İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

SÖZDİZİMİ METODU İLE ELEKTROKARDİYOGRAFİ İŞARETLERİNİN ANALİZİ

YÜKSEK LİSANS TEZİ

Elektronik Müh. Sergin ÖZENÇ

Tezin Enstitüye Verildiği Tarih : 15 Ocak 1996

Tezin Savunulduğu Tarih : 29 Ocak 1996

Tez Danışmanı :Doç. Dr. Mehmet KORÜREK

Diğer Jüri Üyeleri :Prof. Dr. Ahmet Hamdi KAYRAN

Yrd. Doç. Dr. İnci ÇİLESİZ

OCAK 1996

ÖNSÖZ

Bu tez çalışmasındaki yardımlarından dolayı Sayın Doç. Dr. Mehmet Korürek' e ve maddi ve manevi desteklerini esirgemeyen tüm arkadaşlarıma teşekkürü bir borç bilirim.

0cak 1996 Sergin ÖZENÇ

İÇİNDEKİLER

ÖZET

Aritmiler temel olarak iki ana guruba ayrılabilir. Birincisi **ritim bozuklukları** veya düzensizlikleri, ikincisi dalga şeklindeki **şekil bozukluklarıdır**. Aritmilerin kliniksel önemi anlaşıldıkça, bu olayları ortaya çıkartacak güvenilir ve otomatik dedektörlerin geliştirilmesine ihtiyaç duyuldu. EKG' deki şekil bozukluklarının şu anki sistemlerle gerçek zamanda otomatik olarak bulunması oldukça zor olmaktadır. Düşük işaret/gürültü oranları ve hesaplamadaki kısıtlamalardan dolayı, günümüzdeki teknoloji, P ve T dalgalarının gerçek zamanda, standart yüzey monitör elektrodları ile belirlenmesinde yetersiz kalmaktadır. Analizin bu şekilde tamamlanamamasından dolayı, işlem sadece QRS morfolojisi ve zamanlama üzerine yorum getirmektedir.

Sözdizimi metodunun bu zorlukların aşılmasına ve problemin çözümüne ne oranda katkı yapabileceğini görebilmek için, ilk etapta, en kolay problem olan QRS deteksiyonu, sözdizimi metoduyla, bir PC de gerekli programı yazmak suretiyle gerçeklendi. Sözdizimi metodu ile deteksiyon yapmanın ilk basamağı işaretin ilksellerinin belirlenmesidir. Bunun için Gustavo Belforte' un sözdizimi metoduyla QRS deteksiyonu için ortaya attığı algoritmadan yararlanıldı. Buna göre, filtrelenmiş EKG' nin türev karesi alınarak, genlik belli sayıda eşik seviyelerine bölünmüş ve her eşik seviyesi terminal elemanları olarak bir küçük harf ile sembolize edilmişlerdir. Bu şekilde QRS komplekslerinin sözdizimleri çıkartıldı. Sözdizimi metodu ile şekil tanıma probleminde, biri öğrenme modu diğeri de çalışma modu olmak üzere iki ana bölüm vardır. Öğrenme modunda aynı sınıfa ait işaretlerin sözdizimleri çıkartılarak o sınıfa ait bir gramer oluşturulur. Çalışma modunda da gelen işaretlerin sözdizimleri çıkartılarak, bu sözdizimlerinin söz konusu sınıfa ait gramere uyup uymadığı kontrol edilir. Eğer uyuyorsa işaret bu sınıftandır denir. Yapılan bilgisayar programında bu anlatılan işlemler gerçekleştirilerek QRS deteksiyonu başarıldı.

P ve T tepelerinin de dedekte edilebilmesi için Gustavo Belforte tarafından ortaya atılan algoritma geliştirildi. EKG' nin türev karesini almak yerine bölüm 4' te anlatılan formül uygulandı. Bu şekilde hem QRS komplekslerini hem de P ve T tepelerini dedekte etme imkanı elde edildi. Daha ileriki aşamada bir periyodluk EKG işaretinin şeklini tanıyabilmek için, bir periyodluk EKG' nin sözdizimini verebilecek şekilde algoritma geliştirildi. Ancak bu son algoritmanın gramatik yorum bölümünde bazı problemlerle karşılaşıldı. Bu problemlerin sebebi ve çözüm için öneriler son bölümde anlatıldı. Fakat bu çözümleri bilgisayarda deneme imkanı olmadı.

