4_{#2}%
1058 }%
1059 % else
1060 {%
1061 }%

\Hypergeometric Generalized Hypergeometric function.  ${}_pF_q(a_1, \dots, a_p; b_1, \dots, b_q; x)$ 
\Hypergeometric{0}{0}{-}{-}{x}  ${}_0F_0(; ; x)$ 
\Hypergeometric{0}{1}{-}{b}{x}  ${}_0F_1(; b; x)$ 
\Hypergeometric{1}{1}{a}{b}{x}  ${}_1F_1(a; b; x)$ 
\Hypergeometric{1}{1}{1}{1}{x}  ${}_1F_1(1; 1; x)$ 
\Hypergeometric{3}{5}{a}{b}{x}  ${}_3F_5(a_1, a_2, a_3; b_1, b_2, b_3, b_4, b_5; x)$ 
\Hypergeometric{3}{5}{1,2,3}{1,2,3,4,5}{x}  ${}_3F_5(1, 2, 3; 1, 2, 3, 4, 5; x)$ 
\Hypergeometric{p}{5}{a}{b}{x}  ${}_pF_5(a_1, \dots, a_p; b_1, b_2, b_3, b_4, b_5; x)$ 
\Hypergeometric{p}{3}{a}{1,2,3}{x}  ${}_pF_3(a_1, \dots, a_p; 1, 2, 3; x)$ 
\Hypergeometric{p}{q}{a}{b}{x}  ${}_pF_q(a_1, \dots, a_p; b_1, \dots, b_q; x)$ 
1062 \newcommand{\COOL@notation@HypergeometricParen}{p}
1063 \newcommand{\COOL@notation@HypergeometricSymb}{F}
1064 \newcommand{\Hypergeometric}[6][F]{%
1065 \provideboolean{COOL@p@isint}%
1066 \provideboolean{COOL@q@isint}%

```



```

1067 \provideboolean{COOL@a@islist}%
1068 \provideboolean{COOL@b@islist}%
1069 \isint{#2}%
1070 \ifthenelse{\boolean{COOL@isint}}%
1071 {\setboolean{COOL@p@isint}{true}}%
1072 % Else
1073 {\setboolean{COOL@p@isint}{false}}%
1074 \isint{#3}%
1075 \ifthenelse{\boolean{COOL@isint}}%
1076 {\setboolean{COOL@q@isint}{true}}%
1077 % Else
1078 {\setboolean{COOL@q@isint}{false}}%
1079 \listval{#4}{0}%
1080 \ifthenelse{\value{COOL@listpointer}>1}%
1081 {\setboolean{COOL@a@islist}{true}}%
1082 % Else
1083 {\setboolean{COOL@a@islist}{false}}%
    ensure that the submitted list is the same length as p
1084 \ifthenelse{ \boolean{COOL@p@isint} \AND
1085 \boolean{COOL@a@islist} \AND
1086 \NOT \(\ #2 = \value{COOL@listpointer} \) }%
1087 {%
1088 \PackageError{cool}{‘Hypergeometric’ ‘p’-arg mismatch with ‘a’-arg}{}%
1089 }%
1090 % else
1091 {%
1092 \listval{#5}{0}%
1093 \ifthenelse{\value{COOL@listpointer}>1}%
1094 {\setboolean{COOL@b@islist}{true}}%
1095 % Else
1096 {\setboolean{COOL@b@islist}{false}}%
    ensure that the submitted ‘b’ list is the same length as q
1097 \ifthenelse{ \boolean{COOL@q@isint} \AND
1098 \boolean{COOL@b@islist} \AND
1099 \NOT \(\ #3 = \value{COOL@listpointer} \) }%
1100 {%
1101 \PackageError{cool}{‘Hypergeometric’ ‘q’-arg mismatch with ‘b’-arg}%
1102 {‘b’ list is not the same length as ‘q’}%
1103 }%
1104 % else
1105 {%
1106 % troubleshoot
1107 \ifthenelse{ \boolean{COOL@a@islist} \AND \NOT \boolean{COOL@p@isint} }%
1108 {%
1109 \PackageError{cool}{‘Hypergeometric’ ‘a’-arg mismatch with ‘p’-arg}%
1110 {happens if ‘a’-arg is a list and ‘p’-arg isn’t an integer}%
1111 }%
1112 % else
1113 {%

```

```

1114 \ifthenelse{ \boolean{COOL@b@islist} \AND \NOT \boolean{COOL@q@isint} }{%
1115 {%
1116 \PackageError{cool}{‘Hypergeometric’ ‘b’-arg mismatch with ‘q’-arg}%
1117 {happens if ‘b’-arg is a list and ‘q’-arg isn’t an integer}%
1118 }%
1119 % else
1120 {}%
    First print the  ${}_pF_q$ 
1121 {}_{#2}{\COOL@notation@HypergeometricSymb}_{#3}%
1122 \COOL@decide@paren{Hypergeometric}%
1123 {%
1124 \COOL@Hypergeometric@pq@ab@value{p}{#2}{a}{#4};%
1125 \COOL@Hypergeometric@pq@ab@value{q}{#3}{b}{#5};%
1126 #6%
1127 }%
1128 }

\RegHypergeometric Regularized hypergeometric function  ${}_p\tilde{F}_q(a_1, \dots, a_p; b_1, \dots, b_q; x)$ 
1129 \newcommand{\COOL@notation@RegHypergeometricParen}{p}
1130 \newcommand{\COOL@notation@RegHypergeometricSymb}{\tilde{F}}
1131 \newcommand{\RegHypergeometric}[6][\tilde{F}]{%
1132 \provideboolean{COOL@p@isint}%
1133 \provideboolean{COOL@q@isint}%
1134 \provideboolean{COOL@a@islist}%
1135 \provideboolean{COOL@b@islist}%
1136 \isint{#2}%
1137 \ifthenelse{\boolean{COOL@isint}}{%
1138 {\setboolean{COOL@p@isint}{true}}%
1139 % Else
1140 {\setboolean{COOL@p@isint}{false}}%
1141 \isint{#3}%
1142 \ifthenelse{\boolean{COOL@isint}}{%
1143 {\setboolean{COOL@q@isint}{true}}%
1144 % Else
1145 {\setboolean{COOL@q@isint}{false}}%
1146 \listval{#4}{0}%
1147 \ifthenelse{\value{COOL@listpointer}>1}%
1148 {\setboolean{COOL@a@islist}{true}}%
1149 % Else
1150 {\setboolean{COOL@a@islist}{false}}%
    ensure that the submitted list is the same length as p
1151 \ifthenelse{ \boolean{COOL@p@isint} \AND
1152 \boolean{COOL@a@islist} \AND
1153 \NOT \(\ #2 = \value{COOL@listpointer} \) }{%
1154 {%
1155 \PackageError{cool}%
1156 {‘RegHypergeometric’ ‘p’-arg mismatch with ‘a’-arg}{}%
1157 }%

```

```

1158 % else
1159 {}%
1160 \listval{#5}{0}%
1161 \ifthenelse{\value{COOL@listpointer}>1}%
1162 {\setboolean{COOL@b@islist}{true}}%
1163 % Else
1164 {\setboolean{COOL@b@islist}{false}}%
    ensure that the submitted 'b' list is the same length as q
1165 \ifthenelse{ \boolean{COOL@q@isint} \AND
1166 \boolean{COOL@b@islist} \AND
1167 \NOT \{ #3 = \value{COOL@listpointer} \} }%
1168 {%
1169 \PackageError{cool}%
1170 {'RegHypergeometric' 'q'-arg mismatch with 'b'-arg}%
1171 {'b' list is not the same length as 'q'}%
1172 }%
1173 % else
1174 {}%
1175 % troubleshoot
1176 \ifthenelse{ \boolean{COOL@a@islist} \AND \NOT \boolean{COOL@p@isint} }%
1177 {%
1178 \PackageError{cool}%
1179 {'RegHypergeometric' 'a'-arg mismatch with 'p'-arg}%
1180 {happens if 'a'-arg is a list and 'p'-arg isn't an integer}%
1181 }%
1182 % else
1183 {}%
1184 \ifthenelse{ \boolean{COOL@b@islist} \AND \NOT \boolean{COOL@q@isint} }%
1185 {%
1186 \PackageError{cool}%
1187 {'RegHypergeometric' 'b'-arg mismatch with 'q'-arg}%
1188 {happens if 'b'-arg is a list and 'q'-arg isn't an integer}%
1189 }%
1190 % else
1191 {}%
    First print the  ${}_pF_q$ 
1192 {}_{#2}{\COOL@notation@RegHypergeometricSymb}_{#3}%
1193 \COOL@decide@paren{RegHypergeometric}%
1194 {%
1195 \COOL@Hypergeometric@pq@ab@value{p}{#2}{a}{#4};%
1196 \COOL@Hypergeometric@pq@ab@value{q}{#3}{b}{#5};%
1197 #6%
1198 }%
1199 }

```

\AppellFOne Appell Hypergeometric Function

\AppellFOne{a}{b_1,b_2}{c}{z_1,z_2} $F_1(a; b_1, b_2; c; z_1, z_2)$

1200 \newcommand{\COOL@notation@AppellFOneParen}{p}

```

1201 \newcommand{\AppellF0ne}[4]%
1202 {F_{1}\COOL@decide@paren{AppellF0ne}{#1; #2; #3; #4}}

\HypergeometricU Tricomi confluent hypergeometric function
      \HypergeometricU{a}{b}{z} U(a,b,z)
1203 \newcommand{\COOL@notation@HypergeometricUSymb}{U}
1204 \newcommand{\HypergeometricU}[3]%
1205 {\COOL@notation@HypergeometricUSymb\inp{#1, #2, #3}}

\COOL@MeijerG@anp@value This macro is a decision maker for the \MeijerG macro. Despite the name it is
                        used for both  $p$  and  $q$ . It is called as
                        \COOL@MeijerG@anp@value  $\{a|b\}$   $\{n|m\}$   $\{p|q\}$ 
1206 \newcommand{\COOL@MeijerG@anp@value}[3]{%
1207 \isint{#3}%
1208 \ifthenelse{\boolean{COOL@isint}}{%
1209 {%
1210 \isint{#2}%
1211 \ifthenelse{\boolean{COOL@isint}}{%
1212 {%
1213 \forLoop{1}{#3}{COOL@ct}%
1214 {%
1215 \ifthenelse{\value{COOL@ct}=1}{},{,}%
1216 #1_{\arabic{COOL@ct}}}%
1217 }%
1218 }%
1219 % else
1220 {%
1221 #1_1,\ldots,#1_{#2},#1_{#2+1},\ldots,#1_{#3}%
1222 }%
1223 }%
1224 % else
1225 {%
1226 \isint{#2}%
1227 \ifthenelse{\boolean{COOL@isint}}{%
1228 {%
1229 \forLoop{1}{#2}{COOL@ct}%
1230 {%
1231 \ifthenelse{\value{COOL@ct}=1}{},{,}%
1232 #1_{\arabic{COOL@ct}}}%
1233 }%
1234 \setcounter{COOL@ct}{#2}%
1235 \addtocounter{COOL@ct}{1}%
1236 ,#1_{\arabic{COOL@ct}}, \ldots, #1_{#3}%
1237 }%
1238 % else
1239 {%
1240 #1_1,\ldots,#1_{#2},#1_{#2+1},\ldots,#1_{#3}%
1241 }%
1242 }%

```

1243 }

\MeijerG \MeijerG{a₁,...,a_n}{a_{n+1},...,a_p}{b₁,...,b_m}{b_{m+1},...,b_q}{x}
\MeijerG[⟨a list symbol⟩,⟨b list symbol⟩]{⟨n⟩}{⟨p⟩}{⟨m⟩}{⟨q⟩}{⟨x⟩}
\MeijerG[⟨a list symbol⟩]{⟨n⟩}{⟨p⟩}{b₁,...,b_m}{b_{m+1},...,b_q}{⟨x⟩}
\MeijerG[,⟨b list symbol⟩]{a₁,...,a_n}{a_{n+1},...,a_p}{⟨m⟩}{⟨q⟩}{⟨x⟩}

Meijer G-Function

\MeijerG[a,b]{n}{p}{m}{q}{z} $G_{p,q}^{m,n}\left(z \left| \begin{matrix} a_1, \dots, a_n, a_{n+1}, \dots, a_p \\ b_1, \dots, b_m, b_{m+1}, \dots, b_q \end{matrix} \right. \right)$

Meijer G-Function

\MeijerG{1,2}{3}{a,b}{c,d}{z} $G_{3,4}^{2,2}\left(z \left| \begin{matrix} 1,2,3 \\ a,b,c,d \end{matrix} \right. \right)$

Generalized Meijer G-Function

\MeijerG[a,b]{n}{p}{m}{q}{z,r} $G_{p,q}^{m,n}\left(z, r \left| \begin{matrix} a_1, \dots, a_n, a_{n+1}, \dots, a_p \\ b_1, \dots, b_m, b_{m+1}, \dots, b_q \end{matrix} \right. \right)$

Generalized Meijer G-Function

\MeijerG{1,2}{3}{a,b}{c,d}{z,r} $G_{3,4}^{2,2}\left(z, r \left| \begin{matrix} 1,2,3 \\ a,b,c,d \end{matrix} \right. \right)$

1244 \newcommand{\COOL@notation@MeijerGSymb}{G}
1245 \newcommand{\MeijerG}[6][@, @]{%
1246 \listval{#1}{0}
1247 \ifthenelse{\value{COOL@listpointer}>2 \OR \value{COOL@listpointer}<1}%
1248 {%
1249 \PackageError{cool}{‘MeijerG’ Invalid Optional Argument}%
1250 {Must be a comma separated list of length 1 or 2}%
1251 }%
1252 % else
1253 {%
1254 }%
1255 \COOL@notation@MeijerGSymb%
1256 \ifthenelse{\equal{#1}{@, @}}{%
1257 {%
1258 \listval{#2}{0}% n
1259 \setcounter{COOL@ct}{\value{COOL@listpointer}}%
1260 \listval{#4}{0}% m
1261 \setcounter{COOL@ct@}{\value{COOL@listpointer}}%
1262 ~{\arabic{COOL@ct@},\arabic{COOL@ct}}%
1263 \listval{#3}{0}% p - n
1264 \addtocounter{COOL@ct}{\value{COOL@listpointer}}%
1265 \listval{#5}{0}% q - m
1266 \addtocounter{COOL@ct@}{\value{COOL@listpointer}}%
1267 _{\arabic{COOL@ct},\arabic{COOL@ct@}}%
1268 \mathopen{}\left(%
1269 #6%
1270 \left| %
1271 { {#2,#3} \@@atop {#4,#5} }%
1272 \right)\right.%
1273 }%
1274 % else
1275 {%

```

1276 \listval{#1}{0}%
1277 \ifthenelse{\value{COOL@listpointer}=2}%
1278 {%
1279 \provideboolean{COOL@MeijerG@opt@one@blank}%
1280 \def\COOL@MeijerG@sniffer##1,##2\COOL@MeijerG@sniffer@end{%
1281 \ifthenelse{\equal{##1}{}}%
1282 {%
1283 \setboolean{COOL@MeijerG@opt@one@blank}{true}%
1284 }%
1285 % else
1286 {%
1287 \setboolean{COOL@MeijerG@opt@one@blank}{false}%
1288 }%
1289 }%
1290 \expandafter\COOL@MeijerG@sniffer#1\COOL@MeijerG@sniffer@end\relax%
1291 \ifthenelse{\boolean{COOL@MeijerG@opt@one@blank}}%
1292 {%
    this is \MeijerG[,b]{a_1,\dots,a_n}{a_{n++},\dots,a_p}{m}{q}{x}
1293 \listval{#2}{0}% n
1294 \setcounter{COOL@ct}{\value{COOL@listpointer}}%
1295 \sim{#4,\arabic{COOL@ct}}%
1296 \listval{#3}{0}% p
1297 \addtocounter{COOL@ct}{\value{COOL@listpointer}}%
1298 \sim{\arabic{COOL@ct},#5}%
1299 \mathopen{}\left(
1300 #6%
1301 \left|
1302 {
1303 {#2,#3} \@@atop {\COOL@MeijerG@anp@value{\listval{#1}{2}}{#4}{#5}}
1304 }
1305 \right)\right.%
1306 }%
1307 % else
1308 {
1309 \sim{#4,#2}_{#3,#5}%
1310 \mathopen{}\left(
1311 #6%
1312 \left|
1313 {
1314 {\COOL@MeijerG@anp@value{\listval{#1}{1}}{#2}{#3}}
1315 \@@atop
1316 {\COOL@MeijerG@anp@value{\listval{#1}{2}}{#4}{#5}}
1317 }
1318 \right)\right.%
1319 }%
1320 }%
1321 % else
1322 {

```

