YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DENEYSEL BİR STEWART PLATFORMUNUN KONTROL VE SİMULASYON PROGRAMLANMASI

Makine Müh. HİLMİ KÖMÜRLÜOĞLU

FBE Makine Mühendisliği Anabilim Dalı Makine Teorisi ve Kontrol Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Yrd. Doç. Dr. Vasfi Emre ÖMÜRLÜ

İSTANBUL, 2007

İÇİNDEKİLER

SİMGE LİSTESİ

W	Sabit koordinat sistemi
P	Hareketli koordinat sistemi
${}^{W}R_{P}$	Rotasyon matrisi
q	Hareketli platformun sabit koordinat sistemine göre pozisyonunu ifade eden konum matrisi
l	Mafsal uzayı koordinat vectörü
a_i	Her bir linkin hareketli platforma bağlantı noktası
b_i	Her bir linkin sabit platforma bağlantı noktası
L_i	Her bir linkin konumunu ifade eden vektör
n_i	Her bir linkin birim vektörü
J	Jacobian matrisi

KISALTMA LİSTESİ

SPM	Stewart Platform Mekanizması
IK	Ileri Kinematik
TK	Ters Kinematik
NRM	Newton-Rapson Metodu
3DS	Three Dimension Simulation (Üç Boyutlu Simülasyon)
JM	Jacobian Matrisi

ŞEKİL LİSTESİ

TABLO LİSTESİ

ÖNSÖZ

İlk olarak tekerlek test makinasında kullanılmak üzere tasarlanan Stewart Platform Mekanizması, bu gün yüksek pozisyonlama kabiliyeti dolayısı ile önemli uygulama alanlarından biri haline gelmiştir. Stewart Platform Mekanizmaları yüksek hassasiyetli pozisyonlama gerektiren amiyat robotlarının tasarımında, büyük çanak antenlerin hareket mekanizmalarında, teleskopların yönlendirilmesinde ve titreşim obsorbe eden sistemler gibi ileri konstrüksiyonlar için vazgeçilmez mekanik yapıların oluşturulmasında kullanılmaktadır.

Bu çalışmada bütün bu uygulama alanlarında temel olarak kullanılan belirli bir stewart platform mekanizmasının ileri ve geri kinematik analizlerininin matematiksel ifadeleri çıkartılarak bu ifadelerin Visual C++ programlama dilinde derlenip mekanizmanın kontrolünde bize yardımcı olacak sonucu itibari ile uygun bir programın yazılması gerçekleştirilmeye çalışılacak, eş zamanlı olarak 3D State programında simülasyonu gerçekleştirilecek ve sayısal örneklerle doğrulukları araştırılacaktır.

Bu çalışmanın gerçekleştirilmesinde büyük katkıları olan ve bana her aşamasında cesaret veren danışmanım sayın Yrd. Doc. Dr. Vasfi Emre Ömürlü' ye teşekkür ederim.

Son olarak çalışmalarım sırasında bana sonsuz desteklerini esirgemeyen eşime ve aileme teşekkür etmeyi bir borç sayıyorum.

ÖZET

Stewart Platform mekanizması gibi paralel yapılı olan mekanizmaların yüksek hassasiyet, rijitlik ve iyi yük taşıma kabiliyeti açısından avantajları bulunmaktadır. Bu yüzden uçak ve araç simülatörleri, yüksek hassasiyetli takım tezgahları ve işleme merkezleri, kazı makinaları gibi birçok alanda kulanım imkanı bulunmaktadır. Bunlara karşın, kısıtlı çalışma alanlarına ve ileri kinematik hesabında mekanizmanın paralelliğinden dolayı zorluklarla karşılaşılmaktadır.

Stewat Platform mekanizması temel olarak birbirlerine göre seri çalışan altı liner motor ve biri sabit diğeri hareketli olmak üzere iki platformdan oluşmaktadır. Ters kinematik problemi, platformun kartezyen koordinat sistemindeki verilen pozisyon ve açılarna göre plarforma ait 6 adet linkin boy uzunluklarının bulunmasıdır. İleri kinematik problemi, verilen mafsal uzayı pozisyonuna göre kartezyen uzayındaki pozisyon ve açılarının bulunmasıdır. Bu hesaplama ters kinematiğe göre daha karışık bir problemdir.

Bu çalışmada, Stewart platform mekanizmasının simülasyonu için gerekli ileri ve ters kinematiğinin hızlı ve hassas hesap yöntemi problemini araştıracağız. İterativ olarak ileri kinematiğini elde etmek için, ters kinematiğinin matematiksel ifadesi bulunup Newton-Rapson Metodu ile yaklaşık değeri bulunmaya çalışılacaktır. Hesaplamalar sonucu elde edilen değerler aracılığı ile Stewart Platform mekanizmasının gerçek zamanlı 3D simülasyonu gerçekleştirilecektir.

Anahtar kelimeler: Stewart Platform Mekanizması, İleri kinematik, Ters kinematik, Newton-Rapson metodu, C++, 3B Simülasyon.

ABSTRACT

Parallel manipulators such as Stewart Platform mechanism has some advantages of high rigidity, high accuracy, and high load-carrying capacity over serial manipulators. These manipulators have found a variety of applications in flight and vehicle simulators, high-precision machining centers, mining machines, and so on. However, they have some drawbacks of relatively small workspace and difficult forward kinematics problems.

Stewart Platform mechanism consist mainly of two platforms and six linear actuators. One of platform is moveable and the other one is stable. Six linear motor work serially. The inverse kinematic problem for Stewart Platforms, that is, determination of the six link lengths given the position/orientation of the platform or the Cartesian space position. The forward kinematic problem, that is the determination of the Cartesian space position for a given joint space position, is more demanding computationally.

In this paper, we consider the problem of efficient computation of the forward kinematics and the inverse kinematic of Stewart Platform mechanism for its simulation. The solition of the inverse kinematics is formulayted in such a way to optimaze the computation efficiency of the iterative solution of foward kinematics using Newton-Rapson Method. 3D computer simulation of Stewart Platform mechanism is performed via eveluating the computation efficiency of the devoloped computation values.

Keywords: Stewart Platform Mechanism, Forward Kinematics, Invers Kinematics, Newton-Ropson Metod, C++, 3D Simulation.

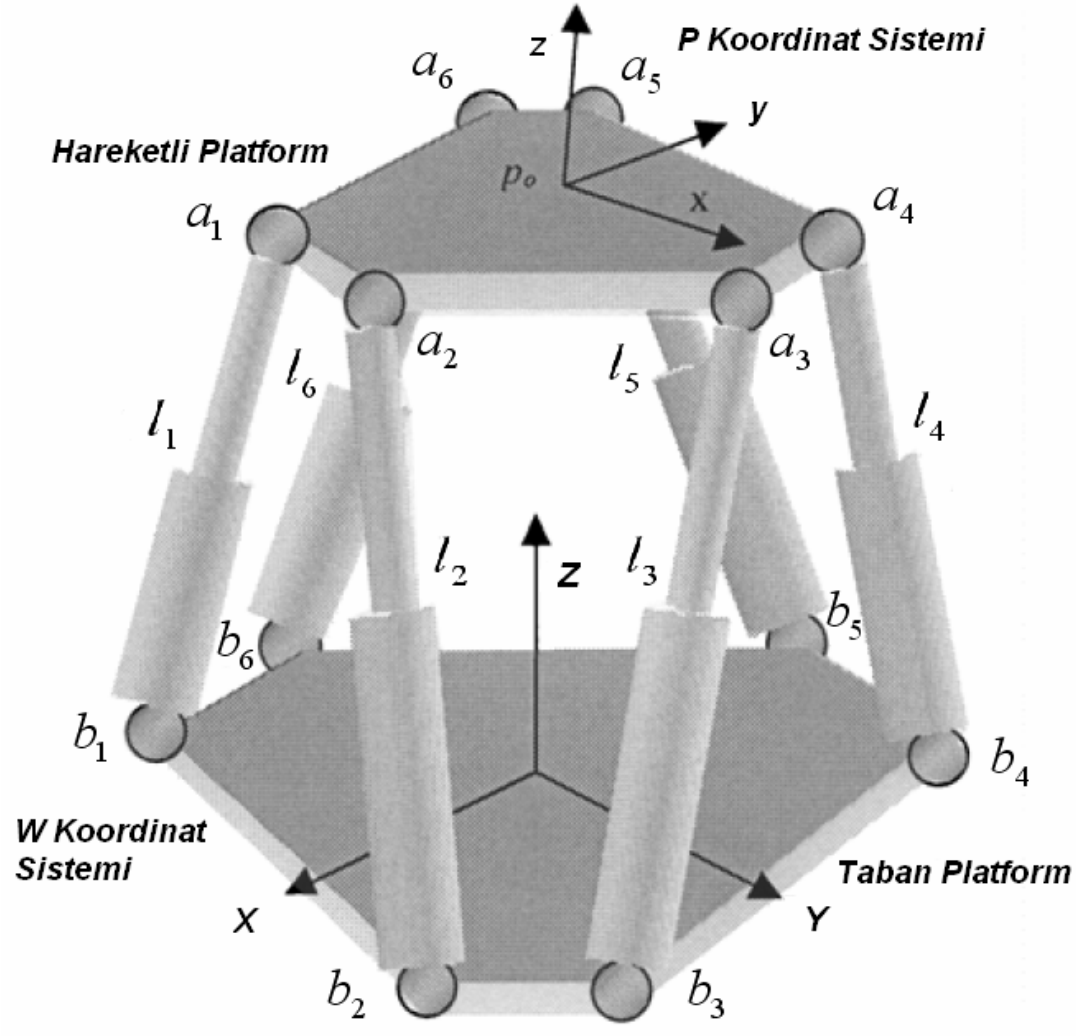
1. GİRİŞ

Stewart Platform aynı zamanda “Gough-Stewart” Platform olarakta bilinir. İlk olarak 1956 yılında Gough tarafından tekerlek test makinası için tasarlamıştır[1], ve daha sonra bu mekanizmayı 1965 yılında Stewart uçak simulatörü olarak kullanmıştır[2]. O zamanlardan bu yana bir çok imalat ve robotik uygulamalarda faydalanıldığından önemli bir araştırma alanı olarak ilgi çekmiştir. Örneğin, Hunt platformu bir paralel robot kolu olarak kullanmayı önermiş[3], Geng ve Haynes ise deneysel titreşim izolatör aleti tasarımında kullanmayı düşünmüşlerdir[4]. Son yıllarda bir çok ticari takım tezgahı üreticisi Stewart Platformunu temel alarak işleme merkezlerini tasarlamaktadır. Bunların en önemli örnekleri Ingersoll Hexapot, Hexel ve Variax işleme merkezleridir. Şekil 1’ de gösteridiği gibi, Stewart Platform yük taşıyabilecek bir platforma parelel olarak monte edilmiş 6 liner motor dan oluşmaktadır. Liner motorların diğer ucu ise sabit bir temele oturtulmuştur. Her bir liner motor temele ve platforma 2 serbestlik dereceli yada 3 serbestlik dereceli mafsallar ile bağlanmaktadır. 6 motorun liner olarak uzanabilme ve kısalabilme kabiliyetleri sayesinde platform 3 tanesi öteleme ve 3 tanesi dönme olmak üzere 6 serbestlik derecesinde hareket kabiliyetine sahip olmaktadır. Motorların liner hareketleri hidrolik veya elektriksel olarak sağlanabilir, ama genellikle servo motorlar ve bu motorların dönme hareketini uzama veya kısalma hareketine çevirebilen bilyalı vida mekanizmaları kullanılmaktadır.

Stewart Platformunu da içine alan paralel bağlantılı mekanizmaların kinematik karakteristikleri seri bağlı mekanizmalara göre farklılık göstermektedir. Stewart Platformunun ters kinematik problemi, platformun kartezyen koordinat sistemindeki verilen pozisyon ve açılarna göre mafsal nokatlarının uzaydaki pazisyon tanımlarını yada plarforma ait 6 adet bacak boy uzunluklarının bulunmasıdır. Ters kinematik problemi yapılabilirlik yönünden açık bir hesaplamadır. İleri kinematik problemi, verilen mafsal uzayı pozisyonuna göre kartezyen uzayındaki pozisyon ve açılarının bulunmasıdır. Bu hesaplama ters kinematiğe göre daha karışık bir problemdir.