Bu tez çalışmasında, Gustavo Belforte' un sözdizimi metoduyla QRS deteksiyonu için ortaya attığı algoritmadan yola çıkmakla beraber, bu algoritmaya bağlı kalınmamış, QRS deteksiyonundan daha ileri amaçları gerçekleştirebilmek için algoritma geliştirilerek konuya yeni bir boyut katılmıştır.

SUMMARY

ANALYSİS OF ECG SİGNALS BY SYNTACTİC METHOD

Cardiologs can classify EKG waves as normal and anormal, by looking at their wave forms. A computer can help a cardiolog in reading recording of EKG. For instance, in intensive care units the EKG taken from many patients is recorded continuosly. Auto-watching programs are used in the stiuations that cardiologs don't make any recording for 24 hours, to determine vital dangers and abnormal heart beats. Today, computer added EKG monitors, recording and analyses of EKG are widely known and used. The function of auto-arrhytmia analyzers can be summerized as choosing datas that have clinical importance out of many non-processed datas.

The datas that a doctor can look through are much more less than the datas which an automatic system analyses. Many automatic systems determine, measure, classify, QRS complexes one by one and according to their timing and beating series, recognize known arrhytmies.

An electronic system, which can notify the vital ones out of many arrhytmies can be disigned in differnt ways. Arrhytmies are divided into two main groups. One is rhythm defects or disorders, the other is shape defects in wave forms. Rhythm defect is known as sinus arrhytmia. Normally heart beats 70 times per minute in adults. This number can change due to age, physological efort and physichological stiuation. Some of these disorders in the rhythm are temporal, if there is no phatological defect. But, if there is a phatological defect, a permanent disorder is observed in the rhythm. If heart beats much more slower than normal, it is called bradycardia, and if it beats much more faster than normal it is called tachycardia. So, the machine must test the heart rhythm if it is between these minimum and maximum values. Therefore it is enaugh to make comperasion between minimum, maximum values and the value of a counter that counts between R-R.

Shape defects are the abnormalities in shapes of some important waves in EKG signals. The must important ones are the distance to QRS complex of P, amplitude of P and the distance to QRS complex of T, amplitude of T; rising and falling that happens due to izoelectrical line at the S-T segment.

Because of low signal/noise ratios and restrictions in the calculations; todays' technology is not enough to determine P and T waves at the real time, by standart surface monitor electrods. This process can only commend on QRS morphology and timing because analyses of EKG is not completed in this way.

Its quite hard to find the EKG shape defects automatically with todays' systems. Syntactic method can ease to overcome these difficulties mentioned in automatical arryhtmia dedection. Because this method uses structural knowladge for recognizing and classifying signals and according to the older approaching of decision theory, it is more likely to diagnoises of human.

To carry out these purposes and to see how much the syntactic method can contribute to this problem, firstly QRS dedection which is the easiest problem is carried out by syntactic method.

First step in dedection by syntactic method is to determine primitives of the signal. For this, the algorythm prepared by Gustave Belforte has been used. Acording to this algorythm, the filtered wave forms are normalized in order to make the samples of each waveform range between the minimum and maximum vallues allowed in a fixed-point representation of the samples.

Let T be the sampling period and x_i (nT) the n th sample of the normalized waveform taken at the i th lead. Then, for each sample of each waveform, the digital derivative y_i (nT), defined as follows:

$$y_i(nT) = k\frac{x_i(nT) - x_i(nT-T)}{T} \forall n,i \qquad (1)$$

is taken.

But, the signal y_i^2 (nT) obtained by these calculations is not regular enough. So the signal is passed through a low-pass filter. This new signal obtained is divided into certain number of threshold and every threshold is sembolized by a letter. These letters are used as terminal elements. It is shown at table 1 which therminal element will be assigned to syntax according to threshold at square of derivated EKG signals and its' stay time on that threshold. According to this algorithm for every QRS complex, one syntax is obtained.

There is two main parts in recognoizing the shape by syntactic method.

a) Training mode
b) Classification mode

These main parts and precedures carried out in these parts are shown in the figure 1. below. Every procedure shown in the figure 1. has been carried out in the computer program developed. As a result of this, the QRS dedection is succeeded.

