

```

1 #####
2 #
3 #           Package sui quaternioni
4 #           -----
5 #
6 # Autore: Claudio Marsan
7 # Ultima revisione: 16 marzo 2003
8 # Versione: Maple V Release 6.02 for Windows 2000
9 #
10 #####
11
12 # Base per i quaternioni
13 alias( q1 = quat(1, 0, 0, 0),
14        qj = quat(0, 1, 0, 0),
15        qk = quat(0, 0, 1, 0),
16        qk = quat(0, 0, 0, 1));
17
18 # "Pretty print" per i quaternioni
19 `print/quat` := proc(a0, a1, a2, a3)
20   a0 + a1*'i' + a2*'j' + a3*'k';
21 end;
22
23 # Estrazione dei quattro coefficienti dalla struttura di dati di un quaternione
24 quaternion[qparts] := proc(a, a0, a1, a2, a3)
25   a0 := op(1, a);
26   a1 := op(2, a);
27   a2 := op(3, a);
28   a3 := op(4, a);
29   NULL;
30 end;
31
32 # Traccia di un quaternione
33 quaternion[qtrace] := proc(a)
34   local a0, a1, a2, a3;
35   quaternion[qparts](a, a0, a1, a2, a3);
36   2*a0;
37 end;
38
39 # Norma di un quaternione
40 quaternion[qnorm] := proc(a)
41   local a0, a1, a2, a3;
42   quaternion[qparts](a, a0, a1, a2, a3);
43   a0^2 + a1^2 + a2^2 + a3^2;
44 end;
45
46 # Valore assoluto di un quaternione
47 quaternion[qabs] := proc(a)
48   local a0, a1, a2, a3;
49   quaternion[qparts](a, a0, a1, a2, a3);
50   sqrt(a0^2 + a1^2 + a2^2 + a3^2);
51 end;
52
53 # Coniugato di un quaternione
54 quaternion[qconj] := proc(a)
55   local a0, a1, a2, a3;
56   quaternion[qparts](a, a0, a1, a2, a3);
57   quat(a0, -a1, -a2, -a3);
58 end;
59
60 # Prodotto di uno scalare con un quaternione:
61 quaternion[qscal] := proc(k, a)
62   local a0, a1, a2, a3;
63   quaternion[qparts](a, a0, a1, a2, a3);
64   quat(k*a0, k*a1, k*a2, k*a3);
65 end;
66
67 # Somma di quaternioni
68 quaternion[qadd] := proc()
69   local n, lista, nuovalista, i, a0, a1, a2, a3, b0, b1, b2, b3 ;
70   n := nargs;
71   lista := [args];
72   if n = 2
73     then
74       quaternion[qparts](lista[1], a0, a1, a2, a3);
75       quaternion[qparts](lista[2], b0, b1, b2, b3);
76       quat(a0 + b0, a1 + b1, a2 + b2, a3 + b3);
77     else
78       quaternion[qparts](lista[1], a0, a1, a2, a3);
79       quaternion[qparts](lista[2], b0, b1, b2, b3);
80       nuovalista := quat(a0 + b0, a1 + b1, a2 + b2, a3 + b3),
81         seq(lista[i], i = 3..n);
82       quaternion[qadd](nuovalista);
83     fi;
84 end;
85
86 # Differenza di due quaternioni
87 quaternion[qsub] := proc(a, b)
88   local a0, a1, a2, a3, b0, b1, b2, b3;
89   quaternion[qparts](a, a0, a1, a2, a3);
90   quaternion[qparts](b, b0, b1, b2, b3);
91   quat(a0 - b0, a1 - b1, a2 - b2, a3 - b3);
92 end;
93
94 # Prodotto di quaternioni

```

```

95 quaternion[qmul] := proc()
96   local n, lista, nuovalista, i, a0, a1, a2, a3, b0, b1, b2, b3 ;
97   n := nargs;
98   lista := [args];
99   if n = 2
100    then
101      quaternion[qparts](lista[1], a0, a1, a2, a3);
102      quaternion[qparts](lista[2], b0, b1, b2, b3);
103      quat(a0*b0 - a1*b1 - a2*b2 - a3*b3,
104           a0*b1 + a1*b0 + a2*b3 - a3*b2,
105           a0*b2 - a1*b3 + a2*b0 + a3*b1,
106           a0*b3 + a1*b2 - a2*b1 + a3*b0);
107    else
108      quaternion[qparts](lista[1], a0, a1, a2, a3);
109      quaternion[qparts](lista[2], b0, b1, b2, b3);
110      nuovalista := quat(a0*b0 - a1*b1 - a2*b2 - a3*b3,
111                       a0*b1 + a1*b0 + a2*b3 - a3*b2,
112                       a0*b2 - a1*b3 + a2*b0 + a3*b1,
113                       a0*b3 + a1*b2 - a2*b1 + a3*b0),
114                seq(lista[i], i = 3..n);
115      quaternion[qmul](nuovalista);
116   fi;
