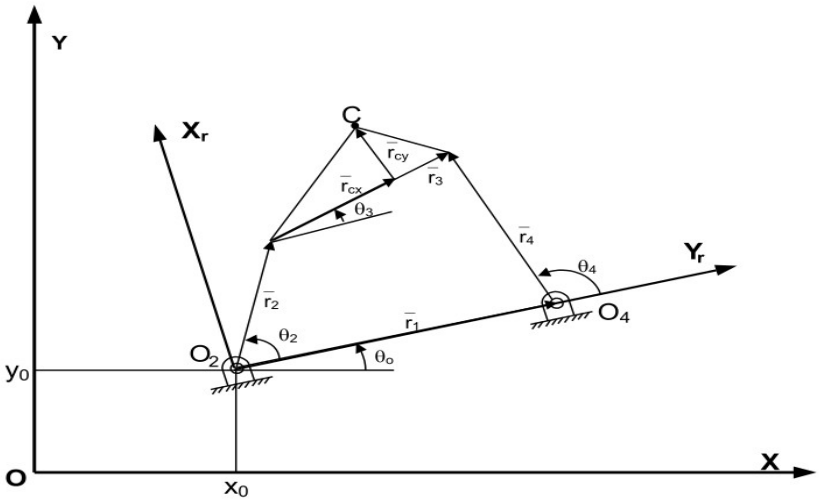
Proyecto 1 Programación Avanzada
Juan Gerardo Fuentes Almeida

Introducción. El objetivo de este proyecto es simular el comportamiento de un mecanismo de 4 brazos y optimizar la aproximación de la terminal C del sistema a ciertos puntos de precisión dados. Es el esquema de abajo, el par motriz es el relativo a θ_2 y partir de este valor, la rotación θ_0 del origen del sistema, las longitudes de los eslabones $r_1, r_2, r_3, r_4, r_{cx}, r_{cy}$ y las coordenadas x_0, y_0 , es posible determinar las coordenadas de los puntos C con los que se medirá la cercanía a los puntos de precisión dados:



Llamaremos $C=[C x_i, C x_i]$ a los puntos que aproximaremos, con $i=1,2,..,n$, siendo n elnumero de puntos de precisión.

Evaluando el vector resultante del origen del sistema a un punto C, las coordenadas cercanas a los puntos de precisión pueden ser determinadas de la siguiente manera respecto al eje de coordenadas relativo a x_0, y_0 y θ_0 :

(1) $\vec{c}_0=\vec{r}_2+\vec{r}_{cx}+\vec{r}_{cy}$
(2) $C_x^0=r_2\cos(\theta_2)+r_{cx}\cos(\theta_3)-r_{cy}\sin(\theta_3)$
(3) $C_y^0=r_2\sin(\theta_2)+r_{cx}\sin(\theta_3)+r_{cy}\cos(\theta_3)$

Evaluando la ecuación de lazo cerrado del mecanismo para calcular θ_2 y θ_3 :

(4) $\vec{r}_2+\vec{r}_3-\vec{r}_4-\vec{r}_1=0$
(5) $r_2\cos(\theta_2)+r_3\cos(\theta_3)-r_4\cos(\theta_4)-r_1=0$
(6) $r_2\sin(\theta_2)+r_3\sin(\theta_3)-r_4\sin(\theta_4)=0$

Elevando al cuadrado (5) y (6), aplicando en θ_4 la identidad $\sin^2(\theta_4)+\cos^2(\theta_4)=1$, y haciendo $k_1=\frac{r_1}{r_2}$, $k_2=\frac{r_1}{r_3}$, y $k_3=\frac{r_4^2-r_1^2-r_2^2-r_3^2}{2r_2r_3}$:

(7) $k_1\cos(\theta_3)+k_2\cos(\theta_2)+k_3=\cos(\theta_2)\cos(\theta_3)+\sin(\theta_2)\sin(\theta_3)$

sea $u=\tan(\frac{\theta_3}{2})$, por tanto tenemos que $\sin(\theta_3)=\frac{2u}{1+u^2}$ y $\cos(\theta_3)=\frac{1-u^2}{1+u^2}$:

(8) $$k_1\frac{1-u^2}{1+u^2}+k_2\cos(\theta_2)+k_3=\cos(\theta_2)\frac{1-u^2}{1+u^2}+\sin(\theta_2)\frac{2u}{1+u^2}$$

de aquí obtenemos la siguiente ecuación de segundo grado $ax^2+bx+c=0$:

(9) $$u^2(\cos(\theta_2)-k_1+k_2\cos(\theta_2)+k_3)+u(-2\sin(\theta_2))+[k_1+(k_2-1)\cos(\theta_2)+k_3]=0$$

donde $a=\cos(\theta_2)-k_1+k_2\cos(\theta_2)+k_3$, $b=-2\sin(\theta_2)$ y $c=k_1+(k_2-1)\cos(\theta_2)+k_3$, aplicando la formula general se deduce que:

(10) $$\theta_3=2\tan^{-1}(\frac{-b\pm\sqrt{b^2-4ac}}{2a})$$

Aquí tenemos que en el caso en que las raíces son complejas, el mecanismo no se conecta, en el caso de raíces positivas tenemos configuración cruzada y en el caso de raíces negativas tenemos configuración abierta.