```

this is \MeijerG[a]{n}{p}{b_1,...,b_m}{b_{m++},...,a_p}{x}
1323 \listval{#4}{0}% m
1324 \setcounter{COOL@ct}{\value{COOL@listpointer}}%
1325 ^{\arabic{COOL@ct}, #2}%
1326 \listval{#5}{0}% q
1327 \addtocounter{COOL@ct}{\value{COOL@listpointer}}%
1328 _{#3, \arabic{COOL@ct}}}%
1329 \mathopen{}\left(
1330 #6%
1331 \left|
1332 {
1333 {\COOL@MeijerG@anp@value{#1}{#2}{#3}} \@@atop {#4,#5}
1334 }%
1335 \right)\right.%
1336 }%
1337 }%
1338 }%

```

1.3.25 Angular Momentum Functions

\ClebschGordon Clebsch-Gordon Coefficients

```

\ClebschGordon{j_1,m_1}{j_2,m_2}{j,m} \langle j_1,j_2;m_1,m_2|j_1,j_2;j,m\rangle
http://functions.wolfram.com/HypergeometricFunctions/ClebschGordan/
1339 \newcommand{\ClebschGordon}[3]{%
1340 \listval{#1}{0}%
1341 \ifthenelse{\NOT \value{COOL@listpointer}=2}%
1342 {%
1343 \PackageError{cool}{‘ClebschGordon’ Invalid Argument}%
1344 {Must have a comma separated list of length two}%
1345 }%
1346 % else
1347 {%
1348 \listval{#2}{0}%
1349 \ifthenelse{\NOT \value{COOL@listpointer}=2}%
1350 {%
1351 \PackageError{cool}{‘ClebschGordon’ Invalid Argument}%
1352 {Must have a comma separated list of length two}%
1353 }%
1354 % else
1355 {%
1356 \listval{#3}{0}%
1357 \ifthenelse{\NOT \value{COOL@listpointer}=2}%
1358 {%
1359 \PackageError{cool}{‘ClebschGordon’ Invalid Argument}%
1360 {Must have a comma separated list of length two}%
1361 }%
1362 % else
1363 {%
1364 \left<

```

```

1365 \listval{#1}{1},\listval{#2}{1};%
1366 \listval{#1}{2},\listval{#2}{2}%
1367 \left| %
1368 \listval{#1}{1},\listval{#2}{1};%
1369 \listval{#3}{1},\listval{#3}{2}%
1370 \right>\right.%
1371 }

\ThreeJSymbol Wigner 3-j Symbol
\ThreeJSymbol{j_1,m_1}{j_2,m_2}{j_3,m_3} 
$$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}$$

http://functions.wolfram.com/HypergeometricFunctions/ThreeJSymbol/
1372 \newcommand{\ThreeJSymbol}[3]{%
1373 \listval{#1}{0}%
1374 \ifthenelse{\NOT \value{COOL@listpointer}=2}%
1375 {%
1376 \PackageError{cool}{‘ThreeJSymbol’ Invalid Argument}%
1377 {Must have comma separated list of length 2}%
1378 }%
1379 % else
1380 {%
1381 \listval{#2}{0}%
1382 \ifthenelse{\NOT \value{COOL@listpointer}=2}%
1383 {%
1384 \PackageError{cool}{‘ThreeJSymbol’ Invalid Argument}%
1385 {Must have comma separated list of length 2}%
1386 }%
1387 % else
1388 {%
1389 \listval{#3}{0}%
1390 \ifthenelse{\NOT \value{COOL@listpointer}=2}%
1391 {%
1392 \PackageError{cool}{‘ThreeJSymbol’ Invalid Argument}%
1393 {Must have comma separated list of length 2}%
1394 }%
1395 % else
1396 {%
1397 \mathchoice{%
1398 % displaystyle
1399 \inp{\!%
1400 \begin{array}{ccc}%
1401 \listval{#1}{1} & \listval{#2}{1} & \listval{#3}{1} \\ \& \& \\
1402 \listval{#1}{2} & \listval{#2}{2} & \listval{#3}{2} \\
1403 \end{array}%
1404 \!}%
1405 }%
1406 {%
1407 % inline
1408 \inp{\!%
1409 {\listval{#1}{1} \@@atop \listval{#1}{2}}%

```



```

1410 {\listval{#2}{1} \@@atop \listval{#2}{2}}%
1411 {\listval{#3}{1} \@@atop \listval{#3}{2}}%
1412      \!}%
1413  }%
1414   {%
1415 % subscript
1416 \inp{\!%
1417 {\listval{#1}{1} \@@atop \listval{#1}{2}}%
1418 {\listval{#2}{1} \@@atop \listval{#2}{2}}%
1419 {\listval{#3}{1} \@@atop \listval{#3}{2}}%
1420      \!}%
1421  }%
1422   {%
1423 % subsubscript
1424 \inp{\!%
1425 {\listval{#1}{1} \@@atop \listval{#1}{2}}%
1426 {\listval{#2}{1} \@@atop \listval{#2}{2}}%
1427 {\listval{#3}{1} \@@atop \listval{#3}{2}}%
1428      \!}%
1429  }%
1430 }

```

\SixJSymbol Racah 6-j Symbol

```

      \SixJSymbol{j_1,j_2,j_3}{j_4,j_5,j_6} \left\{ \begin{matrix} j_1 & j_2 & j_3 \\ j_4 & j_5 & j_6 \end{matrix} \right\}
      http://functions.wolfram.com/HypergeometricFunctions/SixJSymbol/
1431 \newcommand{\SixJSymbol}[2]{%
1432 \listval{#1}{0}%
1433 \ifthenelse{\NOT \value{COOL@listpointer}=3}%
1434 {%
1435 \PackageError{cool}{‘SixJSymbol’ Invalid Argument}%
1436 {Must have a comma separated list of length 3}%
1437 }%
1438 %else
1439 {%
1440 \listval{#2}{0}%
1441 \ifthenelse{\NOT \value{COOL@listpointer}=3}%
1442 {%
1443 \PackageError{cool}{‘SixJSymbol’ Invalid Argument}%
1444 {Must have a comma separated list of length 3}%
1445 }%
1446 %else
1447 {%
1448 \mathchoice{%
1449 % displaystyle
1450 \inbr{\!%
1451 \begin{array}{ccc}%
1452 \listval{#1}{1} & \& \listval{#1}{2} & \& \listval{#1}{3} \\
1453 \listval{#2}{1} & \& \listval{#2}{2} & \& \listval{#2}{3} \\
1454 \end{array}%

```

```

1455      \!}%
1456    }%
1457    {%
1458    % inline
1459    \inbr{\!%
1460    {\listval{#1}{1} \@@atop \listval{#2}{1}}%
1461    {\listval{#1}{2} \@@atop \listval{#2}{2}}%
1462    {\listval{#1}{3} \@@atop \listval{#2}{3}}%
1463      \!}%
1464    }%
1465    {%
1466    % superscript
1467    \inbr{\!%
1468    {\listval{#1}{1} \@@atop \listval{#2}{1}}%
1469    {\listval{#1}{2} \@@atop \listval{#2}{2}}%
1470    {\listval{#1}{3} \@@atop \listval{#2}{3}}%
1471      \!}%
1472    }%
1473    {%
1474    % supersuperscript
1475    \inbr{\!%
1476    {\listval{#1}{1} \@@atop \listval{#2}{1}}%
1477    {\listval{#1}{2} \@@atop \listval{#2}{2}}%
1478    {\listval{#1}{3} \@@atop \listval{#2}{3}}%
1479      \!}%
1480    }%
1481  }

```

1.3.26 Complete Elliptic Integrals

\EllipticK Complete Elliptic Integral of the First Kind

\EllipticK{x} $K(x)$

```

1482 \newcommand{\COOL@notation@EllipticKParen}{p}
1483 \newcommand{\COOL@notation@EllipticKSymb}{K}
1484 \newcommand{\EllipticK}[1]%
1485 {\COOL@notation@EllipticKSymb\COOL@decide@paren{EllipticK}{#1}}%

```

\EllipticE Complete Elliptic Integral of the Second Kind

\EllipticE{x} $E(x)$

```

1486 \newcommand{\COOL@notation@EllipticEParen}{p}
1487 \newcommand{\COOL@notation@EllipticESymb}{E}
1488 \newcommand{\EllipticE}[1]%
1489 \liststore{#1}{\COOL@EllipticE@arg@}%
1490 \listval{#1}{0}%
1491 \ifthenelse{\value{\COOL@listpointer} = 1}%
1492 {%
1493 \COOL@notation@EllipticESymb\COOL@decide@paren{EllipticE}{#1}}%
1494 }%
1495 % ElseIf

```

```

1496 { \ifthenelse{\value{COOL@listpointer} = 2}%
1497 {%
1498 \COOL@notation@EllipticESymb%
1499 \COOL@decide@paren{EllipticE}%
1500 {\COOL@EllipticE@arg@i \left| \, \COOL@EllipticE@arg@ii \!\!\right.}%
1501 }%
1502 % Else
1503 {%
1504 \PackageError{Invalid Argument}%
1505 {'EllipticE' can only accept a comma separated list of length 1 or 2}%
1506 }%
1507 }%
1508 }

```

`\EllipticPi` Complete Elliptic Integral of the Third Kind

$$\text{\EllipticPi}\{n,m\} \quad \Pi(n|m)$$

```

1509 \newcommand{\COOL@notation@EllipticPiParen}{p}
1510 \newcommand{\COOL@notation@EllipticPiSymb}{\Pi}
1511 \newcommand{\EllipticPi}[1]{%
1512 \liststore{#1}{\COOL@EllipticPi@arg@}%
1513 \listval{#1}{0}%
1514 \ifthenelse{\value{COOL@listpointer} = 2}%
1515 {%
1516 \COOL@notation@EllipticPiSymb%
1517 \COOL@decide@paren{EllipticPi}%
1518 {\COOL@EllipticPi@arg@i \left| \, \COOL@EllipticPi@arg@ii \!\!\right.}%
1519 }%
1520 % ElseIf
1521 { \ifthenelse{\value{COOL@listpointer} = 3}%
1522 {%
1523 \COOL@notation@EllipticPiSymb%
1524 \COOL@decide@paren{EllipticPi}%
1525 { \COOL@EllipticPi@arg@i; \, %
1526 \COOL@EllipticPi@arg@ii \left| \, %
1527 \COOL@EllipticPi@arg@iii \!\!\right. %
1528 }%
1529 }%
1530 % Else
1531 {%
1532 \PackageError{cool}{Invalid Argument}%
1533 {'EllipticPi' can only accept a comma separated list of length 2 or 3}%
1534 }%
1535 }%
1536 }

```

1.3.27 Incomplete Elliptic Integrals

`\EllipticF` Incomplete Elliptic Integral of the First Kind
`\IncEllipticF`

```

\EllipticF{z,m}      F(z|m)
\IncEllipticF{z}{m}  F(z|m)
1537 \newcommand{\COOL@notation@EllipticFParen}{p}
1538 \newcommand{\COOL@notation@EllipticFSymb}{F}
1539 \newcommand{\EllipticF}[1]{%
1540 \liststore{#1}{\COOL@EllipticF@arg@}%
1541 \listval{#1}{0}%
1542 \ifthenelse{ \value{COOL@listpointer} = 2 }{%
1543 {%
1544 \COOL@notation@EllipticFSymb%
1545 \COOL@decide@paren{EllipticF}%
1546 {\COOL@EllipticF@arg@i \left| \, \, \COOL@EllipticF@arg@ii \!\!\right.}%
1547 }%
1548 % Else
1549 {%
1550 \PackageError{cool}{Invalid Argument}%
1551 {'EllipticF' can only accept a comma separated list of length 2}%
1552 }%
1553 }
1554 \newcommand{\IncEllipticF}[2]{\EllipticF{#1,#2}}

\IncEllipticE  Incomplete Elliptic Integral of the Second Kind
\IncEllipticE{z}{m}  E(z|m)
\EllipticE{z,m}      E(z|m)
1555 \newcommand{\IncEllipticE}[2]{\EllipticE{#1,#2}}

\IncEllipticPi  Incomplete Elliptic Integral of the Third Kind
\EllipticPi      \IncEllipticPi{n}{z}{m}  \Pi(n; z|m)
\EllipticPi{n,z,m}  \Pi(n; z|m)
1556 \newcommand{\IncEllipticPi}[3]{\EllipticPi{#1,#2,#3}}

\JacobiZeta  Jacobi Zeta Function
\JacobiZeta{z}{m}  Z(z|m)
1557 \newcommand{\COOL@notation@JacobiZetaParen}{p}
1558 \newcommand{\COOL@notation@JacobiZetaSymb}{Z}
1559 \newcommand{\JacobiZeta}[2]{%
1560 \COOL@notation@JacobiZetaSymb
1561 \COOL@decide@paren{JacobiZeta}{#1 \left| \, \, #2 \right. \!\!\!}%
1562 }

```

1.3.28 Jacobi Theta Functions

```

\EllipticTheta  Jacobi Theta Functions
\JacobiTheta      \JacobiTheta{1}{z}{q}  \vartheta_1(z,q)
                  \JacobiTheta{2}{z}{q}  \vartheta_2(z,q)
                  \JacobiTheta{3}{z}{q}  \vartheta_3(z,q)
                  \JacobiTheta{4}{z}{q}  \vartheta_4(z,q)
1563 \newcommand{\COOL@notation@EllipticThetaParen}{p}

```

```

1564 \newcommand{\EllipticTheta}[3]%
1565 {\vartheta_{#1}\COOL@decide@paren{\EllipticTheta}{#2, #3}}
1566 \newcommand{\JacobiTheta}[3]{\{\EllipticTheta{#1}{#2}{#3}\}}

```

1.3.29 Neville Theta Functions

```

\NevilleThetaC Neville Theta Function, \NevilleThetaC{z}{m},  $\vartheta_c(z|m)$ 
1567 \newcommand{\COOL@notation@NevilleThetaCParen}{p}
1568 \newcommand{\NevilleThetaC}[2]{%
1569 \vartheta_{#1}\COOL@decide@paren{\NevilleThetaC}%
1570 {#1 \left| \, , \, #2 \right.\!\!\!}%
1571 }

```

```

\NevilleThetaD Neville Theta Function, \NevilleThetaD{z}{m},  $\vartheta_d(z|m)$ 
1572 \newcommand{\COOL@notation@NevilleThetaDParen}{p}
1573 \newcommand{\NevilleThetaD}[2]{%
1574 \vartheta_{#1}\COOL@decide@paren{\NevilleThetaD}%
1575 {#1 \left| \, , \, #2 \right.\!\!\!}%
1576 }

```

```

\NevilleThetaN Neville Theta Function, \NevilleThetaN{z}{m},  $\vartheta_n(z|m)$ 
1577 \newcommand{\COOL@notation@NevilleThetaNParen}{p}
1578 \newcommand{\NevilleThetaN}[2]{%
1579 \vartheta_{#1}\COOL@decide@paren{\NevilleThetaN}%
1580 {#1 \left| \, , \, #2 \right.\!\!\!}%
1581 }

```

```

\NevilleThetaS Neville Theta Function, \NevilleThetaS{z}{m},  $\vartheta_s(z|m)$ 
1582 \newcommand{\COOL@notation@NevilleThetaSParen}{p}
1583 \newcommand{\NevilleThetaS}[2]{%
1584 \vartheta_{#1}\COOL@decide@paren{\NevilleThetaS}%
1585 {#1 \left| \, , \, #2 \right.\!\!\!}%
1586 }

```

1.3.30 Weierstrass Functions