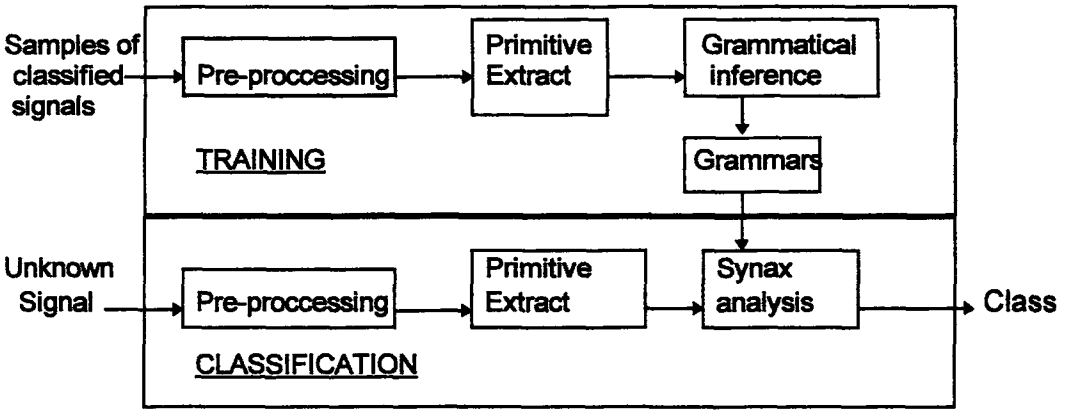


Figure 1. Ageneral syntactic recognition system.

Some changes have been made in the algoryhtm prepared by Gustove Belforte in order to dedect also P and T peaks. For this, the formula below is used to calculate the derivated EKG, instead of the old one.

$$S(K)=\frac{\left|\frac{\sum_{i=0}^{9}x(k+i)}{10}-\frac{\sum_{i=0}^{9}x(k-i)}{10}\right|}{10T} \qquad (2)$$

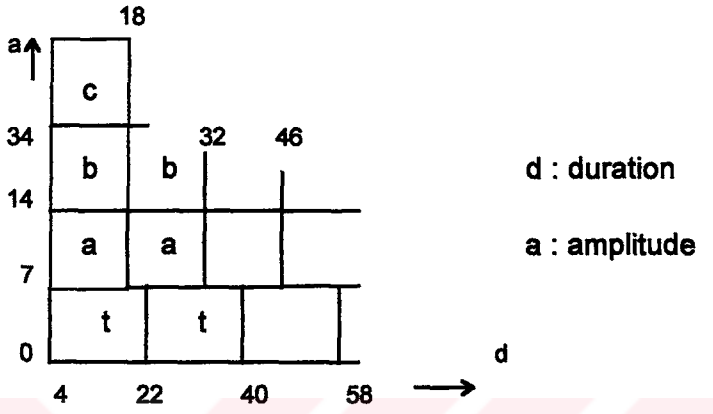
In this formula, the number 9 has been chosen because it has given the best result at the end of various experiments done by some EKG patterns. As a result of this formul the derivates of P and T peaks became more definite, so their syntaxes can be obtained. In this way, for P and T peaks, gramer extraction in training mode and dedection of these in working or classification mode became possible.

Later on, it has been worked on to prepare an algorithm by extracting syntax of EKG signal in one period for obtaining the grammer of EKG signals which belong to a certain group. The table below have been used to extract the syntax of EKG in one period. Here, the thing done in addition is to put "t" therminal element in the spaces between derivates of P, QRS, and T. In this way, after the syntax of EKG in one period has been extracted, the algorithm shown in the figure 1. was carried out also for recognizing the shape of EKG in one period.

In previous studies, gramer extraction process in training mode could have been carried out by entering EKG patterns just for once. There was no possiblity to expand the gramer in the way that it also includes the EKG patterns we will face later on. On the next step, the program is improved in order to give it the ability

later on. On the next step, the program is improved in order to give it the ability to make the necessary additions to the gramer in the way that it can recognize the new EKG patterns entered later on.

Table 1. Coding system for the syntactic recognition of ECG in one periods



In the problem of the EKG signal in the one period, we faced with some difficulties in carrying out the gramatical inference in the figure 1 by computer. These difficulties arose, because the EKG syntax in one period is too long and a large number of ECG samples from one class is wanted to be entered in taining mode. These factors cause a confusion in gramer expressions and insufficiency of capital letters in alphabet used as non-therminal symbols. In order to overcome these difficulties, some solutions are presented in the "Results and Advices" section. But because of lack of time and restrictions in possibilities, these solutions have not been experienced in this study and they are left to be done in the future.