2. STEWART PLATFORM MEKANİZMASININ KİNEMATİĞİ

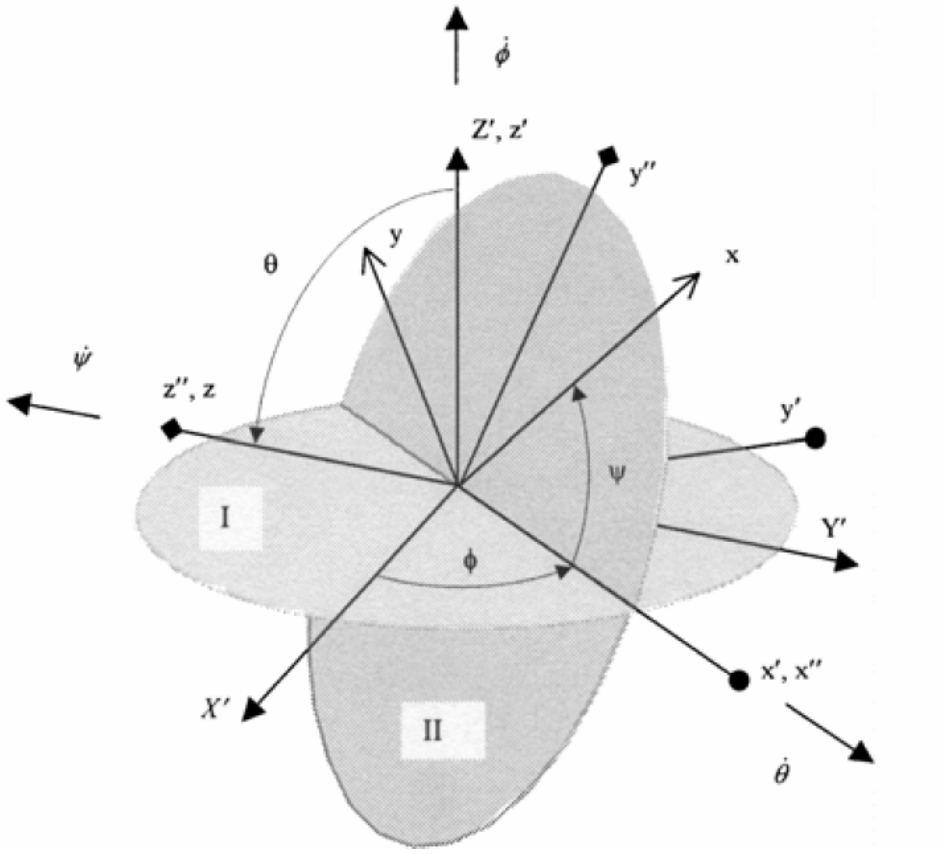
Stewart Platformunuda içine alan paralel bağlantılı mekanizmaların kinematik karakteristikleri seri bağlı mekanizmalara göre farklılık göstermektedir. Stewart Platformunun ters kinematik problemi, platformun kartezyen koordinat sistemindeki verilen pozisyon ve açılarna göre mafsal nokatlarının uzaydaki pazisyon tanımlarını yada plarforma ait 6 adet bacak boy uzunluklarının bulunmasıdır. Ters kinematik problemi yapılabilirlik yönünden sade bir hesaplamadır. İleri kinematik problemi, verilen mafsal uzayı pozisyonuna göre hareketli platformun kartezyen uzayındaki pozisyon ve açılarının bulunmasıdır. Bu hesaplama ters kinematiğe göre daha karışık bir problemdir.



Şekil 2.1 Stewart Platform Mekanizması

2.1 Koordinat Sistemlerinin Belirlenmesi

Stewart platform mekanizmasının hareketli platform kısmı 6 serbestlik derecesine sahip rijit bir cisim olarak düşünebiliriz. Bu kabulden hareket ile platformun bütün açı ve pozisyonlarını tanımlayabilmek için 6 adet koordinat gerekmektedir. Bu koordinatlardan 3 tanesi hareketli platform üzerindeki belli bir noktanın sabit bir koordinat sistemine göre referans alınmış pozisyonel yer değiştirmesidir. Diğer 3 koordinat ise dönme hareketi yapmayan koordinat sistemine göre hareketli platformun açısını belilryen açısal yerdeğiştirmesidir. Euler açıları rijit bir cismin kinematik ve dinamik karateristiğni belirlemede sıkca kullanılanılır. Bizde bu çalışmada Şekil-2.1 de (ϕ, θ, ψ) ile gösterilen bir dizi Euler açısı kullanacağız.



Şekil 2.2 Euler açıları z-x-z

Burada,

- ϕ, hareketli koordinat sisteminin Z' ekseni etrafındaki dönüşünü
- θ, hareketli koordinat sisteminin x' ekseni etrafındaki dönüşünü
- ψ, hareketli koordinat sisteminin z'' ekseni etrafındaki dönüşünü ifade etmektedir.

Şekil 2.1'de $X'Y'Z'$ ile gösterilen koordinat takımı dönme hareketi yapmayan sadece rigit cisinle beraber öteleme hareketi yapan bir koordinat sistemidir. xyz koordinat takımı ise rijit cisimle beraber hem öteleme hareketi hemde dönme hareketi yapmaktadır. $x'y'z'$ ve $x''y''z''$ koordinat takımları bir sonraki bölümde ara geçiş işlemleri sırasında bize yardımcı olacak koordinat sistemleridir.

Şekil 2.1'de görebileceğiniz gibi, θ daima z ve Z' eksenleri arasındaki açıyı gösterir. Ayrıca bu gösterimde, eğer platform düzleminin normali platforma ait koordinat sisteminin z ekseni ile bütünleşik seçilirse, ψ ile gösterilen dönme açısı hareketli platformun 5 eksendeki yerdeğiştirmeleri için gereksiz bir serbestli derecesi olacaktır.

Eğer Stewart platformuna ait 6 adet linkin temel platforma ve hareketli platforma bağlandığı noktaları bilinirse, hareketli platformun pozizyon ve açıları verildiği zaman mekanizmanın mafsal uzay kinematiği kolayca hesaplanabilir. Bu açıklama Stewart platformu gibi paralel mekanizmalarının ters kinematiğinin basitliğini ifade eder.

Hareketli platformun kinematiğini tanımlamak için, Şekil 2.1 'de gösterilen 2 koordinat sistemine ihtiyaç duyacağız. Bunlar temel platform üzerinde bulunan W koordinat sistemi ve hareketli platform üzerindeki bir p_0 noktasına referanslanmış P koordinat sistemi. P eksen takımının doğrusal pozisyonu belirtmek için W eksen takımına göre p_0 noktasının koordinatlarını gösteren $x = (x, y, z)^T$ vektörü tanımlayalım. P eksen takımının, W eksen takımına göre açısal pozisyonu ise ${}^W R_P = (r_1, r_2, r_3)$ vektörü ile ifade edelim. Burada r_1 , r_2 ve r_3 W eksen takımına göre referans alınarak P koordinat sisteminin eksenlerini ifade eden 3x1 boyutundaki birim vektörlerdir. Bir sonraki bölümde ${}^W R_P$ döndürme matrisinin Euler acıları ile ifade edilmiş şeklini türeteceğiz.

Elemanları, platformun pozisyon ve açılarını ifade eden altı değişkenden oluşan genelleştirilmiş bir q vektörü tanımlayalım.

$$q = (X, Y, Z, \phi, \theta, \psi)^T \tag{2.1}$$

Mafsal uzayı koordinat vektötünü ise,

$$l = (l_1, l_2, l_3, l_4, l_5, l_6)^T \tag{2.2}$$

şeklinde tanımlayalım. Burada $l_i \quad i = 1,...,6$ Stewart platformuna ait 6 bacağin boylarını ifade etmektedir. Bir sonraki bölümde ters ve ileri kinematik problemlerin çözümlerini bulabilmek için Stewart platformuna ait iki koordinat takımının birbirlerine göre ifadeleri aranacaktır.

2.2 Hareketli Platformun Kinematiği

Hareketli paltforma ait $x-y-z$ koordinat takımı (yada P, platform koordinat sistemi) ile $X'-Y'-Z'$ koordinat takımı arasındaki ilişkiyi kurmak için 3x3 boyutundaki ${}^{W}R_P$ döndürme matrisi tanımlanır. ${}^{W}R_P$ matrisinin Euler açılı ifadesi aşağıdaki gibidir.

$${}^{W}R_P = \begin{pmatrix} \cos\psi\cos\phi - \cos\theta\sin\phi\sin\psi & -\sin\psi\cos\phi - \cos\theta\sin\phi\cos\psi & \sin\theta\sin\phi \\ \cos\psi\sin\phi + \cos\theta\cos\phi\sin\psi & -\sin\psi\sin\phi + \cos\theta\cos\phi\cos\psi & -\sin\theta\cos\phi \\ \sin\psi\sin\theta & \cos\psi\sin\theta & \cos\theta \end{pmatrix} \tag{2.3}$$

2.2.1 Hareketli Platformun Ters Kinematiği

Stewart platform mekanizmasının ters kinematik problemi, hareketli platformun verilen uygun kartezyen koordinata göre 6 adet bacağın boy uzunluklarının ve zamana göre türevlerinin 3 pozisyonel vede 3 Euler açısal yerdeğiştirmenin ifadesi olarak belirlenmesidir. Şekil 2.1 'e tekrar baktığımızda, hareketli platform üzerindeki i'ninci bağlantı noktasının koordinatları a_i, P koordinat takımı referans alınarak ${}^{P}a_i = (x_{ai}, y_{ai}, z_{ai})^T$ ifadesi ile belirtilir. W sabit koordinat sistemine göre ise

$$a_i = x + {}^{W}R_P \, {}^{P}a_i \tag{2.4}$$

ifadesi kullanılarak elde edilir. Bu ifade ile a_i bağlantı noktasının pozisyonları hesaplandığında i'ninci linkin L_i vektörü kolayca aşağıdaki ifadeyle hesaplanabilir.

$$L_i = a_i - b_i \tag{2.5}$$

Burada b_i, W koordinat takımına göre linkin temel platform ile olan bağlantı noktasının koordinatlarını ifade eden vektördür. i'ninci linkin l_i boyu aşağıdaki ifade ile hesaplanabilir.

$$l_i = \sqrt{L_i . L_i} \tag{2.6}$$

(2.4), (2.5) ve (2.6) numaralı denklemler, hareketli platformu pozisyon ve acısını belirten bilinen bir q kartezyen koordinat vektörü için ters kinematik proleminin çözümünü ifade eder. i'ninci linkin mafsalları arasındaki eksen boyunca uzanan birim vektörde aşağıdaki ifade ile hesaplanabilir.

$$n_i = L_i / l_i \tag{2.7}$$

İleri kinematik prolemine geçmeden önce, Stewart platformunun 6 linki için ters Jacobian matrisini aşağıdaki şekilde yazmakta fayda var.

