YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MATLAB KULLANILARAK D.C. MOTOR KONTROLÜNÜN SİMÜLASYONU

Elektrik Müh. Tolga TUTUM

F.B.E. Elektrik Mühendisliği Anabilim Dalında
Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı:Doç.Dr. Hacı BODUR

Doç.Dr. Hacı BODUR Doç. Dr. Selim Akyokuş Prof. R. Gülgün

İSTANBUL, 2000

İÇİNDEKİLER

SİMGE LİSTESİ

A_1	1. Anot
A_2	2. Anot
Udi_α	Doğrultucu ideal çıkış gerilimi
S	Yol sayısı
Q	Faz sayısı
D_x	Doğrultucudaki endüktif gerilim düşümü
D_r	Doğrultucudaki omik gerilim düşümü
D_T	Doğrultucuların iletim durumundaki gerilim düşümleri
R_K	Bir faz koluna ait transformatör sekonder sargısı ve bağlantı iletkenlerinin direnci
L_K	Bir faz koluna ait self endüksiyon katsayısı
F	Şebeke geriliminin frekansı
ΔU	Doğrultucudaki toplam gerilim düşümü
U_a	Endüvi giriş gerilimi
E_a	Ters elektromotor kuvvet
I_a	Endüvi akımı
L_a	Endüvi devresine ait reaktans
R_a	Endüvi devresine ait direnç
U_f	Uyarma gerilimi
I_f	Uyarma akımı
L_f	Uyarma devresine ait reaktans
R_f	Uyarma devresine ait direnç
w	Motor açısal hızı
n	Motor devir sayısı
B	Viskos sürtünme katsayısı
M_y	Yük momenti
M_d	Motor tarafından endüklenen moment
P_g	Endüviye kaynaktan verilen güç
P_{km}	Motorun milinde meydana gelen sürtünme ve vantilasyon kayıpları
P_m	Üretilen mekaniksel güç
P_{ke}	Endüvi bakır kayıpları
$P_ç$	Mil çıkış gücü
k_Φ	Akı sabiti
k_e	Ters elektromotor kuvvet sabiti
k_m	Moment sabiti

ŞEKİL LİSTESİ

ÖZET

Modern çağımızın hızla gelişen teknolojilerinde maliyet ve zaman kavramları ön plana çıkmaya başlamıştır. Her ikisininde azaltılarak verimin arttırılmasına çalışılmıştır. Bunun sonucu olarak birçok simülasyon paketleri geliştirilmiştir.

Bu tezde ilk olarak temel güç elektroniği sistemleri incelenmiştir. Daha sonra serbest uyartımlı DC motorun yapısı incelenmiş, bu motora ait eşitlikler verilmiştir. Üçüncü kısımda güç elektroniği sistemlerinin simülasyonu ve analizinde kullanılan simülatörler incelenmiş ve birbirleriyle mukayese edilmiştir. Bilgisayar simülasyonlarının avantaj ve dezavantajları ifade edilmiştir. Dördüncü kısımda bu tezde kullanılan MATLAB programı tanıtılmış ve güç elektroniği sistemlerine ait uygulama örnekleri verilmiştir. Son kısımda ise serbest uyartımlı DC motor matematiksel olarak modellenmiş ve MATLAB'la simüle edilmiştir. Pratikteki değerlerle simülasyon sonuçları karşılaştırılmıştır.

Anahtar Kelimeler: Güç elektroniği, MATLAB, Simülasyon

ABSTRACT

In the recent technologies of our modern age cost and time concepts have become considirable than ever. It has been tried to increase the level of profit by decreasing both of them. As a result many simulation packages have been improved.

First of all, basic power electronics systems are studied in this article. Then, the structure of direct current motor is examined and the equations belonging to these motor have been given. In the third section the simulators which are used in the simulations and analyzes of power electronics systems are studied. That simulators are compared with eachothers. The advantages and the disadvantages are also explained. In the fourth section software called MATLAB which takes place in this article has been introduced and the practicle examples of power electronics systems are given. In the last section, direct current motors are patterned and simulated with MATLAB. Outputs of the practical sessions are compared with simulation results.

Keywords: Power electronics, MATLAB, Simulation

1. GİRİŞ

Modern çağımızın hızla gelişen teknolojilerinde maliyet ve zaman kavramları ön plana çıkmaya başlamıştır. Her iki kavramında azaltılarak verimin arttırılmasına çalışılmıştır. Bu sebeple uzun yıllardan beri birçok bilgisayar programları ve simülasyon paketleri geliştirilmiştir.

Simülasyonlarla sistem davranışlarındaki parametreleri analiz etmenin, bu işlemi donanımsal olarak labaratuarda yapmaktan daha kolay olduğunun anlaşılmasından sonra, simülasyonlar endüstride toplam etüd işlemlerinin kısaltılmasında kullanılmaya başlamıştır.

Bugün endüstride ve üniversitelerde bir çok devre tabanlı simülatörler ve eşitlik çözücüler kullanılmaktadır. Mühendislik hesaplamalarında kullanılan en eski ve en kapsamlı bilgisayar dili FORTRAN'dır. FORTRAN ile program hazırlamak bir uzmanlık işi olarak kalmakta ve oldukça zaman alıcı olmaktadır. Daha sonradan ortaya çıkan Pascal ve C programlama dillerinde de benzer zorluklar mevcuttur. Günümüzde mühendislik sistemlerinin simülasyonunda bu programların yerine PSPICE, EMTP gibi devre tabanlı simülatörlerle MATLAB, MathCAD, MACSYMA gibi eşitlik çözücüler kullanılmaktadır.

Bu çalışmada kullanılmış olan MATLAB ilk defa 1985'de C.B. Moler tarafından matematik ve özellikle de matris esaslı matematik ortamında kullanılmak üzere geliştirilmiş bir programlama dilidir. İlk sürümleri FORTRAN diliyle yazılmıştır. Son sürümleri ise C diliyle yazılmıştır. MATLAB mühendislik alanında; sayısal hesaplama, veri çözümleri ve grafik işlemlerinde kullanılabilecek genel amaçlı bir program olmakla beraber özel amaçlı paketlere de sahiptir. Aralık 1999 tarihi itibariyle MATLAB'ın son versiyonu MATLAB 5.3.1'dir. MATLAB'ın tüm hakları The Math Works'e aittir.

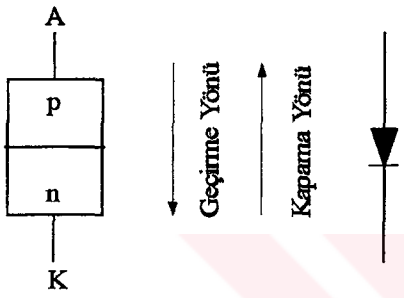
Bu çalışmada ilk olarak serbest uyartımlı DC motorun yapısı, daha sonra da devre tabanlı simülatörler ve eşitlik çözücüler genel olarak incelenmiştir. DC motorun matematiksel modeli oluşturularak MATLAB ile simüle edilmiştir.

2. YARI İLETKEN GÜÇ ELEMANLARI ve TEMEL GÜÇ DÖNÜŞTÜRÜCÜLERİ

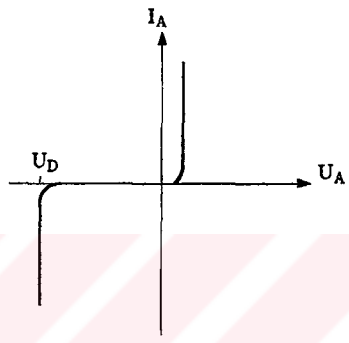
2.1 Yarı İletken Güç Elemanları

2.1.1 Diyot

Diyot, normal yönde akım iletimi ve ters yönde gerilim tutma özelliğine sahip olan yarı iletken elemandır. Şekil 2.1'de diyodun sembolü ve yönleri, Şekil 2.2'de ise diyodun u-i karakteristiği görülmektedir.



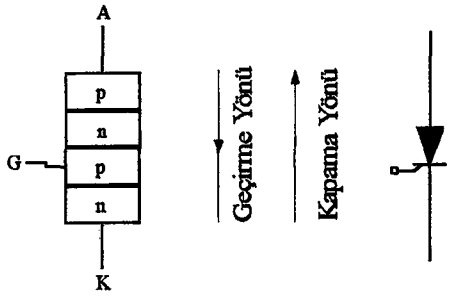
Şekil 2.1 Diyot sembolü ve yönleri



Şekil 2.2 Diyodun u-i karakteristiği

2.1.2 Tristör

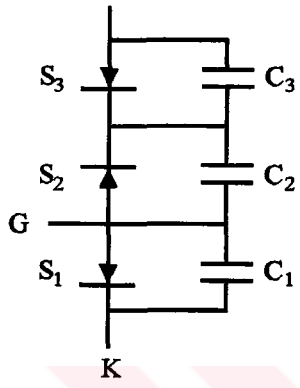
Tristör dört tabakadan oluşmaktadır. Anot, katot ve kapı olarak tanımlanan üç ucu vardır. Kapıdan katoda doğru bir kapı akımı geçirilerek tetiklenir.



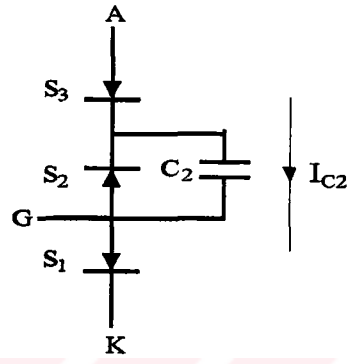
Şekil 2.3 Tristörün sembolü ve yönleri

Tristörler bundan başka kendiliğinden de iletime geçebilirler. Bunun iki nedeni vardır. Şayet anot-katot arası gerilim sıfır devrilme gerilimine erişirse, kapı akımı sıfır olduğu halde tristör iletime geçer. Anot gerilimi dış etkiyle yükselebilir. Ya da devrilme geriliminin değeri sıcaklığın etkisiyle azalabilir. Bunun diğer nedeni de anot gerilimi yükselme hızının belirli bir değere erişmesidir. Jonksiyonlar arasındaki bölgede hareketli yükler azdır ve bu bölge yalıtkandır. Bu da kapasite özelliği gösterir. Yani her jonksiyonun kapasitesi vardır. Bu şartlar

altında tristörü Şekil 2.4'teki gibi modellenebilir. Tristörün iletime geçebilmesi için anot gerilimi positif olmalıdır. Anot-katot arasına bir gerilim uygulayalım. S_3 ve S1 geçirme S2 kapama yönünde kutuplanır. Yani geçen akım C_2 kapasitesi üzerinden geçer. C_1 ve C_3 köprülenir. Bu duruma ait son eşdeğer devre Şekil 2.5'teki gibi olur. Bu I_{C2} akımı U_A'nın ne kadar hızlı uyguladığına bağlıdır. Bu I_{C2} akımı G ile K arasındanda aktığından tristörü tetikleyebilir.



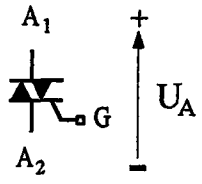
Şekil 2.4 Tristörün jonksiyon modeli



Şekil 2.5 C_1 ve C_3'ün köprülenmesi durumu

2.1.2.1 Triac

Triac, akımı her iki yönde de geçirebilen tristörlerdir. Ters paralel bağlanmış iki tristör gibi davranırlar. Şekil 2.6'da triac'ın sembolü verilmiştir. Akım iki yönde geçtiği için ana uçlar, anot ve katot yerine 1. Anot (A_1) ve 2. Anot (A2) olarak adlandırılırlar. İletime girebilmeleri için kapıdan A_2'ye veya A_2'den kapıya doğru bir kapı akımının geçirilmesi

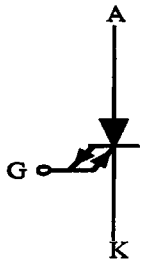


Şekil 2.6 Triac'ın sembolü

gerekir. Triac'larda kapı akımına hassasiyet tristörlere göre daha azdır. Yani tetiklenmeleri için daha büyük bir kapı akımına ihtiyaç duyarlar. 400 Hz'den küçük frekanslarda kullanılırlar. Max. gerilimleri 1000V, akımları ise 50-100A arasındadır.

2.1.2.2 Kapıdan sönen tristör (GTO)

GTO Tristör, normal tristör gibi bir tek kapı akımı darbesi ile iletime geçerilebilir. Ayrıca kapı devresine negatif bir akım darbesi uygulanarak iletimden çıkarılabilme özelliğine de sahiptir. Bu elemanın sembolü Şekil 2.7'de görülmektedir. Güç elektroniğinde zorlamalı

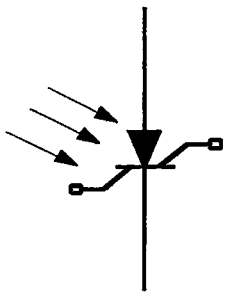


Şekil 2.7 GTO'nun sembolü

komütasyonların gerekli olduğu uygulamalar için tercih edilir. Çok hızlı tristörlerdir. Diğer taraftan GTO'nun iletimden çıkma kazancı düşüktür. Bu nedenle iletimden çıkarılabilmesi için oldukça yüksek bir negatif akım darbesinin uygulanması gerekir.

2.1.2.3 Foto tristör

Foto tristörler, silisyumun doğrudan ışıkla reaksiyonu vasıtasıyla iletime geçerler. Radyasyonun etkisi ile oluşan elektron-delik çiftleri, elektrik alanın etkisi altında tetikleme akımını üretir. Foto tristörler yüksek gerilimli ve büyük akımlı uygulamalarda kullanılırlar.



Şekil 2.8 Dört uçlu foto tristörün sembolü

Büyük güçlü foto tristörlerin yanında kontrol devreleri için geliştirilmiş çok küçük akım ve gerilimli olanları da vardır. Dört uçlu bir foto tristörün sembolü Şekil 2.8'de görülmektedir.

2.1.2.4 Statik endüksiyon tristörü (SITH)

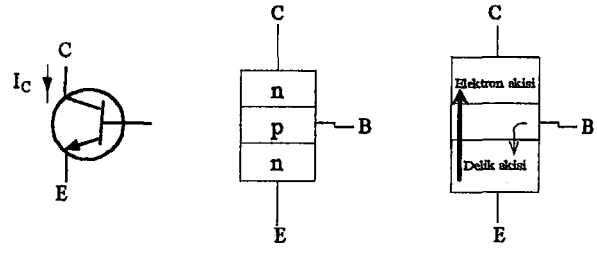
SITH normalde iletimde olan bir elemandır. Yani anot gerilimi positif yapılır ve kapı ucu açık bırakılırsa, eleman tıpkı diyot gibi davranır. Aslında bu durumda iç yapı nedeniyle kapı gerilimi bir miktar positiftir. Anot akımı serbestce akar. Eğer kapıya katoda göre negatif bir gerilim uygulanırsa anot akımı kesilir. SITH'in negatif anot gerilimlerini tutma özelliği yoktur. Bu yapı elemanın yüksek hızda çalısması için gereklidir. İletimden çıkmadaki davranışı GTO'ya benzer. Negatif kapı akımı büyüktür ve anot devresinden bir kuyruk akımı geçer.

2.1.2.5 MOS kontrollü tristör (MCT)

Anot, katot ve kapıdan oluşan üç uçlu tristörlerdir. İletime girebilmesi için, kapısına anadona göre negatif bir gerilim uygulanır. MCT bir kez iletime geçtikten sonra iletimden çıkması için, ya anot akımının kendiliğinden azalarak sıfıra düşmesi veya kapısına pozitif bir gerilim uygulanması gerekir. Davranış bakımından GTO'ya benzer. Ancak iletimden çıkmadaki kazancı çok yüksektir.

2.1.3 Bipolar jonksiyon transistör (BJT)

pnp ve npn olmak üzere iki tür transistör mevcuttur. Güç elektroniği devrelerinde transistörlerden anahtarlama elamanı olarak yararlanılır. npn tipi transistörün sembolü ve



Şekil 2.9 npn tipi transistörün sembolü ve jonksiyonları ile elektron ve deliklerin iki jonksiyon arasındaki geçişleri

jonksiyonları ile jonksiyonlar arası elektron ve deliklerin akışı Şekil 2.9'da görüldügü gibidir. Görüldüğü gibi transistörde iki jonksiyon vardır. Bunlar emiter-baz ve kollektör-baz jonksiyonlarıdır. Transistörü arka arkaya bağlı iki diyot şeklinde kabul ederek, transistörün çalışmasını açıklamak daha kolaydır. Normal çalışmada emiter-baz jonksiyonu iletim yönünde kutuplanır. Bu gerilim küçük olmasına rağmen, jonksiyondan iletim yönünde akan akım büyüktür. Normal çalışmada kollektör-baz jonksiyonu ise tıkama yönünde kutuplanır.

Bundan dolayı, bazdaki çoğunluk taşıyıcılar (delikler) yalnız emiter bölgesine geçebilirler. Bazdaki elektronlar ise kollektör-baz jonksiyonunda yığılırlar.

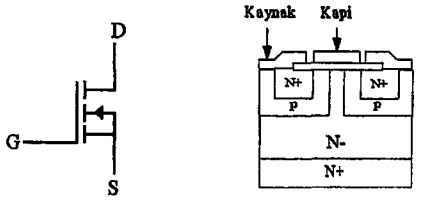
Güç elektroniğinde BJT'ler küçük ve orta güçlü uygulamalarda kullanılmaktadır. Kollektör akımının geçebilmesi için taban devresinden sürekli akım geçirilmesi gerektiğinden, bu akım vasıtasıyla akım sınırlayıcı koruma devreleri gerçekleştirilebilir. Transistörler ters gerilim tutma özelliğine sahip olmadığından, kullanma alanları genellikle dc ile beslenen güç elektroniği sistemleri ile sınırlıdır.

2.1.4 Metal oksit alan etkili transistör (MOSFET)

P ve n kanallı olabilen güç Mosfetleri gerilim kontrollüdür. Azınlıkta hareketli yükler bulunmadığı ve akım çoğunluktaki hareketli yükler tarafından geçirildiği için, bipolar güç transistörlerine göre daha hızlıdır. Bu nedenle MOSFET'ler güç yarı iletkeni olarak, düşük güçlü ve yüksek frekanslı uygulamalarda yaygın olarak kullanılmaktadır.

Kapılarına gerilim uygulanmadığı sürece kesimdedirler. MOSFET'in iletime geçebilmesi için, örneğin n kanallıda kapıya yeterli düzeyde bir positif gerilim uygulanmalıdır. MOSFET'in iletime geçmeye başladığı gerilime "Eşik Gerilimi" adı verilir.

Büyük akımlı MOSFET'lerin mutlaka kısa kanallı olarak gerçekleitirilmesi gerekir. Muhtelif yapılarda kısa kanallı MOSFET'ler üretilmiştir. Kapama geriliminin yükseltilmesinin yanında, iletimdeki direncin düşük tutulması amaçlanmıştır. Şekil 2.10'da büyük gerilimli böyle bir n kanallı MOSFET'in sembolü ve kesiti görülmektedir.

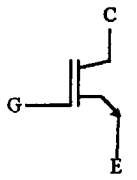


Şekil 2.10 n kanallı MOSFET'in sembolü ve kesiti

Kapıya (G), kaynak (Source, S) ucuna göre pozitif bir gerilim uygulanması ile bir n kanal oluşur ve S'den kanal (Drain, D) ucuna doğru bu kanal üzerinden bir elektron akışı meydana gelir. Tabiatıyla bu elektron akışının sağlanması için D ile S arasına pozitif bir U_{DS} gerilimi uygulanmalıdır.

2.1.5 İzole kapılı bipolar transistör (IGBT)

Gerilim kontrollu bir eleman olan İzole Kapılı bipolar Transistörün (IGBT) sembolü Şekil 2.11'de görülmektedir. BJT'nin ve MOSFET'in iyi taraflarının IGBT'de bir araya getirilmesi amaçlanmıştır. Yapıdaki MOSFET nedeniyle giriş empedansı yüksek, BJT dolayısıyla da iletimdeki gerilim düşümü küçüktür. Anahtarlama hızları MOSFET'ten düşük, BJT'den yüksektir.



Şekil 2.11 IGBT'nin sembolü

Yapısı MOSFET'in yapısına benzer. Aradaki fark burada kanal'daki N^+ tabakasının yerini, kollektördeki P^+ tabakasının almasıdır.

2.1.6 Yarı iletken güç elemanları karşılaştırma tablosu

Yarı iletken güç elemanlarının iyiden kötüye doğru sıralaması Çizelge 2.1'deki gibidir

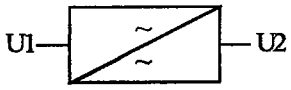
Çizelge 2.1 Yarı iletken elemanların karşılaştırılması

İletimdeki gerilim düşümleri	BJT	SCR	GTO	MCT	IGBT	MOSFET
Anahtarlama hızı	MOSFET	IGBT	MCT	BJT	GTO	SCR
Anahtarlama güç kaybı	MOSFET	IGBT	MCT	GTO	SCR	BJT
Sürme kolaylığı	MOSFET	IGBT	MCT	SCR	GTO	BJT
Akım dayanımı	SCR	GTO	MCT	IGBT	BJT	MOSFET
Gerilim dayanımı	SCR	GTO	MCT	IGBT	BJT	MOSFET

2.2 Temel Güç Elektroniği Devreleri

2.2.1 AC kıyıcılar

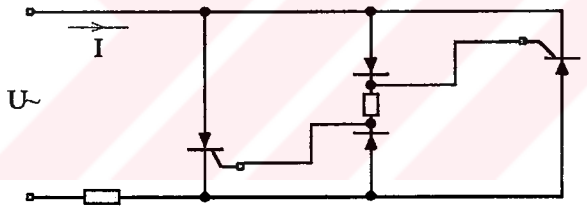
Bu tip devreler girişteki gerilim değerini düşürerek çıkışa yansıtan devrelerdir. Doğal komütasyonlu olup tristörlerle gerçekleştirilirler. Bütün omik yüklerde, vantilatör karakteristikli düşük güçlü AC motorların kontrolünde, gerilim regülatörlerinde, elektronik şalterlerde ve reaktif güç kompanzasyonunda kullanılırlar. AC kıyıcının blok diyagramı şekil 2.12'de görüldüğü gibidir.



