Università di Bologna - Corso di Laurea Magistrale in Matematica

Corso di ALGEBRA E GEOMETRIA PER LE APPLICAZIONI
A.A. 2017/18 - Robotica e codici

1. **Un robot spaziale** (D. Cox, J. Little, D. O'Shea, *Ideals, varieties, and algorithms. An introduction to computational algebraic geometry and commutative algebra.* Fourth edition. Undergraduate Texts in Math., Springer, 2015, pag. 304, exercises 8, 9, pag. 318, exercise 18).

   Si consideri un robot tridimensionale con tre bracci e due giunti di rotazione non planari. Il primo braccio è fisso, il primo giunto permette una rotazione del secondo braccio nel piano perpendicolare al primo braccio e il secondo giunto la rotazione del terzo braccio nel piano perpendicolare al secondo braccio.
   Si introducano tre sistemi di riferimento cartesiani destrorsi come segue. Il primo sistema (x_1, y_1, z_1) è fissato con l'origine nel giunto 1 e il braccio 1 sull'asse z_1. Allora il braccio 2 ruota nel piano (x_1, y_1). Il secondo sistema di coordinate (x_2, y_2, z_2) ha anche l'origine nel primo giunto con l'asse x_2 in direzione del braccio 2 e l'asse y_2 nel piano (x_1, y_1). Il terzo sistema di coordinate (x_3, y_3, z_3) con l'origine nel giunto 2 ha l'asse x_3 in direzione del braccio 3 e l'asse z_3 in direzione del braccio 2. Quindi il braccio 3 ruota nel piano (x_3, y_3) che è parallelo al piano (y_2, z_2).
   Infine si denotino con l_2, l_3 le lunghezze dei bracci $2, 3$ rispettivamente, con θ_1 l'angolo orientato che serve per ruotare in senso antiorario l'asse x_1 nell'asse x_2 e con θ_2 l'angolo orientato per ruotare in senso antiorario l'asse y_2 nell'asse x_3.

   (a) Ponendo $c_i = \cos\theta_i$, $s_i = \operatorname{sen}\theta_i$, $i = 1, 2$, mostrare che

   $$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix} = \begin{pmatrix} -s_1c_2 & s_1s_2 & c_1 & c_1l_2 \\ c_1c_2 & -c_1s_2 & s_1 & s_1l_2 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_3 \\ y_3 \\ z_3 \\ 1 \end{pmatrix}.$$

   (b) Determinare le coordinate della mano (= punto finale del braccio 3) nel sistema di riferimento 3.

   (c) Scrivere l'orientamento della mano, cioè un versore in direzione del braccio 3, nel sistema di riferimento 1 in funzione di s_1, c_1, s_2, c_2.

   (d) Data una posizione (x, y, z) della mano nel sistema 1, trovare un sistema di equazioni le cui soluzioni diano le possibili configurazioni dei giunti per le quali si ottiene tale posizione della mano.

   (e) Risolvere il sistema di (d) usando una base di Gröbner ridotta per l'ideale relativo nell'anello $\mathbb{Q}(l_2, l_3, x, y)[c_2, s_2, c_1, s_1, z]$ rispetto all'ordine lessicografico. Perché non si può lavorare in $\mathbb{Q}(l_2, l_3, x, y, z)[c_2, s_2, c_1, s_1]$?
   Scrivere s_1, c_1, s_2, c_2 in funzione di x, y, z, l_2, l_3.
   Scrivere l'orientamento della mano (c) in funzione di x, y, z, l_2, l_3.

   (f) Calcolare le singolarità cinematiche e descriverle geometricamente.

   (g) Descrivere geometricamente il luogo delle possibili posizioni della mano. Quanti sono i possibili orientamenti della mano in un punto raggiungibile in generale? E nelle singolarità cinematiche?

2. **Un codice di Reed-Solomon (RS).**

(a) Quanti polinomi primitivi di grado 7 ci sono in $\mathbb{F}_2[x]$?