$$J^{-1} = J_1^{-1} J_2^{-1} \tag{2.8}$$

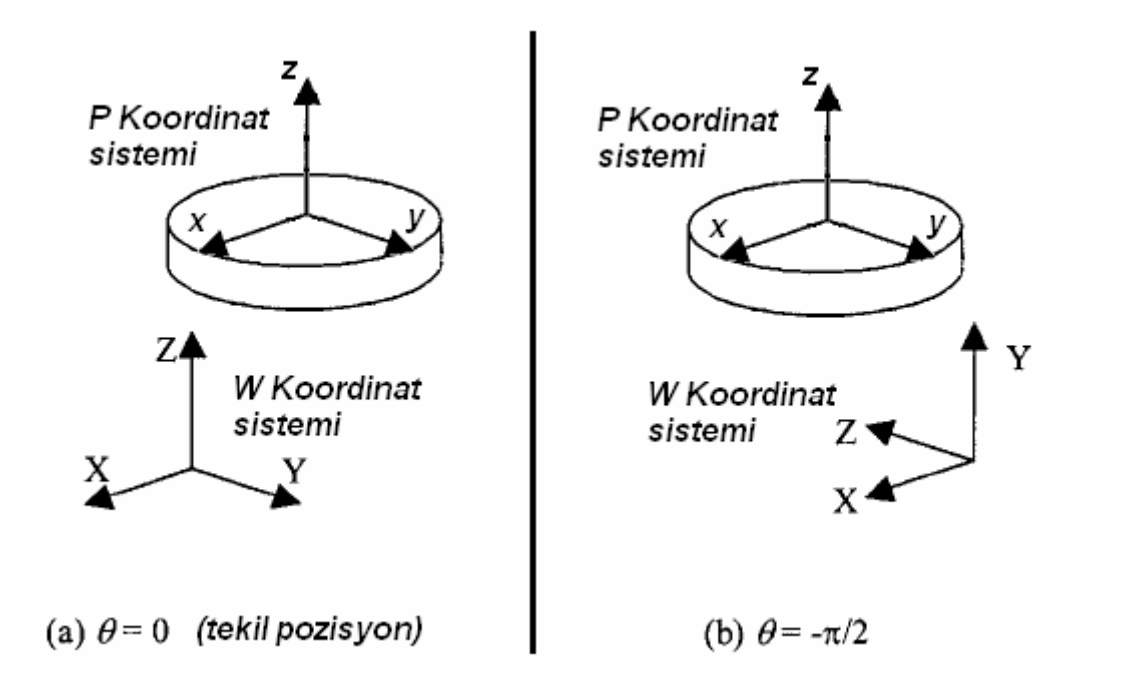
Burada,

$$J_1^{-1} = \begin{pmatrix} n_1^T & \left({}^W R_P \, {}^P a_1 \times n_1\right)^T \\ \vdots & \vdots \\ n_6^T & \left({}^W R_P \, {}^P a_6 \times n_6\right)^T \end{pmatrix} \tag{2.9}$$

$$J_2^{-1} = \begin{pmatrix} I_{3\times3} & & 0_{3\times3} \\ & 0 \quad \cos\phi & \sin\phi\sin\theta \\ 0_{3\times3} & 0 \quad \sin\phi & -\cos\phi\sin\theta \\ & 1 \quad 0 & \cos\theta \end{pmatrix} \tag{2.10}$$

Stewart platform mekanizması tekil bir pozisyonda bulunması için $\det(J^{-1}) = 0$ olmalıdır. Bu durum hem J_1^{-1} hemde J_2^{-1} 'in sıfıra eşit olmasıyla sağlanır. Stewart platform mekanizmasının tüm sınıfları için J_1^{-1} 'in determinantının bir analatik ifadesi mümkün olmadığından J_1^{-1} tekil olduğu durumların analatik olarak bulunması zordur.

Diğer yandan J_2^{-1}, θ 'nın $0,\pi,\ldots,n\pi$ değerleri için tekildir. Bu iki tip tekilsellik, Ma ve Angeles [12] tarafından da belirtiltiği gibi sırasıyla şekilsel ve formülsel tekilliklerdir. Formülsel tekillik daha önce kullanılan Euler açı formülleri ile örtüşür. Daha önce de belirttiğimiz gibi $z-x-z$ Euler formülasyonu için formülsel tekillik J_2^{-1} 'nin tekil yada $\theta = 0,\pi,\ldots,n\pi$ olması durumunda gerçekleşecektir. Şekil 2.3 (a) 'da ki koordinat sistem atamaları ve $z-x-z$ Euler açı formülleri için, bu tip tekilsellik platformun bütün yatay pozisyonları için gerçekleşecektir. Bu çözüm bu uygulamada sağlamamayacağından bu problem ya Euler açı formülasyonunu değiştirerek yada koordinat sistemini atanması sırasında döndürerek çözülebilir. Biz burada $z-x-z$ Euler açı formülasyonu gereksiz oryantasyon yönlerini doğrudan tanımlamaya olanak sağladığından dolayı bu çözümlerden ilkini kullanacağız. W sabit koordinat sisteminin yönünü Şekil 2.3 (b) 'de gösterildiği gibi değiştirilerek platformun dik pozisyonu için formülasyon tekillik sağlanacaktır. Fakat bu durum mekanizmanın normal bir operasyonundan uzak olacaktır ve pratikde gerçekleşmiyecektir.



Şekil 2.3 Koordinat sistemlerinin atanması

2.2.2 Hareketli Platformun İleri Kinematiği

Verilen l mafsal uzayı koordinat vektörü için uygun q kartezyen koordinat vektörünün bulunması şeklinde ileri kinematik problemine bir başlangıç yapılabilinir. Genel bir Stewart Platform mekanizması için ters kinematik probleminin aksine ileri kinematik probleminin çözümü daha zor ve karmaşıktır. Bunun sebebi ise birçok çözümünün bulunmasıdır.

Verilen herbir link uzunluk setleri için çözümlerin sayısı, mekanizmanın alabileceği konfigürasyon sayısına göre değişmektedir. Stewart platform mekanizmasının ters kinematik problemi çözümü (2.4) - (2.6) nolu denklemler ile ifade edilmişti. Verilen l degerine karşılık gelen q değerlerini bulmak için bu eşitliklerde tersden işlem yapılamaz. Çükü (2.4) - (2.6) nolu denklemlerde q açık bir şekilde ifade edilememiştir.

Stewart platform mekanizmasının genel sınıflarında, ileri kinematik problemi için 40'dan fazla çözüm olduğu görülmektedir [13][14]. Buna karşılık gerçekte sadece bir çözüm mekanizmanın asıl duruşu ile uyuşmaktadır. Hesaplama sırasında zamana göre avantaj kazandıracak ve kullanılabilinir yaklaşık bir çözüm sağlayacak basit bir metoda gerek duyulmaktadır. Bunun sağlamak için, ileri kinematik probleminin çözümünde Newton-Raphson metoduna dayanan numerik iterativ tekniği kullanılacaktır.

İleri kinematik probleminin çözümüne geçmeden önce Newton-Raphson metodunun temellerinden bahsetmenin ileri kinematik problemin anlaşılırlığı ve çözülebilirliği üzerine büyük yararı olacaktır. Bu bağlamda bir sonraki bölümde Newton-Raphson metoduna giriş yapılarak metodun işlevselliği üzerine durulacak ve daha sonrasında da ileri kinematik probleminin çözüm yöntemi tartışılacaktır.

2.2.2.1 Newton-Raphson Metodu

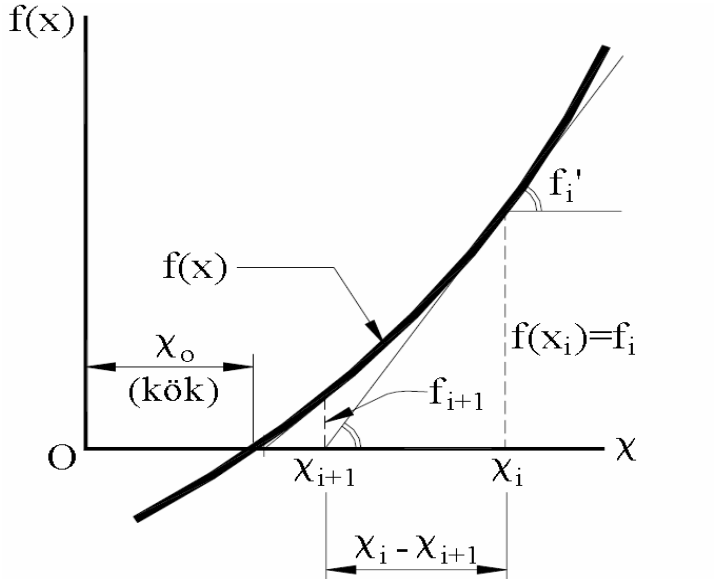
Newton-raphson metodu, bağımsız değişkenin eksen ile kesim noktasının yani kökünün tahmini için foksiyonun türevini yani eğimini kullanmaya dayanır. Örneğin bir f(x) fonksiyonunu ele alalım. Eğer bu fonksiyona ait kökün ilk tahmini x_i ise, Şekil 2.4 de görülebileceği gibi fonksiyonun [x_i, $f(x_i)$] noktasından uzatılan teğetinin x eksenini kestiği nokta köke daha yakın bir değerdir. Ardışık yaklaşımın (i) ve (i+1) sayılı adımlarında arana x_0 değeri için bulunan x_i ve x_{i+1} değerleri arasında

$$f_i' = \frac{f_i}{x_i - x_{i+1}} \tag{2.11}$$

Bağıntısı bulunmaktadır. Buradan,

$$x_{i+1} = x_i - f_i'^{-1} f_i \tag{2.12}$$

yazılabilinir. Buna göre ardışık yaklaşımın her adımında, (2.12) formülünden yararlanarak birsonraki adıma ait bilinmeyen değeri elde edilir. Herhangi bir adımın sonunda bir önceki adımda çıkan değer ile arasındaki fark tolerans değerinden küçük olduğunda işlem sona erdirilir.

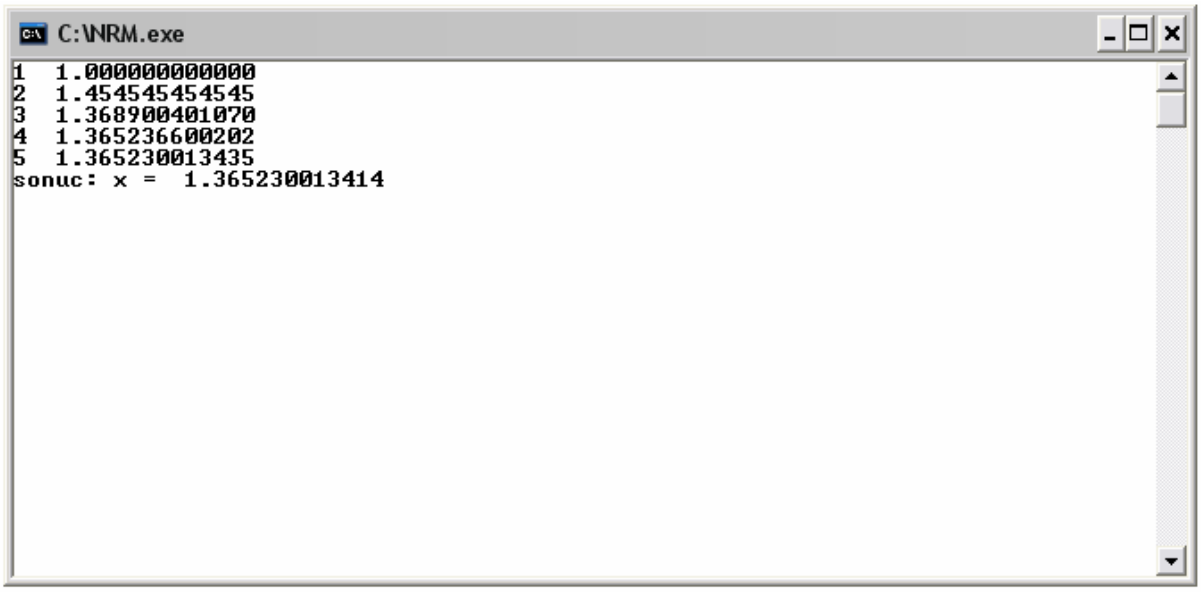


Şekil 2.4 Newton-Raphson Metodu

Bu anlatılan Newton-Rapson metodunu ileri kinematik probleminde kullanmadan önce basit bir fonksiyon üzerinde uygulamasını görmekte ve Visual C++ dilinde nasıl hesap edildiğini örnek bir program üzerinde açıklamakta fayda var.

Örneğimizde f(x) fonksiyonu x3 + 4.0*x2 - 10.0 = 0 olsun. Bu foksiyonun birinci dereceden türevi ise f '(x)=3*x2 + 8.0*x şeklinde ifade edilir. Aslında bu fonksiyonun bir kökünün değeri tam olarak x=1.36523001 dir. Fakat biz Newton-Raphson metodunda kullanacağımız tahmini başlangıç değerini x_i =1.0 olarak kabul edelim.

Programın işlemciyi fazla yormaması için hesaplanacak en fazla iterasyon sayısını ise 10 ile sınırlandıralım ve iterasyonlar sonucu en son bulunan iki değer arasındaki farkın kabul edilebilinir miktarını gösteren tolerans değerini ise 0.0000001 alırsak Visual C++ dilinde yazılabilecek program Tablo 2.1 de göründüğü şekilde olacaktır. Ve programın çıktı görüntüsü ise Şekil 2.5 deki gibi olur.