Şekil 2.12 AC kıyıcı blok diyagramı

2.2.1.1 Tek fazlı AC kıyıcılar

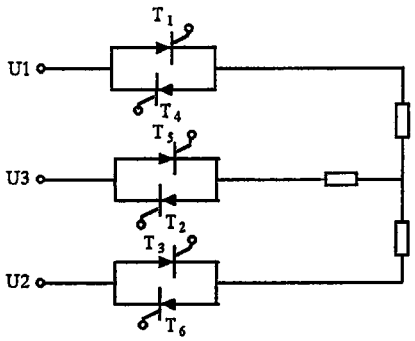
Tek fazlı bir AC kıyıcı devresi Şekil 2.13'de görüldüğü gibidir. Bunun dışında triacla veya diyot köprüsünün çıkışına tristör bağlanarak da gerçekleştirilebilir.



Şekil 2.13 Tek fazlı bir AC kıyıcı

2.2.1.2 Üç fazlı AC kıyıcılar

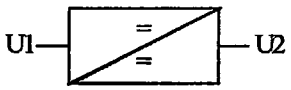
Yıldız noktası nötr noktası ile irtibatlı olan üç fazlı AC kıyıcı, üç adet tek fazlı AC kıyıcıya eşdeğerdir. Yükün üçgen bağlı olması halinde her bir kol fazlararası gerilimle fakat bağımsız tek fazlı gibi çalışır.



Şekil2.14 Yıldız bağlı biryükü besleyen AC kıyıcı devresi

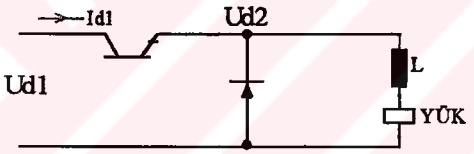
2.2.2 DC kıyıcılar

Zorlamalı komütasyonlu devrelerdir. Bu tür devrelerde genel olarak orta güç ve orta frekanslarda transistör, düşük güç ve yüksek frekanslarda mosfet, yüksek güç ve düşük frekanslarda tristör kullanılır. IGBT ise orta ve yüksek güç, orta ve yüksek frekanslarda kullanılır. DC motor hız kontrolü, anahtarlamalı güç kaynakları, akü şarjı, DC gerilim regülatörleri, DC kaynak makinalarında kullanılırlar. DC kıyıcının blok diyagramı Şekil 2.15'te görüldüğü gibidir.

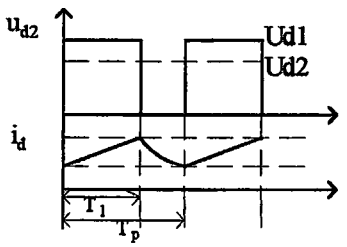


Şekil 2.15 DC kıyıcı blok diyagramı

Temel DC kıyıcı devresi ile çıkış akım ve geriliminin değişimleri Şekil 2.16'da ve Şekil 2.17'de görülmektedir.



Şekil 2.16 Temel DC kıyıcı devresi



Şekil 2.17 DC kıyıcı ile ilgili değişimler

Burada T_1 anahtarlama elemanın iletimde kaldığı süredir. Bu durumda bağıl iletim süresi,

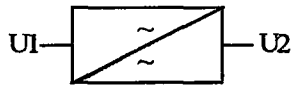
$\lambda=T_1/T$ (2.1)

Böylece çıkış gerilimi Ud_2, $Ud_2=\lambda.Ud_1$ (2.2)

olur.

2.2.3 Doğrultucular

Doğal komütasyonlu devrelerdir. Diyot ve tristörlerle gerçekleştirilirler. DC motor kontrolü, akümülatör şarjı, DC gerilim kaynakları ve galvoteknikle kaplama alanlarında kullanılırlar. Doğrultucunun blok diyagramı Şekil 2.18'de görüldüğü gibidir.



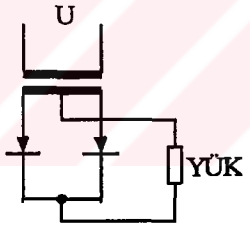
Şekil 2.18 Doğrultucu blok diyagramı

2.2.3.1 Kontrolsüz doğrultucular

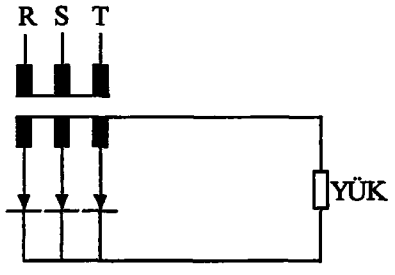
Doğrultma elemanı olarak diyot kullanılır. Çıkış gerilimleri sabittir. Doğrultucunun ideal çıkış gerilimi,

$$U_{d_I}=s.(q/\pi).\sqrt{2}.U.\sin(\pi/q) \qquad (2.3)$$

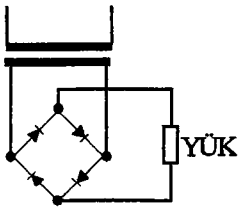
Burada s yol sayısını, q faz sayısını, U ise faz gerilimini ifade eder. Diyotlarla gerçekleştirilen kontrolsüz doğrultucu modelleri Şekil 2.19, 2.20, 2.21, 2.22'de görülmektedir.



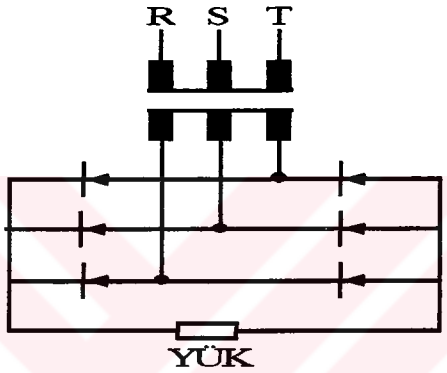
Şekil 2.19 İki fazlı yarım dalga kontrolsüz doğrultucu



Şekil 2.20 Üç fazlı yarım dalga kontrolsüz doğrultucu



Şekil 2.21 Tek fazlı tam dalga kontrolsüz doğrultucu



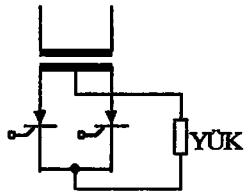
Şekil 2.22 Üç fazlı tam dalga kontrolsüz doğrultucu

2.2.3.2 Kontrollü doğrultucular

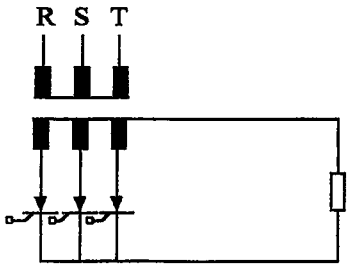
Doğrultma elemanı olarak tristörler kullanılırlar. Çıkış gerilimi sıfır ile Udi arasında istenen değere ayarlanabilir. Çıkış gerilimi 2.4 eşitliğindeki gibi hesaplanabilir.

$$Udi_{\alpha}=S.(q/\pi).\sqrt{2}.U.\sin(\pi/q).\cos\alpha \qquad (2.4)$$

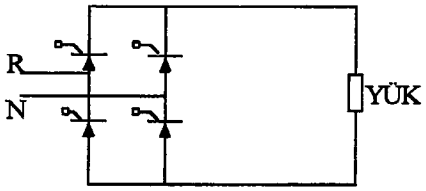
Tristörlerle gerçekleştirilen kontrollü doğrultucu örnekleri Şekil 2.23, 2.24, 2.25, 2.26'te verilmiştir.



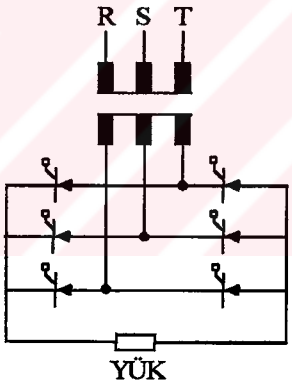
Şekil 2.23 İki fazlı yarım dalga kontrollü doğrultucu



Şekil 2.24 Üç fazlı yarım dalga kontrollü doğrultucu



Şekil 2.25 Tek fazlı tam dalga kontrollü doğrultucu



Şekil 2.26 Üç fazlı tam dalga kontrollü doğrultucu

2.2.3.3 Doğrultuculardaki gerilim düşümleri

Çalışma esnasında hem kontrollü hemde kontrolsüz doğrultucularda ideal çıkış gerilimi doğrudan yük üzerindeki gerilime eşit değildir. Azalmaya sebep olan gerilim düşümleri üç kısımdan oluşmaktadır.

Endüktif gerilim düşümü;

$$D_x=S.f.L_K.Id.q \qquad (2.5)$$

Omik gerilim düşümü;

$$D_r=S.R_K.Id \quad (2.6)$$

Doğrultucuların iletim durumundaki gerilim düşümleri;

$$D_T=S.U_T \quad (2.7)$$

R_K:Bir faz koluna ait transformatör sekonder sargısı ve bağlantı iletkenlerinin direnci
L_K:Bir faz koluna ait self endüksiyon katsayısı
f:Şebeke geriliminin frekansı
Id:DC yük akımı
U_T:Bir doğrultucu elemanın iletim durumundaki gerilim düşümü
Bu durumda toplam gerilim düşümü;

$$\Delta U= D_x+ D_r+ D_T \quad (2.8)$$

Yük uçlarındaki gerilim kontrolsüz doğrultucuda;

$$U_{d\alpha}=Udi-\Delta U \quad (2.9)$$

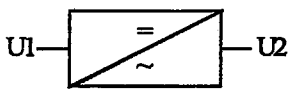
Kontrollü doğrultucuda ise;

$$U_{d\alpha}= U_{di\alpha}- \Delta U \quad (2.10)$$

olur.

2.2.4 İnverterler

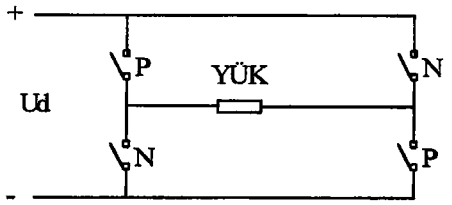
Genel olarak zorlamalı komütasyonludur. Çıkış gerilimi kare dalga, basamaklı kare dalga, sinüsoidal, PWM kontrollü olabilir. Rezonans devreli inverterler doğal komütasyonludur. Yüksek güç ve frekanslarda tristörlü olarak gerçekleştirilirler. İnverterin blok diyagramı Şekil 2.27'da gösterildiği gibidir.



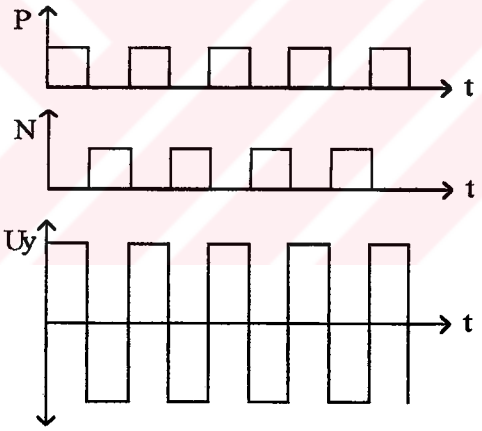
Şekil 2.27 İnverter blok diyagramı

AC motor hız kontrolü, endüksiyonla ısıtma, AC gerilim regülatörleri, UPS ve yüksek gerilim DC taşımada kullanılırlar.

Temel olarak bir inverterin çalışma prensibi Şekil 2.28'deki gibidir. Şekil 2.29'da da görüldüğü gibi ilk yarım peryotta P anahtarlarının kapatıldığını düşünüldüğünde yükün sağ ucu (-) baraya, sol ucuda (+) baraya bağlanır ve yük uçlarında Ud gerilimi oluşur. İkinci yarım peryotta P anahtarlarının açılıp N anahtarlarının kapatıldığı düşünülürse, bu durumda yükün sağ ucu (+) baraya, sol ucu (-) baraya bağlanır ve yük uçlarında (-Ud) gerilimi oluşur.



Şekil 2.28 Temel bir inverter modeli



Şekil 2.29 Temel inverter devresindeki anahtarlama süreleri ve yük geriliminin değişimi

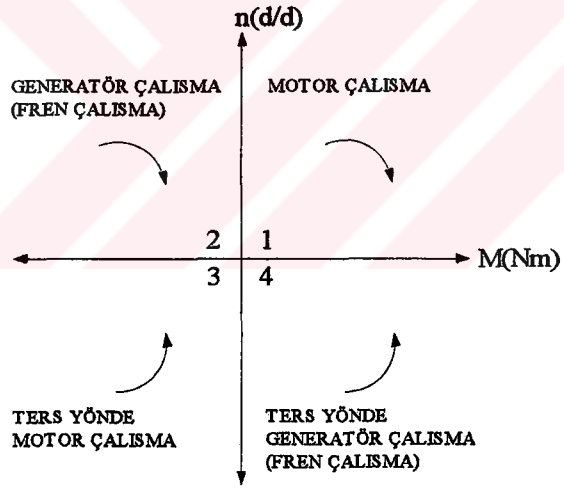
3. DC MOTORUN YAPISI VE KONTROLÜ

3.1 Giriş

Kağıt fabrikalarında, haddehanelerde, çimento fabrikalarında, takım tezgahlarında, tekstil sanayinde ve daha birçok yerlerde hızı iki yönde ayarlanabilen ve belli bir doğrultuda sabit tutulabilen motorlara ihtiyaç vardır. Bu maksatla kullanılan motorların başında doğru akım motorları gelir.

Doğru akım motorlarının en önemli tercih nedenleri mükemmel moment özellikleri ve kontrol karakteristikleridir. Tek önemli dezavantajları ise fırça ve kollektöre sahip olmalarıdır. Kollektör, motor hızını ve gücünü sınırlar, ataletini ve eksenel uzunluğunu artırır ayrıca peryodik bakım gerektirir.

Doğru akım motorlarının hız kontrolünde dört çalışma bölgesi tanımlanır. Bu tanımlamalar Şekil 3.1'de gösterilmiştir.



Şekil 3.1 Doğru akım motorlarının çalışma bölgeleri

Doğru akım motorları yapılarına göre dört ana gruba ayrılırlar.

1)Sebest uyartımlı doğru akım motorları

2)Şönt uyartımlı doğru akım motorları

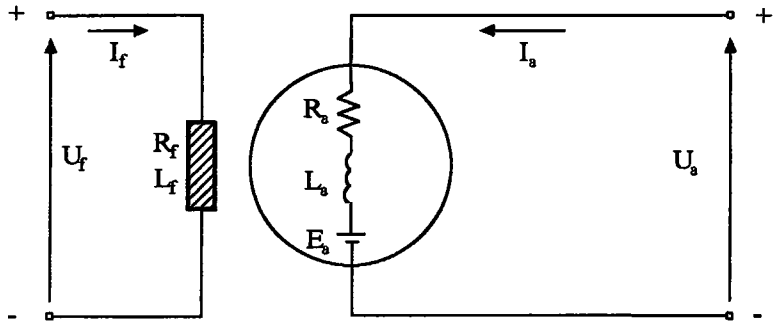
3)Seri uyartımlı doğru akım motorları

4)Kompund doğru akım motorları

Endüstride en çok kullanılan ve hız kontrolü için en uygun yapıya sahip olan doğru akım motor türü serbest uyartımlı doğru akım motorudur.

3.2Serbest Uyartımlı Doğru Akım Motorunun Eşdeğer Devresi ve Karakteristikleri

Serbest uyartımlı bir doğru akım motoru Şekil 3.2'deki gibi bir eşdeğer devre ile ifade edilir.



Şekil 3.2 Serbest uyartımlı doğru akım motorunun eşdeğer devresi

Serbest uyartımlı bir doğru akım motorunun uyarma sargılarından I_f uyarma akımı, rotor devresinden de I_a endüi akımı geçer. Motor çalışma esnasında hıza bağlı bir zıt emk ve yük momentini dengelemek için de milinde endüi akımına bağlı bir mil momenti oluşturur. Serbest uyartımlı doğru akım motorlarında uyartım akımı I_f, endüi akımı I_a'dan tamamen bağımsızdır. Bundan dolayı endüvi devresindeki herhangi bir değişiklik uyarma akımını etkilemez. Uyarma akımı, endüvi akımından çok küçüktür.Eşdeğer devreye ait geçici durum denklemleri aşağıdaki gibi ifade edilebilir.

$$u_f = R_f.i_f + L_f.\frac{di_f}{dt} \tag{3.1}$$

$$u_a = R_a.i_a + L_a.\frac{di_a}{dt} + e \tag{3.2}$$

$$\phi = k_\phi.i_f \tag{3.3}$$

$$e = k_e.\phi.n \tag{3.4}$$

$$e = k_\phi.k_e.\frac{60}{2\pi}i_f.\omega = k_e^{'}.i_f.\omega \tag{3.5}$$

$$m_d = k_m.\phi.i_a = J.\frac{d\omega}{dt} + B.\omega - m_y \tag{3.6}$$

$$m_d = k_m.\phi.i_a = k_m.k_\phi.i_f.i_a = k_m^{'}.i_f.i_a = J.\frac{d\omega}{dt} + B.\omega - m_y \tag{3.7}$$

$$k_e^{'} = k_m^{'} \tag{3.8}$$

Geçici rejim sona erip de kararlı hal başladıktan sonra, yukarıdaki eşitliklerde geçen zamana bağlı ifadeler sıfır olur. Bundan dolayı kararlı hal için aşağıdaki eşitlikler yazılabilir.

$$U_f = R_f . I_f \tag{3.9}$$

$$Ua = R_a . I_a + E \tag{3.10}$$

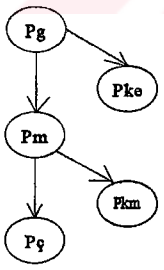
$$\phi = k_\phi . I_f \tag{3.11}$$

$$E = k_e^{'} . I_f . n \tag{3.12}$$

$$M_d = k_m^{'} . I_f . I_a \tag{3.13}$$

Yukarıdaki eşitliklerden de görüldüğü üzere serbest uyartımlı doğru akım motorunun devir sayısı,
Endüi gerilimi U_a kontrol edilerek (Gerilim Kontrolü)
Uyarma akımı I_f kontrol edilerek (Alan Kontrolü)
Belli bir I_f uyarma akımına karşılık gelen I_a endüi akımının oluşturacağı moment (Md) kontrol edilerek (Moment Kontrolü)
şeklinde üç değişik biçimde yapılabilir.

Serbest uyartımlı doğru akım motorunun güç akış diyagramı Şekil 3.3'deki gibidir.



Şekil 3.3.Serbest uyartımlı doğru akım motorunun güç akış diyagramı

Burda P_g giriş gücünü yani endüviye kaynaktan verilen elektriksel gücü, P_{km} motorun milinde meydana gelen sürtünme ve vantilasyon kayıplarını, P_{ke} endüvi bakır kayıplarını, P_m üretilen mekaniksel güç, $P_ç$ ise çıkış mil gücünü ifade eder.Bu güçler matematiksel olarak ifade edilirse,

$$P_g = U_a . I_a \tag{3.14}$$

$$P_{ke} = R_a . I_a^{\,2} \tag{3.15}$$

$$P_{km} = P_{sür} + P_{van} \tag{3.16}$$

$$P_m = P_g - P_{ke} \tag{3.17}$$

Sürtünme kayıpları ihmal edilirse,

$$P_Ç = P_m \tag{3.18}$$

yazılabilir.

Endüvide üretilen mekaniksel güç aşağıdaki gibi ifade edilebilir.

$$P_m = E.I_a \tag{3.19}$$

$P_ç$, yani çıkış gücü mekaniki bir ifadedir. Bu da aşağıdaki gibi ifade edilir.

$$P_ç = \omega M \tag{3.20}$$

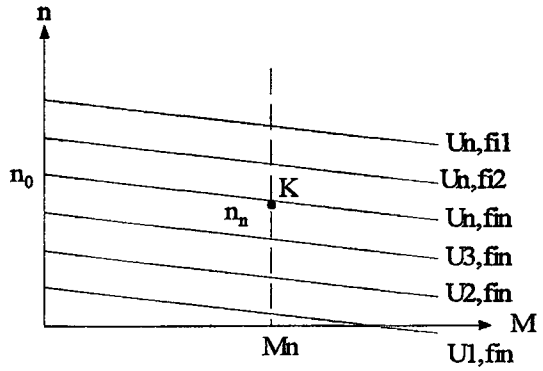
Sürtünme ve vantilasyon kayıpları sıfır kabul edildiğinde P_m ve $P_ç$ birbirlerine eşit olur. Böylece,

$$E.I_a = \omega M \tag{3.21}$$

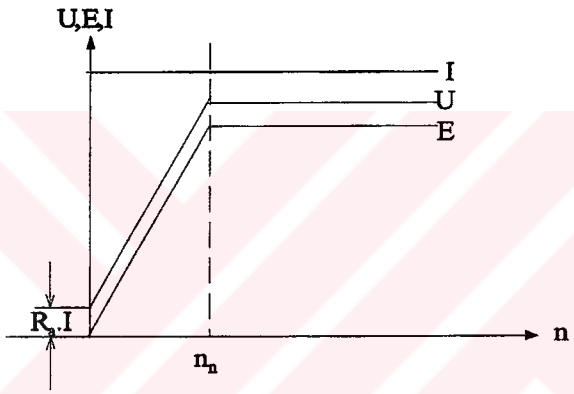
yazılabilir.