```

\WeierstrassP Weierstrass Elliptic Function
\WeiP \WeierstrassP{z}{g_2,g_3}  $\wp(z;g_2,g_3)$ 
\WeiP{z}{g_2,g_3}  $\wp(z;g_2,g_3)$ 
1587 \newcommand{\COOL@notation@WeierstrassPParen}{p}
1588 \newcommand{\WeierstrassP}[2]{%
1589 \liststore{#2}{COOL@WeiP@arg@g}%
1590 \listval{#2}{0}%
1591 \ifthenelse{\NOT \value{COOL@listpointer} = 2}{%
1592 {%
1593 \PackageError{cool}{Invalid Argument}%
1594 {'WeierstrassP' second argument must be%
1595 a comma separated list of length 2}%
1596 }

```

```

1597 % Else
1598 {%
1599 \wp\COOL@decide@paren{WeierstrassP}{#1; #2}
1600 }%
1601 }
1602 \newcommand{\WeiP}[2]{\WeierstrassP{#1}{#2}}

\WeierstrassPInv  Inverse of Weierstrass Elliptic Function
\WeiPInv          Inverse          \WeiPInv{z}{g_2,g_3}       $\wp^{-1}(z; g_2, g_3)$ 
                  Generalized Inverse \WeiPInv{z_1,z_2}{g_2,g_3}  $\wp^{-1}(z_1, z_2; g_2, g_3)$ 

1603 \newcommand{\COOL@notation@WeierstrassPInvParen}{p}
1604 \newcommand{\WeierstrassPInv}[2]{%
1605 \liststore{#1}{COOL@WeiPInv@arg@z}%
1606 \liststore{#1}{COOL@WeiPInv@arg@g}%
1607 \listval{#2}{0}%
1608 \ifthenelse{\NOT \value{COOL@listpointer} = 2}%
1609 {%
1610 \PackageError{cool}{Invalid Argument}%
1611 {'WeierstrassPInv' second argument must be%
1612 a comma separated list of length 2}%
1613 }
1614 % Else
1615 {
1616 \listval{#1}{0}%
1617 \ifthenelse{\value{COOL@listpointer} = 1}%
1618 {%
1619 \wp^{-1}\COOL@decide@paren{WeierstrassPInv}{#1; #2}%
1620 }%
1621 % ElseIf
1622 { \ifthenelse{\value{COOL@listpointer} = 2}%
1623 {%
1624 \wp^{-1}\COOL@decide@paren{WeierstrassPInv}{#1; #2}%
1625 }%
1626 % Else
1627 {%
1628 \PackageError{cool}{Invalid Argument}%
1629 {'WeierstrassPInv' first argument must be%
1630 a comma separate list of length 1 or 2}%
1631 }}%
1632 }%
1633 }
1634 \newcommand{\WeiPInv}[2]{\WeierstrassPInv{#1}{#2}}

\WeierstrassPGenInv  Generalized Inverse of Weierstrass Elliptic Function
\WierstrassPGenInv{z_1}{z_2}{g_1}{g_2}
1635 \newcommand{\WeierstrassPGenInv}[4]{\WeierstrassPInv{#1,#2}{#3,#4}}

\WeierstrassSigma  Wierstrass Sigma Function
\WeiSigma

```

Sigma	$\backslash\text{WeierstrassSigma}\{z\}\{g_2,g_3\}$	$\sigma(z; g_2, g_3)$
	$\backslash\text{WeiSigma}\{z\}\{g_2,g_3\}$	$\sigma(z; g_2, g_3)$
Associated Sigma	$\backslash\text{WeierstrassSigma}\{n,z\}\{g_2,g_3\}$	$\sigma_n(z; g_2, g_3)$
	$\backslash\text{WeiSigma}\{n,z\}\{g_2,g_3\}$	$\sigma_n(z; g_2, g_3)$

```

1636 \newcommand{\WeierstrassSigma}[2]{%
1637 \liststore{#1}{COOL@WeiSigma@arg@z@}%
1638 \liststore{#2}{COOL@WeiSigma@arg@g@}%
1639 \listval{#2}{0}%
1640 \ifthenelse{\NOT \value{COOL@listpointer} = 2}%
1641 {%
1642 \PackageError{cool}{Invalid Argument}%
1643 {'WeierstrassSigma' second argument must be%
1644 a comma separated list of length 2}%
1645 }%
1646 % Else
1647 {%
1648 \listval{#1}{0}%
1649 \ifthenelse{\value{COOL@listpointer} = 1}%
1650 {%
1651 \sigma\inp{#1; #2}%
1652 }%
1653 % ElseIf
1654 { \ifthenelse{\value{COOL@listpointer} = 2}%
1655 {%
1656 \sigma_{\COOL@WeiSigma@arg@z@i}\inp{\COOL@WeiSigma@arg@z@ii; #2}%
1657 }%
1658 % Else
1659 {%
1660 \PackageError{cool}{Invalid Argument}%
1661 {'WeierstrassSigma' first argument must be%
1662 a comma separated list of length 1 or 2}%
1663 }}%
1664 }%
1665 }
1666 \newcommand{\WeiSigma}[2]{\WeierstrassSigma{#1}{#2}}

\AssocWeierstrassSigma Associated Weierstrass Sigma Function
      \AssocWeierstrassSigma{n}{z}{g_2}{g_3}  \sigma_n(z; g_2, g_3)
      \WeiSigma{n,z}{g_2,g_3}                 \sigma_n(z; g_2, g_3)
1667 \newcommand{\AssocWeierstrassSigma}[4]{\WeierstrassSigma{#1,#2}{#3,#4}}

\WeierstrassZeta Weierstrass Zeta Function
  \WeiZeta      \WeierstrassZeta{z}{g_2,g_3}  \zeta(z; g_2, g_3)
               \WeiZeta{z}{g_2,g_3}          \zeta(z; g_2, g_3)
1668 \newcommand{\COOL@notation@WeierstrassZetaParen}{p}%
1669 \newcommand{\WeierstrassZeta}[2]{%
1670 \listval{#2}{0}%
1671 \ifthenelse{\NOT \value{COOL@listpointer} = 2}%
1672 {%

```

```

1673 \PackageError{cool}{Invalid Argument}%
1674 {'WeierstrassZeta' second argument must be%
1675 a comma separated list of length 2}%
1676 }%
1677 % Else
1678 {%
1679 \zeta\COOL@decide@paren{WeierstrassZeta}{#1; #2}%
1680 }%
1681 }
1682 \newcommand{\WeiZeta}[2]{\WeierstrassZeta{#1}{#2}}

\WeierstrassHalfPeriods Weierstrass half-periods
\WeiHalfPeriods \WeierstrassHalfPeriods{g_2,g_3} {\omega_1(g_2,g_3),\omega_3(g_2,g_3)}
\WeiHalfPeriods{g_2,g_3} {\omega_1(g_2,g_3),\omega_3(g_2,g_3)}
1683 \newcommand{\WeierstrassHalfPeriods}[1]{%
1684 \listval{#1}{0}%
1685 \ifthenelse{\NOT \value{COOL@listpointer} = 2}%
1686 {%
1687 \PackageError{cool}{Invalid Argument}%
1688 {'WeierstrassHalfPeriods' can only accept%
1689 a comma separated list of length 2}%
1690 }%
1691 % Else
1692 {%
1693 \{ \omega_1\inp{#1}, \omega_3\inp{#1} \}%
1694 }%
1695 }
1696 \newcommand{\WeiHalfPeriods}[1]{\WeierstrassHalfPeriods{#1}}

\WeierstrassInvariants Weierstrass Invariants
\WeierstrassInvariants{\omega_1,\omega_3} {g_2(\omega_1,\omega_3),g_3(\omega_1,\omega_3)}
\WeiInvars{\omega_1,\omega_3} {g_2(\omega_1,\omega_3),g_3(\omega_1,\omega_3)}
1697 \newcommand{\WeierstrassInvariants}[1]{%
1698 \listval{#1}{0}%
1699 \ifthenelse{\NOT \value{COOL@listpointer} = 2}%
1700 {%
1701 \PackageError{cool}{Invalid Argument}%
1702 {'WeierstrassInvariants' can only accept%
1703 a comma separated list of length 2}%
1704 }%
1705 % Else
1706 {%
1707 \{ g_2\inp{#1}, g_3\inp{#1} \}%
1708 }%
1709 }
1710 \newcommand{\WeiInvars}[1]{\WeierstrassInvariants{#1}}

\COOL@hideOnSF Used to hide inputs or other when style is sf

```



```

        sf    short form
        ff    full form
1711 \newcommand{\COOL@hideOnSF}[2]
1712 {%
1713 \ifthenelse{ \equal{\csname COOL@notation@#1\endcsname}{sf} }{%
1714 }{%
1715 % Else
1716 {#2}%
1717 }

\WeierstrassPHalfPeriodValues    Weierstrass elliptic function values at half-periods
\WeiPHalfPeriodVal              \Style{WeierstrassPHalfPeriodValuesDisplay=sf} (Default)
                                \WeierstrassPHalfPeriodValues{g_2,g_3}
                                \WeiPHalfPeriodVal{g_2,g_3}
                                {e_1,e_2,e_3}

                                \Style{WeierstrassPHalfPeriodValuesDisplay=ff}
                                \WeierstrassPHalfPeriodValues{g_2,g_3}
                                \WeiPHalfPeriodVal{g_2,g_3}
                                {e_1(g_2,g_3),e_2(g_2,g_3),e_3(g_2,g_3)}
1718 \newcommand{\COOL@notation@WeierstrassPHalfPeriodValuesDisplay}{sf}
1719 \newcommand{\WeierstrassPHalfPeriodValues}[1]
1720 {%
1721 \listval{#1}{0}%
1722 \ifthenelse{\NOT \value{COOL@listpointer} = 2}%
1723 {%
1724 \PackageError{cool}{Invalid Argument}%
1725 {'WeierstrassPHalfPeriodValues' can only accept%
1726  a comma separated list of length 2}%
1727 }%
1728 % Else
1729 {%
1730 \{ e_1\COOL@hideOnSF{WeierstrassPHalfPeriodValuesDisplay}{\inp{#1}},%
1731 e_2\COOL@hideOnSF{WeierstrassPHalfPeriodValuesDisplay}{\inp{#1}},%
1732 e_3\COOL@hideOnSF{WeierstrassPHalfPeriodValuesDisplay}{\inp{#1}}%
1733 \}%
1734 }%
1735 }
1736 \newcommand{\WeiPHalfPeriodVal}[1]{\WeierstrassPHalfPeriodValues{#1}}

\WeierstrassZetaHalfPeriodValues    Weierstrass zeta function values at half-periods
\WeiZetaHalfPeriodVal

```

```

\Style{WeierstrassZetaHalfPeriodValuesDisplay=sf} (Default)
\WeierstrassZetaHalfPeriodValues{g_2,g_3}
\WeiZetaHalfPeriodVal{g_2,g_3}
{\eta_1,\eta_2,\eta_3}

\Style{WeierstrassZetaHalfPeriodValuesDisplay=ff}
\WeierstrassZetaHalfPeriodValues{g_2,g_3}
\WeiZetaHalfPeriodVal{g_2,g_3}
{\eta_1(g_2,g_3),\eta_2(g_2,g_3),\eta_3(g_2,g_3)}
1737 \newcommand{\COOL@notation@WeierstrassZetaHalfPeriodValuesDisplay}{sf}
1738 \newcommand{\WeierstrassZetaHalfPeriodValues}[1]
1739 {%
1740 \listval{#1}{0}%
1741 \ifthenelse{\NOT \value{COOL@listpointer} = 2}%
1742 {%
1743 \PackageError{cool}{Invalid Argument}%
1744 {'WeierstrassZetaHalfPeriodValues' can only accept%
1745 a comma separated list of length 2}%
1746 }%
1747 % Else
1748 {%
1749 \{%
1750 \eta_1\COOL@hideOnSF%
1751 {WeierstrassZetaHalfPeriodValuesDisplay}{\inp{#1}},%
1752 \eta_2\COOL@hideOnSF%
1753 {WeierstrassZetaHalfPeriodValuesDisplay}{\inp{#1}},%
1754 \eta_3\COOL@hideOnSF%
1755 {WeierstrassZetaHalfPeriodValuesDisplay}{\inp{#1}}%
1756 \}%
1757 }%
1758 }
1759 \newcommand{\WeiZetaHalfPeriodVal}[1]%
1760 {\WeierstrassZetaHalfPeriodValues{#1}}

```

1.3.31 Jacobi Functions

```

\JacobiAmplitude Amplitude, \JacobiAmplitude{z}{m}, am(z|m)
1761 \newcommand{\COOL@notation@JacobiAmplitudeParen}{p}
1762 \DeclareMathOperator{\JacobiAmplitudeSymb}{am}
1763 \newcommand{\JacobiAmplitude}[2]{%
1764 \JacobiAmplitudeSymb\COOL@decide@paren%
1765 {JacobiAmplitude}{#1 \left| \right. , #2 \right.\!\!}%
1766 }

\JacobiCD Jacobi elliptic function and its inverse
\JacobiCDInv \JacobiCD{z}{m} cd(z|m)
\JacobiCDInv{z}{m} cd^{-1}(z|m)
1767 \newcommand{\COOL@notation@JacobiCDParen}{p}

```

```

1768 \newcommand{\COOL@notation@JacobiCDInvParen}{p}
1769 \DeclareMathOperator{\JacobiCDSymb}{cd}
1770 \newcommand{\JacobiCD}[2]{%
1771 \JacobiCDSymb\COOL@decide@paren%
1772 {JacobiCD}{#1 \left| \, , #2 \right.\!\!}%
1773 }
1774 \newcommand{\JacobiCDInv}[2]{%
1775 \JacobiCDSymb^{-1}\COOL@decide@paren%
1776 {JacobiCDInv}{#1 \left| \, , #2 \right.\!\!}%
1777 }

\JacobiCN Jacobi elliptic function and its inverse
\JacobiCNInv \JacobiCN{z}{m} cn(z|m)
\JacobiCNInv{z}{m} cn^{-1}(z|m)

1778 \newcommand{\COOL@notation@JacobiCNParen}{p}
1779 \newcommand{\COOL@notation@JacobiCNInvParen}{p}
1780 \DeclareMathOperator{\JacobiCNSymb}{cn}
1781 \newcommand{\JacobiCN}[2]{%
1782 \JacobiCNSymb\COOL@decide@paren%
1783 {JacobiCN}{#1 \left| \, , #2 \right.\!\!}%
1784 }
1785 \newcommand{\JacobiCNInv}[2]{%
1786 \JacobiCNSymb^{-1}\COOL@decide@paren%
1787 {JacobiCNInv}{#1 \left| \, , #2 \right.\!\!}%
1788 }

\JacobiCS Jacobi elliptic function and its inverse
\JacobiCSInv \JacobiCS{z}{m} cs(z|m)
\JacobiCSInv{z}{m} cs^{-1}(z|m)

1789 \newcommand{\COOL@notation@JacobiCSParen}{p}
1790 \newcommand{\COOL@notation@JacobiCSInvParen}{p}
1791 \DeclareMathOperator{\JacobiCSSymb}{cs}
1792 \newcommand{\JacobiCS}[2]{%
1793 \JacobiCSSymb\COOL@decide@paren%
1794 {JacobiCS}{#1 \left| \, , #2 \right.\!\!}%
1795 }
1796 \newcommand{\JacobiCSInv}[2]{%
1797 \JacobiCSSymb^{-1}\COOL@decide@paren%
1798 {JacobiCSInv}{#1 \left| \, , #2 \right.\!\!}%
1799 }

\JacobiDC Jacobi elliptic function and its inverse
\JacobiDCInv \JacobiDC{z}{m} dc(z|m)
\JacobiDCInv{z}{m} dc^{-1}(z|m)

1800 \newcommand{\COOL@notation@JacobiDCParen}{p}
1801 \newcommand{\COOL@notation@JacobiDCInvParen}{p}
1802 \DeclareMathOperator{\JacobiDCSymb}{dc}
1803 \newcommand{\JacobiDC}[2]{%
1804 \JacobiDCSymb\COOL@decide@paren%

```