Şekil 2.5 Örnek Newton-Raphson metodu programının çıktısı

Program çıktısı incelendiğinde de anlaşılacağı gibi beşinci iterasyon sonunda heseplanan son iki değer arasındaki fark tolerans değerinden küçük olduğu için program sonlanmış ve sonuç x = 1.365230013414 olarak ekrana yazdırılmıştır.

```
#include <stdio.h>
#include <math.h>
#include <conio.h>
//maksimum iterasyon sayısı
#define N      10
//kabu edilebilinir tolerans miktarı
#define TOL 0.0000001
//Ana fonksiyon  f(x)=x^3 + 4.0*x^2 - 10.0
double f(double x)
{return (x*x*x + 4.0*x*x - 10.0);}
//Ana fonksiyonun birinci dereceden türevi f'(x)=3*x^2 + 8.0*x
double df(double x)
{return (3.0*x*x + 8.0*x);}
int main(void){
double x, x0;
int i;
i=1;
// f(x)'in x kökünün tahmini yaklaşık başlangıç değeri
x0=1.0;
printf("%d %15.12lf\n", i, x0);
while (i<N)
        {x=x0-f(x0)/df(x0);
        if (fabs(x-x0)<TOL)
                {printf("sonuc x = %15.12lf\n", x);
                getch();
                return 0;};
        i+=1;
        x0=x;
        printf("%d %15.12lf\n", i, x);}
printf("Maksimum iterasyon miktarina ulasildi.");
getch();
return 0;}
```

Tablo 2.1 Örnek Newton-Raphson metodu programı

2.2.2.2 İleri Kinematik Proplemin Hesabında Newton-Roaphson Metodunun Kullanımı

Çok değişkenli denklemlerin tekil köklerini bulmak için de Newton-Raphson metodu kullanılabilinir[15]. Bundan sonraki bahsi geçecek metodla belirtilen toleransların hahssasiyeti ölçüsünde yakın sonuçlar elde edilecektir. Bu sayısal bir örnekle gösterilecektir.

Mekanizmanın belirli bir pozisyonu için $F(q)$ fonksiyonu tanımlanır.

$$F(q) = l(q) - l_{verilen} \tag{2.13}$$

Burada $l(q)$, q kartezyen uzayı koordinat vektörünü kullanarak ters kinematik çözümü ile hesaplanmış mafsal uzayı koordinat vektörüdür. $l_{verilen}$ ise bilinen mafsal uzayı koordinat vektörünü ifade etmektedir. Eğer q istenilen ileri kinematik çözüm ise $l(q)$ ve $l_{verilen}$ birbirlerine eşit olacaklardır ve $F(q)$ sıfıra eşit olacaktır. Newton-Raphson metodunu kullanarak yapılacak bir çözüm aşağıdaki gibi ifade edilebilinir.

$$\tilde{q}_i = \tilde{q}_{i-1} - \left(\frac{\partial F(\tilde{q}_{i-1})}{\partial q} \right)^{-1} F(\tilde{q}_{i-1}) \tag{2.14}$$

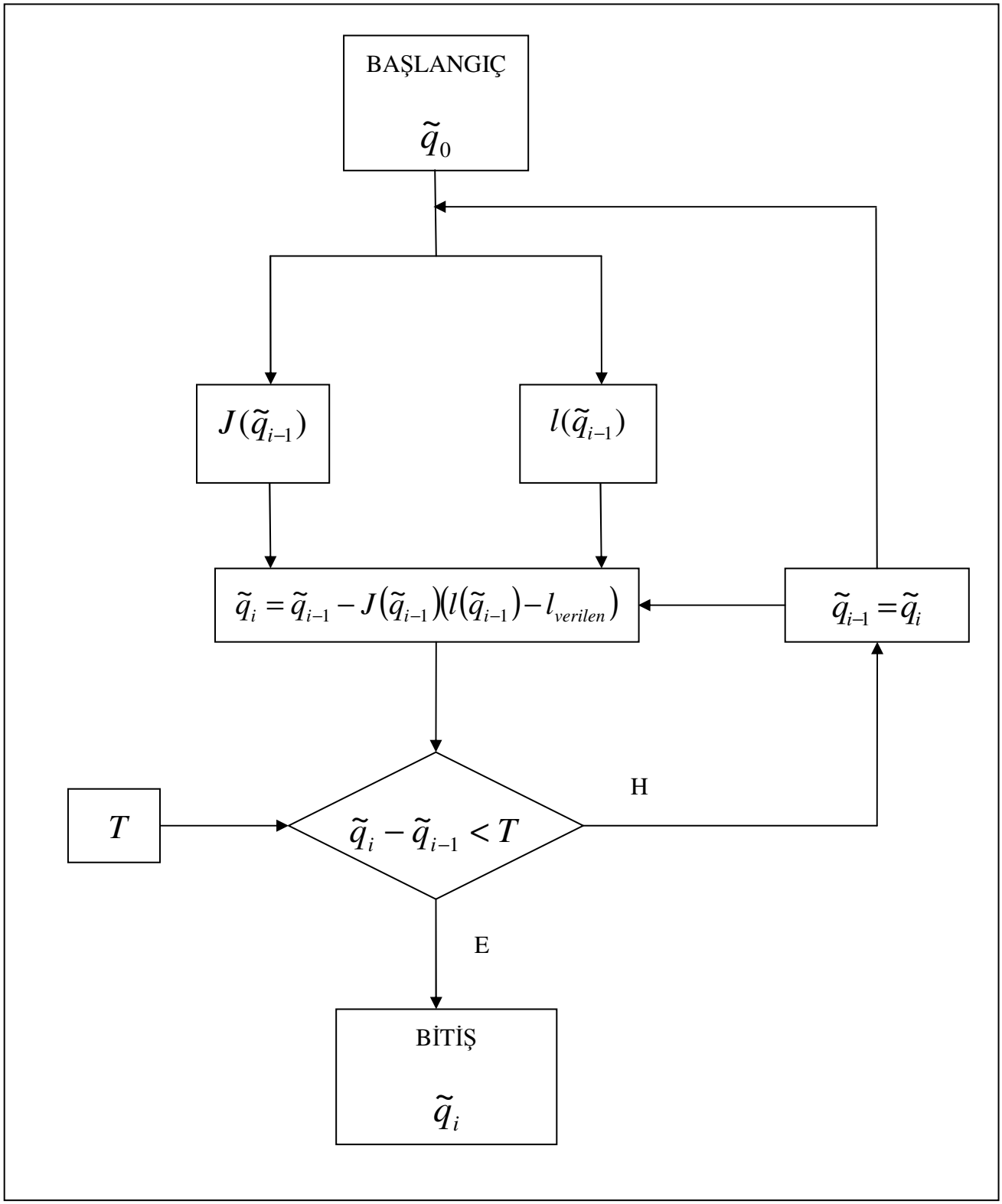
Burada $\tilde{q}_i$, i'ninci iterasyondan sonra elde edilen çözümdür. Bu eşitlikte $\frac{\partial F(\tilde{q}_{i-1})}{\partial q}$ matris ifadesi, Eşitlik 2.8'de verilen $J^{-1}(\tilde{q}_{i-1})$ ters Jacobian matrisi ile aynı gibi görünebilinir.

Buradan hareketle Eşitlik 2.12 aşağıdaki gibi yazılabilinir.

$$\tilde{q}_i = \tilde{q}_{i-1} - J(\tilde{q}_{i-1})(l(\tilde{q}_{i-1}) - l_{verilen}) \tag{2.15}$$

Tahmini bir $\tilde{q}_0$ başlangıç değerinden hareketle Eşitlik 2.12, kabul edilebilinir bir çözüme ulaşana kadar iterativ biçimde kullanılır. Birçok çözüm olduğundan, başlangıç tahmini değerin platformun gerçek değerine mümkün olduğunca yakın seçilmesi çok önemlidir. Başlangıç tahmini değer, ya bilinen kartezyen duruşu yada hareket yörüngesi üzerinde kısa zaman aralığı içindeki bir önceki pozisyonu seçilebilinir.

2.2.2.3 Program Akış Diyagramı



Tablo 2.2 İleri kinematik hesabı için yazılan programın akış diyagramı

3. Stewart Platform Mekanizmasının Kontrol Programlanması

Bir önceki bölümde Stewart Platform mekanizmasına ait hareketli platformun ileri kinematik ve ters kinematik problemlerinin çözümleri matematiksel deklemler ile ifade edilmişti. Bu bölünde ise bu ifadelerin bilgisayar ortamında hesabı gerçekleştirilecektir. İleri ve ters kinematik ifadelerin matematiksel denklemleri C++ dilinde derlenecek. Bunun için derleyici program olarak Visual C++ kullanılacak.

C++, nesne tabanlı dillerden biridir. Bunun anlamı en basit ifade ile anlatılması gerekirse bir fonksiyon arka arkaya kulanılması gerekiyorsa bu fonksiyonu her seferinde ayrı ayrı yazmaya gerek yoktur. Sadece fonksiyonu birkere tanımladıktan sonra ana program içerisinde fonksiyona ihtiyaç duyulduğu zaman birkaç satırla fonksiyonu ve hesaplanması gereken değişkenlerin yazılması yeterli olmaktadır. C++ program dili gibi nesne tabanlı dillerde bu özelliklerden dolayı kütüphane dosyaları yazılmaya başlanmıştır. Bu dosyalar fonksiyonları hazır halde bize vererek tekrar tekrar yazılmasına gerek kalmamıştır.

Bu çalışmada matrislerin vektörel veya scaler çarpımları gibi bellirli fonksiyonların kontrol programını ağırlaştırmaması için ROBOOP – “Robotics object oriented package” kaynak kütüphaneleri kullanılacaktır. Bu kütüphanenin kullanım şekli EK-1’ de gösterilmektedir.

Hesaplanan hareketli platform pozisyon ve açılarını gerçek zamanlı olarak simülasyon etmek için yüksek performansı ve kullanımı kolay olduğundan dolayı 3D STATE grafik motoru kullanılacaktır. 3D STATE grafik motorunun kullanımı EK-2’ de gösterilmektedir. Bundan sonraki bölümlerde kontrol programı için kullanılan fonksiyonların Visual C++ dilinde ifade şekilleri üzerine durulacaktır.

3.1 Harekerli platform üzerindeki bağlantı noktalarının hesap döngüsü

$$a_i = x + {}^{W}R_{P}\,{}^{P}a_i$$

```
ReturnMatrix LinkStewart::Find_a(const Matrix wRp, const ColumnVector q)
    {
    ColumnVector a;
    a = q.Rows(1,3) + wRp*ap;
    a.Release();
    return a;
    }
```

Değişkenler:

wRp : Rotasyon matrisi

q : Hareketli platformun pozisyonu

Burada,

a : Herbir link için hareketli platform üzerindeki bağlantı noktalarının sabit koordinat sistemine göre konumları hesap edilmektedir.

3.2 Link boyunca uzanan birim vektörün hesap döngüsü

$$n_i = L_i / l_i$$

```
ReturnMatrix LinkStewart::Find_UnitV()
    {
    Matrix Tmp (1,3);
    Tmp = (aPos - b)/L;
    Tmp.Release();
    return Tmp;
    }
```

Değişkenler:

aPos : Platform üzerindeki bağlantı noktalarının hareketli koordinat sistemine göre konumları.

b : Temel platform üzerindeki link bağlantı noktalarının sabit koordinat sistemine göre konumları.

L : Herbir linkin boyu.

3.3 Link boyunun hesap döngüsü

$$l_i = \sqrt{L_i . L_i}$$

```
Real LinkStewart::Find_Lenght()
    {
    return sqrt(DotProduct((aPos - b),(aPos - b)));
    }
```

Değişkenler:

aPos : Platform üzerindeki bağlantı noktalarının hareketli koordinat sistemine göre konumları.

b : Temel platform üzerindeki link bağlantı noktalarının sabit koordinat sistemine göre konumları.