3.3 Hız Kontrol Metodları ve Değişimleri

Serbest uyartımlı doğru akım motorunun hız kontrol değişimleri daha önce bölüm 3.2'de de belirtildiği gibi üç yolla yapılır. Sabit uyarma akımı altında endüvi giriş gerilimi değiştirilerek yapılan kontrolde endüi giriş gerilimi arttıkça devir sayısı da artar. Sabit endüvi giriş gerilimi altında uyarma akımı degiştirilerek yapılan kontrolde uyarma akımı azaldıkça devir sayısı artar. Bu yöntemle devir sayısı nominal devrin üzerine çıkarılabilir.



Şekil 3.4 Endüvi gerilimi ve akıya bağlı hız-moment karakteristiği



Şekil 3.5 Hıza göre gerilim, akım, ters emk karakteristiği

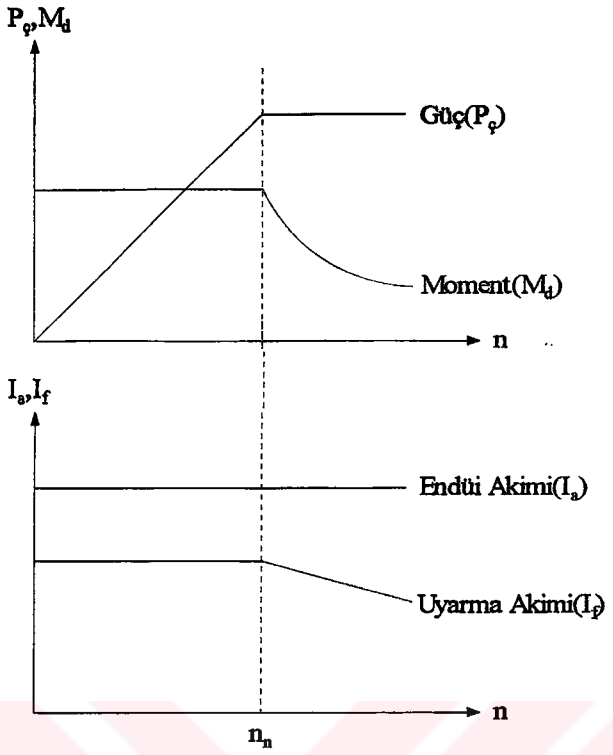
Şekil 3.4'teki K noktası karakteristik noktadır.

3.4 Serbest Uyartımlı Doğru Akım Motorlarında Hız Kontrol Düzenleri

Güç elektroniğindeki gelişmelere paralel olarak günümüzde doğru akım motorlarının çok hassas olarak kontrollerinin yapılması mümkündür. Genellikle doğru akım motorları nominal devir sayıları ve bunun altındaki değerlerde kontrol edilir. Dolayısıyla endüvi gerilimi (U_a) ile hız kontrolü en çok uygulanan metottur.

Pratikte gerilim alternatif akım şebekesinden doğrultucular vasıtasıyla elde edilir. Değişken doğru gerilim ise, doğrudan alternatif akım şebekesinden kontrollü doğrultucuyla ya da kontrolsüz doğrultucu çıkışında doğru gerilim kıyıcısı kullanılarak elde edilir.

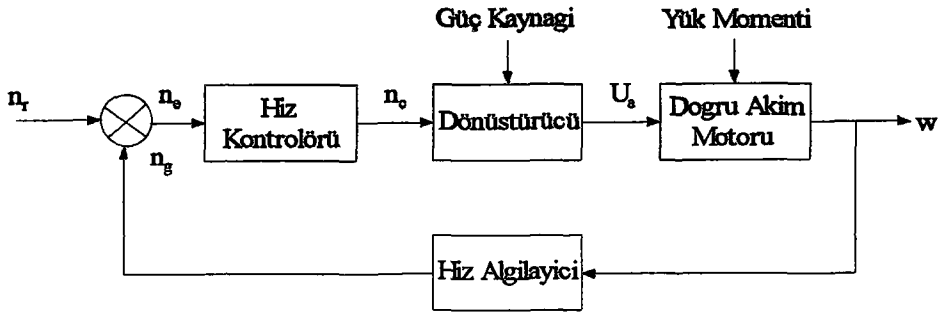
Serbest uyartımlı doğru akım motorlarının hızı yük momentiyle değişir. Belirli bir hızı sürekli olarak korumak için endüvi ya da uyarma devresi gerilimi sürekli olarak ayarlanmalıdır.



Şekil 3.6 Serbest uyartımlı doğru akım motorunda güç ve moment ile uyarma ve endüvi akımlarının devir sayısına bağlı değişimleri.

Serbest uyartımlı doğru akım motoruna ait güç, moment, uyarma ve endüvi akımlarının devir sayısına bağlı olan değişimleri Şekil 3.6'da görülmektedir.

Serbest uyartımlı doğru akım motorlarının hızı yük momentiyle de değişir. Belirli bir hızı sürekli olarak korumak için endüi yada uyarma devresi gerilimi sürekli olarak ayarlanmalıdır. Pratikte motorun sabit bir moment ya da güç ile çalışmasına ilave olarak kontrollü bir hızlanma ve yavaşlama da istenebilir. Endüstride kullanılan serbest uyartımlı doğru akım motor sürücülerinin büyük bir kısmı kapalı çevrim kontrol sistemleri içermektedir. Kolay gerçeklenme, hızlı dinamik cevap, yükten kaynaklanan etkileşimlerin ve sistemden kaynaklanan lineer olmayan etkilerin azaltılması kapalı çevrim transfer fonksiyonunun avantajları arasında sayılabilir. Serbest uyartımlı bir dc motor sürücüsüne ait kapalı çevrim blok diyagramı Şekil 3.7'de gösterilmiştir.

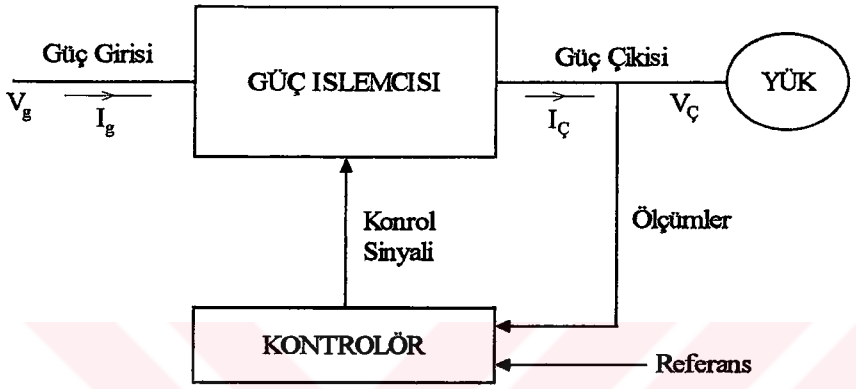


Şekil 3.7 Serbest uyartımlı bir doğru akım motor sürücüsüne ait kapalı çevrim kontrol sistemi blok diyagramı.

Motor hızının yük momentinin artması nedeniyle düşmesi durumunda n_e hata değeri artar. Bu durumda hız kontrolörü, n_c kontrol sinyalinin değerini arttırır. Bu da doğrultucu veya dc kıyıcının iletim sürelerinin değişmesine neden olur. İletim sürelerinin değişmesi ile de endüi devresi gerilimi arttırılmış olur. Endüi gerilimi artması, motor milinde oluşan yeni yük momentini eski devir sayısıyla karşılayabilecek yeni bir mil momenti oluşturur. Gerekli olan bu yeni mil momenti oluşuncaya kadar geçici rejim durumu oluşur.

4.GÜÇELEKTRONİĞİNDEKULLANILAN SİMÜLASYON TEKNİKLERİ ve PROGRAMLARI

Bu bölümün amacı bilgisayar simülasyonlarının güç elektroniği sistemlerinin analizi ve dizayn edilmesindeki rolünü incelemektir.Burada simülasyon işlemleri ve bazı simülasyon yazılım paketlerinin bu uygulama için uygun olup olmadığı tartışılmıştır.



Şekil 4.1 Güç elektroniği sistemleri blok diyagramı

Şekil 4.1'de gösterilen blok diyagramında olduğu gibi güç işlemlerinde kullanılan dönüştürücüler diyot, tristör ve diğer anahtarlamalı basit bileşenlerden oluşmaktadır. Bu nedenle devrenin yapısı kontrolörlerin rehberliğinde anahtarların zamanın bir fonksiyonu olarak açılıp kapanmasıyla değişir.

Kapalı şekildeki akım ve gerilimi zamanın bir fonksiyonu olarak çözmek genelde mümkün değildir. Bilgisayar simülasyonları sayesinde bu tip devreleri modellemek mümkündür. Bilgisayar simülasyonları çoğunlukla yeni devrelerin davranışlarının analizlerini ve bu devrelerin ileri düzeyde anlaşılmasını sağlayan araştırmalarda kulanılır.

Simülasyonlarda sistem davranışlarındaki parametreleri analiz etmenin, bu işlemi donanımsal olarak labaratuarda yapmaktan daha kolay olduğunun anlaşılmasından sonra, simülasyon endüstride toplam etüd işlemlerinin kısaltılmasında kullanılmıştır.

Simülasyonlar çeşitli parçaların akım ve gerilim oranlarının sistemlerin durgun ve dinamik haldeki performanslarının ve devrelerin dalga şekillerinin hesaplanmasında kullanılır. Değişken basamaklar arasında genellikle bir çok iterasyon oluşmaktadır.

Simülasyonların gelişimine güvenin artmasıyla, simülasyonların güç kaybı hesaplamalarını da içerecek duruma genişletilmesi mümkün olacaktır. Ancak bilgisayar simülasyonlarını tamamen donanım prototiplerinin yerini alacak şekilde görmemek gerekir.

4.1 Bilgisayar Simülasyonlarındaki Dezavantajlar

Güç elektroniği sistemleri ile ilgili simülasyon oluşturmada bir çok zorluk vardır. Bunların en önemlisi anahtarlama elemanlarının (diyotlar, tristörler, v.b.) bir durumdan diğerine geçerken lineer olmayan bir durum sergilemeleridir. Simülasyon programları bu duruma uyan anahtarlama konumlarını çözebilecek yetenekte olmalıdır.

Simülasyon uzun zaman alabilir. Zaman sabitleri veya diğer bir ifadeyle sistem içersindeki elemanların cevap verme süreleri, bir çok değişik büyüklükle ifade edilebilir. Örnek olarak motor sürücülerinde yarı iletken anahtarlar μs'ler ya da daha az bir anahtarlama süresine sahiptirler. Aynı zamanda motorun mekanik zaman sabitleri ve cevap verme süresi ile yükü, saniyeler ve dakikalar ile belirtilebilir. Bu da küçük zaman sabitini bile temsil edecek çözünürlüğe sahip olacak ve küçük zaman basamaklarını işleyebilecek simülasyonlara gereksinim gösterir. Aynı simülasyonlarda maksimum simülasyon zamanı genellikle çok büyüktür ve bu da en uzun zaman sabitiyle belirtilir.

İdeal modeller her zaman mümkün olmayabilir. Bu özellikle yarı iletken güç elemanları için doğrudur. Bu durum aynı zamanda transformatörler ve bobinler gibi manyetik elemanlar için bile geçerlidir.

Şekil 4.1'de gösterilen kontrolörün, ister anolog ister dijital olsun, mutlaka güç dönüştürücüleriyle modellenmesi gerekir.

Yalnızca durgun dalga şekilleri ile ilgilenilse bile, simülasyonun başında devre durumlarındaki bilinmeyen değerler nedeniyle simülasyon süresi uzun olabilir.

Genellikle sistemin bütün detaylarının simüle edilmesi arzulanmaz. Sebep ise simülasyon süresinin çok uzun olabileceği ve simülasyon sonucunda elde edilecek çıkış değerlerinin beklenildiği gibi çıkmayabileceğidir. Bu açıdan bakıldığında, en iyi simülasyon acil ihtiyaçlara cevap verebilecek basit simülasyondur. Diğer bir ifadeyle, simülasyon öznesine ulaşabilmesi için sisteminin basitleştirilmesi zorunludur.

4.2.Simülasyon İşlemi

Güç elektroniğinde bir çok analiz türü ele alınmalıdır. Herbir analiz türü için kontrolör ve devre elemanlarının temsil edilebileceği uygun durumlar bulmak mümkündür.

4.2.1 Açık çevrim, geniş sinyal simülasyonu

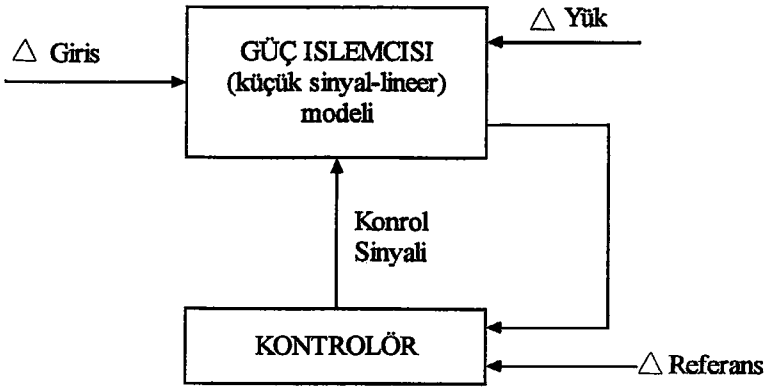
Yeni sistemin davranış biçiminin daha iyi anlaşılması için, güç işlemcilerinin simülasyonuna düzenli olarak verilen kontrol sinyalleri ile başlanır. Önceden yapılan analitik hesaplardan elde edildiği gibi, simülasyonun amacı devrenin düzgün davrandığını doğrulamak ve güç dönüştürücüleri içersindeki gerilim ve akım dalga şekillerini elde etmektir.

Şekil 4.2 Açık çevrim geniş sinyal simülasyonu blok diyagramı

Sonuçta bu işlem, devre yapısı ve elemanların değerlerini elde etmeyi sağlar. Bu simülasyon her bir anahtarın açılıp kapanmasını içerir. Simülasyon çok sayıdaki anahtar grubunun durgun vaziyete geçmesi ile devam eder. Simülasyonun bu bölümünde çoğunlukla devre elemanlarının çok detaylı modellerine sahip olmak hiç bir fayda sağlamaz. Bundan dolayı devre elemanları özelliklede anahtarlama parçaları idealize edilmiş basit modelleri ile temsil edilmelidir. Kontrolörün dizaynının hala devam ettirilme özelliğini korumasından ve bu erken safhalarda sistemin dinamik davranışlarının operasyon koşullarındaki değişiminin dikkate alınmaması gerektiğinden dolayı, kontrolör temsil edilmemiştir. Bundan dolayı buna açık çevrim simülasyonu adı verilir.

4.2.2 Küçük sinyalli (lineer) model ve kontrolör dizaynı

Lineer küçük sinyalli transfer fonksiyonu şeklinde güç işlemcisi modeli geliştirilebilir. Böyle bir modelde en önemli husus anahtarların ortalama karakteristikleri ile temsil edilmesidir.



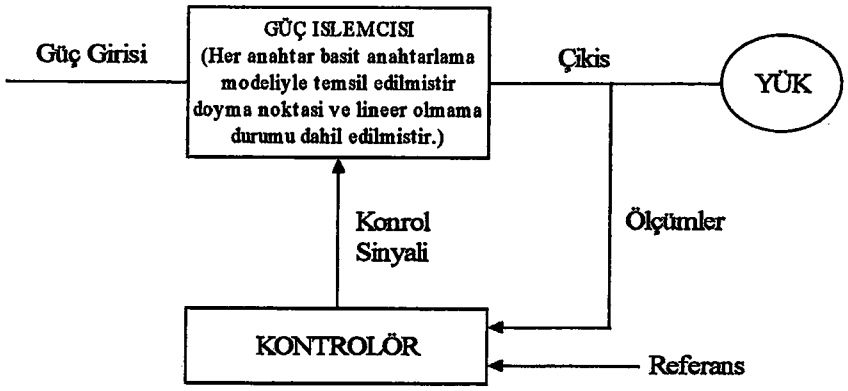
Şekil 4.3 Küçük sinyalli (lineer) model ve kontrolör dizaynı

Bir kere devrenin lineerleştirilmiş modeline sahip olunduğunda kötü etkilere verilen dinamik tepkileri ve dengeyi ya da referans yük ve girişteki Şekil 4.3'de gösterilen küçük değişmeleri sağlaycak kontrolörlerin dizaynı için kontrol teorisinden iyi bilinen metodlara sahip olunur. Kontrolör dizayn işlemini otomatikleştirmek için bir çok özel ticari yazılım paketleri mevcuttur.

4.2.3 Kapalı çevrimli geniş sistemin davranışı

Kontrolör ile tasarlanan sistemin performansı, kapalı çevrili sisteme girişteki ve yükteki bozucu etkiler de dahil edilertek değerlendirilmelidir. Bu durum Şekil 4.4'de gösterilmiştir.

Birçok anahtarlama grubu içeren sistem, uzun zaman aralığında geniş sinyal simülasyonu ile yürütülür. Bundan dolayı da anahtarlama elemanları basit idealleştirilmiş modelleriyle temsil edilmelidir. Bununla birlikte doyma noktası ve uygun lineer olmama durumları ve kayıplar olaya dahil edilmelidir. Karşılaştırıcı opamp gibi detaylı elemanlar ile temsil etmek yerine kontrolörü böyle bir simülasyonda mümkün olduğu kadar basit bir şekilde temsil etmek önemlidir.



Şekil 4.4 Kapalı çevrimli geniş sistemin davranışı

4.2.4 Anahtarlama detayları

Bu bölümdeki hedefimiz güç işlemcileri içersindeki anahtarlama elemanları, bobin ve kondasatörlerin ideal olmayan yapılarından kaynaklanan aşırı gerilim, güç kayıpları v.b. saptamaktır. Bu bilgi eleman oranlarının belirlenmesi, bastırma devreleri gibi koruma devrelerine olan ihtiyacın belirlenmesi ve de bobin, kondansatörlere olan ihtiyacın azaltılabilmesi için gereklidir.

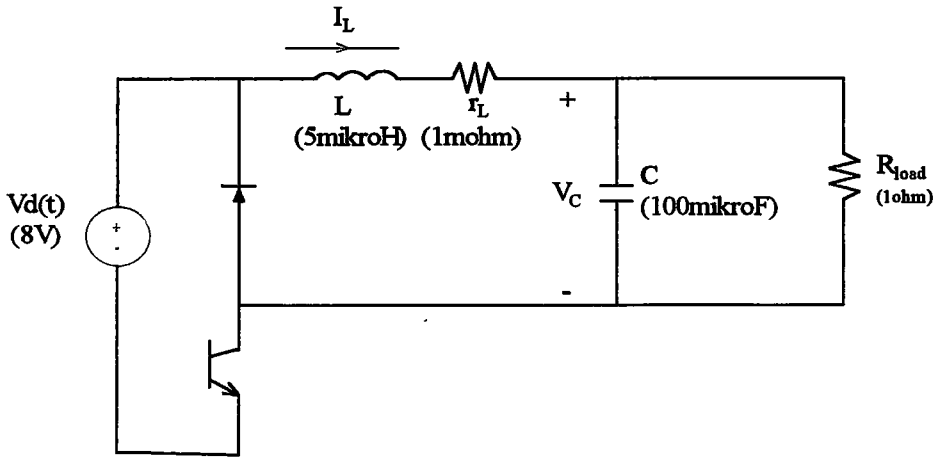
Bu bilgiyi elde etmek için daha önceki simülasyonlarda elde edilmiş gerilim ve akım değerleri ile yalnızca az sayıdaki anahtarlama peryodunun simüle edilmesine ihtiyaç duyulur. Bütün devre yerine yalnızca inceleme altındaki devre bölümü detaylı bir şekilde modellenmelidir. Az sayıdaki peryot boyunca çalışan simülasyon en kötü koşullardaki durumu saptamak için kullanılır. Çünkü bunlar her peryotta tekrarlanır. Bu sonuca göre anahtarlama elemanlarının detaylı ve doğru modellerine ihtiyaç duyarız.

4.3 Geniş Çapta Kullanılan Devre Tabanlı Simülatörler

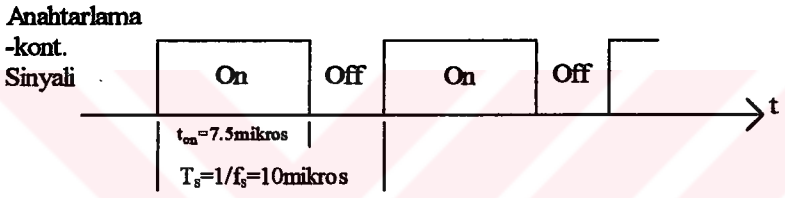
Genel amaçlı bir çok devre tabanlı simülatör mevcuttur. Örnek olarak SPICE, EMTP, Saber ve Krean örnek olarak verilebilir. Bunlardan SPICE ve EMTP kolayca bulunabilir ve geniş alanlarda kullanılır. İkisinin de kuvvetli ve zayıf tarafları mevcuttur. SPICE entegre devreleri simüle etmek için geliştirilmiştir. EMTP ise güç sistemlerinin modellenmesi için geliştirilmiştir.

4.3.1 SPICE

SPICE (Simulation Program Integrated Curcuit Emphasis) Berkeley California üniversitesinde geliştirilmiştir. SPICE lineer olmayan durumları analiz eder ve zaman basamakları üzerinde otomatik kontrol sağlar. SPICE'ın kişisel bilgisayarlar üzerinde çalışan birçok ticari versiyonu mevcuttur. Bunlardan biri olan PSPICE ile devre şeması çizilerek simülasyon yapılmaktadır. Endüstride kullanılmasına ek olarak elektronik ve devre eğitimi veren bir çok yerde de popüler hale gelmiştir. PSPICE'ın popüler olmasındaki bir diğer sebepte kolay bulunabilmesi ve deneme versiyonun ücretsiz olmasıdır. Bu deneme versiyonu güç elektroniği simülasyonlarında başarı ile uygulanmaktadır. Bir devre hakkındaki bilginin devre tabanlı bir programa ve PSPICE'a tanıtılması ile ilgili basit bir örnek mevcuttur. Şekil 4.5'te verilen devrenin açık çevrim altındaki kontrol sinyalinin dalga şekli Şekil 4.6'da gösterilmiştir.