```

1805 {JacobiDC}{#1 \left| \, , #2 \right.\!\!\}%
1806 }
1807 \newcommand{\JacobiDCInv}[2]{%
1808 \JacobiDCSymb^{-1}\COOL@decide@paren%
1809 {JacobiDCInv}{#1 \left| \, , #2 \right.\!\!\}%
1810 }

\JacobiDN Jacobi elliptic function and its inverse
\JacobiDNInv \JacobiDN{z}{m} dn(z|m)
\JacobiDNInv{z}{m} dn^{-1}(z|m)
1811 \newcommand{\COOL@notation@JacobiDNParen}{p}
1812 \newcommand{\COOL@notation@JacobiDNInvParen}{p}
1813 \DeclareMathOperator{\JacobiDNSymb}{dn}
1814 \newcommand{\JacobiDN}[2]{%
1815 \JacobiDNSymb\COOL@decide@paren%
1816 {JacobiDN}{#1 \left| \, , #2 \right.\!\!\}%
1817 }
1818 \newcommand{\JacobiDNInv}[2]{%
1819 \JacobiDNSymb^{-1}\COOL@decide@paren%
1820 {JacobiDNInv}{#1 \left| \, , #2 \right.\!\!\}%
1821 }

\JacobiDS Jacobi elliptic function and its inverse
\JacobiDSInv \JacobiDS{z}{m} ds(z|m)
\JacobiDSInv{z}{m} ds^{-1}(z|m)
1822 \newcommand{\COOL@notation@JacobiDSParen}{p}
1823 \newcommand{\COOL@notation@JacobiDSInvParen}{p}
1824 \DeclareMathOperator{\JacobiDSSymb}{ds}
1825 \newcommand{\JacobiDS}[2]{%
1826 \JacobiDSSymb\COOL@decide@paren%
1827 {JacobiDS}{#1 \left| \, , #2 \right.\!\!\}%
1828 }
1829 \newcommand{\JacobiDSInv}[2]{%
1830 \JacobiDSSymb^{-1}\COOL@decide@paren%
1831 {JacobiDSInv}{#1 \left| \, , #2 \right.\!\!\}%
1832 }

\JacobiNC Jacobi elliptic function and its inverse
\JacobiNCInv \JacobiNC{z}{m} nc(z|m)
\JacobiNCInv{z}{m} nc^{-1}(z|m)
1833 \newcommand{\COOL@notation@JacobiNCParen}{p}
1834 \newcommand{\COOL@notation@JacobiNCInvParen}{p}
1835 \DeclareMathOperator{\JacobiNCSymb}{nc}
1836 \newcommand{\JacobiNC}[2]{%
1837 \JacobiNCSymb\COOL@decide@paren%
1838 {JacobiNC}{#1 \left| \, , #2 \right.\!\!\}%
1839 }
1840 \newcommand{\JacobiNCInv}[2]{%
1841 \JacobiNCSymb^{-1}\COOL@decide@paren%

```

```

1842 {JacobiNCInv}{#1 \left| \, #2 \right.\!\!\}%
1843 }

\JacobiND Jacobi elliptic function and its inverse
\JacobiNDInv \JacobiND{z}{m} nd( $z|m$ )
\JacobiNDInv{z}{m} nd-1( $z|m$ )
1844 \newcommand{\COOL@notation@JacobiNDParen}{p}
1845 \newcommand{\COOL@notation@JacobiNDInvParen}{p}
1846 \DeclareMathOperator{\JacobiNDSymb}{nd}
1847 \newcommand{\JacobiND}[2]{%
1848 \JacobiNDSymb\COOL@decide@paren%
1849 {JacobiND}{#1 \left| \, #2 \right.\!\!\}%
1850 }
1851 \newcommand{\JacobiNDInv}[2]{%
1852 \JacobiNDSymb^{-1}\COOL@decide@paren%
1853 {JacobiNDInv}{#1 \left| \, #2 \right.\!\!\}%
1854 }

\JacobiNS Jacobi elliptic function and its inverse
\JacobiNSInv \JacobiNS{z}{m} ns( $z|m$ )
\JacobiNSInv{z}{m} ns-1( $z|m$ )
1855 \newcommand{\COOL@notation@JacobiNSParen}{p}
1856 \newcommand{\COOL@notation@JacobiNSInvParen}{p}
1857 \DeclareMathOperator{\JacobiNSSymb}{ns}
1858 \newcommand{\JacobiNS}[2]{%
1859 \JacobiNSSymb\COOL@decide@paren%
1860 {JacobiNS}{#1 \left| \, #2 \right.\!\!\}%
1861 }
1862 \newcommand{\JacobiNSInv}[2]{%
1863 \JacobiNSSymb^{-1}\COOL@decide@paren%
1864 {JacobiNSInv}{#1 \left| \, #2 \right.\!\!\}%
1865 }

\JacobiSC Jacobi elliptic function and its inverse
\JacobiSCInv \JacobiSC{z}{m} sc( $z|m$ )
\JacobiSCInv{z}{m} sc-1( $z|m$ )
1866 \newcommand{\COOL@notation@JacobiSCParen}{p}
1867 \newcommand{\COOL@notation@JacobiSCInvParen}{p}
1868 \DeclareMathOperator{\JacobiSCSymb}{sc}
1869 \newcommand{\JacobiSC}[2]{%
1870 \JacobiSCSymb\COOL@decide@paren%
1871 {JacobiSC}{#1 \left| \, #2 \right.\!\!\}%
1872 }
1873 \newcommand{\JacobiSCInv}[2]{%
1874 \JacobiSCSymb^{-1}\COOL@decide@paren%
1875 {JacobiSCInv}{#1 \left| \, #2 \right.\!\!\}%
1876 }

\JacobiSD Jacobi elliptic function and its inverse
\JacobiSDInv

```

```

\JacobiSD{z}{m}      sd(z|m)
\JacobiSDInv{z}{m}   sd^{-1}(z|m)
1877 \newcommand{\COOL@notation@JacobiSDParen}{p}
1878 \newcommand{\COOL@notation@JacobiSDInvParen}{p}
1879 \DeclareMathOperator{\JacobiSDSymb}{sd}
1880 \newcommand{\JacobiSD}[2]{%
1881 \JacobiSDSymb\COOL@decide@paren%
1882 {JacobiSD}{#1 \left| \, , #2 \right.\!\!\}%
1883 }
1884 \newcommand{\JacobiSDInv}[2]{%
1885 \JacobiSDSymb^{-1}\COOL@decide@paren%
1886 {JacobiSDInv}{#1 \left| \, , #2 \right.\!\!\}%
1887 }

```

```

\JacobiSN      Jacobi elliptic function and its inverse
\JacobiSNInv   \JacobiSN{z}{m}      sn(z|m)
               \JacobiSNInv{z}{m}   sn^{-1}(z|m)
1888 \newcommand{\COOL@notation@JacobiSNParen}{p}
1889 \newcommand{\COOL@notation@JacobiSNInvParen}{p}
1890 \DeclareMathOperator{\JacobiSNSymb}{sn}
1891 \newcommand{\JacobiSN}[2]{%
1892 \JacobiSNSymb\COOL@decide@paren%
1893 {JacobiSN}{#1 \left| \, , #2 \right.\!\!\}%
1894 }
1895 \newcommand{\JacobiSNInv}[2]{%
1896 \JacobiSNSymb^{-1}\COOL@decide@paren%
1897 {JacobiSNInv}{#1 \left| \, , #2 \right.\!\!\}%
1898 }

```

1.3.32 Modular Functions

```

\DedekindEta   Dedekind eta modular function, \DedekindEta{z},  $\eta(z)$ 
1899 \newcommand{\COOL@notation@DedekindEtaParen}{p}
1900 \newcommand{\DedekindEta}[1]{\eta\COOL@decide@paren{DedekindEta}{#1}}

\KleinInvariantJ   Klein invariant modular function, \KleinInvariantJ{z},  $J(z)$ 
1901 \newcommand{\COOL@notation@KleinInvariantJParen}{p}
1902 \newcommand{\KleinInvariantJ}[1]{%
1903 {J\COOL@decide@paren{KleinInvariantJ}{#1}}

\ModularLambda   Modular lambda function, \ModularLambda{z},  $\lambda(z)$ 
1904 \newcommand{\COOL@notation@ModularLambdaParen}{p}
1905 \newcommand{\ModularLambda}[1]{%
1906 {\lambda\COOL@decide@paren{ModularLambda}{#1}}

\EllipticNomeQ   Nome and its inverse
\EllipticNomeQInv \EllipticNomeQ{m}       $q(m)$ 
                  \EllipticNomeQInv{m}    $q^{-1}(m)$ 

```

```

1907 \newcommand{\COOL@notation@EllipticNomeQParen}{p}
1908 \newcommand{\COOL@notation@EllipticNomeQInvParen}{p}
1909 \newcommand{\EllipticNomeQ}[1]%
1910 {q\COOL@decide@paren{EllipticNomeQ}{#1}}
1911 \newcommand{\EllipticNomeQInv}[1]%
1912 {q^{-1}\COOL@decide@paren{EllipticNomeQ}{#1}}

```

1.3.33 Arithmetic Geometric Mean

```

\ArithGeoMean Arithmetic Geometric Mean
\AGM \ArithGeoMean{a}{b} agm(a,b)
\AGM{a}{b} agm(a,b)
1913 \newcommand{\COOL@notation@ArithGeoMeanParen}{p}
1914 \DeclareMathOperator{\ArithGeoMeanSymb}{agm}
1915 \newcommand{\ArithGeoMean}[2]%
1916 {\ArithGeoMeanSymb\COOL@decide@paren{ArithGeoMean}{#1, #2}}
1917 \newcommand{\AGM}[2]{\ArithGeoMean{#1}{#2}}

```

1.3.34 Elliptic Exp and Log

```

\EllipticExp Elliptic exponential
\EExp \EllipticExp{z}{a,b} eexp(z;a,b)
\EExp{z}{a,b} eexp(z;a,b)
1918 \newcommand{\COOL@notation@EllipticExpParen}{p}
1919 \DeclareMathOperator{\EllipticExpSymb}{eexp}
1920 \newcommand{\EllipticExp}[2]{%
1921 \liststore{#2}{\COOL@EllipticExp@arg@}
1922 \listval{#2}{0}%
1923 \ifthenelse{\NOT \value{\COOL@listpointer} = 2}%
1924 {%
1925 \PackageError{cool}{Invalid Argument}%
1926 {'EllipticExp' second argument must be
1927 a comma separated list of length 2}%
1928 }%
1929 % Else
1930 {%
1931 \EllipticExpSymb\COOL@decide@paren{EllipticExp}{#1; #2}%
1932 }%
1933 }
1934 \newcommand{\EExp}[2]{\EllipticExp{#1}{#2}}

\EllipticLog Elliptic logarithm
\ELog \EllipticLog{z_1,z_2}{a,b} elog(z_1,z_2;a,b)
\ELog{z_1,z_2}{a,b} elog(z_1,z_2;a,b)
1935 \newcommand{\COOL@notation@EllipticLogParen}{p}
1936 \DeclareMathOperator{\EllipticLogSymb}{elog}
1937 \newcommand{\EllipticLog}[2]{%
1938 \liststore{#1}{\COOL@EllipticLog@arg@z@}%

```

```

1939 \liststore{#2}{COOL@EllipticLog@arg@a@}%
1940 \listval{#1}{0}%
1941 \ifthenelse{\NOT \value{COOL@listpointer} = 2}%
1942 {%
1943 \PackageError{cool}{Invalid Argument}%
1944 {'EllipticLog' first argument must be
1945 a comma separated list of length 2}%
1946 }%
1947 % Else
1948 {%
1949 \listval{#2}{0}%
1950 \ifthenelse{\NOT \value{COOL@listpointer} = 2}%
1951 {%
1952 \PackageError{cool}{Invalid Argument}%
1953 {'EllipticLog' second argument must be
1954 a comma separated list of length 2}%
1955 }%
1956 % Else
1957 {%
1958 \EllipticLogSymb\COOL@decide@paren{EllipticLog}{#1; #2}%
1959 }%
1960 }%
1961 }
1962 \newcommand{\ELog}[2]{\EllipticLog{#1}{#2}}

```

1.3.35 Zeta Functions

\RiemannZeta Riemann Zeta Function

\RiemannZeta{s} $\zeta(s)$
 \Zeta{s} $\zeta(s)$

```
1963 \newcommand{\RiemannZeta}[1]{\Zeta{#1}}
```

\HurwitzZeta Hurwitz Zeta Function

\HurwitzZeta{s}{a} $\zeta(s, a)$
 \Zeta{s,a} $\zeta(s, a)$

```
1964 \newcommand{\HurwitzZeta}[2]{\Zeta{#1,#2}}
```

\Zeta Riemann and Hurwitz Zeta

Riemann Zeta \Zeta{s} $\zeta(s)$
 Hurwitz Zeta \Zeta{s,a} $\zeta(s, a)$

```

1965 \newcommand{\COOL@notation@ZetaParen}{p}
1966 \newcommand{\Zeta}[1]{%
1967 \liststore{#1}{COOL@Zeta@arg@}%
1968 \listval{#1}{0}% get the list length
1969 \ifthenelse{\value{COOL@listpointer} = 2}%
1970 {%
1971 \zeta\COOL@decide@paren{Zeta}{\COOL@Zeta@arg@i,\COOL@Zeta@arg@ii}%
1972 }%

```



```

1973 % else
1974 {%
1975 \ifthenelse{\value{COOL@listpointer} = 1}%
1976 {%
1977 \zeta\COOL@decide@paren{Zeta}{#1}%
1978 }%
1979 % else
1980 {%
1981 \PackageError{cool}{‘Zeta’ Invalid Argument}%
1982 {the Zeta function can only accept%
1983 a comma delimited list of length 1 or 2}
1984 }%
1985 }%
1986 }%

\RiemannSiegelTheta Riemann-Siegel Theta Function, \RiemannSiegelTheta{z},  $\vartheta(z)$ 
1987 \newcommand{\COOL@notation@RiemannSiegelThetaParen}{p}
1988 \newcommand{\RiemannSiegelTheta}[1]%
1989 {\vartheta\COOL@decide@paren{RiemannSiegelTheta}{#1}}

\RiemannSiegelZ Riemann-Siegel Z Function, \RiemannSiegelZ{z},  $Z(z)$ 
1990 \newcommand{\COOL@notation@RiemannSiegelZParen}{p}
1991 \newcommand{\RiemannSiegelZ}[1]%
1992 {Z\COOL@decide@paren{RiemannSiegelZ}{#1}}

\StieltjesGamma Stieltjes Constant, \StieltjesGamma{n},  $\gamma_n$ 
1993 \newcommand{\StieltjesGamma}[1]{\gamma_{#1}}

\LerchPhi Lerch transcendent, \LerchPhi{z}{s}{a},  $\Phi(z, s, a)$ 
1994 \newcommand{\COOL@notation@LerchPhiParen}{p}
1995 \newcommand{\LerchPhi}[3]{\Phi\COOL@decide@paren{LerchPhi}{#1,#2,#3}}

```

1.3.36 Polylogarithms

```

\NielsenPolyLog Nielsen Polylogarithm, \NielsenPolyLog{\nu}{p}{z},  $S_{\nu}^p(z)$ 
1996 \newcommand{\COOL@notation@NielsenPolyLogParen}{p}
1997 \newcommand{%
1998 \NielsenPolyLog}[3]{S_{#1}^{#2}%
1999 \COOL@decide@paren{NielsenPolyLog}{#3}%
2000 }

\PolyLog Polylogarithm
      Nielsen PolyLog \PolyLog{\nu,p,z}  $S_{\nu}^p(z)$ 
      PolyLog \PolyLog{\nu,z}  $Li_{\nu}(z)$ 
2001 \newcommand{\COOL@notation@PolyLogParen}{p}
2002 \DeclareMathOperator{\PolyLogSymb}{Li}
2003 \newcommand{\PolyLog}[1]{%
2004 \liststore{#1}{COOL@PolyLog@arg@}%

```