3.4 Rotasyon matrisinin hesap döngüsü

$$ {}^{W}R_{P} = \begin{pmatrix} \cos\psi\cos\phi - \cos\theta\sin\phi\sin\psi & -\sin\psi\cos\phi - \cos\theta\sin\phi\cos\psi & \sin\theta\sin\phi \\ \cos\psi\sin\phi + \cos\theta\cos\phi\sin\psi & -\sin\psi\sin\phi + \cos\theta\cos\phi\cos\psi & -\sin\theta\cos\phi \\ \sin\psi\sin\theta & \cos\psi\sin\theta & \cos\theta \end{pmatrix} $$

```
ReturnMatrix Stewart::Find_wRp ()
{
Matrix _wRp(3,3);
_wRp(1,1) = cos(q(6))*cos(q(4)) - cos(q(5))*sin(q(4))*sin(q(6));
_wRp(1,2) = -sin(q(6))*cos(q(4)) - cos(q(5))*sin(q(4))*cos(q(6));
_wRp(1,3) = sin(q(5))*sin(q(4));
_wRp(2,1) = sin(q(6))*cos(q(4)) + cos(q(5))*sin(q(4))*cos(q(6));
_wRp(2,2) = -sin(q(6))*sin(q(4)) + cos(q(6))*cos(q(4))*cos(q(5));
_wRp(2,3) = -sin(q(5))*cos(q(4));
_wRp(3,1) = sin(q(6))*sin(q(5));
_wRp(3,2) = sin(q(5))*cos(q(6));
_wRp(3,3) = cos(q(5));
_wRp.Release();
return _wRp;
}
```

Değişkenler :

q(4) : q vektörünün dördüncü elemanı.

q(5) : q vektörünün beşinci elemanı.

q(6) : q vektörünün altıncı elemanı.

3.5 İleri kinematik hesap döngüsü

$$\tilde{q}_i = \tilde{q}_{i-1} - J(\tilde{q}_{i-1})(l(\tilde{q}_{i-1}) - l_{verilen})$$

```
ReturnMatrix Stewart::ForwardKine(const ColumnVector guess_q, const ColumnVector
l_given, const Real tolerance)
{
ColumnVector next_q, tmp_long(6);
Real Diff = 1;
q = guess_q;
while (Diff>tolerance)
{
for(int i=0; i<6; i++)
tmp_long(i+1) = Links[i].L - l_given(i+1);
next_q = q - Jacobian*(tmp_long);
Diff = (next_q - q).MaximumAbsoluteValue();
set_q(next_q);
}
next_q.Release();
return next_q;
}
```

Değişkenler :

guess_q : Hareketli platformun tahmini pozisyonu.

l_given : Linklerin verilen boyları.

tolerance : kabul edilebilinir hata miktarı.

3.6 Jacobian matrisinin hesap döngüsü

$$J^{-1} = J_1^{-1} J_2^{-1}$$

```
ReturnMatrix Stewart::jacobian()
{
Matrix _Jacobi;
_Jacobi = (IJ1*IJ2).i();
_Jacobi.Release();
return _Jacobi;
}
```

Değişkenler :

IJ1 : İlk jacobian matrisi.

IJ2 : İkinci jacobian matrisi.

3.7 İlk jacobian matrisinin tersinin hesap döngüsü

$$J_1^{-1} = \begin{pmatrix} n_1^T & \left({}^W R_P \, {}^P a_1 \times n_1\right)^T \\ \vdots & \vdots \\ n_6^T & \left({}^W R_P \, {}^P a_6 \times n_6\right)^T \end{pmatrix}$$

```
ReturnMatrix Stewart::Find_InvJacob1()
{
Matrix tmp_Jacobi1 (6,6);
for(int i = 0; i<6; i++)
tmp_Jacobi1.Row(i+1)=Links[i].UnitV.t()|
CrossProduct(wRp*Links[i].ap,Links[i].UnitV).t();
tmp_Jacobi1.Release();
return tmp_Jacobi1;
}
```

Değişkenler :

UnitV : Herbir linkin birim vektörü.

wRp : Rotasyon matrisi.

ap : Platform üzerindeki herbir linke ait bağlantı noktasının hareketli koordinat sistemine göre konumu.

3.8 İkinci jacobian matrisinin tersinin hesap döngüsü

$$J_2^{-1} = \begin{pmatrix} I_{3\times3} & & 0_{3\times3} \\ & 0 & \cos\phi & \sin\phi\sin\theta \\ 0_{3\times3} & 0 & \sin\phi & -\cos\phi\sin\theta \\ & 1 & 0 & \cos\theta \end{pmatrix}$$

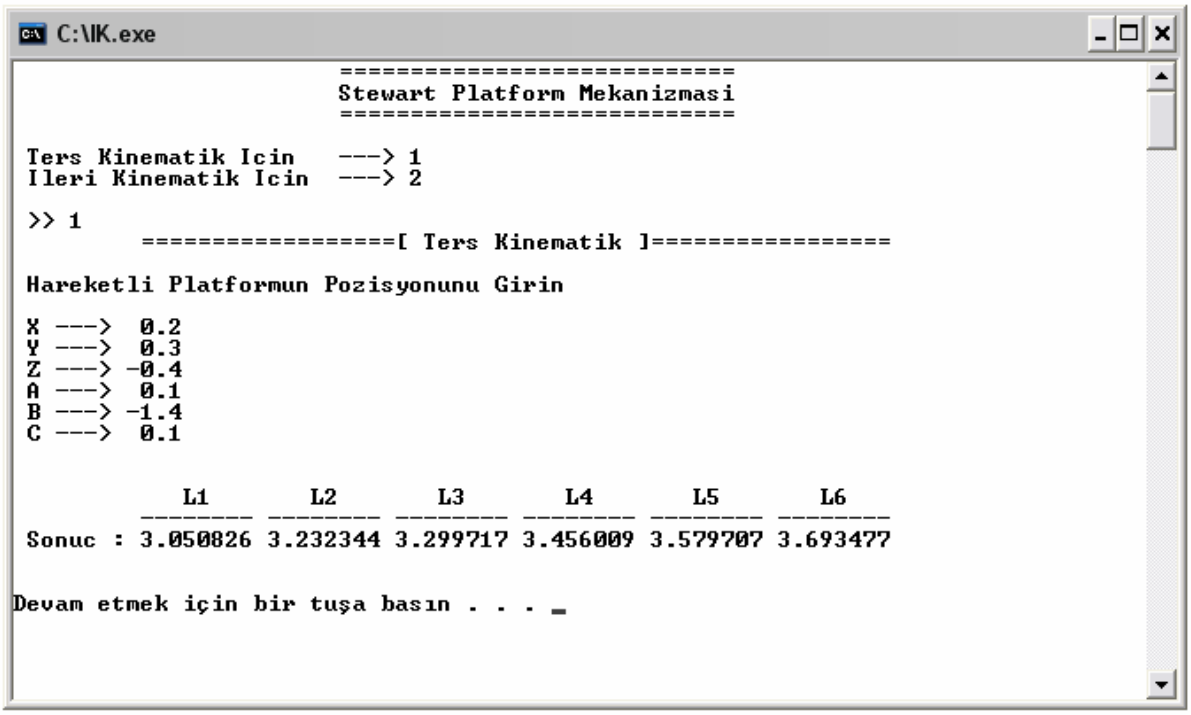
```
ReturnMatrix Stewart::Find_InvJacob2()
{
Matrix tmp_Jacobi2;
tmp_Jacobi2 = IdentityMatrix(6);
tmp_Jacobi2(4,4) = 0;
tmp_Jacobi2(6,4) = 1;
tmp_Jacobi2(4,5) = cos(q(4));
tmp_Jacobi2(5,5) = sin(q(4));
tmp_Jacobi2(4,6) = sin(q(4))*sin(q(5));
tmp_Jacobi2(5,6) = -cos(q(4))*sin(q(5));
tmp_Jacobi2(6,6) = cos(q(5));
tmp_Jacobi2.Release();
return tmp_Jacobi2;
}
```

Değişkenler :

q(4) : q, pozisyon vektörünün dördüncü elemanı.

q(5) : q, pozisyon vektörünün beşinci elemanı.

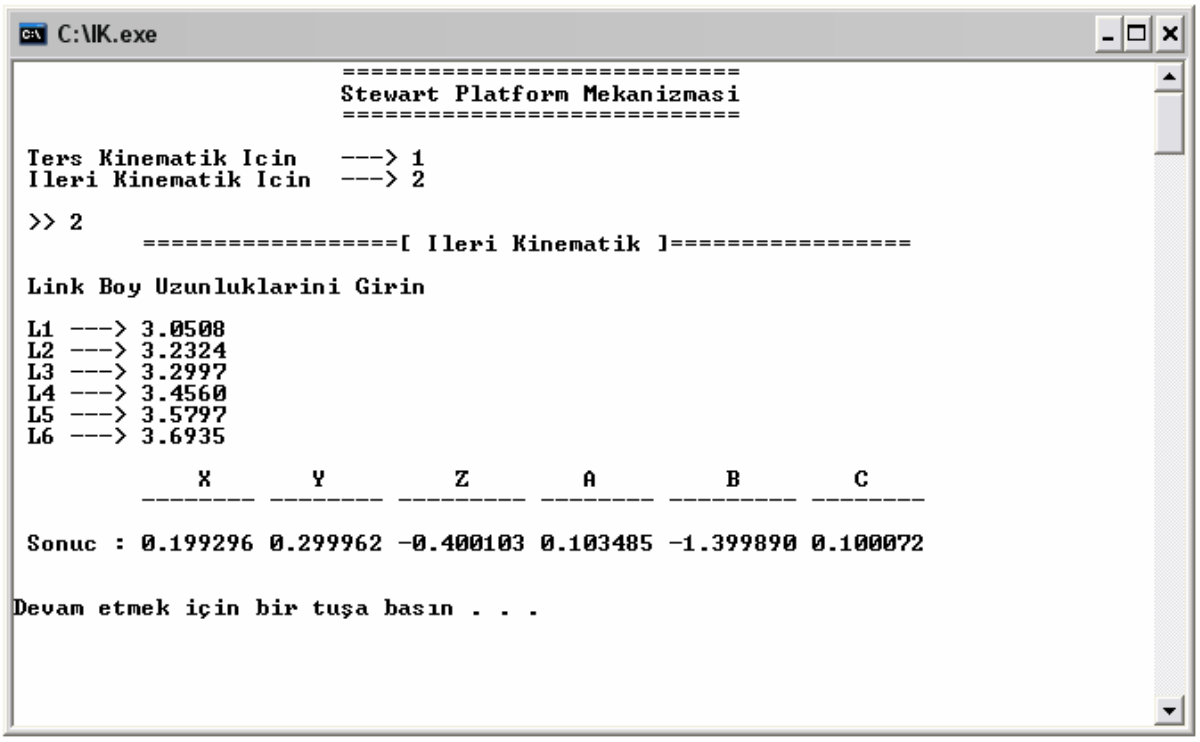
3.9 Ters kinematik programının kullanımı



Şekil 3.1 Stewart platform mekanizmasının ters kinematik hesabı için Visual C++ ile yazılan kontrol programının görüntüsü

IK.EXE programı çalıştırıldığında ilk olarak platformun ileri kinematiğin mi yoksa ters kinematiğin mi hesabını istendiğinin girilmesi gerekmektedir. Eğer ters kinematik isteniyorsa 1 yazılarak ENTER tuşuna basılır. Bundan sonra programa ters kinematik hesabı için gerekli olan "q" hareketli platformun sabit koordinat sistemine göre pozisyonunu ifade eden konum matrisi ile tanımlanan X, Y, Z, konumları ile A, B, C, euler açılarının girilmesi gerekmektedir. Hesabın sonunda sonuç olan herbir linkin uzunluğu Şekil 3.1 de görüldüğü gibi ekrana yansır.

3.10 İleri kinematik programının kullanımı



Şekil 3.2 Stewart platform mekanizmasının ileri kinematik hesabı için Visual C++ ile yazılan kontrol programının görüntüsü

IK.EXE programı çalıştırıldığında ilk olarak platformun ileri kinematiğin mi yosa ters kinematiğin mi hesabını istendiğinin girilmesi gerekmektedir. Eğer ileri kinematik isteniyorsa 2 yazılarak ENTER tuşuna basılır. Bundan sonra programa ileri kinematik hesabı için gerekli olan "l"matrisi ile tanımlanan mafsal uzayı koordinat vectörü ifade eden link boylarının girilmesi gerekmektedir. Hesabın sonunda sonuç olan hareketli platformun X, Y, Z, konumları ile A, B, C, euler açıları Şekil 3.2 de görüldüğü gibi ekrana yansır.