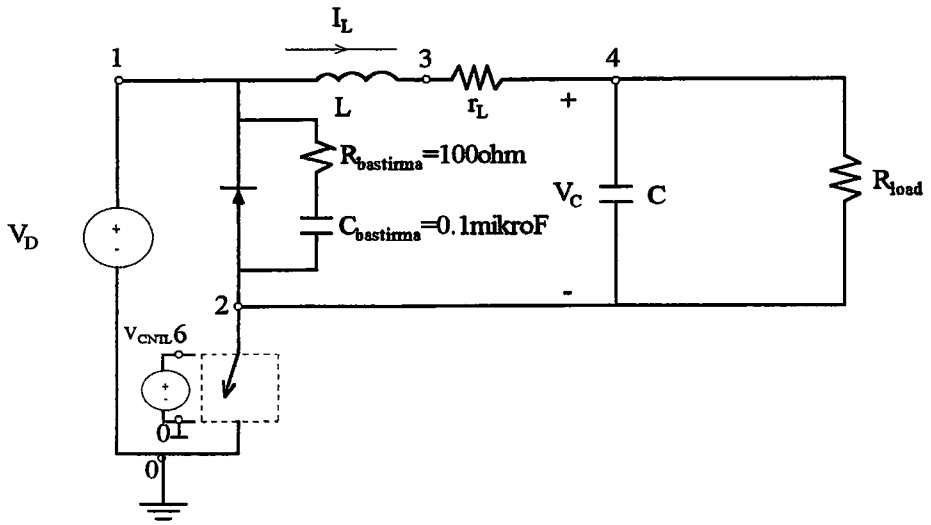


Şekil 4.5 Simülasyonu yapılacak olan devre



Şekil 4.6 Anahtarlama kontrol sinyali

Devre tabanlı simülatör bobin akımının değişimini ve diğer değişken devre durumlarını otomatik olarak hesaplar. Bu simülasyonda diyot PSPICE'daki basit bir modellemeyle, anahtarlama elemanı ise basit bir gerilim kontrollü anahtar ile gösterilmiştir. PSPICE gibi devre tabanlı simülatörler içerisinde eğer anahtarlamayı ayrıntılı olarak analiz etmek istersek detaylı eleman modelleri basit eleman modelleriyle değiştirilir. İlk olarak düğümlerden bir tanesi toprak yani sıfır düğümü seçilmek üzere Şekil 4.7'de gösterildiği gibi düğümleri belirlemeliyiz. Şekil 4.5'teki transistör gerilim kontrollü bir anahtar ile modellenmiştir. Kontrol gerilimi V_{on}'dan büyük ise (=1 V kabul edilen değer) anahtar küçük bir R_{on} direncine (=1Ω kabul edilen değer) sahiptir. Kontrol gerilimi V_{off}'dan küçük ise (=0 V kabul edilen değer) anahtar kapalı durumdadır ve büyük bir direnç olan R_{off} ile temsil edilir (=10^6ohm). Kapalı kabul edilen bu değerler daha farklı seçilebilir.



Şekil 4.7 PSPICE'a göre şekil 4.5'de ki devrenin modellenmesi

Kullanıcı birden fazla değer belirleyebilir. PSPICE'ın bu devreye göre yazılımı ise aşağıda verilmiştir.

```
DIODE 2  1  POWER_DIODE
Rsnub   1  5  100.0
Csnub   5  2  0.1µF
*
SW      2 0 6   0 SWITCH
VCNTL 6  0  PULSE(0V,1V,0s,1ns,1ns,7.5us,10us)
*
L135µH IC=4A
RL        3  4  1m
C         4  2  100µF  IC=5.5V
RLOAD  4  2  1.0
VD        1  0  8.0V
*
MODEL POWER_DIODE D (RS=0.01,CJO=10pF)
.MODEL SWITCH VSWITCH (RON=0.01)
.TRAN    10µs  500.0µs  0s  0.2µs  uic
.PROBE
.END
```

SW anahtarının durumunu ifade eden Şekil 4.6'da gösterilen kontrol gerilimi PSPICE'da VCNTL adı verilen kaynak gerilimi ile modellenmiştir. Değiştirilebilir sıfır tabanlı jonksiyon

kapasitesi C_{JO} ve açma durumundaki direnci R_S' olan diyot modelleri mevcuttur. Bunun dışında varsayılan değerler program tarafından kullanılır. SPICE'daki ani bir süreksizlik bazı düğümlerde zaman basamaklarında aynı noktaya yaklaşamama gibi problemleri ve programın çok küçük zaman basamakları ile durumu çözmeye çalışmasına neden olabilir. Böyle bir durum olduğunda simülasyon hata mesajı verip duracaktır. Aynı noktaya yaklaşamama gibi problemlerin önüne geçmek için bazı kurallar vardır. Diyot akımının aniden sıfıra düşmesini yumuşatan amacıyla Şekil 4.7' de ki diyoda paralel RC bastırma devresi kullanma gibi ani kesinti ve devamsızlıkları önlemek faydalıdır. Buna benzer şekilde Şekil 4.7'deki VCNTL'nin yükselme ve düşüş zamanlarının programda yazıldığı gibi PULS sıfır yerine herbiri için 1 ns ile belirtilmesi uygundur.

4.3.2 EMTP simülasyon programı

Yaygın olarak kullanılan bir diğer simülasyon programı EMTP'dir (Elektromagnetik Trasient Program). Mikroelektronik modellemenin oluşturduğu EMTP programı Oregon Portland'da güç endüstrisi için geliştirilmiştir. ATP (Alternativ Transient Program) MS-DOS işletim sistemi altında çalışan PC'ler için uygun olan EMPT'nin bir versiyonudur. Spice'ye benzer olarak EMPT integrasyonun trapezoidal kuralını kullanır fakat integrasyonun zaman basamakları sabit olarak tutulur. Üç fazlı iletim hatları gibi çeşitli güç sistem parçaları için üretilmiş modellerden dolayı EMTP güç sistemlerindeki güç elektroniği uygulamalarının modellenmesinde kullanılan güçlü bir programdır. PSPICE'la karşılaştırdığında EMTP'deki anahtarlar biraz farklı olarak karşımıza çıkar. Anahtarların açılıp kapanması devre matrisi oluşturur. Bu özellik yüksek seviyeli dillerde dijital ve analog kontrolörler ile temsil edilmesine benzeyen çok güçlü bir özelliktir. Elektrik devresi ve kontrolörler her zaman basamağında çeşitli değişkenlerin değerlerini ileri ve geri hareket ettirebilirler.

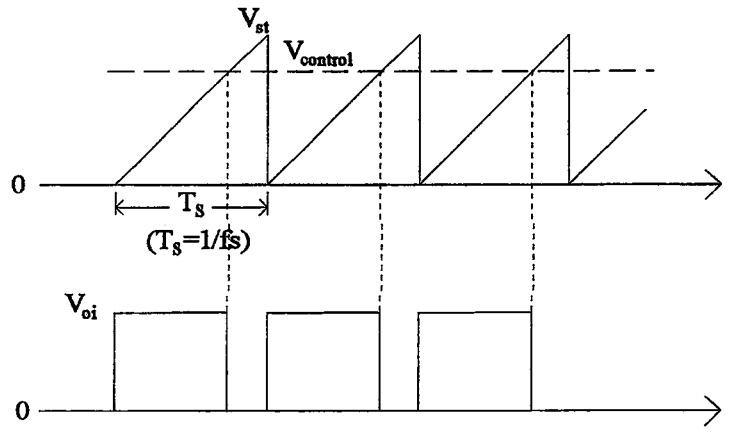
4.3.3 PSPICE ve EMPTnin uyumluluğu

Güç elektroniği simülasyonları için hem PSPICE hem EMTP çok uygundur. Güç elektroniği konusunda bir çok nedenden dolayı PSPICE daha üstündür. Diğer bir taraftan EMTP güç sistemlerindeki yüksek güç elektroniği sistemlerinin simülasyonu için daha uygundur. Yüksek seviyeli dillerdeki gibi kontrolörlerin aynı kolaylıkta sunulması yeteneğine sahiptir. Δt zaman aralıkları kullanılarak elde edilen sonuçlar kabul edilebilir doğruluktadır. Yukarıda listelenen özelliklerden dolayı EMTP mantıksal tanımlamalar, transfer fonksiyonları, kontrolörler ve ideal anahtarlar gibi elemanları modellemek için kompleks güç elektroniği sistemlerinin analizi açısından uygundur.

Güç elektroniğini öğrenme konusunda ilk yardım özelliği taşıyan çok geniş sayıdaki güç elektroniği örnekleri PSPICE ve EMPT'nin deneme versiyonlarında mevcuttur.

4.4 Eşitlik Çözücüler

Eğer bir eşitlik çözücü seçersek devre durumunu ifade eden kontrolörler içerisinde mantıksal tanımlama ve çeşitli devre durumlarını izah etmek amacıyla diferansiyel eşitlikler yazmak durumunda kalırız. Bu durumda diferansiyel eşitlikler zamanın bir fonksiyonu olarak çözülmek durumunda kalır. En basit şekilde bu eşitlikleri Pascal, C, Fortran gibi yüksek seviyeli dillerdeki programlama ile çözebiliriz. Matris inversiyonu yada integrasyonu sağlayabilen özel uygulamalardan oluşan bu dillerdeki kütüphanelerden yararlanmak mümkündür. Bu özelliklerin bulunduğu MATLAB gibi paketleri ya da diğer paket türlerini kullanmak daha akıllıca olabilir. Bu paketlerin her biri bahsedilen uygulamalarda kendi formatlarını kullanırlar. a ve b gibi iki değişkenin çarpımı olan y=a.b örneğindeki gibi matris ve değişken uygulamaları MATLAB ile kolayca gerçekleştirir. Benzer olarak bir matrisin tersini almak için kullanılacak komut y=inv(x)'tir. MATLAB endüstride geniş olarak kullanılır. Aynı zamanda bu tür programlar sinyal üretme ve kontrol sistemlerinin öğretildiği yerlerde de kullanılırlar. Matlabın içinde yer alan Simulink, dinamik sistemleri kolay blok diyagramı şeklinde tanımladığı için, MATLAB kullanıcı ara yüzü teşkil eden güçlü bir grafiksel işlemcidir. MATLAB'a örnek olarak Şekil 4.5' te verilen devrenin çözümünde Şekil 4.8'de ki integrasyonun tropoizoidal metoda göre çözümü kullanılmıştır. Şekil 4.8'de gösterildiği gibi V_{oi} giriş gerilimi MATLAB'da fs anahtarlama frekansındaki V_{st} dalga şeklinin $V_{kontrol}$ DC kontrol geriliminin karşılaştırılmasıyla elde edilir. Kontrol gerilimi V_{st}, V_{oi}=Vd den büyük olduğunda sıfır olur.



Şekil 4.8 MATLAB'la yapılan simülasyonda kullanılan dalga şekli

Şekil 4.5'te ki devrenin her iki programla da simüle edilmesinde yazılan simülasyon programları birbirlerinden farklılık gösterir. MATLAB'la bu devrenin simülasyonu için kullanılan komut listesi aşağıda verilmiştir.

```
Vd=8; L=5e-6; C=100e-6; rL=1e-3; R=1.0; fs=100e3; vcontrol=o.75;
Ts=1/fs; tmax=50*Ts; deltat=Ts/50;
Time=0:deltat:tmax;
Vst=time/Ts-fix(time/Ts);
Voi=Vd*(vcontrol>vst);
A=[-rL/L –1/L; 1/C –1/(C*R)];
b=[1/L O]';
MN=inv (eye(2)-delta/2 * A);
M=MN * (eye+deltat/2 * A);
N=MN * delta/2 * b;
İL(1)=4.0; vC(1)=5.5;
Timelenght=length(time);
for k =2:timelength
x=M * [iL(k-1) vC(k-1)]' + N * (voi(k) + voi(k-1));
iL(k)=x(1); vC(k)=x(2);
end
plot (time,iL,time,vC)
```

4.5Sonuç

Modelleme ve bilgisayar simülasyonu güç elektroniği sistemlerinin öğrenilmesi, dizaynı ve analizinde öneli bir rol oynar. Bu tür simülasyonlarda bahsedilen durumlardan dolayı simüle edilen sistemin basitleştirilmesi önemlidir. Uzun yıllardan beri birçok simülasyon paketi geliştirilmiştir. Bir sistemin simüle edilmesinde mevcut simülasyon paketleri incelenerek sisteme en uygun olanının seçilmesi gerekmektedir.

5. MATLAB

5.1. MATLAB'ın Tarihçesi

MATLAB, (MATrix LABoratory) ilk defa 1985'de C.B. Moler tarafından matematik ve özellikle de matris esaslı matematik ortamında kullanılmak üzere geliştirilmiş etkileşimli bir programlama dilidir. İlk sürümleri Fortran diliyle yazılmıştır. Son sürümleri ise C diliyle yazılmaktadır. MATLAB mühendislik alanında; sayısal hesaplama, veri çözümleri ve grafik işlemlerinde kullanılabilecek genel amaçlı bir program olmakla beraber özel amaçlı modüler paketlere de sahiptir. Aralık 1999 tarihi itibariyle MATLAB'ın son versiyonu MATLAB 5.3.1'dir. MATLAB'ın tüm hakları The Math Works'e aittir. Diğer tüm modüler paketleri ve araç kutuları (toolbox) gene bu firma tarafından yazılmaktadır.

5.2. MATLAB'ın Endüstride Başlıca Kullanım Alanları ve Büyük Kullanıcı Firmalar

Günümüzde MATLAB endüstride yaygın olarak çeşitli sektörlerde kullanılmaktadır. Başlıca kullanılan sektörler, kullanıcı firmalar ve üniversitelerden bazıları aşağıdaki gibidir.

Aeorodinamik sektörü: Boeing, Calspan, Daimler Chrysler Aerospace, Duke University, Gencorp Aerojet, Honeywell, Saab Military Aircraft, Sheet Dynamics, Washington University

Otomotiv: Delphi Chassis Systems, Newman-Haas Racing, Simcar.com, Toyota

Biotıp: Taiwan Neurology Clinic, Univerity of Graz, Washington University

Haberleşme: Motorola, Quantum, Texas Instruments

Elektronik: Spatializer Audio Laboratories

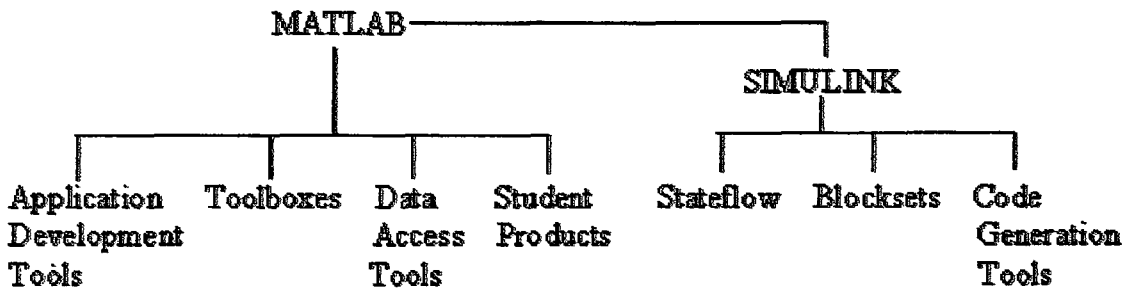
Finans: Deutsche Morgen Grenfell, Lionhart Investments Ltd.

Fabrika: BHP Steel Rolling Mills

5.3. MATLAB Ürün Ailesi

Math Works tarafından hazırlanan ve MATLAB'la beraber kullanılan birçok araç kutuları ve modüler paketler mevcuttur. Bunları temel olarak Şekil 5.1'de görüldüğü gibidir. Şekil 5.1'de görülen ifadeleri açıklayalım.

Matlab: MATLAB bir dil ve teknik bilgi- işlem ortamıdır. Uygulama geliştirmek, algoritma ve data analizi yapmak için gerekli olan gelişmiş grafiksel araçları ve temel matematiksel işlemleri sağlar.



Şekil 5.1. Math Works Ürün ailesi

Application Development Tools (Uygulama Geliştirme Araçları): Uygulama geliştirme araçları MATLAB derleyicisi, C/C++ programlama dili, matematik-grafik kütüphaneleri ve MATLAB web sunucusudan oluşur. Bu araçlar MATLAB uygulamalarının kendimize özgü versiyonlarını yaratmamızı ve dağıtmamızı, diğer tabanlarda çalışan kişilerle MATLAB'ı paylaşmamızı sağlar duruma getirir.

Toolboxes (Araç Kutuları): Araç Kutuları, MATLAB ve Simulink'i genişleten yüksek derecede optimize edilmiş, uygulamaya yönelik fonksiyonların bileşimidir. Araç kutuları, sinyal ve imaj üretme, kontrol sistemleri dizaynı, optimizasyon, finansal mühendislik, sembolik metematik, network ve diğerlerinide içeren uygulamaları destekler. Araç kutusu fonksiyonları MATLAB dili ile yazılmıştır. Bunlar kullanıcı tarafından değiştirilebilir.

Data Access Products (Veritabanı Erişimi Ürünleri): Bu ürünler, harici kaynaklardan mevcut datalara kolayca erişim imkanı sağlarlar. The Data Acquisition Toolbox (Veri kazanç araç kutusu) harici bir donanımdan direkt olarak MATLAB'a canlı veriyi getirmeyi sağlar. Veritabanı araç kutuları, MATLAB kullanıcılarının JDBC/ODBC uyumlu veritabanları ile arayüzler vasıtasıyla iletişimini sağlar.

Simulink: Simulink prototip simülasyonunu gerçekleştirebilen, gerçek dünya analizi yapan dinamik sistemlerdir. Simulink çekirdek MATLAB nümerik fonksiyonları ile inşa edilmiş blok diyagram ara yüzü sunar.

Stateflow: Stateflow, event-driven (Süren olaylar tarafından kontrol edilen) sistemlerin dizaynı ve modellenmesi için grafiksel simülasyon ortamıdır. Stateflow, oturmuş sistemler içindeki kontrol ve protokol (ağ üzerindeki özel iletişim dili) dizaynında uygun çözümler sunar.

Blocksets: Blocksets, elektrik güç sistem modelleme, dijital sinyal işleme, sabit noktalı algoritma geliştirme ve diğerlerini içeren birden fazla dizayn bölgelerini destekleyen uygulamaya yönelik bloklar bileşimidir. Bu bloklar direkt olarak Simulink modellerine dahil edilebilirler.

Code Generation Tools (Kod Oluşturma Araçları): Gerçek zamanlı Workshop (Çalışma Arayüzü) ve Stateflow kodlayıcısı direkt olarak Simulink modellerinden ve Stateflow diyagramlarından, hızlı prototipleme döngü simülasyonları ve oturmuş sistem uygulamaları için kişiselleştirilen C ve ara kodları oluşturur.

Math Works Partner Products (Math Works Ortak Ürünleri): Math Works araçları ile uyumlu 200'den fazla ek donanım ve yazılım mevcuttur. Bu yardımcı sunumlar araç kutuları, bloksetleri, gerçek zamanlı workshop hedefleri, eğitim ve danışmanlık servislerini kapsar.

5.4. Temel MATLAB Komutları ve Birkaç Basit Uygulama Örneği

MATLAB, küçük büyük harf'e dikkat eden bir programdır. Bu sebeple komutların yazımında buna dikkat edilmelidir. MATLAB'da basit uygulamalar için gerekli olan işaret ve komutlar aşağıdaki gibidir.

[ ]: Köşeli parantez satır ya da sütun vektörlerini ve matrisleri boyutlandırma amacıyla kullanılır.
( ): Normal parantez çok amaçlı kullanıma sahiptir. Matematiksel önceliğin belirtilmeşinde, fonksiyon değişkenlerini belirtmekte, vektör ve matrislerde indisleri göstermekte kullanılır.
.: Nokta işareti kesir ayırma işlemlerinde kullanılır.
...: Tek satıra sığmayan ifadelerin bir alt satırda devam ettiğini gösterir.
,: Virgül işareti matrislerde indisleri ayırmak amacıyla kullanılır.
;: Noktalı virgül sonuçların ekrana yazılmasını önler, matrislerde satırı sona erdirir.
%: Açıklama yapılan durumları ifade eder. MATLAB tarafından icra edilmez.
': Tırnak matrisin transpozesini alır.
+: Artı işareti iki matrisin toplamını alır. Ancak matrisler eşit boyutta olmalıdır. Skaler sayıları toplamak içinde kullanılır.

-: Eksi işareti iki matrisi birbirinden çıkarmada kullanılır. Skaler sayıların çıkartılmasında da kullanılır.

*: Çarpı işareti matrissel çarpımı gösterir.

\: Bölme işareti soldan bölmeyi temsil eder.

/: Bölme işareti sağdan bölmeyi temsil eder.

^: İşareti matrissel kuvvet almada kullanılır.

<: İşareti ...den küçük anlamında kullanılır.

>: İşareti ...den büyük anlamında kullanılır.

|: İşareti or işleminde kullanılır.

&: İşareti and işleminde kullanılır.

= =: İşareti denk işlemine karşılık gelir. Yani matrissel eşitlik söz konusudur.