117 end:
118
119 # Inversa moltiplicativa di un quaternione non nullo
120 quaternion[qinv] := proc(a)
121   local a0, a1, a2, a3, q;
122   quaternion[qparts](a, a0, a1, a2, a3);
123   q := quaternion[qnorm](a);
124   if q = 0
125     then ERROR(`Il quaternione 0 non e' invertibile!`);
126     else quat(a0/q, -a1/q, -a2/q, -a3/q);
127   fi;
128 end:
129
130 # Testo di aiuto per il package dei quaternioni
131 `help/text/quaternion` := TEXT(
132 ``,
133 `HELP FOR: quaternion`,
134 ``,
135 `      Introduzione al package dei quaternioni`,
136 `-----`,
137 ``,
138 `Il seguente package permette di operare con i quaternioni classici, ossia`,
139 `della forma  $a_0 + a_1i + a_2j + a_3k$  ( $a_0, a_1, a_2, a_3$  sono scalari reali) e`,
140 `con le usuali regole per i prodotti tra  $i, j, k$ .`,
141 `Un quaternione e' definito mediante una struttura di dati della forma`,
142 `quat( $a_0, a_1, a_2, a_3$ ), dove  $a_0, a_1, a_2, a_3$  sono scalari reali rappresentanti`,
143 `i coefficienti di, rispettivamente,  $1, i, j, k$ .`,
144 `I quaternioni speciali  $1, i, j, k$  possono essere richiamati mediante,`,
145 `rispettivamente,  $q1, qi, qj, qk$ .`,
146 `I risultati vengono restituiti nella forma  $a_0 + a_1i + a_2j + a_3k$ .`,
147 ``,
148 `REMARK:`,
149 `Come punto di partenza per la programmazione di questo package e' stato`,
150 `utilizzato il package di Greg Fee dell'aprile 1992, pubblicato su "First`,
151 `Leaves: A Tutorial Introduction to Maple V" che fa parte della documentazione`,
152 `ufficiale di Maple V. Questo package contiene molte piu' funzioni di quello`,
153 `originale che conteneva solo qparts, qadd, qmul e inoltre qadd e qmul sono`,
154 `state estese per poter supportare la somma e il prodotto di piu' di due`,
155 `quaternioni con una sola chiamata di funzione.`,
156 ``,
157 `SEE ALSO:`,
158 `  qparts, qtrace, qnorm, qabs, qconj, qscal, qadd, qsubs, qmul, qinv`);
159
160 # Testo di aiuto per la funzione qparts
161 `help/text/qparts` := TEXT(
162 ``,
163 `HELP FOR: qparts`,
164 ``,
165 `CALLING SEQUENCE:`,
166 `  qparts( $a, a_0, a_1, a_2, a_3$ )`,
167 ``,
168 `PARAMETERS:`,
169 `  -  $a$  e' un quaternione`,
170 `  -  $a_0, a_1, a_2, a_3$  sono le variabili alle quali verranno assegnati i`,
171 `    coefficienti di  $a$ ,
172 ``,
173 `SYNONPMSIS:`,
174 `qparts estrae i coefficienti del quaternione  $a$  e li assegna alle variabili`,
175 ` $a_0, a_1, a_2, a_3$ . Non ritorna alcun valore sul terminale.`,
176 ``,
177 `EXAMPLE:`,
178 ` $\>$  with(quaternion):`,
179 ` $\>$  a := quat(3, -1, 4, 2);`,
180 `      3 - i + 4j + 2k`,
181 ` $\>$  qparts(a, a0, a1, a2, a3);`,
182 ``,
183 ` $\>$  a0; a1; a2; a3;`,
184 `      3`,
185 `      -1`,
186 `      4`,
187 `      2`,
188 ``,

```

```

189 `SEE ALSO:`,
190 `    quaternion`):
191
192 # Testo di aiuto per la funzione qtrace
193 `help/text/qtrace` := TEXT(
194 ``,
195 `HELP FOR: qtrace`,
196 ``,
197 `CALLING SEQUENCE:`,
198 `    qtrace(a)`,
199 ``,
200 `PARAMETERS:`,
201 `    a e' un quaternione`,
202 ``,
203 `SYNONPSIS:`,
204 `qtrace calcola la traccia del quaternione a. Il risultato e' un reale.`,
205 ``,
206 `EXAMPLE:`,
207 `> with(quaternion):`,
208 `> a := quat(3, -1, 4, 2);`,
209 `    3 - i + 4j + 2k`,
210 `> qtrace(a);`,
211 `    6`,
212 ``,
213 `SEE ALSO:`,
214 `    quaternion`):
215
216 # Testo di aiuto per la funzione qnorm
217 `help/text/qnorm` := TEXT(
218 ``,
219 `HELP FOR: qnorm`,
220 ``,
221 `CALLING SEQUENCE:`,
222 `    qnorm(a)`,
223 ``,
224 `PARAMETERS:`,