4. Sayısal Örnek

Bu örnekte geliştirilmiş Newton-Raphson metodunu uygulamak için bir bilgisayar programı yazılmıştır. Stewart platformunun kinematik parametreleri Tablo 4.1 ve Tablo 4.2 'de gösterilmektedir. Ve platformun kartezyen koordinatlarındaki duruşu aşağıdaki matris ile ifade edilsin.

$$q = (0.2,0.3,-0.4,0.1,-1.4,0.1)^T \tag{4.1}$$

Burada doğrusal yer değiştirmeler metre ile açısal yer değiştirmeler ise radyan cinsinden ifade edilmektedir. Ters pozisyon kinematiğinin hesaplanmış sonucu olan mafsal uzayı koordinat vektörü aşağıdaki gibi hesaplanmıştır.

$$l = (3.0508,3.2324,3.2997,3.4560,3.5797,3.6935)^T \tag{4.2}$$

Burada bütün link uzunlukları metre cinsinden verilmiştir. Bundan sonra yapılacak hareket, l'nin Eşitlik 2.15 ile verildiğini kabul etmek olacak, mafsal uzayı ölçümlerinden tahminen, q için yaklaşık sayısal çözümün bulunması gerekmektedir.

$$q_0 = (0.25,0.25,-0.45,0.07,-1.7,0.07)^T \tag{4.3}$$

Bu bölümde anlatılan nümerik tekniğin q_0 başlangıç tahmini değeri ve 1×10^{-6} kriteri ile uygulanması ile 3 iterasyon sonucu Tablo 4.3 'de görüldüğü gibi istenilen değere ulaşılır.

Sayısal girdiler; K. Harib ve K. Srinivasan "Kinematic and dynamic analysis of Stewart platform-based machine tool structures" [19] makalesinden alınmıştır.

Tablo 4.1 Hareketli platform üzerindeki mafsal noktalarının P eksen takımına göre konumları

	$a_1(m)$	$a_2(m)$	$a_3(m)$	$a_4(m)$	$a_5(m)$	$a_6(m)$
x	0.225	0.1125	-0.1125	-0.225	-0.1125	0.1125
y	0.0	0.1949	0.1949	0.0	-0.1949	-0.1949
z	-0.228	-0.228	-0.228	-0.228	-0.228	-0.228

Tablo 4.2 Sabit platform üzerideki mafsal noktalarının W koordinat eksenine göre konumları

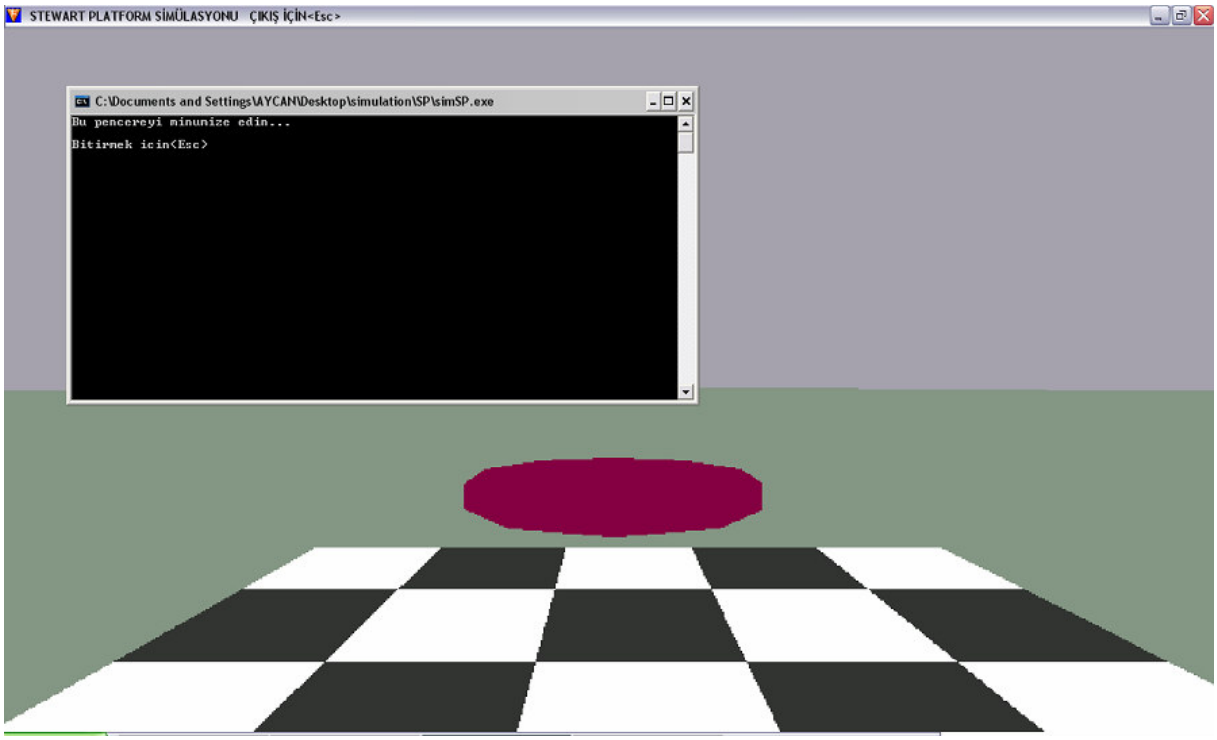
	$b_1(m)$	$b_2(m)$	$b_3(m)$	$b_4(m)$	$b_5(m)$	$b_6(m)$
X	1.7580	1.6021	-1.7580	-1.6021	0.0	0.0
Y	2.8	3.07	2.8	3.07	2.8	3.07
Z	-1.015	-0.925	-1.015	-0.925	2.03	1.85

Tablo 4.3 İleri kinematik problemini çözmek için kullanılan Newton-Raphson metodunun sonuçları

	$X(m)$	$Y(m)$	$Z(m)$	$\phi(rad)$	$\theta(rad)$	$\psi(rad)$
0	0.25	0.25	-0.45	0.07	-1.7	0.07
1	0.1933094	0.3077386	-0.4089530	0.1126940	-1.404225	0.1023974
2	0.2000163	0.2999979	-0.4000039	0.0999466	-1.400029	0.0999672
3	0.2000000	0.3000000	-0.4000000	0.1000000	-1.400000	0.1000000

5. Stewart Platform Mekanizmasının Simülasyon Programlanması

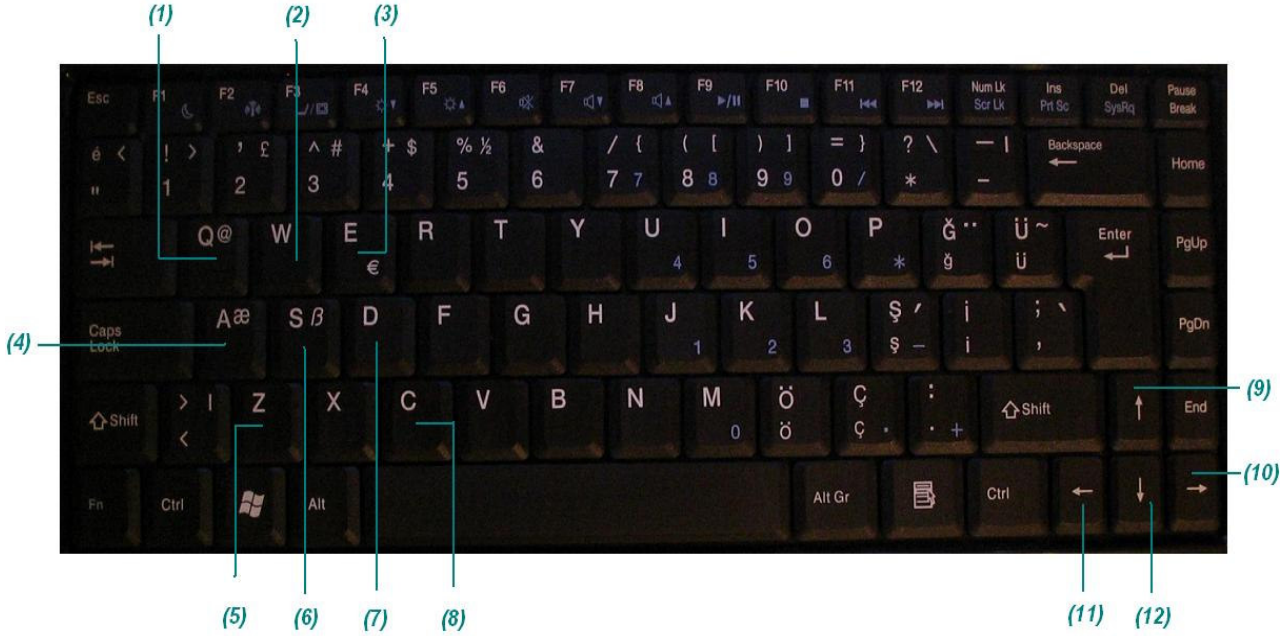
Stewart Platfor mekanizmasının görsel olarak simüle etmek için, Visula C++ programı ile uyumlu çalışma kabiliyeti ve basit kullanımı dolayısı ile 3D EDITOR Version 2 programı kullanılmıştır. 3D EDITOR programı ara hiçbir programa ve çeviriciye ihtiyaç duymadan Visual C++ programından gelen komutlarla çizilen cismi 3 boyutlu dünyasında kareket ettirebilmektedir. Bizde stewart platformun mekanizmasının hareketlerini görsel olarak kavrayabilmek için daha önceden ters ve ileri kinematik hesabını geliştirerek bulunan platform pozisyonlarını 3 boyutlu bir disk ile gösterdik. Bu programın ilk açılış görünümü ve hareketli platformu temsil eden 3 boyutlu diskin duruşu Şekil 5,1'deki gibidir.



Şekil 5.1 Simülasyon programının ilk açılış görüntüsü

Program ilk açıldığında iki pencere halinde ekrana gelir. İlk pencere 3D EDITOR programının komut penceresidir, ikinci pencere ise simülasyonu görebileceğimiz penceredir. Biz komutları doğrudan Visual C++ programından alacağımız için komut penceresine ihtiyaç duymayacağız. Bu yüzden bu pencereyi minimize ederek bizim için arka planda kalması yeterli olacaktır. Programda işimiz bittiği zaman çıkmak için herhangi bir durumda klavyedeki "ESC" tuşuna basmak yeterli olacaktır.

5.1 Simülasyon Programının Klavye İle Kontrolü



Şekil 5.2 Klavye kullanımı

Simülasyon programı, hem klavyeden hem de yönetme kolu ile kontrol edilebilmektedir. Şekil 5.2 de görünen numaralandırılmış klavye tuşları ile Şekil 5.3 de görünen yönetme kolu üzerindeki aynı numaralı tuşlar benzer işlevleri yerine getirmektedir. Bu işlevlerin görsel sonuçları ise Şekil 5.4 de gösterilmiştir.

(3) numaralı işlev, hareketli platformun kendi üzerindeki hareketli eksene göre x etrafında pozitif yöndeki açısal dönüşünü ifade etmektedir.

(1) numaralı işlev, hareketli platformun kendi üzerindeki hareketli eksene göre x etrafında negatif yöndeki açısal dönüşünü ifade etmektedir.

(4) numaralı işlev, hareketli platformun kendi üzerindeki hareketli eksene göre y etrafında pozitif yöndeki açısal dönüşünü ifade etmektedir.

(7) numaralı işlev, hareketli platformun kendi üzerindeki hareketli eksene göre y etrafında negatif yöndeki açısal dönüşünü ifade etmektedir.

(5) numaralı işlev, hareketli platformun kendi üzerindeki hareketli eksene göre z etrafında pozitif yöndeki açısal dönüşünü ifade etmektedir.