~: İşareti komplement yani değil olarak kullanılır.

cd: Dizin değiştirmek için kullanılır.

demo: Demo dosyalarını görüntülemek amacıyla kullanılır.

save: komut penceresindeki verileri saklamak için kullanılır.

load: Kaydedilen verileri çağırmak için kullanılır.

exit: Matlab'dan çıkışta kullanılır.

clear: Çalışma alanında ki değişkenleri sıfırlar.

clr: komut penceresini temizler.

clg: Grafik penceresini temizler.

who: Kullanılan değişkenleri gösterir.

whos: Kullanılan değişkenlerin boyutlarını gösterir.

ans: Sonucu yazdırır.

size: Matrisin boyutunu gösterir.

what: Dizindeki tüm dosyaları gösterir.

input: Değer ya da herhangi bir şeyin girilmesini açıklamak için kullanılır.

plot: Grafiği çizdirmek için kullanılır.

title: Başlık koymak amacıla kullanılır.

xlabel: X eksenine etiket değeri koyar.

ylabel: Y eksenine etiket değeri koyar.

gtext: İşaretle istenilen noktaya açıklama yazılır.

grid: Şekile grid ekler.

if: Eğer mantıksal bildirim gerçek ise if bildirimi ve end bildirimi arasındaki bildirimler icra edilir.

İf mantıksal deyim bildirim gurubu

end

if else: Else hükmü, bir mantıksal deyimin gerçek olması halinde belli bir bildirimler grubunu, mantıksal deyimin gerçek olmaması halinde ise başka bir bildirim takımının icrasını sağlar.

İf mantıksal deyim grubu A

else bildirim grubu B

end

for: Bu deyim bir matris ve bildirim grubu A içindeki bildirimleri bildirim matrisi içindeki sütunların sayısı kadar tekrar eder. Her defasında döngü boyunca, indis deyim matrisi içindeki elemanlardan birinin değerini alır.

for indis=deyim bildirim grubu A

end

Aşağıda MATLAB'a ait temel komutların kullanıldığı örnekler verilmiştir.

Örnek-1: 1,2,3,4,5,6,7,8,9 elemanlarından oluşan bir matris giriniz.

Çözüm-1: a=[1 2 3 4 5 6 7 8 9] şeklinde girilir ve MATLAB komut satırında cevap olarak

a=

1 2 3 4 5 6 7 8 9

ifadesi görülür.

Örnek-2: Örnek-1 de girilen a matrisi elemanlarını 2 ile toplayınız.

Çözüm-2:b=a+2 ifadesi girilir ve cevap olarak

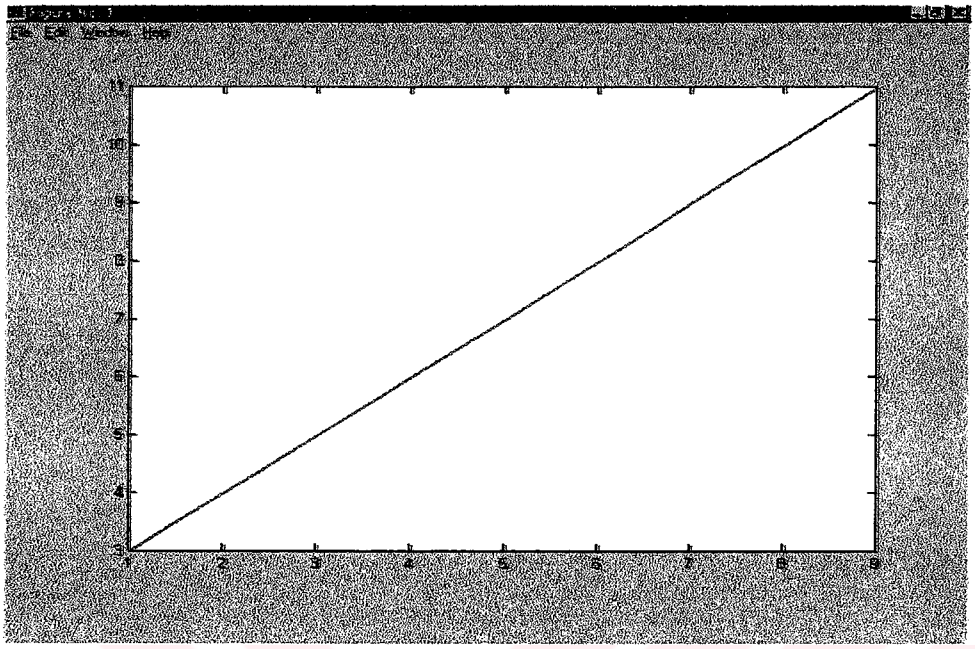
b=

3 4 5 6 7 8 9 10 11

ifadesi görülür.

Örnek-3: a ve b matrislerini grafik olarak çizdiriniz.

Çözüm-3:plot (a,b) ifadesi girilir.



Şekil 5.2 a ve b matrislerinin çizdirilmesi

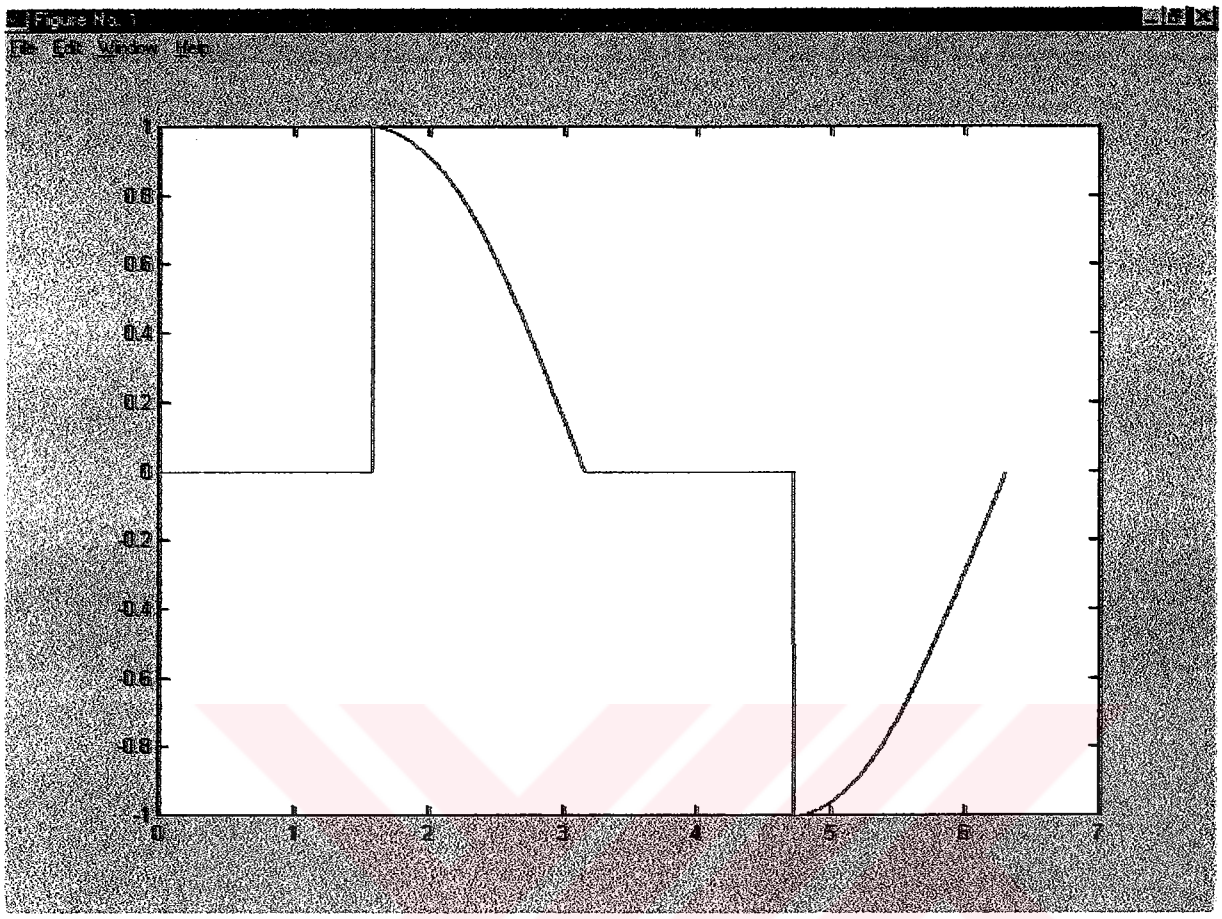
Şekil 5.2'de görüldüğü gibi grafik elde edilir.

Örnek-4: $\alpha=90^{o}$ olacak şekilde tek fazlı bir A.C. kıyıcının çıkış dalga şeklini çizdiriniz.

Çözüm-4:

```
Clear
k=0
For t=0:0.001*pi:2*pi;
k=k+1;
if((t>=0 & t<=0.5*pi) | (t>=pi & t<(3*pi/2)))
y(k)=0;
elseif(t>0.5*pi)
y(k)=sin(t);
end
time(k)=t;
end
```

plot(time,y);şeklinde program yazılır ve çalıştırılırsa;



Şekil 5.3 Tek fazlı A.C kıyıcının çıkış dalga şekli

Şekil 5.3'de görüldüğü gibi çizim elde edilir.

Örnek-5: Üç fazlı kontrolsüz doğrultucu için her bir fazın çıkışını ve Ud geriliminin değişimini çizdirin.

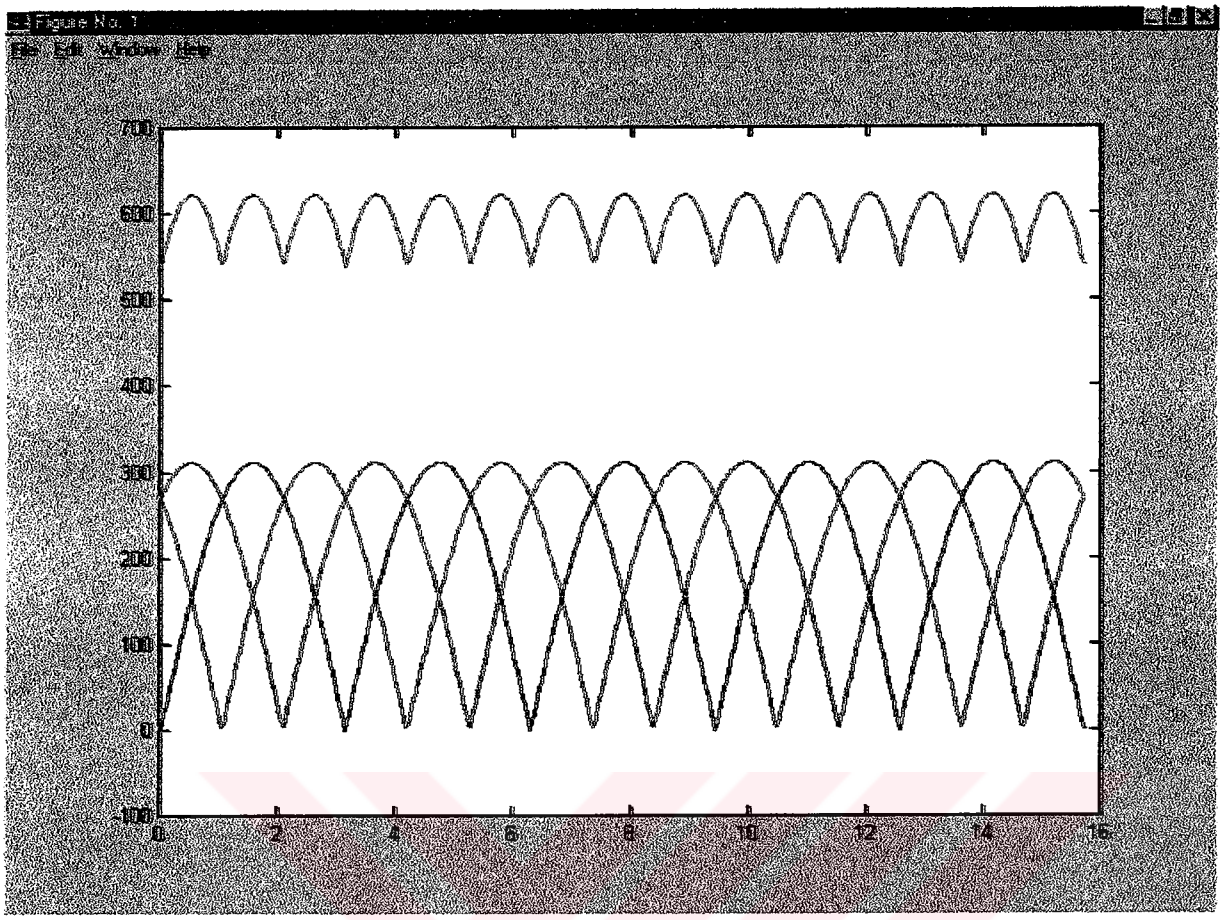
Çözüm-5:

```
Clear
k=0;
for z=0:0.01*pi:5*pi;
k=k+1;
X(k)=z;
%r(k)=(220*sin(z));
%s(k)=(220*sin(z+(2*pi/3)));
%t(k)=(220*sin(z-(2*pi/3)));
%ud(k)=r(k)+s(k)+t(k);
%*****    T FAZI İÇİN ÇİZİM YAPTIRLIYOR    *********
if ((z>=0*pi-(2*pi/3) & z<1*pi-(2*pi/3))...
|(z>=2*pi-(2*pi/3)&z<3*pi-(2*pi/3))...
|(z>=4*pi-(2*pi/3) & z<=5*pi-(2*pi/3)))
```

```
T(k)=(sqrt(2)*220*sin(z+(2*pi/3)));    else  ((z>=1*pi-(2*pi/3)  &  z<=2*pi-
(2*pi/3)) | (z>=3*pi-(2*pi/3) & z<=4*pi-(2*pi/3)));

%*****    S FAZI İÇİN ÇİZİM YAPTIRLIYOR    *********
if ((z>=0*pi+(2*pi/3) & z<1*pi+(2*pi/3))...
| (z>=2*pi+(2*pi/3) & z<3*pi+(2*pi/3))...
| (z>=4*pi+(2*pi/3) & z<=5*pi+(2*pi/3)))
S(k)=(sqrt(2)*220*sin(z-(2*pi/3)));
else  ((z>=1*pi+(2*pi/3)  &  z<=2*pi+(2*pi/3))  |  (z>=3*pi+(2*pi/3)  &
z<=4*pi+(2*pi/3)));
S(k)=-(sqrt(2)*220*sin(z-(2*pi/3)));
end
%*****    R FAZI İÇİN ÇİZİM YAPTIRLIYOR    *********
if ((z>=0*pi & z<1*pi) | (z>=2*pi & z<3*pi) | (z>=4*pi & z<=5*pi));
R(k)=(sqrt(2)*220*sin(z));
else ((z>=1*pi & z<=2*pi) | (z>=3*pi & z<=4*pi));
R(k)=-(sqrt(2)*220*sin(z));
end
Ud(k)=R(k)+S(k)+T(k);
end
plot(X,R,X,S,X,T,X,Ud)
```

şeklinde program yazılır ve çalıştırılırsa, şekil 5.4'te görüldüğü gibi çizim elde edilir.



Şekil 5.4 Üç fazlı kontrolsüz doğrultucunun çıkış dalga şekli

6. MATLABLA DC MOTORUN SİMÜLASYONU

6.1. Giriş

Bu bölümde MATLAB'ın bir yardımcı programı olan Simulink kullanılarak serbest uyartımlı bir DC motorun modellenerek boşta ve yükte çalışması simüle edilmiştir. Bölüm.5'de de anlatıldığı üzere Simulink prototip simülasyonu gerçekleştirebilen, gerçek dünya analizi yapan dinamik bir sistemdir. Simulink bu işlemleri yaparken iterasyon yöntemleri kullanır. Bu yöntemler Bölüm 6.2'de açıklanmıştır.

6.2. Simulink'in Kullandığı İterasyon Yöntemleri

Simulink modellerinin simülasyonu diferansiyel denklem takımlarının sayısal integrasyonu ile alakalıdır. Simulink içinde bu tür denklemlerin simülasyonunda kullanılan çeşitli integrasyon alt programları mevcuttur. Sistemlerin dinamik davranışlarındaki farklılıklardan dolayı, her türden modelin doğru ve verimli bir biçimde simülasyonunu sağlayan tek bir yöntem verilememektedir. Doğru sonuçlar elde edilmesi açısından seçilen yöntemin uygunluğu ve simülasyon parametrelerinin dikkatle seçilmesi çok önemlidir. Bu alt programlar program üreticisi tarafından kodlandığından dolayı Simulink'le alakalı kaynaklarda ilgili algoritmalar açık olarak verilmemiş, sadece kullanım ile alakalı bilgiler sunulmuştur. Simulink'de kullanılan belli başlı simülasyon integrasyon alt programları aşağıdaki gibi özetlenebilir.

linsim: Doğrusal dinamikleri ortaya çıkaran yöntemdir. Dolayısıyla doğrusal modeller için kullanılır. Doğrusal modeller Simulink toolbox'nın blok kütüphanesinde (block library) yer alan transfer fonksiyonu (transfer function), durum uzayı ve durum denklemleri (state space), sıfır-kutup-kazanç (zero-pole-gain), toplama (sum) ve kazanç (gain) bloklarından ibarettir.

Birkaç doğrusal olmayan blok ile birlikte temelde doğrusal olan bir sistem Lsim (sürekli zaman sistemi keyfi giriş cevabını veren komut) ile iyi sonuçlar verebilir. Lsim, özellikle hem hızlı hem de yavaş dinamiklere sahip doğrusal bloklarda kullanıldığında diğer integrasyon alt programlarına göre daha iyi neticeler vermektedir.

rk23 ve **rk45:** Üçüncü ve beşinci dereceden Runge-Kutta yöntemleridir. Runge-Kutte yöntemleri aşırı doğrusal olmayan ve\veya kesikli (dijital) sistemlerde diğer sistemlere göre çok daha iyi sonuçlar sağlanmaktadır. Yalnız bu yöntemler hem hızlı hem de yavaş dinamiklere (sıkı sistemler) sahip sistemler de iyi verim vermezler. Bu tür sistemlerde de

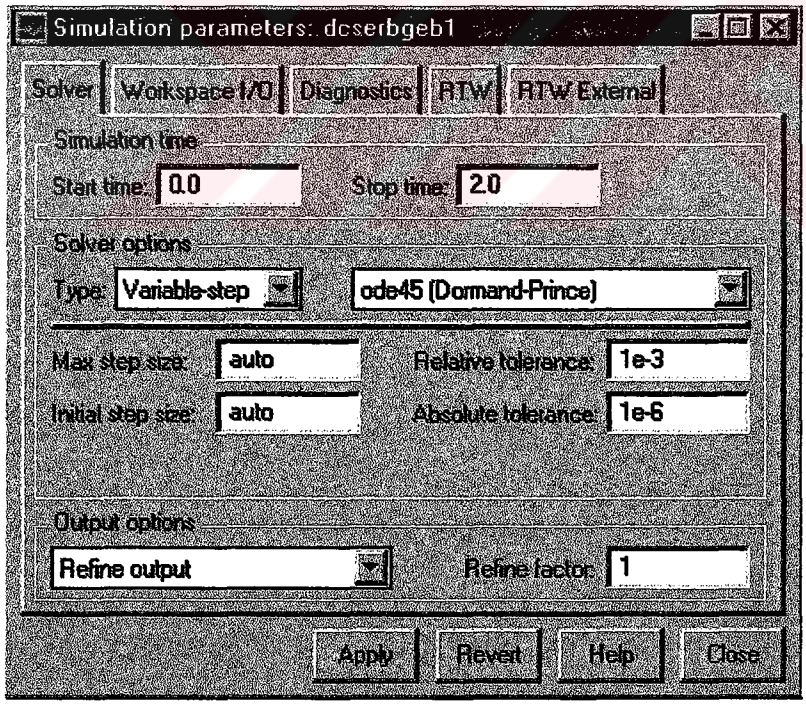
linsim ve gear gibi sıkı yöntemler kullanılır. Rk23 ve rk45 yöntemleri karışık sistemlerde (sürekli ve kesikli zaman) iyi neticeler vermektedir.

gear: Sıkı sistemler için gear'ın bulunduğu kestirimci-düzeltici (predictor- corrector) yöntemidir. Gear yöntemi düzgün geçişli ve doğrusal olmayan sistemler için kullanılır. Sıkı sistemler (aynı anda hem çok küçük ve hem de çok büyük zaman sabitlerine sahip sistemler) için tasarlanan gear yöntemi sıkı olmayan sistemlerden iyi sonuçlar vermezler. Tekil noktalara sahip sistemlerde de iyi sonuçlar vermeyebilirler.

adams: Adams kestirimci-düzeltici yöntemi. Düzgün dağılımlı (smooth) ve doğrusal olmayan (non-linear) ve aynı zamanda zaman sabitleri çok farklı büyüklüklerde, olmayan sistemlerde kullanılır.

euler: Euler yöntemi yalnız sonuçları tahkik etmek için kullanılır.

Bu bölümde yapılan simülasyonlarda rk45 (5. dereceden Range-Kutta) metodu kullanılmıştır.



Şekil 6.1 Simülasyon parametrelerinin görsel olarak belirlendiği pencere

Şekil 6.1'de görülen pencerede parametrelerin isimlerinin ve işlevlerinin tanımlanması aşağıda verilmiştir.

Simulation time (Simülasyon süresi): Simülasyon başlangıç ve bitiş zamanın tanımlandığı bölümdür.