225 `    a e' un quaternione`,
226 ``,
227 `SYNONPSIS:`,
228 `qnorm calcola la norma del quaternione a. Il risultato e' un reale.`,
229 ``,
230 `EXAMPLE:`,
231 `> with(quaternion):`,
232 `> a := quat(3, -1, 4, 2);`,
233 `    3 - i + 4j + 2k`,
234 `> qnorm(a);`,
235 `    30`,
236 ``,
237 `SEE ALSO:`,
238 `    quaternion`):
239
240 # Testo di aiuto per la funzione qabs
241 `help/text/qabs` := TEXT(
242 ``,
243 `HELP FOR: qabs`,
244 ``,
245 `CALLING SEQUENCE:`,
246 `    qabs(a)`,
247 ``,
248 `PARAMETERS:`,
249 `    a e' un quaternione`,
250 ``,
251 `SYNONPSIS:`,
252 `qabs calcola il valore assoluto del quaternione a. Il risultato e' un reale.`,
253 ``,
254 `EXAMPLE:`,
255 `> with(quaternion):`,
256 `> a := quat(3, -1, 4, 2);`,
257 `    3 - i + 4j + 2k`,
258 `> qabs(a);`,
259 `    sqrt(30)`,
260 ``,
261 `SEE ALSO:`,
262 `    quaternion`):
263
264 # Testo di aiuto per la funzione qconj
265 `help/text/qconj` := TEXT(
266 ``,
267 `HELP FOR: qconj`,
268 ``,
269 `CALLING SEQUENCE:`,
270 `    qconj(a)`,
271 ``,
272 `PARAMETERS:`,
273 `    a e' un quaternione`,
274 ``,
275 `SYNONPSIS:`,
276 `qconj calcola il quaternione coniugato di a. Il risultato e' un quaternione.`,
277 ``,
278 `EXAMPLE:`,
279 `> with(quaternion):`,
280 `> a := quat(3, -1, 4, 2);`,
281 `    3 - i + 4j + 2k`,
282 `> qconj(a);`,

```

```

283 `      3 + i - 4j - 2k`,
284 ``,
285 `SEE ALSO:`,
286 `  quaternion`):
287
288 # Testo di aiuto per la funzione qscal
289 `help/text/qscal` := TEXT(
290 ``,
291 `HELP FOR: qscal`,
292 ``,
293 `CALLING SEQUENCE:`,
294 `  qscal(k, a)`,
295 ``,
296 `PARAMETERS:`,
297 `  - k e' uno scalare (un numero reale)`,
298 `  - a e' un quaternione`,
299 ``,
300 `SYNONPSIS:`,
301 `qscal calcola il prodotto tra lo scalare k e il quaternione a. Il risultato`,
302 `e' un quaternione`,
303 ``,
304 `EXAMPLE:`,
305 `> with(quaternion):`,
306 `> a := quat(3, -1, 4, 2);`,
307 `      3 - i + 4j + 2k`,
308 `> qscal(-5, a);`,
309 `      -15 + 5i - 20j - 10k`,
310 ``,
311 `SEE ALSO:`,
312 `  quaternion`):
313
314 # Testo di aiuto per la funzione qadd
315 `help/text/qadd` := TEXT(
316 ``,
317 `HELP FOR: qadd`,
318 ``,
319 `CALLING SEQUENCE:`,
320 `  qadd(args)`,
321 ``,
322 `PARAMETERS:`,
323 `  args e' una successione di quaternioni`,
324 ``,
325 `SYNONPSIS:`,
326 `qadd somma i quaternioni che appaiono nella successione degli argomenti. Il`,
327 `risultato e' un quaternione`,
328 ``,
329 `EXAMPLE:`,
330 `> with(quaternion):`,
331 `> a := quat(2, 3, 1, 5);`,
332 `      2 + 3i + j + 5k`,
333 `> b := quat(9, 4, 7, 6);`,
334 `      9 + 4i + 7j + 6k`,
335 `> c := quat(0, 2, -1, 2);`,
336 `      2i - j + 2k`,
337 `> qadd(a, b, c);`,
338 `      11 + 9i + 7j + 13k`,
339 ``,
340 `SEE ALSO:`,
341 `  quaternion`):
342
343 # Testo di aiuto per la funzione qsub
344 `help/text/qsub` := TEXT(
345 ``,
346 `HELP FOR: qsub`,
347 ``,
348 `CALLING SEQUENCE:`,
349 `  qsub(a, b)`,
350 ``,
351 `PARAMETERS:`,
352 `  a, b sono due quaternioni`,
353 ``,
354 `SYNONPSIS:`,
355 `qsub calcola a-b. Il risultato e' un quaternione`,
356 ``,
357 `EXAMPLE:`,
358 `> with(quaternion):`,
359 `> a := quat(2, 3, 1, 5);`,
360 `      2 + 3i + j + 5k`,
361 `> b := quat(9, 4, 7, 6);`,
362 `      9 + 4i + 7j + 6k`,
363 `> qsub(a, b);`,
364 `      -7 - 5i - 6j - k`,
365 ``,
366 `SEE ALSO:`,
367 `  quaternion`):
368
369 # Testo di aiuto per la funzione qmul
370 `help/text/qmul` := TEXT(
371 ``,
372 `HELP FOR: qmul`,
373 ``,
374 `CALLING SEQUENCE:`,
375 `  qmul(args)`,
376 ``,

```