```

2005 \listval{#1}{0}%
2006 \ifthenelse{\value{COOL@listpointer} = 3}%
2007 {%
2008 \NielsenPolyLog{\COOL@PolyLog@arg@i}%
2009 {\COOL@PolyLog@arg@ii}{\COOL@PolyLog@arg@iii}%
2010 }%
2011 % else
2012 {%
2013 \ifthenelse{ \value{COOL@listpointer} = 2 }%
2014 {%
2015 \PolyLogSymb_{\COOL@PolyLog@arg@i}%
2016 \COOL@decide@paren{PolyLog}{\COOL@PolyLog@arg@ii}%
2017 }%
2018 % else
2019 {%
2020 \PackageError{cool}{‘PolyLog’ Invalid Argument}%
2021 {This function returns either the Polylogarithm or the%
2022 Nielsen Polylogarithm. It therefore only accepts a comma%
2023 delimited list of length two or three (1 or 2 commas)}%
2024 }%
2025 }%
2026 }

\DiLog Dilogarithm (alias for \PolyLog{2,x}); \DiLog{x},  $\text{Li}_2(x)$ 
2027 \newcommand{\DiLog}[1]{\PolyLog{2,#1}}

```

1.3.37 Mathieu Functions

```

\MathieuC Even Mathieu Function, \MathieuC{a}{q}{z},  $\text{Ce}(a, q, z)$ 
2028 \newcommand{\COOL@notation@MathieuCParen}{p}
2029 \DeclareMathOperator{\MathieuCSymb}{Ce}
2030 \newcommand{\MathieuC}[3]%
2031 {\MathieuCSymb\COOL@decide@paren{MathieuC}{#1,#2,#3}}

\MathieuS Odd Mathieu Function, \MathieuS{a}{q}{z},  $\text{Se}(a, q, z)$ 
2032 \newcommand{\COOL@notation@MathieuSParen}{p}
2033 \DeclareMathOperator{\MathieuSSymb}{Se}
2034 \newcommand{\MathieuS}[3]%
2035 {\mathord{\MathieuSSymb}\COOL@decide@paren{MathieuS}{#1,#2,#3}}

```

1.3.38 Mathieu Characteristics

```

\MathieuCharacteristicA Characteristic Value of Even Mathieu Function
\MathieuCharisticA      \MathieuCharacteristicA{r}{q}   $a_r(q)$ 
                        \MathieuCharisticA{r}{q}       $a_r(q)$ 
2036 \newcommand{\COOL@notation@MathieuCharacteristicAParen}{p}
2037 \newcommand{\MathieuCharacteristicA}[2]%
2038 {a_{#1}\COOL@decide@paren{MathieuCharacteristicA}{#2}}
2039 \newcommand{\MathieuCharisticA}[2]{\MathieuCharacteristicA{#1}{#2}}

```

`\MathieuCharacteristicB` Characteristic Value of Even Mathieu Fucntion
`\MathieuCharisticB` $\mathcal{M}_r(q)$
 $\mathcal{M}_r(q)$

```

2040 \newcommand{\COOL@notation@MathieuCharacteristicBParen}{p}
2041 \newcommand{\MathieuCharacteristicB}[2]%
2042 {b_{#1}\COOL@decide@paren{MathieuCharacteristicB}{#2}}
2043 \newcommand{\MathieuCharisticB}[2]{\MathieuCharacteristicB{#1}{#2}}

```

`\MathieuCharacteristicExponent` Characteristic Exponent of a Mathieu Fucntion
`\MathieuCharisticExp` $r(a, q)$
 $r(a, q)$

```

2044 \newcommand{\COOL@notation@MathieuCharacteristicExponentParen}{p}
2045 \newcommand{\MathieuCharacteristicExponent}[2]%
2046 {r\COOL@decide@paren{MathieuCharacteristicExponent}{#1,#2}}
2047 \newcommand{\MathieuCharisticExp}[2]%
2048 {\MathieuCharacteristicExponent{#1}{#2}}

```

1.3.39 Complex variables

`\Abs` Absolute value, `\Abs{z}`, $|z|$

```

2049 \newcommand{\Abs}[1]{\left|#1\right| }

```

`\Arg` Argument, `\Arg{z}`, $\arg(z)$

```

2050 \newcommand{\Arg}[1]{\arg\inp{#1} }

```

`\Conjugate` Complex Conjugate

`\Conj` z^*
 z^*

```

2051 \def\COOL@notation@Conjugate{star}
2052 \newcommand{\COOL@notation@ConjugateParen}{inv}
2053 \newcommand{\Conjugate}[1]{\Conj{#1}}
2054 \newcommand{\Conj}[1]{%
2055 \ifthenelse{\equal{\COOL@notation@Conjugate}{bar}}{%
2056 {%
2057 \bar{#1}%
2058 }%
2059 % ElseIf
2060 { \ifthenelse{\equal{\COOL@notation@Conjugate}{overline}}{%
2061 {%
2062 \overline{#1}%
2063 }%
2064 % ElseIf
2065 { \ifthenelse{\equal{\COOL@notation@Conjugate}{star}}{%
2066 {%
2067 \COOL@decide@paren{Conjugate}{#1}^*%
2068 }%
2069 % Else
2070 {%

```

```

2071 \PackageError{cool}{Invalid Option Sent}%
2072 {'Conjugate' can only be set at 'star', 'bar', or 'overline'}}%
2073 }%
2074 }}%
2075 }

\Real    Real Part, \Real{z},  $\operatorname{Re} z$ 
2076 \newcommand{\COOL@notation@RealParen}{none}
2077 \DeclareMathOperator{\RealSymb}{Re}
2078 \newcommand{\Real}[1]{%
    we put a space if there is no parentheses, or leave it out if there are
2079 \ifthenelse{\equal{\COOL@notation@ImagParen}{none}}{%
2080 {%
2081 \RealSymb{#1}%
2082 }%
2083 % Else
2084 {%
2085 \RealSymb\COOL@decide@paren{Imag}{#1}%
2086 }%
2087 }

\Imag    Imaginary Part, \Imag{z},  $\operatorname{Im} z$ 
2088 \newcommand{\COOL@notation@ImagParen}{none}
2089 \DeclareMathOperator{\ImagSymb}{Im}
2090 \newcommand{\Imag}[1]{%
    we put a space if there is no parentheses, or leave it out if there are
2091 \ifthenelse{\equal{\COOL@notation@ImagParen}{none}}{%
2092 {%
2093 \ImagSymb{#1}%
2094 }%
2095 % Else
2096 {%
2097 \ImagSymb\COOL@decide@paren{Imag}{#1}%
2098 }%
2099 }

\Sign    Sign function, \Sign{x},  $\operatorname{sgn}(x)$ 
2100 \newcommand{\COOL@notation@SignParen}{p}
2101 \newcommand{\Sign}[1]{\operatorname{sgn}\COOL@decide@paren{Sign}{#1}}
```

1.3.40 Number Theory Functions

```

\FactorInteger Prime decomposition, \Factors{n},  $\operatorname{factors}(n)$ 
\Factors
2102 \newcommand{\COOL@notation@FactorIntegerParen}{p}
2103 \DeclareMathOperator{\FactorIntegerSymb}{factors}
2104 \newcommand{\FactorInteger}[1]{%
2105 {\FactorIntegerSymb\COOL@decide@paren{FactorInteger}{#1}}
2106 \newcommand{\Factors}[1]{\FactorInteger{#1}}
```

```

\Divisors Divisors, \Divisors{n}, divisors(n)
2107 \newcommand{\COOL@notation@DivisorsParen}{p}
2108 \DeclareMathOperator{\DivisorsSymb}{divisors}
2109 \newcommand{\Divisors}[1]%
2110 {\mathord{\DivisorsSymb}\COOL@decide@paren{Divisors}{#1}}

\Prime The nth Prime, \Prime{n}, prime(n)
2111 \newcommand{\COOL@notation@PrimeParen}{p}
2112 \DeclareMathOperator{\PrimeSymb}{prime}
2113 \newcommand{\Prime}[1]%
2114 {\mathord{\PrimeSymb}\COOL@decide@paren{Prime}{#1}}

\PrimePi Prime counting function, \PrimePi{x},  $\pi(x)$ 
2115 \newcommand{\COOL@notation@PrimePiParen}{p}
2116 \newcommand{\PrimePi}[1]{\pi\COOL@decide@paren{PrimePi}{#1}}

\DivisorSigma Sum of divisor powers, \DivisorSigma{k}{n},  $\sigma_k(n)$ 
2117 \newcommand{\COOL@notation@DivisorSigmaParen}{p}
2118 \newcommand{\DivisorSigma}[2]%
2119 {\sigma_{#1}\COOL@decide@paren{DivisorSigma}{#2}}

\EulerPhi Euler Totient Function, \EulerPhi{x},  $\varphi(x)$ 
2120 \newcommand{\COOL@notation@EulerPhiParen}{p}
2121 \newcommand{\EulerPhi}[1]{\varphi\COOL@decide@paren{EulerPhi}{#1}}

\MoebiusMu Moebius Function, \MoebiusMu{x},  $\mu(x)$ 
2122 \newcommand{\COOL@notation@MoebiusMuParen}{p}
2123 \newcommand{\MoebiusMu}[1]{\mu\COOL@decide@paren{MoebiusMu}{#1}}

\JacobiSymbol Jacobi Symbol, \JacobiSymbol{n}{m},  $\left(\frac{n}{m}\right)$ 
2124 \newcommand{\JacobiSymbol}[2]{\inp{\frac{#1}{#2}}}

\CarmichaelLambda Carmichael Lambda Function, \CarmichaelLambda{x},  $\lambda(x)$ 
2125 \newcommand{\COOL@notation@CarmichaelLambdaParen}{p}
2126 \newcommand{\CarmichaelLambda}[1]%
2127 {\lambda\COOL@decide@paren{CarmichaelLambda}{#1}}

\DigitCount Count the digits of an integer n for a given base b
\DigitCount{n}{b}

$$\{s_b^{(1)}(n), s_b^{(2)}(n), \dots, s_b^{(b)-1}(n), s_b^{(0)}(n)\}$$

2128 \newcommand{\DigitCount}[2]{%
2129 \isint{#2}%
2130 \ifthenelse{\boolean{COOL@isint}}{%
2131 {%
2132 \{%
2133 \setcounter{COOL@ct@}{#2}%
2134 \addtocounter{COOL@ct@}{-1}%
2135 \forLoop{1}{\arabic{COOL@ct@}}{COOL@ct}%

```

```

2136 {%
2137 s^{\arabic{COOL@ct}}_{#2}\inp{#1},
2138 }%
2139 s^{\inp{0}}_{#2}\inp{#1}%
2140 \}%
2141 }%
2142 % else
2143 {%
2144 \{%
2145 s^{\inp{1}}_{#2}\inp{#1},%
2146 s^{\inp{2}}_{#2}\inp{#1},%
2147 \ldots,%
2148 s^{\inp{#2} - 1}_{#2}\inp{#1},%
2149 s^{\inp{0}}_{#2}\inp{#1}%
2150 \}%
2151 }%
2152 }

```

1.3.41 Generalized Functions

`\DiracDelta` Dirac Delta Function, `\DiracDelta{x}`, $\delta(x)$

```

2153 \newcommand{\COOL@notation@DiracDeltaParen}{p}
2154 \newcommand{\DiracDelta}[1]{\delta\COOL@decide@paren{DiracDelta}{#1}}

```

`\HeavisideStep` Heaviside Step Function

<code>\UnitStep</code>	<code>\HeavisideStep{x}</code>	$\theta(x)$
	<code>\UnitStep{x}</code>	$\theta(x)$

```

2155 \newcommand{\COOL@notation@HeavisideStepParen}{p}
2156 \newcommand{\HeavisideStep}[1]{}
2157 {\theta\COOL@decide@paren{HeavisideStep}{#1}}
2158 \newcommand{\UnitStep}[1]{\HeavisideStep{#1}}

```

1.3.42 Calculus

`\COOL@notation@DDisplayFunc` Both `\D` and `\pderiv` are controlled by these keys.

`\COOL@notation@DShorten` `DDisplayFunc` controls how the function is displayed, it can take the values:

inset	Display as $\frac{df}{dx}$
outset	Display as $\frac{d}{dx}f$

`DShorten` is for multiple derivatives. it can take the values

true	force derivatives to be consolidated, as in $\frac{d^2}{dx^2}f$
false	expand derivatives as in $\frac{d}{dx}\frac{d}{dx}f$

```

2159 \newcounter{COOL@multideriv}
2160 \newcommand{\COOL@notation@DDisplayFunc}{inset}
2161 \newcommand{\COOL@notation@DShorten}{true}

```

`\COOL@derivative` Both `\D` and `\pderiv` have the same basic operation, so a macro is defined that does the internals

```

\COOL@derivative{\derivative power(s)}{\function}{\wrt}{\symbol}

```

$\langle wrt \rangle$ is a comma separated list of length ≥ 1 .
 $\langle symbol \rangle$ is passed by $\backslash D$ or $\backslash pderiv$ and is $\backslash COOL@notation@DSymb$ or ∂ respectively

$\backslash COOL@derivative\{2,3\}\{f\}\{x,y,z\}\{d\}$	$\frac{d^8 f}{dx^2 dy^3 dz^3}$
$\backslash COOL@derivative\{2,3,4,5\}\{f\}\{x,y,z\}\{d\}$	$\frac{d^9 f}{dx^2 dy^3 dz^4}$
$\backslash COOL@derivative\{2,n,1\}\{f\}\{x,y,z\}\{d\}$	$\frac{d^{2+n+1} f}{dx^2 dy^n dz}$
$\backslash COOL@derivative\{2,n\}\{f\}\{x,y,z\}\{d\}$	$\frac{d^{2+n+n} f}{dx^2 dy^n dz^n}$
$\backslash Style\{DDisplayFunc=outset\}$	
$\backslash COOL@derivative\{2,n\}\{f\}\{x,y,z\}\{d\}$	$\frac{d^{2+n+n} f}{dx^2 dy^n dz^n}$
$\backslash Style\{DShorten=false,DDisplayFunc=inset\}$	
$\backslash COOL@derivative\{2,n\}\{f\}\{x,y,z\}\{d\}$	$\frac{d^2}{dx^2} \frac{d^n}{dy^n} \frac{d^n}{dz^n} f$
$\backslash COOL@derivative\{2,3,4,5\}\{f\}\{x,y,z\}\{d\}$	$\frac{d^2}{dx^2} \frac{d^3}{dy^3} \frac{d^4}{dz^4} f$
$\backslash Style\{DShorten=false,DDisplayFunc=outset\}$	
$\backslash COOL@derivative\{2,n\}\{f\}\{x,y,z\}\{d\}$	$\frac{d^2}{dx^2} \frac{d^n}{dy^n} \frac{d^n}{dz^n} f$

2162 $\backslash newcommand\{\backslash COOL@notation@DSymb\}\{d\}$

2163 $\backslash newcommand\{\backslash COOL@derivative\}[4]\{\%$

Get the length of $\langle wrt \rangle$ argument. $\backslash listval\{\#3\}\{0\}$ gives the length of the list since lists begin indexing at 1.

2164 $\backslash listval\{\#3\}\{0\}\%$

2165 $\backslash setcounter\{COOL@listlen\}\{\backslash value\{COOL@listpointer\}\}\%$

Store the $\langle wrt \rangle$ list and get the length of $\langle derivative power(s) \rangle$.

2166 $\backslash liststore\{\#3\}\{COOL@deriv@wrt@0\}\%$

2167 $\backslash listval\{\#1\}\{0\}\%$

2168 $\backslash setcounter\{COOL@ct\}\{\backslash value\{COOL@listpointer\}\}\%$

2169 $\backslash ifthenelse\{\backslash value\{COOL@ct\}>\backslash value\{COOL@listlen\}\}\%$

2170 $\{\backslash setcounter\{COOL@ct\}\{\backslash value\{COOL@listlen\}\}\}\{\}\%$

2171 $\backslash liststore\{\#1\}\{COOL@deriv@powers@0\}\%$

Check to see if all of the powers are integers—if they are, then we may sum them in the usual sense

2172 $\backslash isint\{\backslash COOL@deriv@powers@i\}\%$

2173 $\backslash setcounter\{COOL@multideriv\}\{2\}\%$

2174 $\backslash whiledo\{\backslash boolean\{COOL@isint\}\ \backslash AND$

2175 $\backslash NOT\ \backslash value\{COOL@multideriv\}>\backslash value\{COOL@ct\}\ \}\%$

2176 $\{\%$

2177 $\backslash def\backslash COOL@tempd\%$

2178 $\{\backslash csname\ COOL@deriv@powers@\roman\{COOL@multideriv\}\backslash endcsname\}\%$

2179 $\backslash isint\{\backslash COOL@tempd\}\%$

2180 $\backslash stepcounter\{COOL@multideriv\}\%$

2181 $\}\%$

If the length of $\langle derivative\ power(s) \rangle$ is less than the length of $\langle wrt \rangle$, then we assume that the last value applies to *all* the remaining derivatives.