(8) numaralı işlev, hareketli platformun kendi üzerindeki hareketli eksene göre z etrafında negatif yöndeki açısal dönüşünü ifade etmektedir.

(10) numaralı işlev, hareketli platformun sabit eksene göre X doğrultusunda pozitif yöndeki ilerlemesini ifade etmektedir.

(11) numaralı işlev, hareketli platformun sabit eksene göre X doğrultusunda negatif yöndeki ilerlemesini ifade etmektedir.

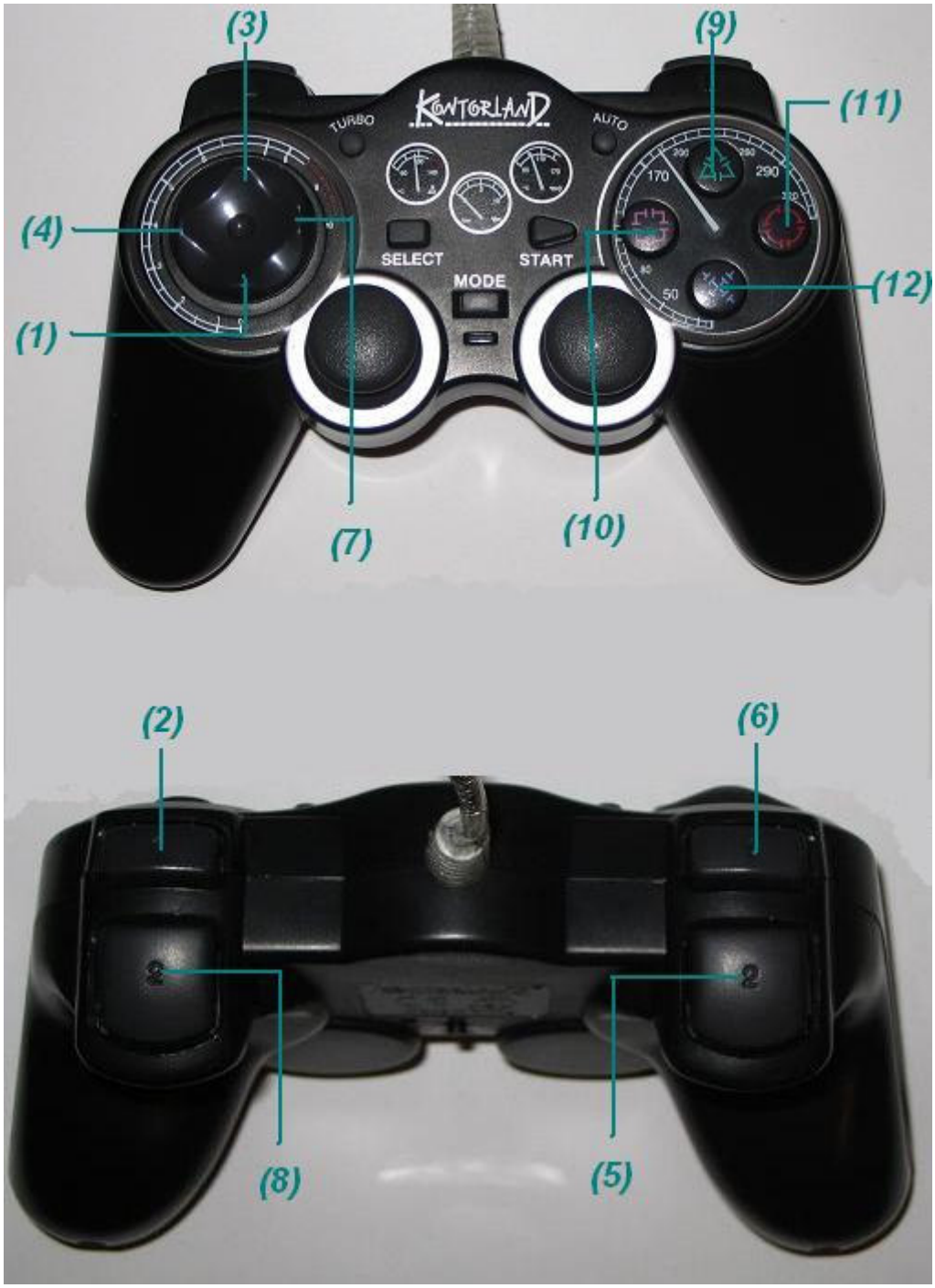
(9) numaralı işlev, hareketli platformun sabit eksene göre Y doğrultusunda pozitif yöndeki ilerlemesini ifade etmektedir.

(12) numaralı işlev, hareketli platformun sabit eksene göre Y doğrultusunda negatif yöndeki ilerlemesini ifade etmektedir.

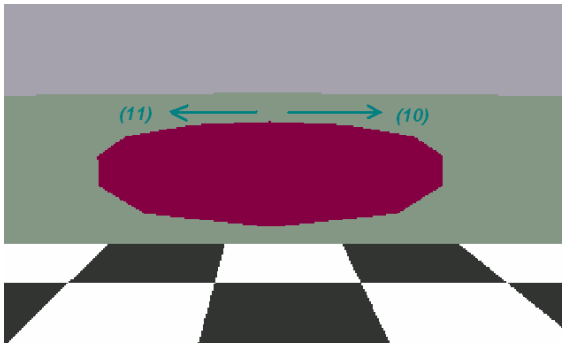
(2) numaralı işlev, hareketli platformun sabit eksene göre Z doğrultusunda pozitif yöndeki ilerlemesini ifade etmektedir.

(6) numaralı işlev, hareketli platformun sabit eksene göre Z doğrultusunda negatif yöndeki ilerlemesini ifade etmektedir.

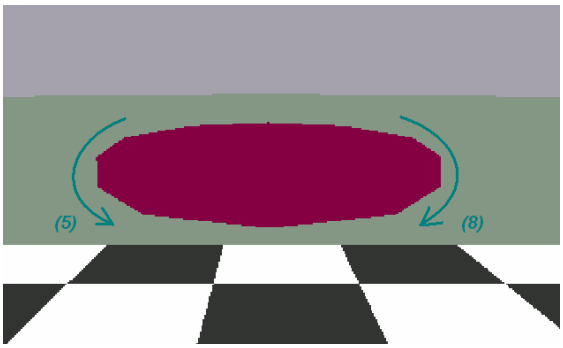
5.2 Simülasyon Programının Joystick İle Kontrolü



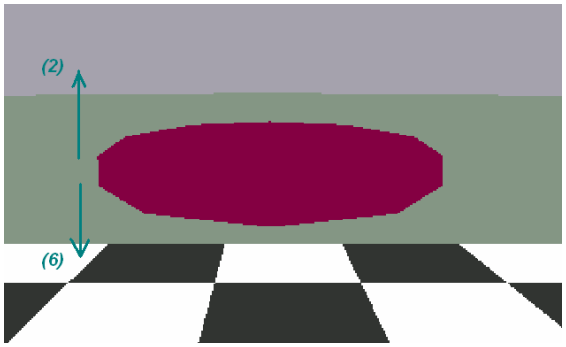
Şekil 5.3 Yönetme kolu kullanımı



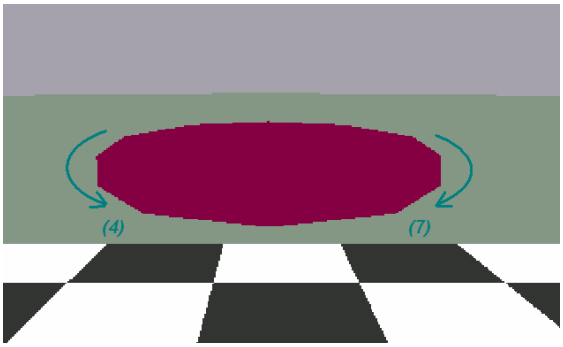
Şekil 5.4-a (10) ve (11) nolu hareket



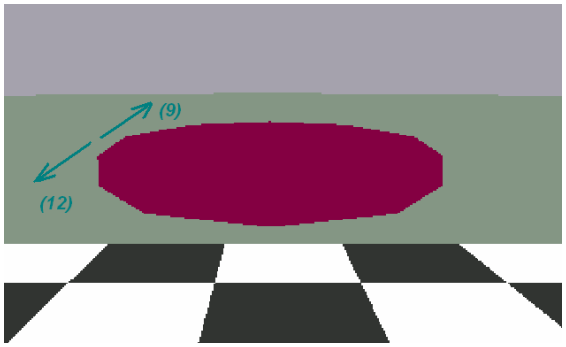
Şekil 5.4-b (5) ve (8) nolu hareket



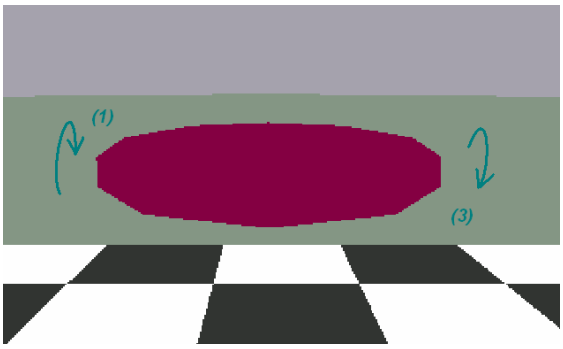
Şekil 5.4-c (2) ve (6) nolu hareket



Şekil 5.4-d (4) ve (7) nolu hareket



Şekil 5.4-e (9) ve (12) nolu hareket



Şekil 5.4-f (1) ve (3) nolu hareket

KAYNAKLAR

1. V.E. Gough and S.G. Whitehall, "Universal Tyre Test Machine", *Proc. 9th International Technical Congress F.I.S.I.T.A.* (1962) pp. 117–137.

2. D. Stewart, "A Platform with Six Degrees of Freedom", *Proceedings of the Institution of Mechanical Engineers*, London **180**, No. 15, 371–386 (1965).

3. K.H. Hunt, *Kinematic Geometry of Mechanisms* (Oxford University, London, 1978).

4. Z. Geng and L.S. Haynes, "Six-Degree-of-Freedom Active Vibration Isolation Using a Stewart Platform Mechanism", *J. Robotic Systems* **10**, No. 5, 725–744 (1993).

5. P. Nanua, K.J. Waldron and V. Murthy, "Direct Kinematic Solution of a Stewart Platform", *IEEE Trans. on Robotics and Automation* **6**, No. 4, 438–443 (1990).

6. W.Q.D. Do and D.C.H. Yang, "Inverse Dynamic Analysis and Simulation of a Platform Type of Robot", *J. Robotic Systems* **5**, No. 3, 209–227 (1988).

7. Z. Ji, "Study of the Effect of Leg Inertia in Stewart Platform", *Proc. of the IEEE Conf. on Robotics and Automation* (1993) **Vol. 1**, pp. 212–226.

8. B. Dasgupta and T.S. Mruthyunjaya, "A Newton-Euler Formulation for the Inverse Dynamics of the Stewart Platform Manipulator", *Mechanism and Machine Theory* **33**, No. 8, 1135–1152 (1998).

9. C.C. Nguyen and F.J. Pooran, "Dynamic Analysis of a 6 DOF CKCM Robot End-Effector for Dual-Arm Telerobot Systems", *Robotics and Autonomous Systems* **5**, 377–394 (1989).

10. Z. Geng, L.S. Haynes, J.D. Lee and R.L. Carroll, "On the Dynamic Model and Kinematic Analysis of a Class of Stewart Platforms", *Robotics and Autonomous Systems* **9**, 237–254 (1992).

11. G. Lebret, K. Liu and F.L. Lewis, "Dynamic Analysis and Control of a Stewart Platform Manipulator", *J. Robotic Systems* **10**, No. 5, 629–655 (1993).

12. O. Ma and J. Angeles, "Archticture Singudarities of Parallel Manipulators", *Int. J. Of Robotics and Automation* **7**, No. 1, 23–29 (1992).

13. M.L. Husty, An Algorithm for Solvmg the Direct Kinematics of General Stewart-Gough Platform, *Mech. Mach. Theory* **31**, No. 4, 365–380, (1996).

14. C.W. Wampler, Forward Displacement Analysis of General Six-in-Parallel SPS (Stewart) Platform Manipulators Using Soma Coordinates, *Mech. Mach. Theory* **31**, No. 3, 331–337 (1996).

