**Crank-Arm Kinematics**

Crank-arm mechanisms are used in many devices to convert rotary motion into reciprocating linear motion. Presses involving crank arms are starting to use servo motors to provide greater flexibility in the press cycles they provide. Many users would like to program these devices directly in terms of the linear dimension, not of the underlying rotary crank angle. Turbo PMAC’s kinematic algorithms permit this method of programming by automatically computing the transformation between the linear end-effector position and the rotary crank angle.

The geometry in this mechanism is quite simple. The trickiest aspect of the kinematics algorithms is to decide which of the two possible crank angles should be used for a given end-effector position. There are many possible methods for making this decision; in this example we make the choice that minimizes crank-wheel acceleration.

**Mechanism Description**

The mechanism has an arm of length $L$ attached by a pivoting joint at one end to an end-effector that is constrained to move in only one direction – with the line between this pivot and the center of the crank collinear with the motion of the end-effector. At the other end of the arm is a pivot joint connecting the arm to the rotating crank wheel at a distance $R$ from the center of the crank wheel. The crank wheel is driven by a servo motor.

![Crank-Arm Kinematics Diagram](image)

**Forward Kinematics**

In the forward kinematics, we want to compute the linear $X$ position from the rotary $\theta$ position. We start with the law of cosines for the triangle with sides of lengths $R$, $L$ and $X$:

$$L^2 = X^2 + R^2 - 2RX \cos \theta$$

We convert this to a quadratic equation in $X$, and solve:

$$X^2 - 2R \cos \theta X + R^2 - L^2 = 0$$

$$X = R \cos \theta \pm \sqrt{R^2 \cos^2 \theta - R^2 + L^2}$$

Due to the physical constraints – the arm can only go out to the right in our drawing – only the positive solution is valid. The following setup and program can be used to implement the forward kinematics:

**Macro Substitution Definitions**

Status/control bits using suggested M-variable definitions

```c
#define Mtr1Homed M145 ; Motor 1 home complete bit
Mtr1Homed->Y:$0000C0,10,1 ; Bit 10 in Motor 1 status word
#define CS1RunTimeErr M5182 ; CS 1 run-time error bit
CS1RunTimeErr->Y:$00203F,22,1 ; Bit 22 in C.S. 1 status word
```
Definitions to Variables with Fixed Functions
#define Mtr1KinPos P1 ; #1 pos in cts for kinematics
#define XkinPos Q7 ; X-axis pos in deg for kin

Definitions to Variables with Open Functions
#define Mtr1Scale Q120 ; Counts per degree of crank
#define Rad Q121 ; Radius of crank
#define Len Q122 ; Length of crank arm
#define L2MinR2 Q123 ; Len^2 – Rad^2
#define Theta Q124 ; Intermediate angle term
#define CosTheta Q125 ; Intermediate trig term
#define Dtheta Q126 ; Velocity term

Setup for Program
I15=0 ; Trig calcs in degrees
I5150=1 ; Enable kinematics for C.S.1

Compute System Constants (these may go in power-on PLC)
Mtr1Scale=200 ; Counts per degree
Rad=400 ; mm
Len=750 ; mm
L2MinR2=Len*Len-Rad*Rad ; Pre-compute for efficiency

Forward Kinematic Program Buffer for Repeated Execution
&1 OPEN FORWARD CLEAR
IF(Mtr1Homed=1) ; Valid position reference?
   Theta=(Mtr1KinPos/Mtr1Scale)%-180
   CosTheta=COS(Theta) ; Compute once for efficiency
   XKinPos=Rad*CosTheta+SQRT(Rad*Rad*CosTheta*CosTheta-L2MinR2)
   DTheta=0 ; Starts at zero velocity
ENDIF
CLOSE

This forward-kinematics program will be executed automatically every time Coordinate System 1 starts a motion program. It can also be executed on a PMATCH command. Note that variables Theta and DTheta are computed and stored for use by the inverse kinematic program.

Inverse Kinematics
In the inverse kinematics, we want to compute the rotary $\theta$ position from the linear $X$ position. We rearrange the law-of-cosine equation used in the forward kinematics to isolate and solve for $\theta$:

$$2RX \cos \theta = X^2 + R^2 - L^2$$

$$\cos \theta = \frac{X^2 + R^2 - L^2}{2RX}$$

$$\theta = \pm \cos^{-1} \left( \frac{X^2 + R^2 - L^2}{2RX} \right)$$

Here, both the positive and negative arc-cosine solutions are potentially valid, and we must devise a method to choose between them. In this example, we will choose the solution that produces the lowest acceleration in $\theta$. When we are not close to 0 or 180 degrees, this strategy will keep us on the same “side” (as would choosing the lowest velocity). As we pass through 0 or 180 degrees, this will keep us moving in the same direction (using the lowest velocity would not necessarily do this).

Because the crank wheel can rotate many revolutions, we cannot simply use the arc-cosine solution. Instead, we must extend this value by computing the change in angle, and adding this to an accumulated multi-turn angle.
Additional Definitions for Inverse Kinematic Program

```c
#define ThetaP Q131 ; Possible positive soln
#define ThetaM Q132 ; Possible negative soln
#define DthetaP Q133 ; Velocity if positive soln
#define DthetaM Q134 ; Velocity if negative soln
#define D2ThetaP Q135 ; Accel if positive soln
#define D2ThetaM Q136 ; Accel if negative soln
```

Additional Setup for Inverse Kinematic Program

```c
&1
#1->I ; Motor 1 assigned to inverse kinematic axis
        ; in CS 1
I5113=10 ; Solve inv. kin. every 10 msec

&1 OPEN INVERSE CLEAR

CosTheta=(XKinPos*XKinPos-L2MinR2)/(2*Rad*XKinPos)

IF(CosTheta!>1.0 AND CosTheta!<-1.0) ; Valid position?
    ThetaP=ACOS(CosTheta) ; Tentative positive soln
    ThetaM=-ThetaP ; Tentative negative soln
    DThetaP=(ThetaP-Theta)%-180 ; Velocity for positive soln
    DThetaM=(ThetaM-Theta)%-180 ; Velocity for negative soln
    D2ThetaP=DThetaP-DTheta ; Accel for positive soln
    D2ThetaM=DThetaM-DTheta ; Accel for negative soln
    IF(ABS(D2ThetaP)<ABS(D2ThetaM)) ; Use positive soln
        Theta=ThetaP ; Select and save for next cycle
        DTheta=DThetaP ; Select and save for next cycle
    ELSE ; Use negative soln
        Theta=ThetaM ; Select and save for next cycle
        DTheta=DThetaM ; Select and save for next cycle
    ENDIF

    Mtr1KinPos=Mtr1KinPos+DTheta*Mtr1Scale ; Accumulated angle
ELSE ; Command out of range
    CSI1RunTimeErr=1 ; Set to stop
ENDIF

CLOSE
```

Note that by selecting the positive solution only if its resulting acceleration is less than the acceleration for the negative solution, when we start from a stop at 0° or 180°, we will choose the negative solution. If we had used the !> (not greater than) comparator instead of the < (less than) comparator, we would have chosen the positive solutions in these cases instead.