```

2182 \ifthenelse{ \equal{\COOL@notation@DShorten}{true} \AND
2183 \equal{\COOL@notation@DDisplayFunc}{inset} }{%
2184 {%
2185 \ifthenelse{ \boolean{COOL@isint} }{%
2186 {%
2187 \def\COOL@temp@D@bot{}%
2188 \setcounter{COOL@ct@}{0}%
2189 \forLoop{1}{\value{COOL@ct@}}{COOL@multideriv}%
2190 {%
2191 \edef\COOL@power@temp%
2192 {\csname COOL@deriv@powers@\roman{COOL@multideriv}\endcsname}%
2193 \edef\COOL@wrt@temp%
2194 {\csname COOL@deriv@wrt@\roman{COOL@multideriv}\endcsname}%
2195 \addtocounter{COOL@ct@}{\COOL@power@temp}%
2196 \ifthenelse{ \value{COOL@multideriv}=1 }{}%
2197 {\edef\COOL@temp@D@bot{\COOL@temp@D@bot \,}%}
2198 \ifthenelse{ \equal{\COOL@power@temp}{1} }%
2199 {%
2200 \edef\COOL@temp@D@bot%
2201 {\COOL@temp@D@bot {#4} \COOL@wrt@temp}%
2202 }%
2203 % Else
2204 {%
2205 \edef\COOL@temp@D@bot%
2206 {\COOL@temp@D@bot {#4} \COOL@wrt@temp^{\COOL@power@temp}%
2207 }%
2208 }%

```

we're done with the length of the $\langle derivative\ power(s) \rangle$ argument, and we want to start at it + 1 to add the remainders

```

2209 \ifthenelse{\value{COOL@ct}<\value{COOL@listlen}}{%
2210 {%
2211 \edef\COOL@power@temp%
2212 {\csname COOL@deriv@powers@\roman{COOL@ct}\endcsname}%
2213 \stepcounter{COOL@ct}%
2214 \forLoop{\value{COOL@ct}}{\value{COOL@listlen}}{COOL@multideriv}%
2215 {%
2216 \edef\COOL@wrt@temp%
2217 {\csname COOL@deriv@wrt@\roman{COOL@multideriv}\endcsname}%
2218 \addtocounter{COOL@ct@}{\COOL@power@temp}%
2219 \ifthenelse{ \value{COOL@multideriv}=1 }{}%
2220 {\edef\COOL@temp@D@bot{\COOL@temp@D@bot \,}%}
2221 \ifthenelse{ \equal{\COOL@power@temp}{1} }%
2222 {%
2223 \edef\COOL@temp@D@bot%
2224 {\COOL@temp@D@bot {#4} \COOL@wrt@temp}%
2225 }%
2226 % Else

```



```

2227 {%
2228 \edef\COOL@temp@D@bot%
2229 {\COOL@temp@D@bot {#4} \COOL@wrt@temp^\COOL@power@temp}%
2230 }%
2231 }%
2232 }%
2233 % Else
2234 {%
2235 \ifthenelse{\value{COOL@ct@}=1}%
2236 {%
2237 \frac{{#4} #2}{\COOL@temp@D@bot}%
2238 }%
2239 % Else
2240 {%
2241 \frac{{#4}^{\arabic{COOL@ct@}} #2}{\COOL@temp@D@bot}%
2242 }%
2243 }%
2244 % Else
2245 {%
    Powers are not all Integers
2246 \edef\COOL@temp@D@bot{}%
2247 \def\COOL@temp@D@top@power{}%
2248 \forLoop{1}{\value{COOL@ct@}}{\COOL@multideriv}%
2249 {%
2250 \edef\COOL@power@temp%
2251 {\csname COOL@deriv@powers@\roman{COOL@multideriv}\endcsname}%
2252 \edef\COOL@wrt@temp%
2253 {\csname COOL@deriv@wrt@\roman{COOL@multideriv}\endcsname}%
2254 \ifthenelse{ \value{COOL@multideriv} = 1}%
2255 {%
2256 \edef\COOL@temp@D@top@power{\COOL@power@temp}%
2257 }%
2258 % Else
2259 {%
2260 \edef\COOL@temp@D@top@power%
2261 {\COOL@temp@D@top@power + \COOL@power@temp}%
2262 \edef\COOL@temp@D@bot{\COOL@temp@D@bot \,}%
2263 }%
2264 \ifthenelse{ \equal{\COOL@power@temp}{1} }%
2265 {%
2266 \edef\COOL@temp@D@bot%
2267 {\COOL@temp@D@bot {#4} \COOL@wrt@temp}%
2268 }%
2269 % Else
2270 {%
2271 \edef\COOL@temp@D@bot%
2272 {\COOL@temp@D@bot {#4} \COOL@wrt@temp^\COOL@power@temp}%
2273 }%
2274 }%

```

we're done with the length of the $\langle derivative\ power(s) \rangle$ argument, and we want to start at it + 1 to add the remainders

```

2275 \ifthenelse{\value{COOL@ct}<\value{COOL@listlen}}{%
2276 {%
2277 \edef\COOL@power@temp%
2278 {\csname COOL@deriv@powers@\roman{COOL@ct}\endcsname}%
2279 \stepcounter{COOL@ct}%
2280 \forLoop{\value{COOL@ct}}{\value{COOL@listlen}}{COOL@multideriv}%
2281 {%
2282 \edef\COOL@wrt@temp%
2283 {\csname COOL@deriv@wrt@\roman{COOL@multideriv}\endcsname}%
2284 \ifthenelse{ \value{COOL@multideriv} = 1}%
2285 {%
2286 \edef\COOL@temp@D@top@power{\COOL@power@temp}%
2287 }%
2288 % Else
2289 {%
2290 \edef\COOL@temp@D@top@power%
2291 {\COOL@temp@D@top@power + \COOL@power@temp}%
2292 \edef\COOL@temp@D@bot{\COOL@temp@D@bot \,}%
2293 }%
2294 \ifthenelse{ \equal{\COOL@power@temp}{1} }%
2295 {%
2296 \edef\COOL@temp@D@bot%
2297 {\COOL@temp@D@bot {#4} \COOL@wrt@temp}%
2298 }%
2299 % Else
2300 {%
2301 \edef\COOL@temp@D@bot%
2302 {\COOL@temp@D@bot {#4} \COOL@wrt@temp^{\COOL@power@temp}%
2303 }%
2304 }%
2305 }%
2306 % Else
2307 {}%
2308 \frac{{#4}^{\COOL@temp@D@top@power} #2}{\COOL@temp@D@bot}%
2309 }%
2310 }%

2311 % Else If
2312 { \ifthenelse{ \equal{\COOL@notation@DShorten}{true} \AND
2313 \equal{\COOL@notation@DDisplayFunc}{outset} }%
2314 {%
2315 \ifthenelse{ \boolean{COOL@isint} }%
2316 {%
2317 \def\COOL@temp@D@bot{}%
2318 \setcounter{COOL@ct}{0}%
2319 \forLoop{1}{\value{COOL@ct}}{COOL@multideriv}%
2320 {%
2321 \edef\COOL@power@temp%

```

```

2322 {\csname COOL@deriv@powers@\roman{COOL@multideriv}\endcsname}%
2323 \edef\COOL@wrt@temp%
2324 {\csname COOL@deriv@wrt@\roman{COOL@multideriv}\endcsname}%
2325 \addtocounter{COOL@ct@}{\COOL@power@temp}%
2326 \ifthenelse{ \value{COOL@multideriv}=1 }{}%
2327 {\edef\COOL@temp@D@bot{\COOL@temp@D@bot \,}}%
2328 \ifthenelse{ \equal{\COOL@power@temp}{1} }%
2329 {%
2330 \edef\COOL@temp@D@bot%
2331 {\COOL@temp@D@bot {#4} \COOL@wrt@temp}%
2332 }%
2333 % Else
2334 {%
2335 \edef\COOL@temp@D@bot%
2336 {\COOL@temp@D@bot {#4} \COOL@wrt@temp~\COOL@power@temp}%
2337 }%
2338 }%

```

we're done with the length of the $\langle derivative\ power(s) \rangle$ argument, and we want to start at it + 1 to add the remainders

```

2339 \ifthenelse{\value{COOL@ct}<\value{COOL@listlen}}%
2340 {%
2341 \edef\COOL@power@temp%
2342 {\csname COOL@deriv@powers@\roman{COOL@ct}\endcsname}%
2343 \stepcounter{COOL@ct}%
2344 \forLoop{\value{COOL@ct}}{\value{COOL@listlen}}{\COOL@multideriv}%
2345 {%
2346 \edef\COOL@wrt@temp%
2347 {\csname COOL@deriv@wrt@\roman{COOL@multideriv}\endcsname}%
2348 \addtocounter{COOL@ct@}{\COOL@power@temp}%
2349 \ifthenelse{ \value{COOL@multideriv}=1 }{}%
2350 {\edef\COOL@temp@D@bot{\COOL@temp@D@bot \,}}%
2351 \ifthenelse{ \equal{\COOL@power@temp}{1} }%
2352 {%
2353 \edef\COOL@temp@D@bot%
2354 {\COOL@temp@D@bot {#4} \COOL@wrt@temp}%
2355 }%
2356 % Else
2357 {%
2358 \edef\COOL@temp@D@bot%
2359 {\COOL@temp@D@bot {#4} \COOL@wrt@temp~\COOL@power@temp}%
2360 }%
2361 }%
2362 }%
2363 % Else
2364 {%
2365 \ifthenelse{\value{COOL@ct@}=1}%
2366 {%
2367 \frac{#4}{\COOL@temp@D@bot} #2%
2368 }%

```

```

2369 % Else
2370 {%
2371 \frac{{#4}^{\arabic{COOL@ct}}}{COOL@temp@D@bot} #2%
2372 }%
2373 }%
2374 % Else
2375 {%
    Powers are not all Integers
2376 \edef\COOL@temp@D@bot{%
2377 \def\COOL@temp@D@top@power{%
2378 \forLoop{1}{\value{COOL@ct}}{COOL@multideriv}%
2379 {%
2380 \edef\COOL@power@temp%
2381 {\csname COOL@deriv@powers@\roman{COOL@multideriv}\endcsname}%
2382 \edef\COOL@wrt@temp%
2383 {\csname COOL@deriv@wrt@\roman{COOL@multideriv}\endcsname}%
2384 \ifthenelse{ \value{COOL@multideriv} = 1}%
2385 {%
2386 \edef\COOL@temp@D@top@power{\COOL@power@temp}%
2387 }%
2388 % Else
2389 {%
2390 \edef\COOL@temp@D@top@power%
2391 {\COOL@temp@D@top@power + \COOL@power@temp}%
2392 \edef\COOL@temp@D@bot{\COOL@temp@D@bot \,}%
2393 }%
2394 \ifthenelse{ \equal{\COOL@power@temp}{1} }%
2395 {%
2396 \edef\COOL@temp@D@bot%
2397 {\COOL@temp@D@bot {#4} \COOL@wrt@temp}%
2398 }%
2399 % Else
2400 {%
2401 \edef\COOL@temp@D@bot%
2402 {\COOL@temp@D@bot {#4} \COOL@wrt@temp^{\COOL@power@temp}%
2403 }%
2404 }%

    we're done with the length of the  $\langle derivative\ power(s) \rangle$  argument, and we want to
    start at it + 1 to add the remainders
2405 \ifthenelse{\value{COOL@ct}<\value{COOL@listlen}}%
2406 {%
2407 \edef\COOL@power@temp%
2408 {\csname COOL@deriv@powers@\roman{COOL@ct}\endcsname}%
2409 \stepcounter{COOL@ct}%
2410 \forLoop{\value{COOL@ct}}{\value{COOL@listlen}}{COOL@multideriv}%
2411 {%
2412 \edef\COOL@wrt@temp%
2413 {\csname COOL@deriv@wrt@\roman{COOL@multideriv}\endcsname}%
2414 \ifthenelse{ \value{COOL@multideriv} = 1}%

```

```

2415 {%
2416 \edef\COOL@temp@D@top@power{\COOL@power@temp}%
2417 }%
2418 % Else
2419 {%
2420 \edef\COOL@temp@D@top@power%
2421 {\COOL@temp@D@top@power + \COOL@power@temp}%
2422 \edef\COOL@temp@D@bot{\COOL@temp@D@bot \,}%
2423 }%
2424 \ifthenelse{ \equal{\COOL@power@temp}{1} }%
2425 {%
2426 \edef\COOL@temp@D@bot%
2427 {\COOL@temp@D@bot {#4} \COOL@wrt@temp}%
2428 }%
2429 % Else
2430 {%
2431 \edef\COOL@temp@D@bot%
2432 {\COOL@temp@D@bot {#4} \COOL@wrt@temp^{\COOL@power@temp}%
2433 }%
2434 }%
2435 }%
2436 % Else
2437 {}%
2438 \frac{{#4}^{\COOL@temp@D@top@power} }{\COOL@temp@D@bot} #2%
2439 }%
2440 }%

2441 % Else If
2442 { \ifthenelse{ \equal{\COOL@notation@DShorten}{false} \AND
2443 \equal{\COOL@notation@DDisplayFunc}{inset} }%
2444 {%
2445 \def\COOL@temp@D@result{%
2446 \def\COOL@temp@D@bot{%
2447 \def\COOL@temp@D@top{%
2448 \setcounter{COOL@ct@}{\value{COOL@ct}}%
2449 \addtocounter{COOL@ct@}{-1}
2450 \forLoop{1}{\value{COOL@ct@}}{\COOL@multideriv}%
2451 {%
2452 \edef\COOL@power@temp%
2453 {\csname COOL@deriv@powers@\roman{COOL@multideriv}\endcsname}%
2454 \edef\COOL@wrt@temp%
2455 {\csname COOL@deriv@wrt@\roman{COOL@multideriv}\endcsname}%
2456 \ifthenelse{ \equal{\COOL@power@temp}{1} }%
2457 {%
2458 \edef\COOL@temp@D@top{#4}%
2459 \edef\COOL@temp@D@bot{{#4} \COOL@wrt@temp}%
2460 }%
2461 % Else
2462 {%
2463 \edef\COOL@temp@D@top{{#4}^{\COOL@power@temp}%

```

```

2464 \edef\COOL@temp@D@bot{{#4} \COOL@wrt@temp^\COOL@power@temp}%
2465 }%
2466 \edef\COOL@temp@D@result%
2467 {\COOL@temp@D@result \frac{\COOL@temp@D@top}{\COOL@temp@D@bot}}%
2468 }%

```

we're done with the length of the $\langle derivative\ power(s) \rangle$ argument, and we want to start at it + 1 to add the remainders

```

2469 \ifthenelse{\value{COOL@ct}<\value{COOL@listlen}}%
2470 {%

```

Must pick up the one for \value{COOL@ct}

```

2471 \edef\COOL@power@temp%
2472 {\csname COOL@deriv@powers@\roman{COOL@ct}\endcsname}%
2473 \edef\COOL@wrt@temp%
2474 {\csname COOL@deriv@wrt@\roman{COOL@ct}\endcsname}%
2475 \ifthenelse{ \equal{\COOL@power@temp}{1} }%
2476 {%
2477 \edef\COOL@temp@D@top{{#4}^\COOL@power@temp}%
2478 \edef\COOL@temp@D@bot{{#4} \COOL@wrt@temp}%
2479 }%
2480 % Else
2481 {%
2482 \edef\COOL@temp@D@top{{#4}^\COOL@power@temp}%
2483 \edef\COOL@temp@D@bot{{#4} \COOL@wrt@temp^\COOL@power@temp}%
2484 }%
2485 \edef\COOL@temp@D@result%
2486 {\COOL@temp@D@result \frac{\COOL@temp@D@top}{\COOL@temp@D@bot}}%

```

Now add the ones beyond