Luego para calcular las coordenadas del punto C con respecto a los ejes absolutos:

(11) $$C_x=x_0+C_x^0\cos(\theta_0)-C_y^0\sin(\theta_0)$$
(12) $$C_y=y_0+C_x^0\sin(\theta_0)+C_y^0\cos(\theta_0)$$

Implementación.

- Este algoritmo se implemento en dos etapas, un primer programa que lee de un archivo los puntos de precisión a estimar y devuelve en un archivo las medidas del mecanismo que se aproxima a esos puntos, luego se tiene un segundo programa que lee el archivo generado por el primero y simula el mecanismo con WxWidgets.

Programa 1

- El programa abre un archivo “target.txt” y lee del archivo los puntos de precisión que se van a estimar, se genera un estructura que almacena cada uno de los puntos. También se tiene un estructura que almacena los parámetros de un mecanismo.

```
typedef struct typetarget{      //Estructura que contiene los puntos de precision
    double x[6],y[6];
}target;

typedef struct typepoint{       //Estructura que contiene los resultados de analizar un mecanismo
    double x1,y1,theta3_1,x2,y2,theta3_2;
    int flag;
}point;

typedef struct MECHA{           //Estructura que contiene las medidas de un mecanismo
    double r1,r2,r3,r4,rcx,rcy,x0,y0,theta0,result1,result2;
}mechanism;
```

- Se genera un arreglo de 1000 mecanismos cuyos parámetros son asignados pseudo-aleatoriamente, por medio la función `HybridTaus()`, que opera con una distribución uniforme en (0,1). Algo importante que hay que aclarar es que se agrego una restricción a las medidas que se generan aleatoriamente, y esta es que para cualquier mecanismo de 4 brazos, suma de la longitud del brazo mas corto y del brazo mas largo no debe ser mayor a la suma de las longitudes de los otros dos brazos.

```
mechanism *GenerateRandomMecha(mechanism *system, int inf, int sup) //Genera N mecanismos aleatorios
{
    for (int i=inf;i<sup;i++)
    {
        system[i].r3=max*HybridTaus();
        system[i].r4=max*HybridTaus();
        system[i].r1=(system[i].r3+system[i].r4)*HybridTaus();
        system[i].r2=(system[i].r3+system[i].r4-system[i].r1)*HybridTaus();
        system[i].rcx=system[i].r3*HybridTaus();
        system[i].rcy=max*HybridTaus();
        system[i].x0=max*HybridTaus();
        system[i].y0=max*HybridTaus();
        system[i].theta0=2*PI*HybridTaus();
    }
    return system;
}
```

- Posteriormente se corre un ciclo en el cual cada uno de los mecanismos son evaluados para saber si generan soluciones reales del sistema, y se calcula la distancia mas corta a los puntos de precisión, este calculo de distancias se almacena en la misma estructura en cada mecanismo después de esto hacemos un `qsort()` al arreglo de mecanismos para ordenarlos conforme a la menor distancia obtenida a partir de los puntos de precisión, quedando el mejor mecanismo en la posición 0, luego se calcula la media y varianza de los mejores mecanismos elegidos conforme a un criterio establecido, y para la siguiente iteración, el programa genera los mecanismos a partir de la media y varianza obtenida, los cuales son probados nuevamente, generando nuevas medias y varianzas en cada iteración. Al final del numero de iteraciones el programa imprime en un archivo el mejor mecanismo que pudo encontrar.

El algoritmo que se trato de seguir para encontrar las medidas óptimas del mecanismo que pasa por los puntos de precisión consiste en el siguiente:

Después de ordenar los mecanismos, el programa conserva los mejores de acuerdo a la menor distancia del punto C en cada ángulo de rotación a los puntos de precisión dados, para esto se seleccionan aquellos mecanismos cuya distancia este a un múltiplo del mejor mecanismo. A estos mecanismos seleccionados son a los que se calcula la media y la varianza, para sustituir así los mecanismos discriminados por mecanismos generados por una distribución normal con esta media y varianza.

Compilación/Ejecución: Como la implementación se realizo en dos etapas, se tienen dos programas diferentes, uno codificado en C que realiza todos los cálculos para obtener un mecanismo óptimo y generar un archivo con las coordenadas de los puntos de precisión. El comando “make” genera todos los archivos necesarios para la ejecucion y el comando “make run” realiza la ejecucion del programa.

El archivo que genera el primer programa debe colocarse en la carpeta del segundo programa para que este programa lo lea, el segundo programa recibe los datos de los puntos de presicion y las medidas del mecanismo que se genero durante la ejecución del primer programa, de la misma manera el comando

“make” genera todos los archivos necesarios para la ejecución y el comando “make run” realiza la ejecución.

Observaciones/Conclusiones. El proyecto no se completo con éxito, ya que no siempre se obtiene unas medidas útiles para el mecanismo, a pesar de las restricciones que se agregaron, tampoco converge rápido para un valor pequeño (menor a 2) de distancia a los puntos de precisión, esto puede deberse a que el algoritmo no es evolutivo y por tanto depende mucho de la aleatoriedad de los datos generados.