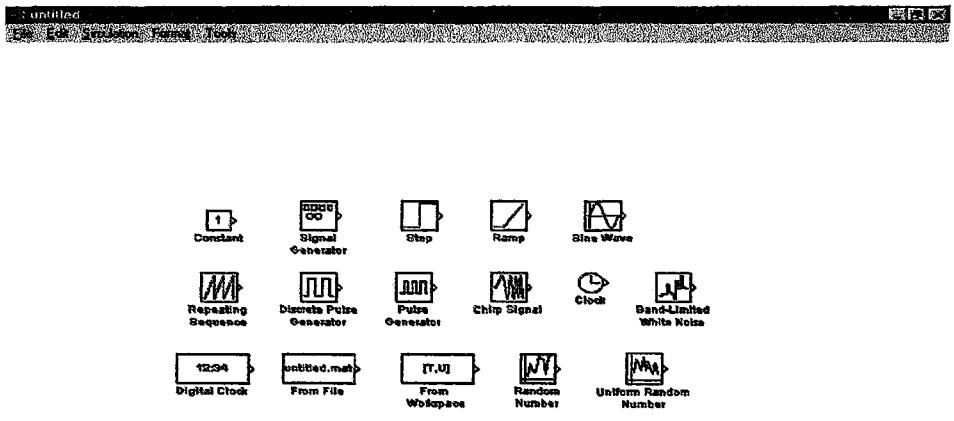
Solver options (Çözücü opsiyonu): Sayısal integrasyon çözüm yönteminin belirlendiği bölümdür. Type (tip) ile integrasyon aralık tipi belirlenir. Bu bölümde yapılan simülasyonda program tarafından otomatik olarak belirlenen variable step (değişken adım)-ode45 Range-Kutta metodu seçilmiştir. Bunun sebebi sistemimizin sürekli zaman sistemi (anolog) olmasıdır.

Output options (çıkış opsiyonları): Refine output (çıkış düzenleme) ve refine factor (düzenleme faktörü) ile simülasyon çıktıları görsel olarak uygun biçimde elde edilir. Refine factor'ün 1 alınması yazılımcı firma tarafından tavsiye edilmiştir.

6.3. Simulink ve Kütüphaneleri

Tüm bu analiz ve simulasyon işlemlerini yaparken Simulink işlemleri kolaylaştıran Library (kütüphaneler) sunar. Bu kütüphanelerle bir çok matematiksel işlemlerin kolaylıkla çözülmesi mümkündür. Bu kütüphanelerin temel başlıkları ise; Sources (kaynaklar), Sinks (izleme araçları), Discrete (fonksiyonlar), Linear (lineer), Non-linear (lineer olmayan), Connections (bağlantılar) ve Blocksets&Toolboxes'dır. (bloksetler ve araç kutuları)

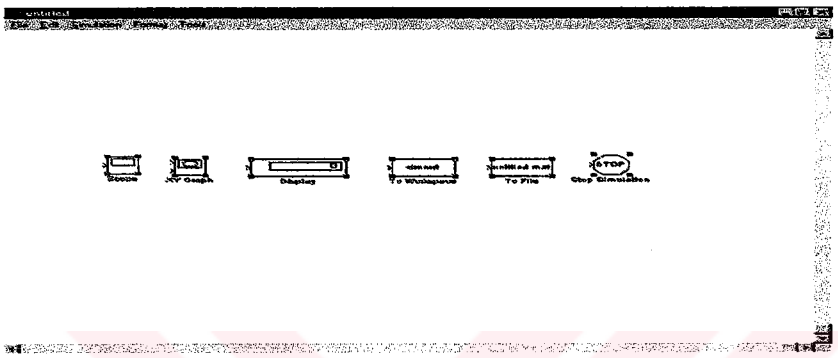
Sources kısmında sisteme uygulayabileceğimiz çeşitli kaynak fonksiyonları mevcuttur.



Şekil 6.2 Sources library

Sources kısmında bulunan elemanlar Şekil 6.2'de de görülmektedir. Görüldüğü üzere sinyal generatöründen darbe genaratörüne kadar birçok eleman mevcuttur. Bu elemanların değerlerini ve şekillerini de ayarlamak mümkündür. Örnek olarak sinyal generatörü ele alınırsa bu elemanın uyguladığı darbe şeklini sinüs, testere, kare olarak ayarlamak bu darbenin frekans değerini ve genliğini belirlemek mümkündür.

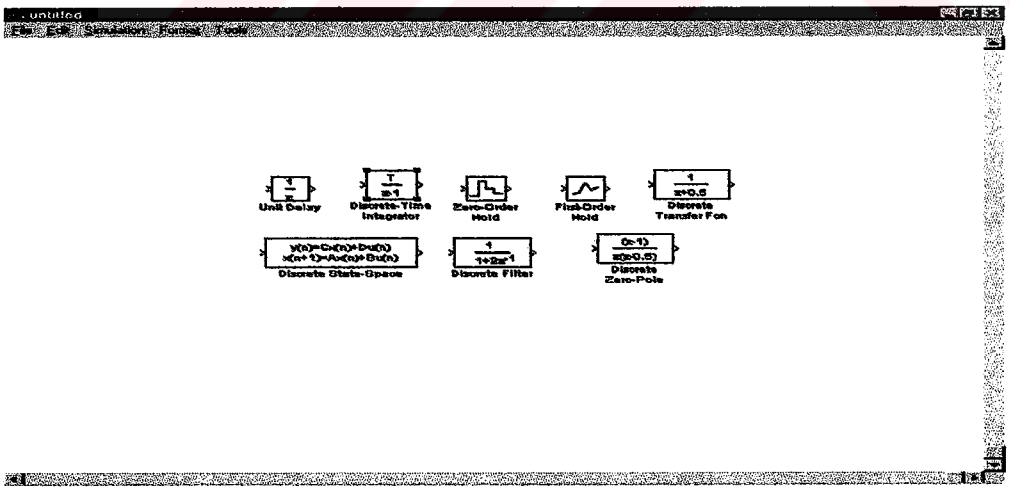
Sinks kısmında ise sistemin giriş ve çıkışlarını izlemek için çeşitli elemanlar yer almaktadır.



Şekil 6.3 Sinks library

Bu elemanlar Şekil 6.3'de de görülmektedir.

Discrete kısmında ise çeşitli filtreler ve özel fonksiyonlar mevcuttur. Bu kütüphaneninde elemanları Şekil 6.4'de görüldüğü gibidir.

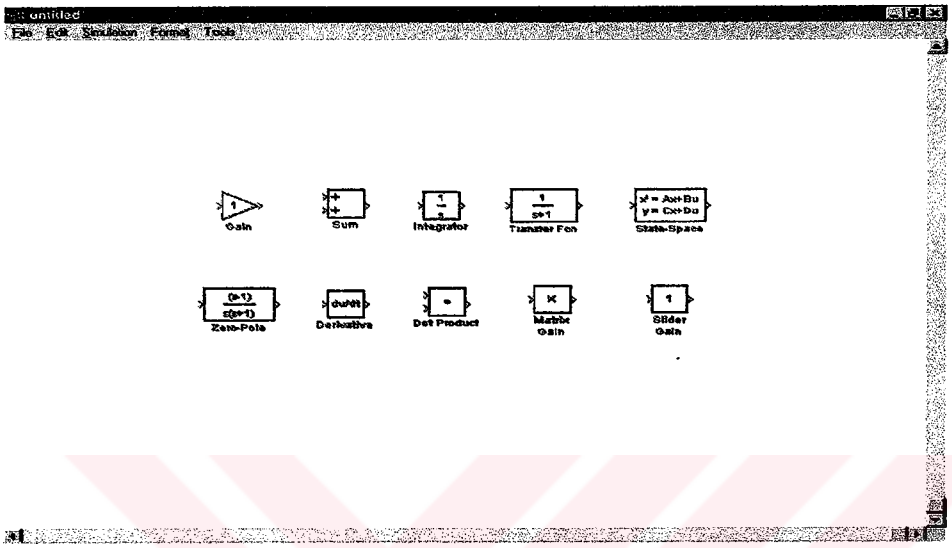


Şekil 6.4 Discrete library

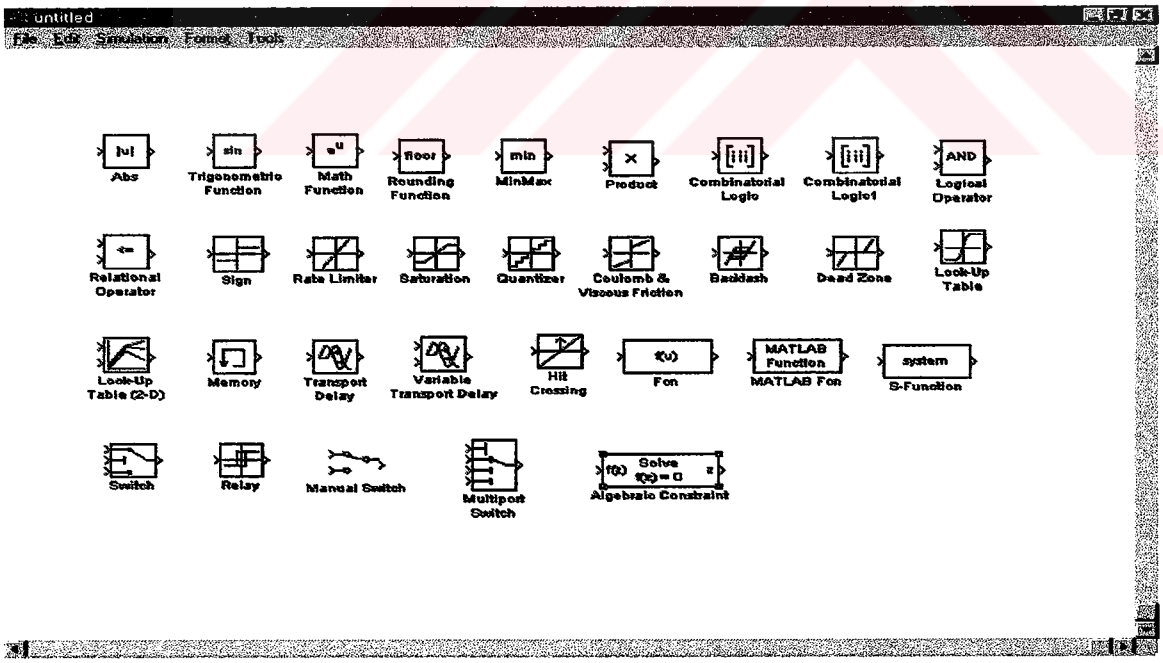
Linear kısmında ise çeşitli lineer matematiksel fonksiyonlar mevcuttur. Bu kütüphane DC motor simülasyonunda en çok kullanılan kütüphanedir. Bu kütüphanede toplama elemanından integral alma elemanına kadar matematikte sık kullanılan fonksiyonlar mevcuttur. Bu kütüphanenin elemanları Şekil 6.5'de ki gibidir.

Non-Linear kısmında ise lineer olmayan matematiksel fonksiyonlar mevcuttur. Bunlar arasında trigonometrik fonksiyonlardan üssel foksiyona birçok lineeer olmayan fonksiyonlar mevcuttur. Bu kütüphanenin elemanları Şekil 6.6'da ki gibidir.

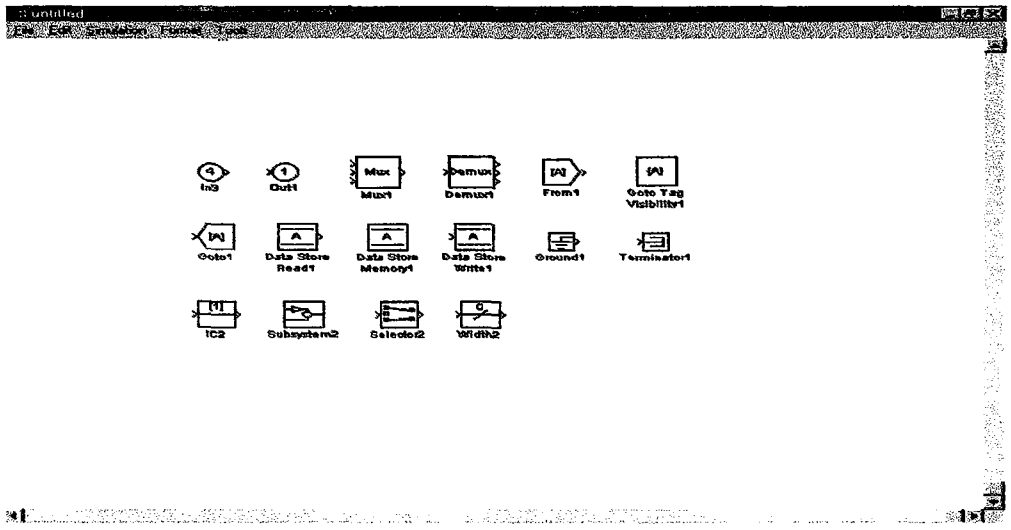
Connections kısmında ise toprak elemanından multiplexer (çoklama) elemanına kadar birçok eleman mevcuttur. Bu kütüphaneye ait elemanlar Şekil 6.7'de görülmektedir.



Şekil 6.5 Linear library



Şekil 6.6 Non-Linear library



Şekil 6.7 Connections library

Blocksets&Toolboxes kütüphanesinde ise bazı yardımcı araç kutuları ve bloksetler bulunmaktadır. Burdaki elemanlarında kendi içinde kütüphaneleri bulunmaktadır. Bu elemanlardan en önemlisi Stateflow'dur. Stateflow süren olaylar tarafından kontrol edilen sistemlerin dizaynı ve modellenmesinde kullanılır. Stateflow'dan başka haberleşme ve fuzzy-logic kontrole yönelik kütüphaneler de içermektedir.

6.4. DC Motorun matematiksel Modelinin Oluşturulması ve Simulink'le Modellenmesi

DC motoru Simulink'le modelleyebilmek için öncelikle motorun matematiksel modelini oluşturmak gerekir. Etiket değerleri belli olan motorun bu değerleri bir m-file hazırlanarak sisteme girilmelidir. Bu m-file ise aşağıda yazıldığı gibidir.

```
Pn=325      %w
Ua=120      %v
Ian=3.5     %A
nn=2650     %d/d
Ra=0.6      %ohm
Rf=33       %ohm
La=0.010    %h
Lf=0.460    %h
J=0.015     %Nms^2/rad
B=0.00156   %Nms/rad
ke=0.1168   %ke=ke'=km' endüklenen gerilim ve moment katsayısı birimsiz.
```

Burada P_n motorun nominal gücünü, u_a rotor uçlarına uygulanan gerilimi, i_{an} nominal endüi akımını, n_n nominal motor devrini, R_a endüvi direncini, R_f uyarma direncini, L_a endüvi self değerini, L_f uyarma self değerini, J motorun atalet momentini, B sürtünme katsayısını, ke ise endüvide endüklenen gerilimin katsayısı ve moment temsil eder. Bu durumda motorun diğer değişken ifadelerinin matematiksel modellerini oluşturmak gerekir. Bu değişken ifadelerde m_d makinanın endüvisinde endüklenen momenti, m_y yük momentini, ω açısal hızı, ϕ akıyı k_ϕ ise akı sabitini ifade eder.

İlk olarak I_a'nın matematiksel modeli oluşturulur.

$$v_a = (R_a \times i_a) + \left(L_a \times \frac{di_a}{dt}\right) + e_a \tag{6.1}$$

$$\frac{v_a}{L_a} = \left(\frac{R_a}{L_a} \times i_a\right) + \frac{di_a}{dt} + \frac{e_a}{L_a} \tag{6.2}$$

$$\frac{di_a}{dt} = \frac{v_a}{L_a} - \left(\frac{R_a}{L_a} \times i_a\right) - \frac{e_a}{L_a} \tag{6.3}$$

$$i_a = \int \left(\frac{v_a}{L_a} - \left(\frac{R_a}{L_a} \times i_a\right) - \frac{e_a}{L_a}\right) dt \tag{6.4}$$

i_a'nın matematiksel modeli 6.4 eşitliğinde görüldüğü gibi elde edilir. ω'nın matematiksel modelini aşağıdaki gibidir.

$$m_d - m_y = \left(J \times \frac{d\omega}{dt}\right) + (B \times \omega) \tag{6.5}$$

$$\frac{d\omega}{dt} = \frac{m_d}{J} - \frac{m_y}{J} - \frac{B \times \omega}{J} \tag{6.6}$$

$$\omega = \int \left(\frac{m_d}{J} - \frac{m_y}{J} - \frac{B \times \omega}{J}\right) dt \tag{6.7}$$

ω'nın matematiksel modeli de 6.7 eşitliğindeki gibi olur. i_f'nin matematiksel modeli ise aşağıdaki gibidir.

$$u_f = (R_f \times i_f) + \left(L_f \times \frac{di_f}{dt}\right) \qquad (6.8)$$

$$\frac{di_f}{dt} = \frac{u_f}{L_f} - \frac{R_f \times i_f}{L_f} \qquad (6.9)$$

$$i_f = \int \left(\frac{u_f}{L_f} - \frac{R_f \times i_f}{L_f}\right) dt \qquad (6.10)$$

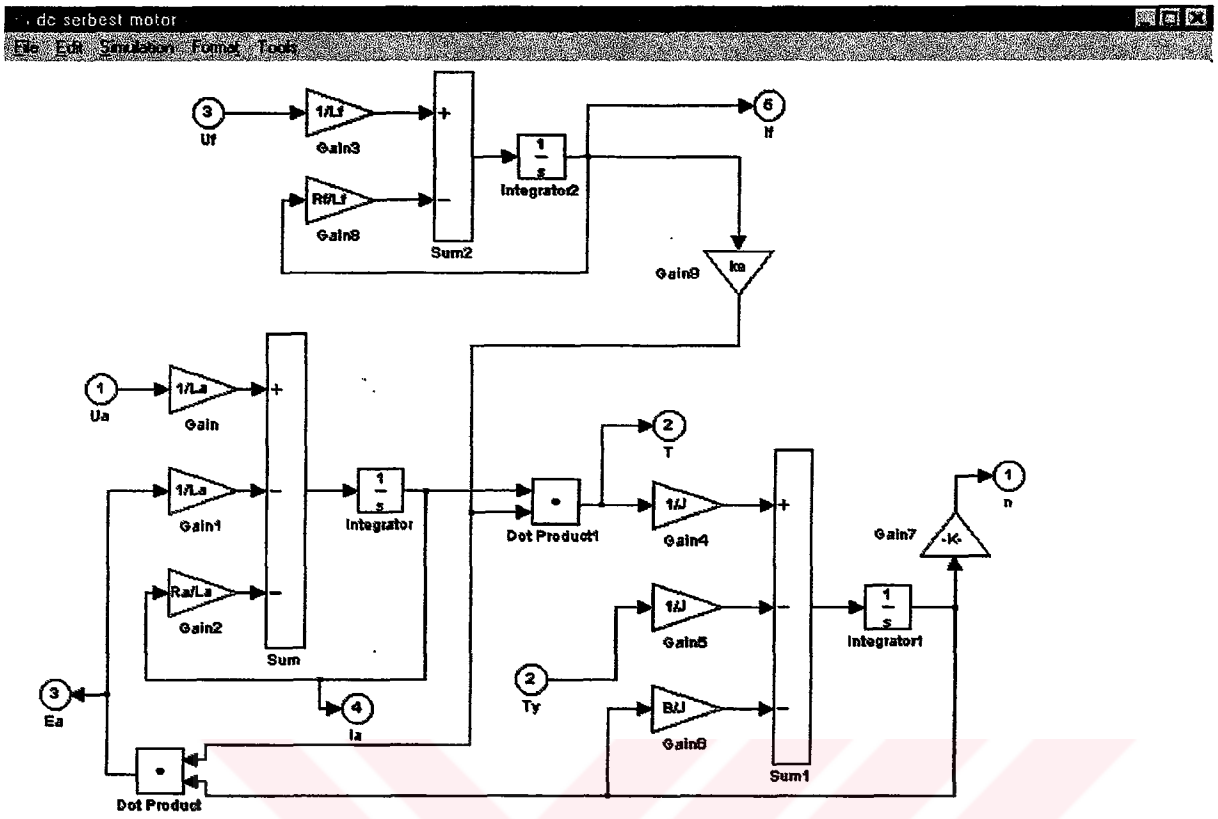
i_f'ninde matematiksel modeli 6.10 eşitliğinde ki gibi olur. Endüvi'de endüklenen gerilimin eşitliği ise 3.3. eşitliğindeki, akınınki de 3.1 eşitliğindeki gibidir. Bu şartlar altında m_d'nin yeni matematiksel modeli oluşturulabilir.

$$m_d = k_n \times \phi \times i_a \qquad (6.11)$$

$$\phi = k_\phi i_f \qquad (6.12)$$

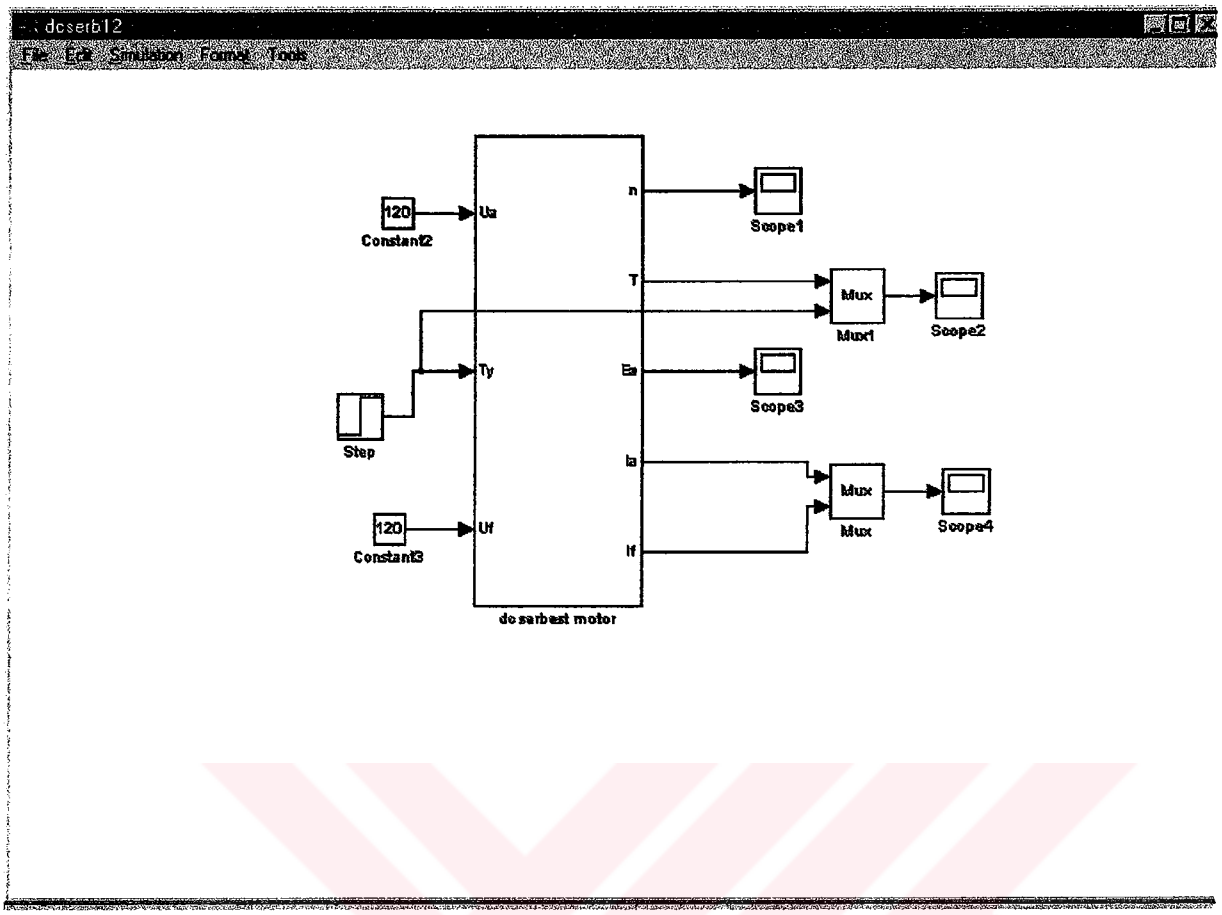
$$m_d = k_n' \times i_f \times i_a \qquad (6.13)$$

Böylece tüm değişkenlerin matametiksel modelleri oluşturulmuş oldu. Şimdi Simulink kütüphaneleri kullanılarak bu sistemi modellenebilir. Bu model Şekil 6.8'de ki gibi olur. Buradaki K sabiti $(60/2\pi)$dir. Bu değer ω'yı devir sayısına dönüştürmek için kullanılmıştır. Oluşturulan bu modele dışarıdan bir yük momenti girilerek n, Md, Ea, Ua ve I_f'nin incelenmesi ise aşağıda anlatıldığı gibidir. Bu değişimler çıkışlara bağlanan osiloskoplarla izlenebilir. Burada $T_y=M_y$ ve $T=M_d$'dir. Burada uygulanan yük momenti motor çalışmaya başladıktan 5sn sonra devreye gircek şekilde ayarlanmıştır. Uyarma ve giriş gerilimleri 120V olarak alınmıştır. Böylelikle etiket değerleri belli olan motorun yük altında çalışması simüle edilmiş olur.