```

2487 \stepcounter{COOL@ct}%
2488 \setcounter{COOL@ct@}{\value{COOL@listlen}}%
2489 \addtocounter{COOL@ct@}{-1}%
2490 \forLoop{\value{COOL@ct}}{\value{COOL@ct@}}{COOL@multideriv}%
2491 {%
2492 \ifthenelse{ \equal{\COOL@power@temp}{1} }%
2493 {%
2494 \edef\COOL@temp@D@top{{#4}^\COOL@power@temp}%
2495 \edef\COOL@temp@D@bot{{#4} \COOL@wrt@temp}%
2496 }%
2497 % Else
2498 {%
2499 \edef\COOL@temp@D@top{{#4}^\COOL@power@temp}%
2500 \edef\COOL@temp@D@bot{{#4} \COOL@wrt@temp^\COOL@power@temp}%
2501 }%
2502 \edef\COOL@temp@D@result%
2503 {\COOL@temp@D@result \frac{\COOL@temp@D@top}{\COOL@temp@D@bot}}%
2504 }%

```

Must pick up the one for \value{COOL@listlen}

```

2505 \edef\COOL@wrt@temp%

```

```

2506 {\csname COOL@deriv@wrt@\roman{COOL@listlen}\endcsname}%
2507 \ifthenelse{ \equal{\COOL@power@temp}{1} }%
2508 {%
2509 \edef\COOL@temp@D@top{#4}%
2510 \edef\COOL@temp@D@bot{{#4} \COOL@wrt@temp}%
2511 }%
2512 % Else
2513 {%
2514 \edef\COOL@temp@D@top{{#4}^\COOL@power@temp}%
2515 \edef\COOL@temp@D@bot{{#4} \COOL@wrt@temp^\COOL@power@temp}%
2516 }%
2517 \edef\COOL@temp@D@result%
2518 {\COOL@temp@D@result \frac{\COOL@temp@D@top #2}{\COOL@temp@D@bot}}%
2519 }%
2520 % Else
2521 {%
    Must pick up the one for \value{COOL@ct}
2522 \edef\COOL@power@temp%
2523 {\csname COOL@deriv@powers@\roman{COOL@ct}\endcsname}%
2524 \edef\COOL@wrt@temp%
2525 {\csname COOL@deriv@wrt@\roman{COOL@ct}\endcsname}%
2526 \ifthenelse{ \equal{\COOL@power@temp}{1} }%
2527 {%
2528 \edef\COOL@temp@D@top{#4}%
2529 \edef\COOL@temp@D@bot{{#4} \COOL@wrt@temp}%
2530 }%
2531 % Else
2532 {%
2533 \edef\COOL@temp@D@top{{#4}^\COOL@power@temp}%
2534 \edef\COOL@temp@D@bot{{#4} \COOL@wrt@temp^\COOL@power@temp}%
2535 }%
2536 \edef\COOL@temp@D@result%
2537 {\COOL@temp@D@result \frac{\COOL@temp@D@top #2}{\COOL@temp@D@bot}}%
2538 }%
2539 \COOL@temp@D@result%
2540 }%

2541 % Else If
2542 { \ifthenelse{ \equal{\COOL@notation@DShorten}{false} \AND
2543 \equal{\COOL@notation@DDisplayFunc}{outset} }%
2544 {%
2545 \def\COOL@temp@D@result{}%
2546 \def\COOL@temp@D@bot{}%
2547 \def\COOL@temp@D@top{}%
2548 \forLoop{1}{\value{COOL@ct}}{\COOL@multideriv}%
2549 {%
2550 \edef\COOL@power@temp%
2551 {\csname COOL@deriv@powers@\roman{COOL@multideriv}\endcsname}%
2552 \edef\COOL@wrt@temp%
2553 {\csname COOL@deriv@wrt@\roman{COOL@multideriv}\endcsname}%

```

```

2554 \ifthenelse{ \equal{\COOL@power@temp}{1} }%
2555 {%
2556 \edef\COOL@temp@D@top{#4}%
2557 \edef\COOL@temp@D@bot{#4} \COOL@wrt@temp}%
2558 }%
2559 % Else
2560 {%
2561 \edef\COOL@temp@D@top{#4}^{\COOL@power@temp}%
2562 \edef\COOL@temp@D@bot{#4} \COOL@wrt@temp^{\COOL@power@temp}%
2563 }%
2564 \edef\COOL@temp@D@result%
2565 {\COOL@temp@D@result \frac{\COOL@temp@D@top}{\COOL@temp@D@bot}}%
2566 }%

    we're done with the length of the  $\langle derivative\ power(s) \rangle$  argument, and we want to
    start at it + 1 to add the remainders
2567 \ifthenelse{\value{COOL@ct}<\value{COOL@listlen}}%
2568 {%
2569 \edef\COOL@power@temp%
2570 {\csname COOL@deriv@powers@\roman{COOL@ct}\endcsname}%
2571 \stepcounter{COOL@ct}%
2572 \forLoop{\value{COOL@ct}}{\value{COOL@listlen}}{\COOL@multideriv}%
2573 {%
2574 \edef\COOL@wrt@temp%
2575 {\csname COOL@deriv@wrt@\roman{COOL@multideriv}\endcsname}%
2576 \ifthenelse{ \equal{\COOL@power@temp}{1} }%
2577 {%
2578 \edef\COOL@temp@D@top{#4}%
2579 \edef\COOL@temp@D@bot{#4} \COOL@wrt@temp}%
2580 }%
2581 % Else
2582 {%
2583 \edef\COOL@temp@D@top{#4}^{\COOL@power@temp}%
2584 \edef\COOL@temp@D@bot{#4} \COOL@wrt@temp^{\COOL@power@temp}%
2585 }%
2586 \edef\COOL@temp@D@result%
2587 {\COOL@temp@D@result \frac{\COOL@temp@D@top}{\COOL@temp@D@bot}}%
2588 }%
2589 }%
2590 % Else
2591 {%
2592 }%
2593 \COOL@temp@D@result #2
2594 }%

2595 % Else
2596 {%
2597 \PackageError{cool}{Invalid Option Sent}%
2598 {DShorten can only be 'true' or 'false';%
2599 DDisplayFunc can only be 'inset' or 'outset'}}%
2600 }%

```



```

2601 } } } %
2602 }

\D Derivatives
\pderiv \Style{DSymb={\mathrm d}}
\{f\}{x}  $\frac{d}{dx} f$ 
\{n\}{f\}{x}  $\frac{d^n}{dx^n} f$ 
\{f\}{x,y,z}  $\frac{d}{dx} \frac{d}{dy} \frac{d}{dz} f$ 
\{1,2,1\}{f\}{x,y,z}  $\frac{d}{dx} \frac{d^2}{dy^2} \frac{d}{dz} f$ 
\pderiv{f}{x}  $\frac{\partial}{\partial x} f$ 
\pderiv[n]{f}{x}  $\frac{\partial^n}{\partial x^n} f$ 
\pderiv{f}{x,y,z}  $\frac{\partial}{\partial x} \frac{\partial}{\partial y} \frac{\partial}{\partial z} f$ 
\pderiv[1,2,1]{f}{x,y,z}  $\frac{\partial}{\partial x} \frac{\partial^2}{\partial y^2} \frac{\partial}{\partial z} f$ 

2603 \newcommand{\D}[3][1]{\COOL@derivative{#1}{#2}{#3}{\COOL@notation@DSymb}}
2604 \newcommand{\pderiv}[3][1]{\COOL@derivative{#1}{#2}{#3}{\partial}}

\Integrate Integrate
\Int This has the option IntegrateDisplayFunc which can be inset or outset:
\Style{IntegrateDisplayFunc=inset} (Default)
\Integrate{f}{x}  $\int f dx$ 
\Int{f}{x}  $\int f dx$ 
\Integrate{f}{x,A}  $\int_A f dx$ 
\Int{f}{x,A}  $\int_A f dx$ 
\Integrate{f}{x,a,b}  $\int_a^b f dx$ 
\Int{f}{x,a,b}  $\int_a^b f dx$ 

\Style{IntegrateDisplayFunc=outset,IntegrateDifferentialDSymb=\text{d}}
\Integrate{f}{x}  $\int dx f$ 
\Int{f}{x}  $\int dx f$ 
\Integrate{f}{x,A}  $\int_A dx f$ 
\Int{f}{x,A}  $\int_A dx f$ 
\Integrate{f}{x,a,b}  $\int_a^b dx f$ 
\Int{f}{x,a,b}  $\int_a^b dx f$ 

2605 \newcommand{\COOL@notation@IntegrateDisplayFunc}{inset}
2606 \newcommand{\COOL@notation@IntegrateDifferentialDSymb}{d}
2607 \newcommand{\Integrate}[2]{%
2608 \listval{#2}{0}%

```

```

record the length of the list
2609 \setcounter{COOL@listlen}{\value{COOL@listpointer}}%
2610 \ifthenelse{ \value{COOL@listlen} = 1 }{%
2611 {%
2612 \ifthenelse{\equal{COOL@notation@IntegrateDisplayFunc}{outset}}%
2613 {%
2614 \int \! \COOL@notation@IntegrateDifferentialDSymb{}#2 \, #1%
2615 }%
2616 % ElseIf
2617 { \ifthenelse{\equal{COOL@notation@IntegrateDisplayFunc}{inset}}%
2618 {%
2619 \int #1 \, \COOL@notation@IntegrateDifferentialDSymb{}#2%
2620 }%
2621 % Else
2622 {%
2623 \PackageError{cool}{Invalid Option Sent}%
2624 {'DisplayFunc' can only be 'inset' or 'outset'}%
2625 }}%
2626 }%
2627 % ElseIf
2628 { \ifthenelse{ \value{COOL@listlen} = 2 }%
2629 {%
2630 \ifthenelse{\equal{COOL@notation@IntegrateDisplayFunc}{outset}}%
2631 {%
2632 \int_{\listval{#2}{2}} \!
2633 \COOL@notation@IntegrateDifferentialDSymb{}{\listval{#2}{1}} \, #1%
2634 }%
2635 % ElseIf
2636 { \ifthenelse{\equal{COOL@notation@IntegrateDisplayFunc}{inset}}%
2637 {%
2638 \int_{\listval{#2}{2}} #1 \,
2639 \COOL@notation@IntegrateDifferentialDSymb{}{\listval{#2}{1}}%
2640 }%
2641 % Else
2642 {%
2643 \PackageError{cool}{Invalid Option Sent}%
2644 {'DisplayFunc' can only be 'inset' or 'outset'}%
2645 }}%
2646 }%
2647 % ElseIf
2648 { \ifthenelse{ \value{COOL@listlen} = 3 }%
2649 {%
2650 \ifthenelse{\equal{COOL@notation@IntegrateDisplayFunc}{outset}}%
2651 {%
2652 \int_{\listval{#2}{2}}^{\listval{#2}{3}} \!
2653 \COOL@notation@IntegrateDifferentialDSymb{}{\listval{#2}{1}} \, #1%
2654 }%
2655 % ElseIf
2656 { \ifthenelse{\equal{COOL@notation@IntegrateDisplayFunc}{inset}}%

```

```

2657 {%
2658 \int_{\listval{#2}{2}}^{\listval{#2}{3}} #1 \,
2659 \COOL@notation@IntegrateDifferentialDSymb{\listval{#2}{1}}%
2660 }%
2661 % Else
2662 {%
2663 \PackageError{cool}{Invalid Option Sent}%
2664 {'DisplayFunc' can only be 'inset' or 'outset'}%
2665 }%
2666 }%
2667 % Else
2668 {%
2669 \PackageError{cool}{'Integrate' has invalid parameter list}%
2670 {this happens when the second argument has more than two commas}%
2671 }%
2672 }%
2673 \newcommand{\Int}[2]{\Integrate{#1}{#2}}

\Sum Sum

$$\sum_n a_n$$


$$\sum_{n=1}^N a_n$$

2674 \newcommand{\Sum}[2]{%
2675 \listval{#2}{0}%
    record the length of the list
2676 \setcounter{COOL@listlen}{\value{COOL@listpointer}}
2677 \ifthenelse{ \value{COOL@listlen} = 1 }{%
2678 {%
2679 \sum_{#2} #1%
2680 }%
2681 % else
2682 {%
2683 \ifthenelse{ \value{COOL@listlen} = 3 }{%
2684 {%
2685 \sum_{ \listval{#2}{1} = \listval{#2}{2} }^{\listval{#2}{3}} #1
2686 }%
2687 % else
2688 {%
2689 \PackageError{cool}{Invalid list length for 'Sum'}%
2690 {can only have none or two commas for second argument}%
2691 }%
2692 }%
2693 }

\Prod Product

$$\prod_n a_n$$


$$\prod_{n=1}^N a_n$$

2694 \newcommand{\Prod}[2]{%
2695 \listval{#2}{0}%

```

```

record the length of the list
2696 \setcounter{COOL@listlen}{\value{COOL@listpointer}}
2697 \ifthenelse{ \value{COOL@listlen} = 1 }{%
2698 {%
2699 \prod_{#2} #1%
2700 }%
2701 % else
2702 {%
2703 \ifthenelse{ \value{COOL@listlen} = 3 }{%
2704 {%
2705 \prod_{ \listval{#2}{1} = \listval{#2}{2} }{ \listval{#2}{3} } #1
2706 }%
2707 % else
2708 {%
2709 \PackageError{cool}{Invalid list length for ‘Prod’}%
2710 {can only have none or two commas for second argument}%
2711 }%
2712 }%
2713 }

```

1.3.43 Vector Operators

\DotProduct The dot product, $\text{\DotProduct}\{\text{\vec{A}}\}\{\text{\vec{B}}\}$, $\vec{A} \cdot \vec{B}$
2714 \newcommand{\DotProduct}[2]{#1 \cdot #2}

\Cross The cross product, $\text{\Cross}\{\text{\vec{A}}\}\{\text{\vec{B}}\}$, $\vec{A} \times \vec{B}$
2715 \newcommand{\Cross}[2]{#1 \times #2}

\Div the divergence, $\text{\Div}\{\text{\vec{A}}\}$, $\nabla \cdot \vec{A}$
2716 \newcommand{\Div}[1]{\nabla \cdot #1}

\Grad The gradient, $\text{\Grad}\{f\}$, ∇f
2717 \newcommand{\Grad}[1]{\nabla #1}

\Curl The curl, $\text{\Curl}\{\text{\vec{A}}\}$, $\nabla \times \vec{A}$
2718 \newcommand{\Curl}[1]{\nabla \times #1}

\Laplacian The laplacian, $\text{\Laplacian}\{f\}$, $\nabla^2 f$
2719 \newcommand{\Laplacian}[1]{\nabla^2 #1}

1.3.44 Matrix Operations

\Transpose Transpose of a matrix, $\text{\Transpose}\{A\}$, A^T
2720 \newcommand{\COOL@notation@TransposeParen}{inv}
2721 \newcommand{\Transpose}[1]{\COOL@decide@paren{Transpose}\{#1\}^T }

\Dagger Conjugate Transpose of a matrix, $\text{\Dagger}\{A\}$, A^\dagger
2722 \newcommand{\COOL@notation@DaggerParen}{inv}
2723 \newcommand{\Dagger}[1]{\COOL@decide@paren{Dagger}\{#1\}^\dagger }

```

\Det    determinant of a matrix
        \Style{DetDisplay=det} (Default)
        \Det{A}                  det A
        \Style{DetDisplay=barenc}
        \Det{A}                  |A|
2724 \newcommand{\COOL@notation@DetParen}{none}
2725 \newcommand{\COOL@notation@DetDisplay}{det}
2726 \newcommand{\Det}[1]{%
2727 \ifthenelse{\equal{\COOL@notation@DetDisplay}{det}}%
2728 {%
2729 \det\COOL@decide@paren{Det}{#1}%
2730 }%
2731 % ElseIf
2732 { \ifthenelse{\equal{\COOL@notation@DetDisplay}{barenc}}%
2733 {%
2734 \left|#1\right|}%
2735 }%
2736 % Else
2737 {%
2738 \PackageError{cool}{Invalid Option Sent}%
2739 {'DetDisplay' can only be 'det' or 'barenc'}%
2740 }%
2741 }

```