The algoryhtim planned so far, is not able to differantiate the negative and pzitive parts in derivate of ECG. But in dedection of some aryhtmies, this ability is necessary. For example, an aryhtmia which contains a reversed T wave is shown at the figure 2.10.E. In order to make the algoryhtim able to recognize also these aryhtmies, the changes that can be made in the algoryhtim are explained in the "Results and Advices" section, but they couldn't be experienced by computer.

In this thises, the studies were based on the algoryhtim prepared by Gustavo Belforte for QRS dedection by syntactic method, but not depended on it completely in order to reach the aims which are more sophisticated than QRS dedection.

BÖLÜM 1

GİRİŞ

Elektrokardiografi işareti (EKG), kalp hastalarından kaydedilen gerekli ve en önemli işaretlerden biridir. Kardiologlar, EKG dalga şekillerine bakarak onları normal ve anormal olarak sınıflandırabilirler. Bilgisayar, EKG kaydı ve okuması sırasında kardioloğa yardım edebilir. Örneğin yoğun bakım ünitelerinde bir çok hastadan alınan EKG devamlı olarak kayıt edilmektedir. Oluşabilecek hayati tehlikelerin bulunması için kardiologların 24 saat kayıt alamadıkları durumlarda kullanılan otomatik izleme programları, anormal kalp atışlarının bulunmasında kullanılır. Günümüzde bilgisayarla desteklenmiş, EKG izleme cihazları ile EKG kayıt ve değerlendirmeleri, yaygın olarak bilinmekte ve kullanılmaktadır. [1]

1960' ların başlarında, şiddetli miyokardiyal enfarktüs hastası olan kişilere gereken özenli bakımı sağlayabilmek için kalp bakım üniteleri (coronary care units, CCUs) kurulmaya başlanmıştır. Bunun nedeni, hastaların enfarktüsü izleyen saat ve günlerde yüksek ani ölüm riski taşımaları ve ventriküler fibrilasyon ile ani olarak yaşamlarını yitirebilmeleridir.

Ventriküler fibrilasyona girilip girilmediğinin belirlenmesi için EKG işaretleri, büyük boyutlardaki monitörlerde, hemşireler tarafından sürekli olarak özenle incelendi. Daha sonra EKG' lerin daha detaylı olarak incelenmesi sonunda erken ventriküler atımların, daha ciddi aritmiler olan ventriküler taşikardi ve fibrilasyonun habercileri olduğu gözlendi. Ventriküler aritmilerin kliniksel önemi anlaşıldıkça, bu olayları ortaya çıkartacak güvenilir ve otomatik dedektörlerin geliştirilmesine ihtiyaç duyuldu.

Otomatik aritmi analizatörünün hizmeti, kliniksel olarak önem taşıyan verilerin. çok miktardaki işlenmemiş veri arasından seçilmesi olarak özetlenebilir. Bir doktorun gözden geçirebileceği veri, otomatik sistemin analiz ettiği veri miktarından çok daha azdır. Pekçok otomatik sistem tek tek QRS komplekslerini

belirler, ölçer, sınıflandırır ve zamanlamalarına ve atış dizilerine göre belli aritmileri ayırdeder.

Düşük işaret/gürültü oranları ve hesaplamadaki kısıtlamalardan dolayı, günümüzdeki teknoloji, P ve T dalgalarının gerçek zamanda, standart yüzey monitör elektrotları ile belirlenmesinde yetersiz kalmaktadır. Ritmin analizinin bu şekilde tamamlanamamasından dolayı, işlem sadece QRS morfolojisi ve zamanlama üzerine yorum getirir. Yanlış iletimden ileri gelen supraventriküler erken atımlar, birinci ve ikinci derece kalp blokları ve aralıklı gidip gelen demet kolu blokları gibi aritmilerin şu anki sistemlerle otomatik olarak bulunması çok zor veya imkansızdır [2].

Sözdizimi metodu (syntactic method), otomatik aritmi deteksiyonunda bahsedilen bu zorlukların aşılmasında önemli kolaylıklar sağlayabilir. Çünkü, bu metod, işaretleri tanımak ve sınıflamak için yapısal bilgi kullanır ve daha eski olan karar teorisi yaklaşımına göre, insanın teşhis yapma işlemine daha yakındır[3].