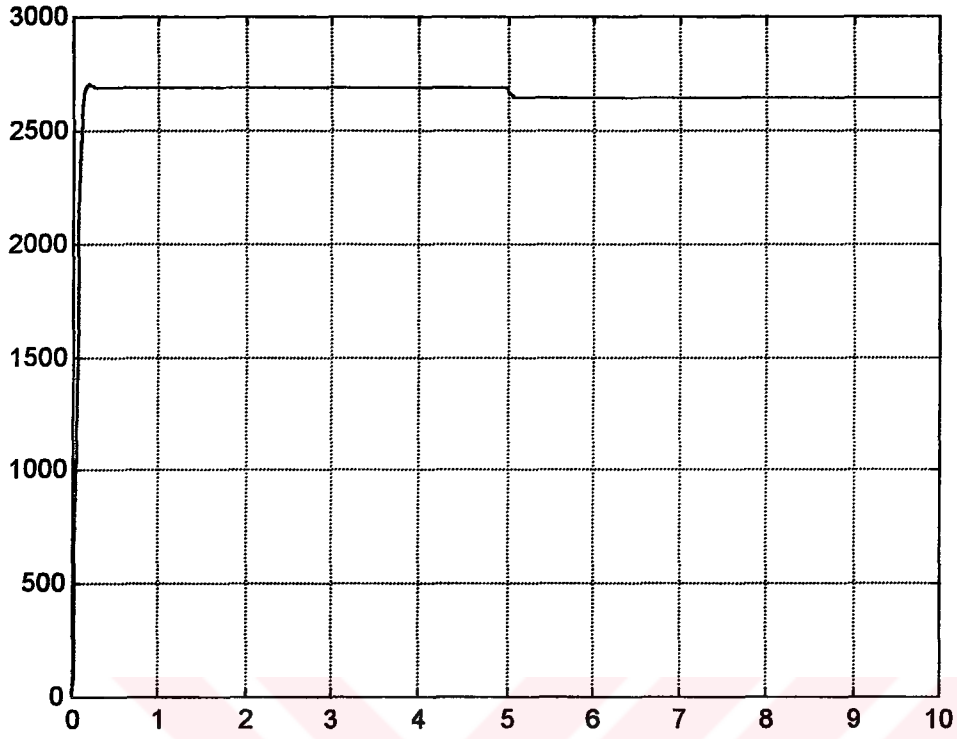


Şekil 6.8 Simulink'le gerçekleştirilen DC motor modeli

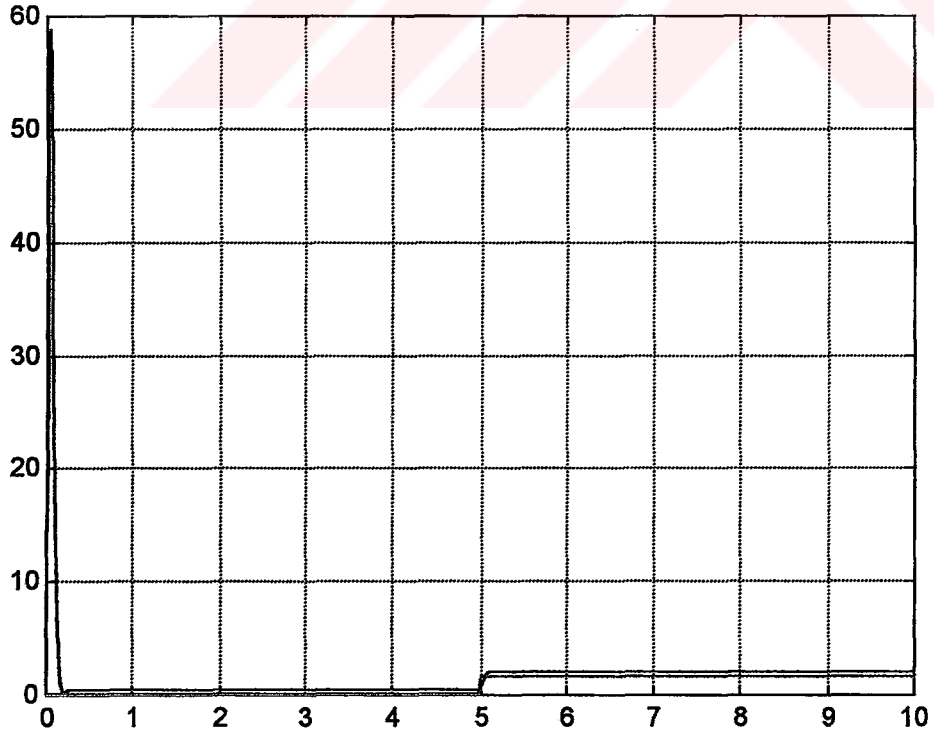
Bu devre çalıştırıldığında Scope 1'in gösterdiği değişim devir sayısının değişimi, Scope2'nin gösterdiği değişim yük momentinin değişimi, Scope 3'ün gösterdiği değişim endüvide endüklenen gerilimin değişimi, Scope 4'ün gösterdiği değişim ise endüvi akımı ve uyarma akımının değişimi olur. Bu değişimler Şekil 6.10, Şekil 6.11, Şekil 6.12 ve Şekil 6.13'de ki gibi olurlar.



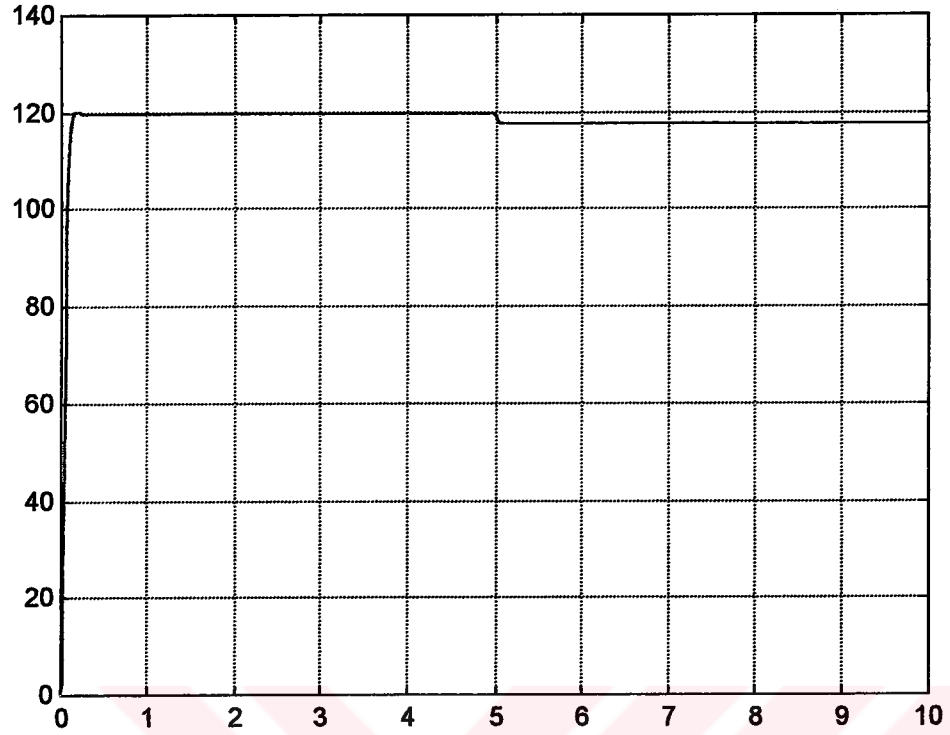
Şekil 6.9 Yük momenti altında DC motor modelinin oluşturulması ve değişimlerinin izlenmesi



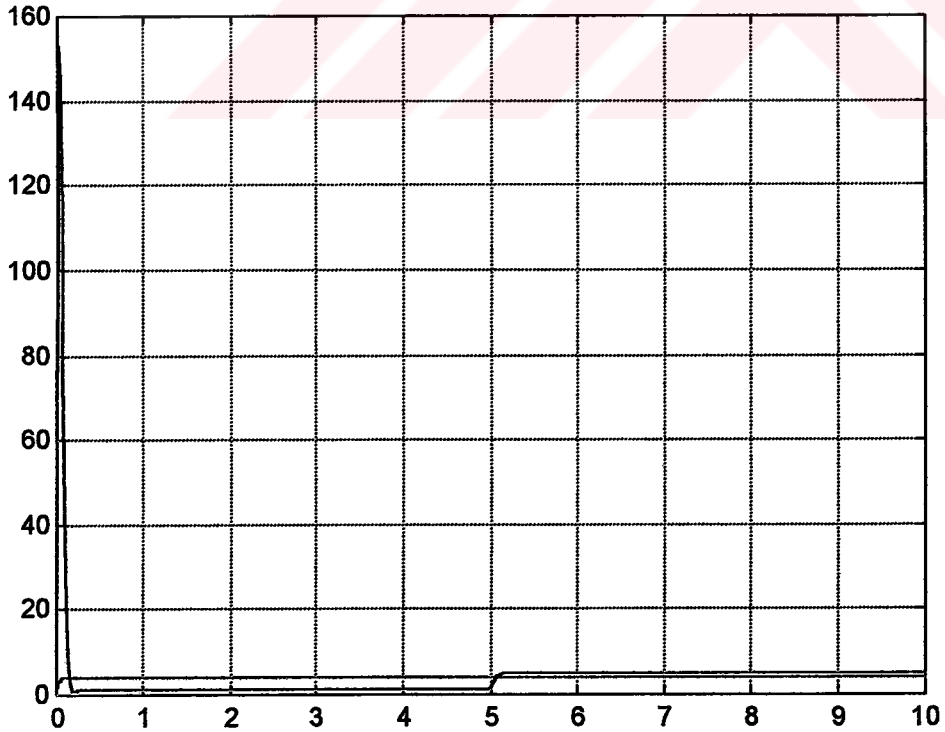
Şekil 6.10 Devirin zamana bağlı değişimi (osiloskop 1)



Şekil 6.11 Endüvide endüklenen moment ve yük momentinin zamana bağlı değişimleri (osiloskop 2)



Şekil 6.12 Endüvide endüklenen geriliminin zamana bağlı değişimi (osiloskop 3)

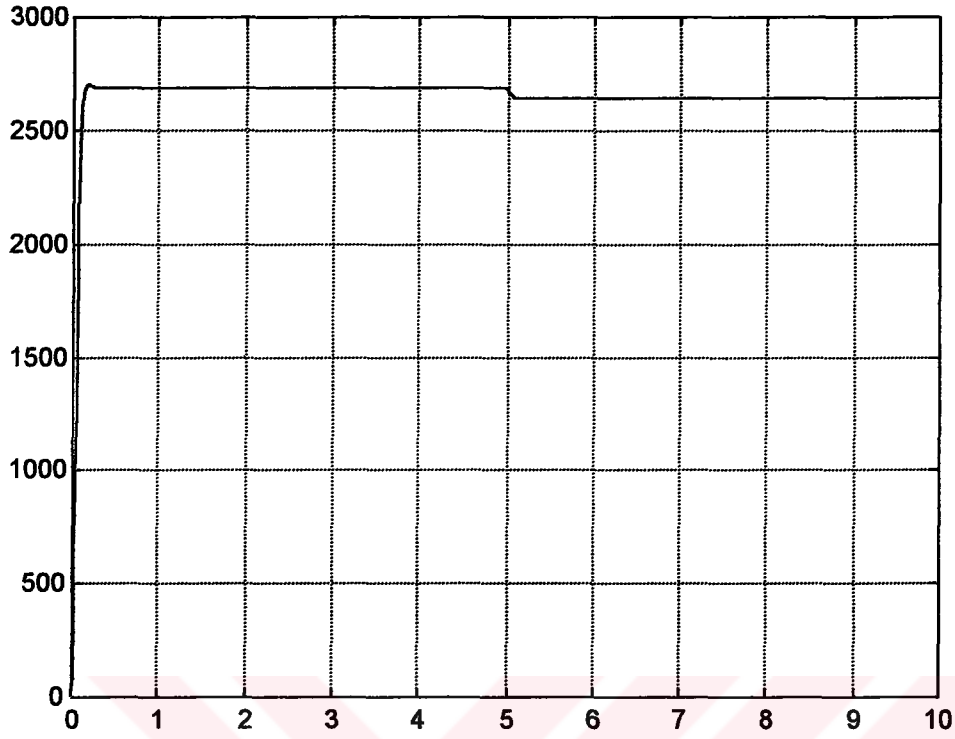


Şekil 6.13 Uyarma akımı ve endüvi akımının zamana bağlı değişimleri (osiloskop 4)

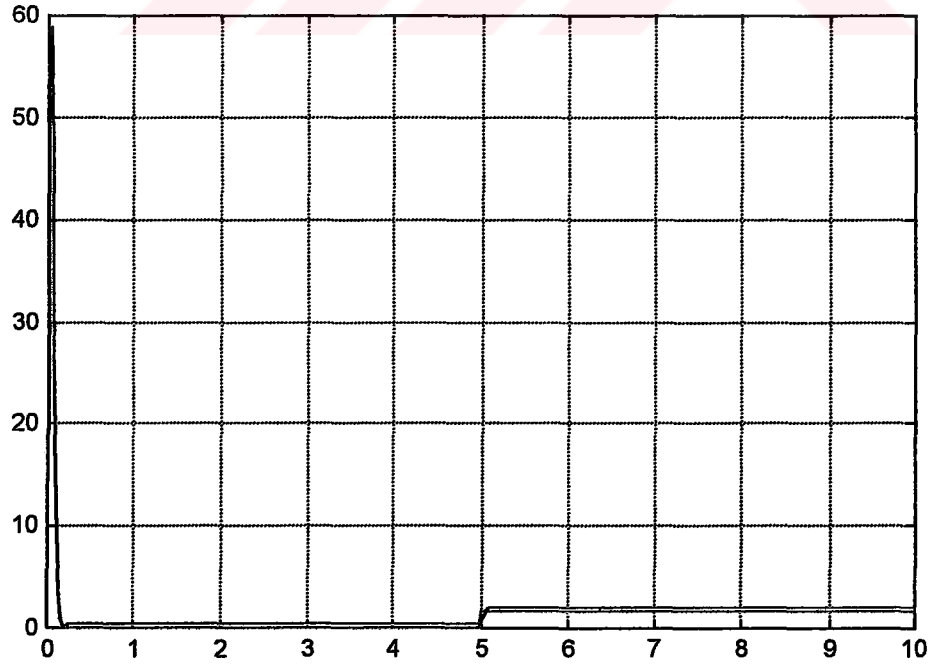
Gerçekleştirilen bu simülasyonda DC motorun mıknatıslanma karakterististiği ve endüvi karakteristiği dikkate alınmamıştır. Ayrıca yaklaşık motor parametreleri kullanılmıştır. Endüvi geriliminin zamana bağlı değişimi ve devrin zamana bağlı değişimleri pratikteki değerlere oldukça yakındır. Akım ve moment'in başlangıç değerleri pratikteki değerlerden fazla çıkmıştır. Ancak unutulmamalıdır ki DC motor kalkış anında nominal akımının 5 ila 50 katı arasında akımlar çekebilir. Bu sebeple akımın değişiminin ve akıma bağlı olan moment değişimilerinin değerleri normal kabul edilebilir.

Simulink'le görsel olarak gerçekleştirilen bu simülasyonu program olarak m-file ile de simüle etmek mümkündür. Ancak bu işlem Simulink'e oranla daha zordur. Simulink'le gerçekleştirilen bu simülasyonun m-file hali Ek.1'de verilmiştir.

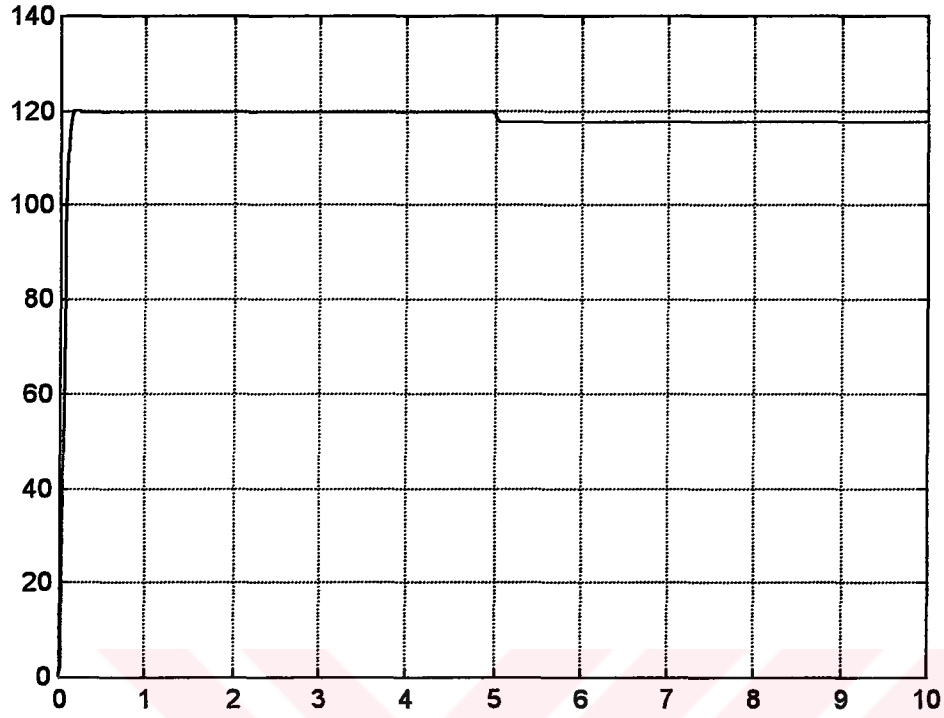
Aşağıda verilen simülasyon örneğinde ise yük, motor çalışmaya başladıktan 2.5sn sonra devreye girecek şekilde ayarlanmıştır. Uyarma geriliminin değeri ise 100V olarak ayarlanmıştır.