```

\Tr    Trace of a Matrix, \Tr{A}, Tr A
2742 \newcommand{\COOL@notation@TrParen}{none}
2743 \newcommand{\Tr}[2][ ]{%
2744 \ifthenelse{\equal{#1}{}}%
2745 {%
2746 \operatorname{Tr}\COOL@decide@paren{Tr}{#2}%
2747 }%
2748 % Else
2749 {%
2750 \operatorname{Tr}_{#1}\COOL@decide@paren{Tr}{#2}%
2751 }%
2752 }

```

1.3.45 Matricies

```

\IdentityMatrix The Identity Matrix
        \IdentityMatrix           $\mathbb{I}$ 
        \IdentityMatrix[2]        $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ 
2753 \newcommand{\COOL@notation@IdentityMatrixParen}{p}
2754 \newcounter{COOL@row}%
2755 \newcounter{COOL@col}%
2756 \newcommand{\COOL@notation@IdentityMatrixSymb}{\mathbbm{1}}
2757 \newcommand{\IdentityMatrix}[1][0]{%

```

```

2758 \isint{#1}%
2759 \ifthenelse{\boolean{COOL@isint}}{%
2760 {%
2761 \ifthenelse{ #1=0 }{%
2762 {%
2763 \COOL@notation@IdentityMatrixSymb%
2764 }%
2765 % Else
2766 {%
2767 \setcounter{COOL@ct}{\value{MaxMatrixCols}}%
2768 \setcounter{MaxMatrixCols}{#1}%
2769 \ifthenelse{\equal{\COOL@notation@IdentityMatrixParen}{p}}{%
2770 {%
2771 \begin{pmatrix}%
2772 }%
2773 % ElseIf
2774 { \ifthenelse{\equal{\COOL@notation@IdentityMatrixParen}{b}}%
2775 {%
2776 \begin{bmatrix}%
2777 }%
2778 % ElseIf
2779 { \ifthenelse{\equal{\COOL@notation@IdentityMatrixParen}{br}}%
2780 {%
2781 \begin{Bmatrix}%
2782 }%
2783 % Else
2784 {%
2785 \begin{matrix}%
2786 }}%
2787 \forLoop{1}{#1}{COOL@row}%
2788 {%
2789 \ifthenelse{\NOT \value{COOL@row} = 1}{\{\}\}%
2790 \forLoop{1}{#1}{COOL@col}%
2791 {%
2792 \ifthenelse{ \NOT \value{COOL@col} = 1 }{\&}{}%
2793 \ifthenelse{ \value{COOL@row}=\value{COOL@col} }{1}{0}%
2794 }%
2795 }%
2796 \ifthenelse{\equal{\COOL@notation@IdentityMatrixParen}{p}}%
2797 {%
2798 \end{pmatrix}%
2799 }%
2800 % ElseIf
2801 { \ifthenelse{\equal{\COOL@notation@IdentityMatrixParen}{b}}%
2802 {%
2803 \end{bmatrix}%
2804 }%
2805 % ElseIf
2806 { \ifthenelse{\equal{\COOL@notation@IdentityMatrixParen}{br}}%
2807 {%

```

```

2808 \end{Bmatrix}%
2809 }%
2810 % Else
2811 {%
2812 \end{matrix}%
2813 }}%
2814 \setcounter{MaxMatrixCols}{\value{COOL@ct}}%
2815 }%
2816 }%
2817 % Else
2818 {%
2819 \COOL@notation@IdentityMatrixSymb%
2820 }%
2821 }%

```

Change History

v0	General: pre-Initial version [tentative edition] 1	v1.3	General: Redefined the <code>in*</code> commands to have a <code>mathopen</code> before the <code>left</code> . Added <code>IntegrateDifferentialDSymb</code> and <code>DSymb</code> options for <code>Integrate</code> and <code>D</code> . Added <code>IdentityMatrixSymb</code> for <code>IdentityMatrix</code> and changed the default to display a double-struck 1. Added <code>ESymb</code> , <code>ISymb</code> , <code>PISymb</code> , and <code>EulerGammaSymb</code> for fundamental constants 1
v1.0	General: Initial Release 1		
v1.1	General: Added <code>listlenstore</code> to package to allow storing of the list length 1		
v1.2	General: Split off the list, string, and <code>forloop</code> parts to separate packages 1		

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols		<code>\AiryBi</code> <u>334</u>
<code>\{</code> 7, 1693, 1707, 1730, 1749, 2132, 2144		<code>\AiryBiSymb</code> 335, 336
<code>\}</code> 7, 1693, 1707, 1733, 1756, 2140, 2150		<code>\AppellF0ne</code> <u>1200</u>
		<code>\ArcCos</code> <u>222</u>
	A	<code>\ArcCosh</code> <u>280</u>
<code>\Abs</code> <u>2049</u>		<code>\ArcCot</code> <u>262</u>
<code>\AGM</code> <u>1913</u>		<code>\ArcCoth</code> <u>288</u>
<code>\AiryAi</code> 331		<code>\ArcCsc</code> <u>258</u>
<code>\AiryAiSymb</code> 332, 333		<code>\ArcCsch</code> <u>284</u>

\ArcSec	260	\COOL@derivative ...	2162, 2603, 2604
\ArcSech	286	\COOL@hideOnSF	
\ArcSin	204	1711, 1730–1732, 1750, 1752, 1754	
\ArcSinh	278	\COOL@Hypergeometric@pq@ab@value	998
\ArcTan	240	\COOL@keyeater	98, 101, 106
\ArcTanh	282	\COOL@keyend	98, 101, 106
\Arg	2050	\COOL@keystop	98, 100, 102
\ArithGeoMean	1913	\COOL@MeijerG@anp@value	
\ArithGeoMeanSymb	1914, 1916	1206, 1303, 1314, 1316, 1333	
\AssocLegendreP	670	\COOL@notation@ArcTrig	203
\AssocLegendreQ	671	\COOL@notation@DDisplayFunc ...	2159
\AssocWeierstrassSigma	1667	\COOL@notation@DShorten	2159
B			
\Bernoulli	435, 691	\COOL@notation@DSymb	2162, 2603
\BernoulliB	691	\COOL@notation@ESymb	112, 113
\BesselI	323	\COOL@notation@EulerGammaSymb ..	
\BesselJ	315	117, 118	
\BesselK	327	\COOL@notation@IdentityMatrixSymb	
\BesselY	319	2756, 2763, 2819	
\Beta	831, 858, 859, 919	\COOL@notation@IntegrateDifferentialDSymb	
\BetaReg	860	2606,	
\BetaRegInv	887	2614, 2619, 2633, 2639, 2653, 2659	
\BetaRegularized	860	\COOL@notation@ISymb	110, 111
\Binomial	694	\COOL@notation@PISymb	114, 115
C			
\CarmichaelLambda	2125	\COOL@tempd	2177, 2179
\Catalan	119	\Cos	193
\Ceiling	342	\Cosh	266
\ChebyshevT	656	\CoshInt	995
\ChebyshevU	660	\CoshIntSymb	996, 997
\CIfty	132	\CosInt	989
\ClebschGordon	1339	\CosIntSymb	990, 991
\ComplexInfinity	132	\Cot	201
\Conj	2051	\Coth	276
\Conjugate	2051	\Cross	2715
\COOL@arg@temp	470, 477, 478,	\Csc	197
482, 483, 488, 489, 492, 499,		\Csch	270
506, 507, 511, 512, 517, 518, 521		\Curl	2718
\COOL@Bernoulli@arg@i	446	\CyclotomicC	686
\COOL@Bernoulli@arg@ii	447	D	
\COOL@decide@indicies	68	\D	2603
\COOL@decide@indicies@placement .		\Dagger	2722
69, 71, 79, 85		\DblFactorial	693
\COOL@decide@paren	13	\DedekindEta	1899
\COOL@decide@paren@no@type ...	13–15	\Det	2724
\COOL@decide@paren@type		\DiGamma	808
17, 21, 23, 29, 35, 41, 47, 53		\DigitCount	2128
\COOL@deriv@powers@i	2172	\DiLog	2027
		\DiracDelta	2153
		\DirectedInfinity	130
		\DirInfty	130

\DiscreteDelta	462	\fPart	348
\Div	2716	\fPartSymb	349, 350
\Divisors	2107	\FractionalPart	348
\DivisorSigma	2117	\FresnelC	975
\DivisorsSymb	2108, 2110	\FresnelS	973
\DotProduct	2714		
E			
\E	112, 142, 146	\GammaFunc	732, 756, 757
\EExp	1918	\GammaReg	758
\EGCD	385	\GammaRegInv	780
\EllipticE	1486, 1555	\GammaRegularized	758, 802
\EllipticExp	1918	\GCD	383
\EllipticExpSymb	1919, 1931	\GegenbauerC	672
\EllipticF	1537	\GenErf	962
\EllipticK	1482	\GenErfInv	962
\EllipticLog	1935	\GenIncBeta	859
\EllipticLogSymb	1936, 1958	\GenIncGamma	757
\EllipticNomeQ	1907	\GenRegIncBeta	919
\EllipticNomeQInv	1907	\GenRegIncBetaInv	920
\EllipticPi	1509, 1556	\GenRegIncGamma	802
\EllipticTheta	1563	\GenRegIncGammaInv	803
\ELog	1935	\Glaisher	120
\Erf	921, 962	\GoldenRatio	116
\Erfc	964	\Grad	2717
\ErfcInv	968		
\ErfcSymb	965, 966, 969	H	
\Erfi	970	\HarmNum	813
\ErfInv	942, 963	\HeavisideStep	2155
\ErfiSymb	971, 972	\HermiteH	524
\ErfSymb	922, 928, 933, 948, 953	\HurwitzZeta	1964
\Euler	414, 690	\Hypergeometric	1062
\EulerE	690	\HypergeometricU	1203
\EulerGamma	117		
\EulerPhi	2120	I	
\Exp	134	\I	110
\ExpIntE	977	\IdentityMatrix	2753
\ExpIntEi	979	\Imag	2088
\ExpIntEiSymb	980, 982	\ImagSymb	2089, 2093, 2097
\ExtendedGCD	385	\IncBeta	858
\ExtendedGCDSymb	386, 388	\IncEllipticE	1555
		\IncEllipticF	1537
F		\IncEllipticPi	1556
\Factorial	692	\IncGamma	756
\FactorInteger	2102	\Indeterminant	123
\FactorIntegerSymb	2103, 2105	\Infinity	122
\Factors	2102	\Int	2605
\Fibonacci	393, 689	\IntegerPart	344
\FibonacciF	689	\Integrate	2605
\Floor	341	\InverseBetaRegularized	887, 920
		\InverseGammaRegularized	780, 803

<code>\iPart</code>	<u>344</u>	<code>\KleinInvariantJ</code>	<u>1901</u>
<code>\iPartSymb</code>	345, 346	<code>\KroneckerDelta</code>	<u>465</u>

J

<code>\JacobiAmplitude</code>	<u>1761</u>
<code>\JacobiAmplitudeSymb</code>	1762, 1764
<code>\JacobiCD</code>	<u>1767</u>
<code>\JacobiCDInv</code>	<u>1767</u>
<code>\JacobiCDSymb</code>	1769, 1771, 1775
<code>\JacobiCN</code>	<u>1778</u>
<code>\JacobiCNInv</code>	<u>1778</u>
<code>\JacobiCNSymb</code>	1780, 1782, 1786
<code>\JacobiCS</code>	<u>1789</u>
<code>\JacobiCSInv</code>	<u>1789</u>
<code>\JacobiCSSymb</code>	1791, 1793, 1797
<code>\JacobiDC</code>	<u>1800</u>
<code>\JacobiDCInv</code>	<u>1800</u>
<code>\JacobiDCSymb</code>	1802, 1804, 1808
<code>\JacobiDN</code>	<u>1811</u>
<code>\JacobiDNInv</code>	<u>1811</u>
<code>\JacobiDNSymb</code>	1813, 1815, 1819
<code>\JacobiDS</code>	<u>1822</u>
<code>\JacobiDSInv</code>	<u>1822</u>
<code>\JacobiDSSymb</code>	1824, 1826, 1830
<code>\JacobiNC</code>	<u>1833</u>
<code>\JacobiNCInv</code>	<u>1833</u>
<code>\JacobiNCSymb</code>	1835, 1837, 1841
<code>\JacobiND</code>	<u>1844</u>
<code>\JacobiNDInv</code>	1851
<code>\JacobiNDInv</code>	<u>1844</u>
<code>\JacobiNDSymb</code>	1846, 1848, 1852
<code>\JacobiNS</code>	<u>1855</u>
<code>\JacobiNSInv</code>	<u>1855</u>
<code>\JacobiNSSymb</code>	1857, 1859, 1863
<code>\JacobiP</code>	<u>664</u>
<code>\JacobiSC</code>	<u>1866</u>
<code>\JacobiSCInv</code>	<u>1866</u>
<code>\JacobiSCSymb</code>	1868, 1870, 1874
<code>\JacobiSD</code>	<u>1877</u>
<code>\JacobiSDInv</code>	<u>1877</u>
<code>\JacobiSDSymb</code>	1879, 1881, 1885
<code>\JacobiSN</code>	<u>1888</u>
<code>\JacobiSNInv</code>	<u>1888</u>
<code>\JacobiSNSymb</code>	1890, 1892, 1896
<code>\JacobiSymbol</code>	<u>2124</u>
<code>\JacobiTheta</code>	<u>1563</u>
<code>\JacobiZeta</code>	<u>1557</u>

K

<code>\Khinchin</code>	<u>121</u>
------------------------------	------------

L

<code>\LambertW</code>	<u>290</u>
<code>\Laplacian</code>	<u>2719</u>
<code>\LaugerreL</code>	<u>528</u>
<code>\LCM</code>	<u>390</u>
<code>\LCMSymb</code>	391, 392
<code>\LegendreP</code>	<u>550</u> , 670
<code>\LegendreQ</code>	<u>603</u> , 671
<code>\LerchPhi</code>	<u>1994</u>
<code>\LeviCivita</code>	<u>494</u>
<code>\Log</code>	<u>139</u>
<code>\LogGamma</code>	<u>805</u>
<code>\LogGammaSymb</code>	806, 807
<code>\LogInt</code>	<u>983</u>
<code>\LogIntSymb</code>	984, 985

M

<code>\mathbbm</code>	2756
<code>\MathieuC</code>	<u>2028</u>
<code>\MathieuCharacteristicA</code>	<u>2036</u>
<code>\MathieuCharacteristicB</code>	<u>2040</u>
<code>\MathieuCharacteristicExponent</code>	<u>2044</u>
<code>\MathieuCharisticA</code>	<u>2036</u>
<code>\MathieuCharisticB</code>	<u>2040</u>
<code>\MathieuCharisticExp</code>	<u>2044</u>
<code>\MathieuCSymb</code>	2029, 2031
<code>\MathieuS</code>	<u>2032</u>
<code>\MathieuSSymb</code>	2033, 2035
<code>\mathopen</code>	3, 5, 7, 9, 11, 1268, 1299, 1310, 1329
<code>\Max</code>	<u>311</u>
<code>\MeijerG</code>	<u>1244</u>
<code>\Min</code>	<u>313</u>
<code>\Mod</code>	<u>352</u>
<code>\ModularLambda</code>	<u>1904</u>
<code>\MoebiusMu</code>	<u>2122</u>
<code>\Multinomial</code>	<u>695</u>

N

<code>\NevilleThetaC</code>	<u>1567</u>
<code>\NevilleThetaD</code>	<u>1572</u>
<code>\NevilleThetaN</code>	<u>1577</u>
<code>\NevilleThetaS</code>	<u>1582</u>
<code>\NielsenPolyLog</code>	<u>1996</u> , 2008

P

<code>\PartitionsP</code>	<u>458</u>
---------------------------------	------------