15. C.E. Wilson and J.P. Sadler, *Kinematics and Dynamics of Machinery* (Harper Collins College Publishers, New York, NY, 1993).

16. K. Harib, "Dynamic Modeling and Control of Stewart Platform-Based Machine Tools", *Ph.D. Thesis* (Department of Mechanical Engineering, The Ohio State University, Columbus, Ohio, 1997).

17. M.W. Walker and D.E. Orin, "Efficient Dynamic Computer Simulation of Robotic Mechanisms", *J. of Dynamic System, Measurement, and Control* **104**, 205–211 (1982).

18. L.J. Gutkowski, "A General, Robust Procedure for the Kinematic and Friction Force Analysis of Single Loop, One DOF Spatial Mechanism", *Ph.D. Thesis* (Department of Mechanical Engineering, TheOhio State University, Columbus, Ohio, 1990).

19. K. Harib and K. Srinivasan "Kinematic and dynamic analysis of Stewart platform-based machine tool structures" Robotica volume 21, pp. 541–554. (2003)

INTERNET KAYNAKLARI

[1] http://www.cours.polymtl.ca/roboop/

[2] http://www.3dstate.com

EKLER

Ek 1 SPM ters ve ileri kinematik problemi hesabı için yazılan Visual C++ kodu

```
#include <windows.h>
#include "robot.h"
#include "stewart.h"

int stewart_ters(void){
 Real Stewart_Ini[] =
{
   1.758 ,  2.8 ,   -1.015,  0.225 ,   0.0   ,    -0.228,
   1.6021, 3.07,  -0.925,  0.1125,  0.1949, -0.228,
 -1.7580,  2.8 ,  -1.015, -0.1125,  0.1949, -0.228,
 -1.6021,  3.07, -0.925, -0.225 ,   0.0,       -0.228,
   0.0,        2.8 ,   2.03 , -0.1125, -0.1949, -0.228,
   0.0,       3.07,   1.85 ,  0.1125, -0.1949, -0.228
 };
   cout<<"        ==================[ Ters Kinematik ]=================\n\n";
   cout<<" Hareketli Platformun Pozisyonunu Girin\n\n";
   float   X,Y,Z,A,B,C;
   cout<<" X ---> ";cin>>   X;
   cout<<" Y ---> ";cin>>   Y;
   cout<<" Z ---> ";cin>>   Z;
   cout<<" A ---> ";cin>>   A;
   cout<<" B ---> ";cin>>   B;
   cout<<" C ---> ";cin>>   C;

Real Stewart_q[] ={X, Y, Z, A, B, C};

   Matrix InitPlatt(6,6);
   Stewart platt_Stewart;
```

```
 Matrix
     _q(6,1),
      L(6,1);

  InitPlatt<<Stewart_Ini;
  _q << Stewart_q;
  platt_Stewart = Stewart(InitPlatt);
  platt_Stewart.set_q(_q);

//==========TERS KİNEMATİK=====================================
  cout<<"\n";
    L = platt_Stewart.InvPosKine();
  cout<<"\n";
  cout<<"          L1       L2       L3       L4       L5       L6\n";
  cout<<"        -------- -------- -------- -------- -------- --------\n";
  cout<<" Sonuc : "<<L.t()<<endl;
  cout<<"\n";
//============================================================
  return 0;
}

int stewart_ileri(void){
 Real Stewart_Ini[] =
{
      1.758 ,  2.8 ,  -1.015,  0.225 ,   0.0   ,    -0.228,
      1.6021, 3.07, -0.925,  0.1125,  0.1949,  -0.228,
     -1.7580, 2.8 , -1.015, -0.1125,  0.1949,  -0.228,
     -1.6021, 3.07, -0.925, -0.225 ,   0.0   ,    -0.228,
       0.0   ,    2.8 ,   2.03 , -0.1125, -0.1949, -0.228,
       0.0   ,    3.07,  1.85 ,  0.1125,  -0.1949, -0.228
};
```

```
  cout<<"  ==================[ Ileri Kinematik ]================\n\n";
  cout<<" Link Boy Uzunluklarini Girin\n\n";
  float   L1,L2,L3,L4,L5,L6;
  cout<<" L1 ---> ";cin>>   L1;
  cout<<" L2 ---> ";cin>>   L2;
  cout<<" L3 ---> ";cin>>   L3;
  cout<<" L4 ---> ";cin>>   L4;
  cout<<" L5 ---> ";cin>>   L5;
  cout<<" L6 ---> ";cin>>   L6;

Real Stewart_l[] =   {L1,L2,L3,L4,L5,L6};
Real Stewart_q[] =  {0.25, 0.25, -0.45, 0.07, -1.7, 0.07};
Real Stewart_qg[] ={0.25, 0.25, -0.45, 0.07, -1.7, 0.07};

Matrix InitPlatt(6,6);
Stewart platt_Stewart;
Matrix
  _q(6,1),
  qg(6,1),
  _l(6,1),
  L(6,1) ;

 InitPlatt<<Stewart_Ini;
  _q << Stewart_q;
  qg << Stewart_qg;
  _l << Stewart_l;

  platt_Stewart = Stewart(InitPlatt);
  platt_Stewart.set_q(_q);
```

```
//=================== Ileri Kinematik ================================
  cout<<"\n";
platt_Stewart.set_q(qg);
L = platt_Stewart.ForwardKine(qg,_l);
  cout<<"            X       Y        Z       A        B       C\n";
  cout<<"        -------- -------- --------- -------- --------- --------\n";
  cout<<"\n Sonuc : "<<L.t()<<endl;
  cout<<"\n";
//====================================================================
  return 0;
}

 int main(void){
cout<<"                    ==============================\n";
cout<<"                    Stewart Platform Mekanizmasi   \n";
cout<<"                    ==============================\n\n";
int sec;
cout<<" Ters Kinematik Icin   ---> 1\n";
cout<<" Ileri Kinematik Icin  ---> 2\n";
cout<<"\n";
cout<<" >> ";
cin>>sec;
//if (sec==1)
stewart_ters() ;
//if (sec==2)
stewart_ileri();
system("pause");
return 0;
}
```

Ek 2 SPM simülasyanu için yazılan Visual C++ kodu

```
#include "C:\\Program Files\\3DSTATE\\3D Developer Studio 6.0 (for Visual C++)\\Engine\\Include\\3DSTATE.H"
#include <mmsystem.h>
#include <iostream.h>
#include "stdio.h"

void main(void)
{
void MoveCar(DWORD car_handle);
 int rc=STATE_engine_load_world("SP.wld",".","Bitmaps",USER_DEFINED_BEHAVIOR);
if(rc==VR_ERROR) {
cout << "Failed to load world, aborting\n";
cout << "look at error.log to see why\n";
return;
}
printf("Bu pencereyi minumize edin...\n");
printf("\n");
printf("Bitirmek icin<Esc>\n");

STATE_engine_set_default_rendering_window_title
("STEWART PLATFORM SİMÜLASYONU   ÇIKIŞ İÇİN<Esc>");
STATE_engine_maximize_default_rendering_window();
ShowCursor(TRUE);

DWORD camera=STATE_camera_get_default_camera();
DWORD car=STATE_object_get_object_using_name("my_car");
STATE_object_set_direction(car,-1,0,0);
while( (GetAsyncKeyState(VK_ESCAPE)&1) ==0 ) {
MoveCar(car);
```

```
STATE_engine_render(NULL,camera);}
}

void MoveCar(DWORD car_handle)
{
#define BORDER_LOW (32768-4096)
#define BORDER_HIGH (32768+4096)
const int speed = 10;

JOYINFOEX ji;
MMRESULT isJoyPlugged;
ji.dwSize=sizeof(JOYINFOEX);
ji.dwFlags=JOY_RETURNBUTTONS | JOY_RETURNX | JOY_RETURNY;

isJoyPlugged=joyGetPosEx(JOYSTICKID1,&ji);

if(isJoyPlugged==JOYERR_NOERROR){
if(ji.dwButtons & JOY_BUTTON1)
STATE_object_move(car_handle,WORLD_SPACE,-speed,0,0);
}

if(isJoyPlugged==JOYERR_NOERROR){
if (ji.dwButtons & JOY_BUTTON3)
STATE_object_move(car_handle,WORLD_SPACE,speed,0,0);
}
if(isJoyPlugged==JOYERR_NOERROR){
if (ji.dwButtons & JOY_BUTTON2)
STATE_object_move(car_handle,WORLD_SPACE,0,speed,0);
}
if(isJoyPlugged==JOYERR_NOERROR){
if (ji.dwButtons & JOY_BUTTON4)
```

```
STATE_object_move(car_handle,WORLD_SPACE,0,-speed,0);
}
if(isJoyPlugged==JOYERR_NOERROR){
if (ji.dwButtons & JOY_BUTTON5)
STATE_object_move(car_handle,WORLD_SPACE,0,0,-speed);
}
if(isJoyPlugged==JOYERR_NOERROR){
if (ji.dwButtons & JOY_BUTTON6)
STATE_object_move(car_handle,WORLD_SPACE,0,0,speed);
}
if(isJoyPlugged==JOYERR_NOERROR){
if (ji.dwButtons & JOY_BUTTON7)
STATE_object_rotate_z(car_handle,2,OBJECT_SPACE);
}
if(isJoyPlugged==JOYERR_NOERROR){
if (ji.dwButtons & JOY_BUTTON8)
STATE_object_rotate_z(car_handle,-2,OBJECT_SPACE);
}
if(isJoyPlugged==JOYERR_NOERROR){
if(ji.dwXpos<BORDER_LOW)
STATE_object_rotate_x(car_handle,2,OBJECT_SPACE);
}
if(isJoyPlugged==JOYERR_NOERROR){
if(ji.dwXpos>BORDER_HIGH)
STATE_object_rotate_x(car_handle,-2,OBJECT_SPACE);
}
if(isJoyPlugged==JOYERR_NOERROR){
if(ji.dwYpos<BORDER_LOW)
STATE_object_rotate_y(car_handle,-2,OBJECT_SPACE);
}
```

```
if(isJoyPlugged==JOYERR_NOERROR){
if(ji.dwYpos>BORDER_HIGH)
STATE_object_rotate_y(car_handle,2,OBJECT_SPACE);
}
if (GetAsyncKeyState(VK_UP)<0)
STATE_object_move(car_handle,WORLD_SPACE,-speed,0,0);
if (GetAsyncKeyState(VK_DOWN)<0)
STATE_object_move(car_handle,WORLD_SPACE,speed,0,0);
if (GetAsyncKeyState(VK_LEFT)<0)
STATE_object_move(car_handle,WORLD_SPACE,0,-speed,0);
if (GetAsyncKeyState(VK_RIGHT)<0)
STATE_object_move(car_handle,WORLD_SPACE,0,speed,0);
if (GetAsyncKeyState('S')<0)
STATE_object_move(car_handle,WORLD_SPACE,0,0,-speed);
if (GetAsyncKeyState('W')<0)
STATE_object_move(car_handle,WORLD_SPACE,0,0,speed);
if (GetAsyncKeyState('A')<0)
STATE_object_rotate_x(car_handle,5,OBJECT_SPACE);
if (GetAsyncKeyState('D')<0)
STATE_object_rotate_x(car_handle,-5,OBJECT_SPACE);
if (GetAsyncKeyState('Q')<0)
STATE_object_rotate_y(car_handle,5,OBJECT_SPACE);
if (GetAsyncKeyState('E')<0)
STATE_object_rotate_y(car_handle,-5,OBJECT_SPACE);
if (GetAsyncKeyState('Z')<0)
STATE_object_rotate_z(car_handle,5,OBJECT_SPACE);
if (GetAsyncKeyState('C')<0)
STATE_object_rotate_z(car_handle,-5,OBJECT_SPACE);
return;
}
```

ÖZGEÇMİŞ

Doğum tarihi	06.05.1979	
Doğum yeri	Sivas	
Lise	1993–1997	Sivas Kongre Lisesi
Lisans	1998–2003	Dokuz Eylül Üniversitesi Mühendislik Fakültesi Makine Mühendisliği Bölümü

Çalıştığı kurum

2003-Devam ediyor
Mustek Takım Tezgâhları Ltd. Şti.