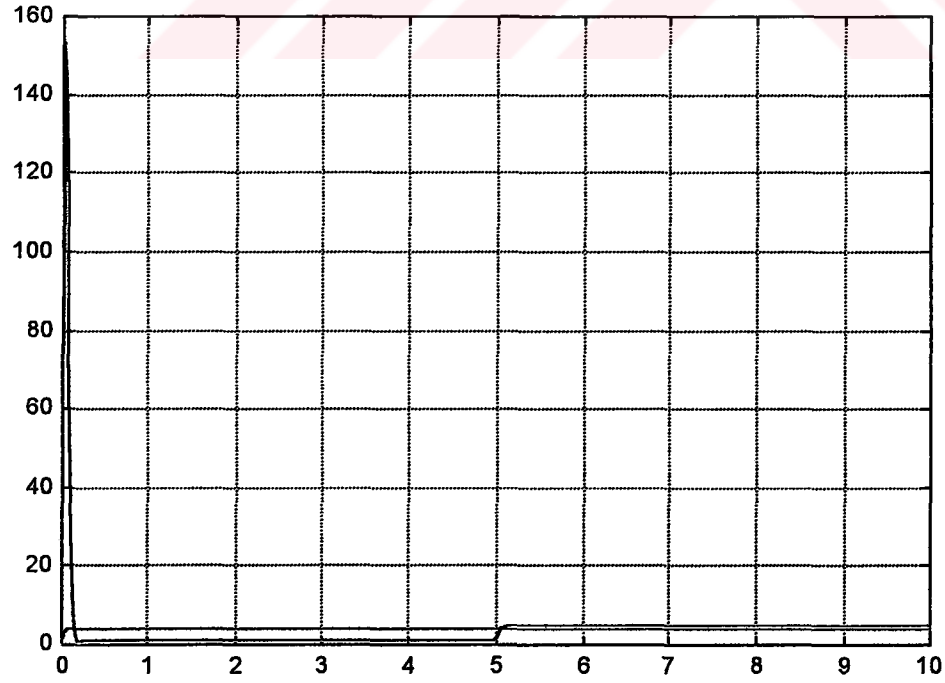
Şekil 6.14 Uyarma akımının 100V ve yük momentinin başlangıç süresinin 2.5sn alındığı simülasyonda devirin zamana bağlı değişimi



Şekil 6.15 Uyarma akımının 100V ve yük momentinin başlangıç süresinin 2.5sn alındığı simülasyonda endüvide endüklenen moment ve yük momentlerinin zamana bağlı değişimi



Şekil 6.16 Uyarma akımının 100V ve yük momentinin başlangıç süresinin 2.5sn alındığı simülasyonda endüvide endüklenen geriliminin zamana bağlı değişimi



Şekil 6.17 Uyarma akımının 100V ve yük momentinin başlangıç süresinin 2.5sn alındığı simülasyonda uyarma akımı ve endüvi akımlarının zamana bağlı değişimleri

7. SONUÇ

Son yıllarda modelleme ve bilgisayar simülasyonları mühendislik alanında sistemlerin öğrenilmesi, dizaynı ve analizinde önemli rol oynamaya başlamıştır. Simülasyonlarla gerek maliyet gerek zaman açısından verimin artması sağlanmıştır. Bu tür simülasyonlarda simüle edilen sistemin basitleştirilmesi önemlidir.

MATLAB'ın uygun toolbox'larından dolayı motor simülasyonu kolaylıkla gerçeklenebilir. Ancak program içindeki integrasyon ve diferansiyel denklem algoritmaları açık olarak yazılımcı firma tarafından ifade edilmemiştir. Bu yüzden simülasyon esnasında algoritma tarafından parametrelerde yapılan yuvarlamalar kesin olarak bilinmemektedir. Simüle edilen sistem yalınlaştırılarak hata oranı azaltılmalıdır.

KAYNAKLAR

Fitzgerald, A.E., Kingsley, C. ve Umans, S.D. (1986), Electric Machinery, McGraw-Hill Book Company

Gülgün, R. (1995), Güç Elektroniği, Yıldız Teknik Üniversitesi Yayınları, 302, İstanbul

Güneş, A. ve Yıldız, K. (1997), Matlab for Windows,Türkmen Kitabevi, İstanbul

Lander, C.W. (1993), Power Electronics, McGraw-Hill Book Company, Third Edition

Mohan, N. , Robbins, W.P. ve Undeland, T.M. ,Power Electronics Converters, Applications and Design

Yüksel, İ. (1996), Matlab ile Mühendislik Sistemlerinin Analizi ve Çözümü, Uludağ Üniversitesi Bilimsel Araştırma Basım ve Yayın İşletmesi, Bursa

Vithayathill, J. (1995), Power Electronics Principles and Applications, McGraw-Hill Book Company, International Edition

The Mathworks Inc. (1997), Simulink Dynamic System Simulation for Matlab User's Guide

Ek-1 Simulink'le Gerçekleştirilen DC Motor Simülasyonun m-file Şeklinde Yazılımı

```
Model {
  Name          "dcserb12"
  Version       2.09
  SimParamPage          Solver
  SampleTimeColors      off
  InvariantConstants    off
  WideVectorLines       off
  ShowLineWidths  off
  PaperOrientation  landscape
  PaperType         usletter
  PaperUnits        inches
  StartTime       "0.0"
  StopTime        "10.0"
  Solver          ode45
  RelTol          "1e-3"
  AbsTol          "1e-6"
  Refine          "1"
  MaxStep         "auto"
  InitialStep           "auto"
  FixedStep             "auto"
  MaxOrder              5
  OutputOption              RefineOutputTimes
  OutputTimes       "[]"
  LoadExternalInput     off
  ExternalInput       "[t, u]"
  SaveTime            on
  TimeSaveName          "tout"
  SaveState       off
  StateSaveName         "xout"
  SaveOutput          on
  OutputSaveName        "yout"
  LoadInitialState    off
  InitialState          "xInitial"
  SaveFinalState    off
  FinalStateName    "xFinal"
  LimitMaxRows          off
  MaxRows           "1000"
  Decimation        "1"
  AlgebraicLoopMsg    warning
  MinStepSizeMsg  warning
  UnconnectedInputMsg     warning
  UnconnectedOutputMsg  warning
  UnconnectedLineMsg      warning
  ConsistencyChecking       off
  ZeroCross           on
  SimulationMode  normal
  RTWSystemTargetFile     "grt.tlc"
```

```
RTWInlineParameters     off
RTWRetainRTWFile        off
RTWTemplateMakefile     "grt_vc.tmf"
RTWMakeCommand          "make_rtw"
RTWGenerateCodeOnly     off
ExtModeMexFile          "ext_comm"
ExtModeBatchMode        off
BlockDefaults {
  Orientation           right
  ForegroundColor       black
  BackgroundColor       white
  DropShadow            off
  NamePlacement         normal
  FontName              "Helvetica"
  FontSize              10
  FontWeight            normal
  FontAngle             normal
  ShowName              on
}
AnnotationDefaults {
  HorizontalAlignment   center
  VerticalAlignment     middle
  ForegroundColor       black
  BackgroundColor       white
  DropShadow            off
  FontName              "Helvetica"
  FontSize              10
  FontWeight            normal
  FontAngle             normal
}
LineDefaults {
  FontName              "Helvetica"
  FontSize              9
  FontWeight            normal
  FontAngle             normal
}
System {
  Name                  "dcserb12"
  Location              [0, 38, 800, 588]
  Open                  on
  ScreenColor           white
  Block {
    BlockType           Constant
    Name                "Constant1"
    Position            [75, 180, 95, 200]
    Value               "0"
  }
  Block {
    BlockType           Constant
```

```
  Name            "Constant2"
  Position            [245, 85, 265, 105]
  Value           "120"
 }
 Block {
  BlockType           Constant
  Name            "Constant3"
  Position            [240, 295, 260, 315]
  Value           "120"
 }
 Block {
  BlockType           Mux
  Name            "Mux"
  Ports           [2, 1, 0, 0, 0]
  Position            [540, 262, 575, 298]
  Inputs          "2"
 }
 Block {
  BlockType           Mux
  Name            "Mux1"
  Ports           [2, 1, 0, 0, 0]
  Position            [540, 132, 575, 168]
  Inputs          "2"
 }
 Block {
  BlockType           Scope
  Name            "Scope"
  Ports           [1, 0, 0, 0, 0]
  Position            [490, 65, 520, 95]
  Floating            off
  Location            [182, 104, 616, 366]
  Open            on
  Grid            on
  TickLabels      on
  ZoomMode              yonly
  TimeRange             "10"
  YMin            "-500"
  YMax            "3500"
  SaveToWorkspace       off
  SaveName            "ScopeData"
  LimitMaxRows          on
  MaxRows             "5000"
  Decimation          "1"
  SampleInput         off
  SampleTime          "0"
 }
 Block {
  BlockType           Scope
  Name            "Scope1"
```

```
  Ports            [1, 0, 0, 0, 0]
  Position         [610, 135, 640, 165]
  Floating         off
  Location         [5, 30, 809, 583]
  Open             off
  Grid             on
  TickLabels       on
  ZoomMode         on
  TimeRange        "auto"
  YMin             "-5"
  YMax             "5"
  SaveToWorkspace  off
  SaveName         "ScopeData"
  LimitMaxRows     on
  MaxRows          "5000"
  Decimation       "1"
  SampleInput      off
  SampleTime       "0"
}
Block {
  BlockType        Scope
  Name             "Scope2"
  Ports            [1, 0, 0, 0, 0]
  Position         [490, 185, 520, 215]
  Floating         off
  Location         [161, 216, 485, 455]
  Open             on
  Grid             on
  TickLabels       on
  ZoomMode         on
  TimeRange        "auto"
  YMin             "-5"
  YMax             "5"
  SaveToWorkspace  off
  SaveName         "ScopeData"
  LimitMaxRows     on
  MaxRows          "5000"
  Decimation       "1"
  SampleInput      off
  SampleTime       "0"
}
Block {
  BlockType        Scope
  Name             "Scope4"
  Ports            [1, 0, 0, 0, 0]
  Position         [615, 265, 645, 295]
  Floating         off
  Location         [5, 30, 809, 583]
  Open             off
```

```
  Grid            on
  TickLabels      on
  ZoomMode            on
  TimeRange           "auto"
  YMin            "-5"
  YMax            "10"
  SaveToWorkspace         off
  SaveName            "ScopeData"
  LimitMaxRows        on
  MaxRows             "5000"
  Decimation          "1"
  SampleInput         off
  SampleTime          "0"
}
Block {
  BlockType           Step
  Name            "Step"
  Position            [215, 215, 245, 245]
  Time            "5"
  Before          "0"
  After           "1.5"
}
Block {
  BlockType           SubSystem
  Name            "dc serbest motor"
  Ports           [3, 5, 0, 0, 0]
  Position            [305, 44, 415, 356]
  ShowPortLabels      on
  System {
  Name            "dc serbest motor"
  Location        [0, 38, 800, 604]
  Open            on
  ScreenColor         white
  Block {
    BlockType           Inport
    Name            "Ua"
    Position        [55, 215, 75, 235]
    Port            "1"
    PortWidth           "-1"
    SampleTime          "-1"
  }
  Block {
    BlockType           Inport
    Name            "Ty"
    Position        [340, 415, 360, 435]
    Port            "2"
    PortWidth           "-1"
    SampleTime          "-1"
  }
```

```
Block {
  BlockType       Inport
  Name            "Uf"
  Position        [125, 25, 145, 45]
  Port            "3"
  PortWidth       "-1"
  SampleTime      "-1"
}
Block {
  BlockType       Reference
  Name            "Dot Product"
  Ports           [2, 1, 0, 0, 0]
  Position        [90, 472, 120, 503]
  Orientation     left
  SourceBlock     "simulink/Linear/Dot Product"
  SourceType      "Dot Product"
}
Block {
  BlockType       Reference
  Name            "Dot Product1"
  Ports           [2, 1, 0, 0, 0]
  Position        [350, 287, 380, 318]
  SourceBlock     "simulink/Linear/Dot Product"
  SourceType      "Dot Product"
}
Block {
  BlockType       Gain
  Name            "Gain"
  Position        [110, 209, 155, 241]
  Gain            "1/La"
}
Block {
  BlockType       Gain
  Name            "Gain1"
  Position        [110, 279, 155, 311]
  Gain            "1/La"
}
Block {
  BlockType       Gain
  Name            "Gain2"
  Position        [110, 349, 155, 381]
  Gain            "Ra/La"
}
Block {
  BlockType       Gain
  Name            "Gain3"
  Position        [200, 19, 245, 51]
  Gain            "1/Lf"
}
```

```
Block {
  BlockType        Gain
  Name             "Gain4"
  Position         [430, 289, 475, 321]
  Gain             "1/J"
}
Block {
  BlockType        Gain
  Name             "Gain5"
  Position         [430, 359, 475, 391]
  Gain             "1/J"
}
Block {
  BlockType        Gain
  Name             "Gain6"
  Position         [430, 429, 475, 461]
  Gain             "B/J"
}
Block {
  BlockType        Gain
  Name             "Gain7"
  Position         [607, 310, 653, 340]
  Orientation      up
  NamePlacement    alternate
  Gain             "60/(2*3.14159)"
}
Block {
  BlockType        Gain
  Name             "Gain8"
  Position         [200, 79, 245, 111]
  Gain             "Rf/Lf"
}
Block {
  BlockType        Gain
  Name             "Gain9"
  Position         [462, 130, 508, 160]
  Orientation      down
  NamePlacement    alternate
  Gain             "ke"
}
Block {
  BlockType        Integrator
  Name             "Integrator"
  Ports            [1, 1, 0, 0, 0]
  Position         [235, 280, 265, 310]
  ExternalReset    none
  InitialConditionSource  internal
  InitialCondition    "0"
  LimitOutput      off
```

```
  UpperSaturationLimit      "inf"
  LowerSaturationLimit      "-inf"
  ShowSaturationPort    off
  ShowStatePort         off
  AbsoluteTolerance     "auto"
}
Block {
  BlockType         Integrator
  Name          "Integrator1"
  Ports         [1, 1, 0, 0, 0]
  Position      [570, 360, 600, 390]
  ExternalReset         none
  InitialConditionSource  internal
  InitialCondition    "0"
  LimitOutput       off
  UpperSaturationLimit      "inf"
  LowerSaturationLimit      "-inf"
  ShowSaturationPort    off
  ShowStatePort         off
  AbsoluteTolerance     "auto"
}
Block {
  BlockType         Integrator
  Name          "Integrator2"
  Ports         [1, 1, 0, 0, 0]
  Position      [340, 50, 370, 80]
  ExternalReset         none
  InitialConditionSource  internal
  InitialCondition    "0"
  LimitOutput       off
  UpperSaturationLimit      "inf"
  LowerSaturationLimit      "-inf"
  ShowSaturationPort    off
  ShowStatePort         off
  AbsoluteTolerance     "auto"
}
Block {
  BlockType         Sum
  Name          "Sum"
  Ports         [3, 1, 0, 0, 0]
  Position      [175, 187, 200, 403]
  Inputs        "+--"
}
Block {
  BlockType         Sum
  Name          "Sum1"
  Ports         [3, 1, 0, 0, 0]
  Position      [510, 267, 535, 483]
  Inputs        "+--"
```

```
 }
 Block {
  BlockType        Sum
  Name             "Sum2"
  Ports            [2, 1, 0, 0, 0]
  Position         [285, 7, 310, 123]
  Inputs           "+-"
 }
 Block {
  BlockType        Outport
  Name             "n"
  Position         [655, 275, 675, 295]
  Port             "1"
  OutputWhenDisabled      held
  InitialOutput        "0"
 }
 Block {
  BlockType        Outport
  Name             "T"
  Position         [430, 240, 450, 260]
  Port             "2"
  OutputWhenDisabled      held
  InitialOutput        "0"
 }
 Block {
  BlockType        Outport
  Name             "Ea"
  Position         [25, 425, 45, 445]
  Orientation         left
  Port             "3"
  OutputWhenDisabled      held
  InitialOutput        "0"
 }
 Block {
  BlockType        Outport
  Name             "Ia"
  Position         [225, 435, 245, 455]
  Port             "4"
  OutputWhenDisabled      held
  InitialOutput        "0"
 }
 Block {
  BlockType        Outport
  Name             "If"
  Position         [495, 20, 515, 40]
  Port             "5"
  OutputWhenDisabled      held
  InitialOutput        "0"
 }
```

```
Line {
  SrcBlock       "Gain7"
  SrcPort        1
  Points         [0, -20]
  DstBlock       "n"
  DstPort        1
}
Line {
  SrcBlock       "Dot Product1"
  SrcPort        1
  Points         [10, 0]
  Branch {
    Points         [0, -55]
    DstBlock           "T"
    DstPort        1
  }
  Branch {
    DstBlock           "Gain4"
    DstPort        1
  }
}
Line {
  SrcBlock       "Dot Product"
  SrcPort        1
  Points         [-15, 0; 0, -55]
  Branch {
    DstBlock           "Ea"
    DstPort        1
  }
  Branch {
    Points         [0, -140]
    DstBlock           "Gain1"
    DstPort        1
  }
}
Line {
  SrcBlock       "Ty"
  SrcPort        1
  Points         [40, 0; 0, -50]
  DstBlock       "Gain5"
  DstPort        1
}
Line {
  SrcBlock       "Ua"
  SrcPort        1
  DstBlock       "Gain"
  DstPort        1
}
Line {
```

```
  SrcBlock      "Integrator1"
  SrcPort       1
  Points        [25, 0]
  Branch {
   DstBlock         "Gain7"
   DstPort          1
  }
  Branch {
   Points           [0, 130; -230, 0]
   Branch {
    Points          [0, -60]
    DstBlock        "Gain6"
    DstPort         1
   }
   Branch {
    Points          [-265, 0]
    DstBlock        "Dot Product"
    DstPort         2
   }
  }
}
Line {
  SrcBlock      "Gain6"
  SrcPort       1
  DstBlock      "Sum1"
  DstPort       3
}
Line {
  SrcBlock      "Gain5"
  SrcPort       1
  DstBlock      "Sum1"
  DstPort       2
}
Line {
  SrcBlock      "Gain4"
  SrcPort       1
  DstBlock      "Sum1"
  DstPort       1
}
Line {
  SrcBlock      "Sum1"
  SrcPort       1
  DstBlock      "Integrator1"
  DstPort       1
}
Line {
  SrcBlock      "Sum"
  SrcPort       1
  DstBlock      "Integrator"
```

```
  DstPort        1
}
Line {
  SrcBlock       "Gain2"
  SrcPort        1
  DstBlock       "Sum"
  DstPort        3
}
Line {
  SrcBlock       "Gain1"
  SrcPort        1
  DstBlock       "Sum"
  DstPort        2
}
Line {
  SrcBlock       "Gain"
  SrcPort        1
  DstBlock       "Sum"
  DstPort        1
}
Line {
  SrcBlock       "Gain3"
  SrcPort        1
  DstBlock       "Sum2"
  DstPort        1
}
Line {
  SrcBlock       "Gain8"
  SrcPort        1
  DstBlock       "Sum2"
  DstPort        2
}
Line {
  SrcBlock       "Sum2"
  SrcPort        1
  DstBlock       "Integrator2"
  DstPort        1
}
Line {
  SrcBlock       "Integrator2"
  SrcPort        1
  Points         [10, 0]
  Branch {
    Points         [100, 0]
    DstBlock           "Gain9"
    DstPort        1
  }
  Branch {
    Points         [0, 80; -200, 0]
```

```
    DstBlock        "Gain8"
    DstPort         1
   }
   Branch {
    Points          [0, -35]
    DstBlock        "If"
    DstPort         1
   }
  }
  Line {
   SrcBlock        "Integrator"
   SrcPort         1
   Points          [10, 0]
   Branch {
    DstBlock        "Dot Product1"
    DstPort         1
   }
   Branch {
    Points          [0, 130; -70, 0]
    Branch {
     Points          [-115, 0]
     DstBlock        "Gain2"
     DstPort         1
    }
    Branch {
     DstBlock        "Ia"
     DstPort         1
    }
   }
  }
  Line {
   SrcBlock        "Gain9"
   SrcPort         1
   Points          [0, 30; -175, 0; 0, 115]
   Branch {
    Points          [0, 160; -175, 0]
    DstBlock        "Dot Product"
    DstPort         1
   }
   Branch {
    DstBlock        "Dot Product1"
    DstPort         2
   }
  }
  Line {
   SrcBlock        "Uf"
   SrcPort         1
   DstBlock        "Gain3"
   DstPort         1
```

```
    }
    }
  }
  Line {
    SrcBlock        "Constant2"
    SrcPort         1
    DstBlock        "dc serbest motor"
    DstPort         1
  }
  Line {
    SrcBlock        "dc serbest motor"
    SrcPort         1
    DstBlock        "Scope"
    DstPort         1
  }
  Line {
    SrcBlock        "dc serbest motor"
    SrcPort         3
    DstBlock        "Scope2"
    DstPort         1
  }
  Line {
    SrcBlock        "dc serbest motor"
    SrcPort         4
    Points          [105, 0]
    DstBlock        "Mux"
    DstPort         1
  }
  Line {
    SrcBlock        "dc serbest motor"
    SrcPort         5
    Points          [105, 0]
    DstBlock        "Mux"
    DstPort         2
  }
  Line {
    SrcBlock        "Mux"
    SrcPort         1
    DstBlock        "Scope4"
    DstPort         1
  }
  Line {
    SrcBlock        "Mux1"
    SrcPort         1
    DstBlock        "Scope1"
    DstPort         1
  }
  Line {
    SrcBlock        "dc serbest motor"
```

```
      SrcPort              2
      DstBlock             "Mux1"
      DstPort              1
    }
    Line {
      SrcBlock             "Constant3"
      SrcPort              1
      DstBlock             "dc serbest motor"
      DstPort              3
    }
    Line {
      SrcBlock             "Step"
      SrcPort              1
      Points               [15, 0; 0, -30; 5, 0]
      Branch {
      DstBlock             "dc serbest motor"
      DstPort              2
      }
      Branch {
      Points               [0, -40]
      DstBlock             "Mux1"
      DstPort              2
      }
    }
  }
}
```

ÖZGEÇMİŞ

Doğum tarihi	10.10.1975	
Doğum yeri	İzmir	
Lise	1986-1993	Bornova Anadolu Lisesi
Lisans	1993-1997	Yıldız Teknik Üniversitesi Elektrik-Elektronik Fak. Elektrik Müh. Bölümü
Yüksek Lisans	1997-2000	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektrik Müh. Ana Bilim Dalı

Çalıştığı kurumlar

1997-1999	Cegelec AEG Tesis ve Otomasyon San. Ve Tic.
2000-Devam ediyor	Novatek İnşaat San.ve Tic.