Crampette Olivier
Danton-Lalo Solal

Rapport TP2

Partie 1 : Identifications des longueurs des corps

Observation : Dans cette première partie, nous avons ouvert le fichier de mesures contenant des données angulaires des articulations d'un robot RRR (Rotational, Rotational, Rotational). À l'aide de ces données, nous avons calculé les longueurs des liaisons du robot à partir des angles mesurés. Les longueurs calculées ont été obtenues en utilisant une méthode d'optimisation analytique. Avec X*=AX+B tq:

```
qrad = [q1data[i], q2data[i], q3data[i]]



        Xd = mgdreel(qrad, [1.0, 1.0, 1.0])

        A[2 * i, :] = [np.cos( qrad[0]), np.cos(qrad[0] + qrad[1]),
np.cos(qrad[0] + qrad[1] + qrad[2])]

        A[2 * i + 1, :] = [np.sin(qrad[0]), np.sin(qrad[0] + qrad[1]),
np.sin(qrad[0] + qrad[1] + qrad[2])]

        b[2 * i, 0] = xdata[i][0]



        b[2 * i + 1, 0] = xdata[i][1]
```

Resultats: Avec le MGD on a vérifier les résultats nous obtenons:

Longueurs l1,l2,l3 analytiques: [6.10698524 18.33314853 9.67050546]

Verification de la position calculee: [np.float64(-11.593464401099832), np.float64(9.919539306006989), -5.235987]

Cela correspond approximativement aux valeurs du fichier angle 32.

Partie 2 : Identifications des longueurs des corps et du décalage codeur

Observation:
Ici on fait sensiblement la même chose qu au 1 mais nous fixons l3 et ajoutons d. Les formules de A des moindre carrées analytiques sont adaptées au vue des approximation données.

```
q123 = q1data[i] + q2data[i] + q3data[i]

        A[2 * i, 0] = np.cos(q1data[i])
        A[2 * i, 1] = np.cos(q1data[i] + q2data[i])
        A[2 * i, 2] = -l3 * np.sin(q123)
        B[2 * i] = l3 * np.cos(q123)

        A[2 * i + 1, 0] = np.sin(q1data[i])
        A[2 * i + 1, 1] = np.sin(q1data[i] + q2data[i])
        A[2 * i + 1, 2] = l3 * np.cos(q123)
        B[2 * i + 1] = l3 * np.sin(q123)
```

On obtient

```
Valeurs analytiques:  16.501271018767405 8.43723982744557
0.1402018149371322
```

Avec la mgd nous trouvons qq chose de cohérent aussi.

Partie 3 : Utilisation de scipy.optimize

Observations: Ici on utilise la fonction de la librairie scipy afin de voir les differences avec nos calculs de la partie 2.

On trouve avec la fonction least squares :

```
Valeurs optimisees avec least_squares: l1 = 16.501271018724015 , l2 =
8.437239827408419 , d = 0.140201814939468
Differences :
```

Et en erreur (difference) :

```
Differences :

l1 : -0.000000000004338929216, l2 : -0.000000000003715072694, d :
0.000000000000233579822
```

On peut voir que la methode analytique se rapproche grandement de la fonction de scipy. Selon moi l'erreur est dû à des choix d'approximations/convergence differents.