(b) Dimostrare che il polinomio $x^7 + x^3 + 1 \in \mathbb{F}_2[x]$ è primitivo.
(Se $\alpha := [x] \in \mathbb{F}_2[x]/(x^7+x^3+1)$, si possono studiare le potenze α^i con Singular utilizzando un ciclo while: `int i = 1; while (a^i != 1) {i++;}; i;`)

(c) Si consideri il codice RS di dimensione 117 sull'alfabeto

$$\mathbb{F}_{128} \cong \mathbb{F}_2(\alpha) \cong \mathbb{F}_2[x]/(x^7 + x^3 + 1) \cong {\mathbb{F}_2}^7.$$

Quindi una parola codice si può vedere come un polinomio $c(x)$ (di quale grado?) o come una sequenza di bit (di quale lunghezza?). Quale è la distanza minima del codice, quanti errori possono essere rilevati e quanti possono essere corretti?

(d) Si consideri un errore che consiste di bit consecutivi errati ("error burst"). Quale lunghezza di bit può avere un tale errore al massimo affinché esso sia sempre correggibile?

(e) Scrivere il polinomio generatore g, la prima riga della matrice generatrice G in forma standard $(A|I)$ e la undicesima riga della matrice di controllo H. Quante righe e quante colonne hanno G e H, quante sindromi ci sono?

(f) Codificare il messaggio

"cane" = `ASCII(99)+ASCII(97)+ASCII(110)+ASCII(101);`

identificando `ASCII(i)` (procedura Singular di `general.lib`) con α^i, cioè codificare la parola

$$w(x) = \alpha^{99}x^{10} + \alpha^{97}x^{11} + \alpha^{110}x^{12} + \alpha^{101}x^{13}.$$

(g) Decodificare la parola codice del file
`http://www.dm.unibo.it/~achilles/algeom/word_received`
e dire quanti errori la contiene.

Qui i primi 7 bit consecutivi `1,0,1,1,1,1,0` $= 1 + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^5$ costituiscono il termine di grado 0, i secondi 7 bit il coefficiente di grado 1, e così via, del polinomio parola codice. La parola può essere caricata nella sessione Singular con il copia e incolla o con il comando

```
execute("matrix m[127][7]="+read("word_received")+";");
```

se il file `word_received` è stato salvato nella cartella dove Singular lavora.

Poi bisogna trasformare la matrice `m` in un polinomio $y \in \mathbb{F}_2(\alpha)[x]$ (le righe di `m` sono i coefficienti di $y \in {\mathbb{F}_2}^7[x] \cong \mathbb{F}_2(\alpha)[x]$). Scrivere una procedura `bin2pol(matrix m)` che ha come input la matrice `m` contenente il messaggio in forma binaria e come output il polinomio y.

Infine bisogna trasformare `poly c = decodifica(d,y)` nel messaggio in chiaro. I coefficienti non nulli di $c \in \mathbb{F}_2(\alpha)[x]$ possono essere scritti nella forma α^i , $1 \leq i \leq 127$, dove i è il codice decimale ASCII. Poiché Singular non rappresenta gli elementi di $\mathbb{F}_2(\alpha)$ in forma di potenze di α, è utile la seguente procedura

```
proc dlog(poly b, poly x)
//---logaritmo discreto
  {
   int i;
   i = 0;
   while (b^i - x != 0 and i < 32767) {i++;};
   return(i);
  };
```

che permette di riscrivere il messaggio in chiaro:

```
proc pol2mess(poly c)
//---dal polinomio c al messaggio in chiaro
  {
   matrix m[127][1] = coeffs(c,x);
   string messaggio;
   int i;
   for (i=11;i<=127;i++)
     {
      messaggio = messaggio + ASCII(dlog(a,m[i,1]));
     };
   return(messaggio);
   };

pol2mess(c);
```