Bu tez çalışmasında, otomatik aritmi deteksiyonunda sözdizimi metodundan nasıl istifade edilebileceği araştırılmıştır. Başka metodlarla da kolayca yapılabilen QRS deteksiyonunun yanısıra, yukarıda belirtilen zorluklara çözüm getirebilmek amacıyla, tüm bir EKG şeklini tanıma, sözdizimi metodu ile gerçekleştirilmeye çalışılmış, bunun için bir algoritma geliştirilmiş ve bu algoritma, yazılan bilgisayar programları ile denenerek olgunlaştırılmaya çalışılmıştır.

Bunu için ilk önce, Bölüm 2' de EKG işareti ve aritmiler hakkında genel bir bilgi verilmiş, Bölüm 3' te sözdizimi metodunun teorisi anlatılmış, Bölüm 4' te de yapılan bilgisayar uygulaması ve bunun için kullanılan algoritma anlatılmıştır.

BÖLÜM 2

ELEKTROKARDİYOGRAM İŞARETİ VE BOZUKLUKLARI HAKKINDA GENEL AÇIKLAMALAR

2.1. GİRİŞ

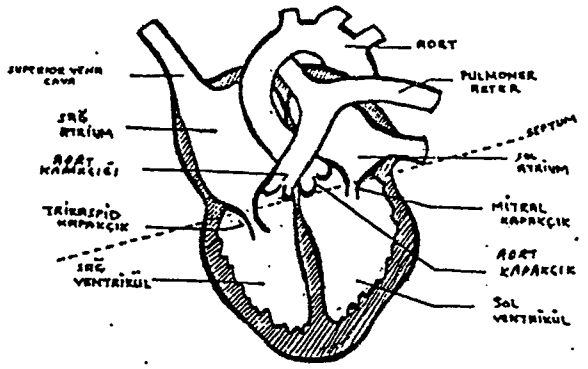
İnsan vücudu üzerinden algılanan ve kalbin elektriksel aktivitesininin sonucu olarak ortaya çıkan belli tipteki biyolojik işaretlere elektrokardiyogram, elektrokardiyografik işaret, EKG işareti veya kısaca EKG adı verilir. EKG işaretlerinin gösterilmesini veya kaydedilmesini sağlayan cihazlara elektrokardiyograf ve EKG ile ilgili sistem ve olgulara da genel olarak elektrokardiyografi denir [4].

Bu bölümde EKG işaretlerinin kaynağı, oluşumu değişim şekli, ölçümü ve bozuklukları üzerinde kısaca durulacaktır.

2.2. KALP VE DOLAŞIM SİSTEMİ

Kanın vücudda dolaşımı bir pompa görevi gören kalbin, sıkışması sonucuyla oluşan basınç yardımıyla sağlanır. Şekil 2.1' de kalbin yapısı görülmektedir. Kalp kabaca, sağ kulakcık (atrium), sol kulakcık, sağ karıncık (ventrikül) ve solkarıncık olmak üzere dört odacıktan meydana gelmiştir. Karıncıklarla kulakcıklar arasında, kulakcıktan karıncığa kan akışını sağlayan fakat tersi yönde akışı engelleyen kapakcıklar vardır.

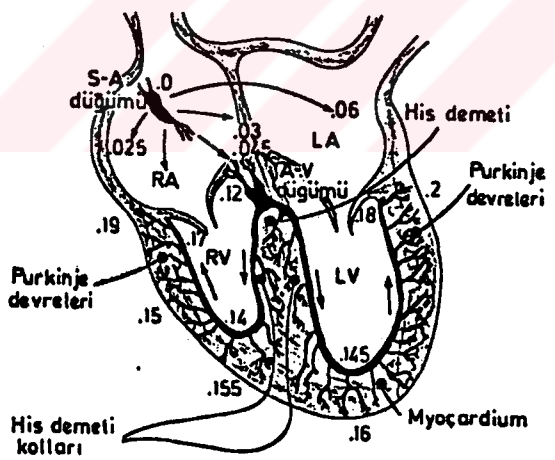
Vücuttan toplanan oksijensiz pis kan, sağ kulakcığa gelir. kulakcıkların kasılmasıyla bu pis kan sağ karıncığa geçer. Karıncıkların kasılmasıyla da pulmanör arter üzerinden akciğerlere pompalanır. Akciğerlerde CO_2 - O_2 alışverişi sonucunda temizlenen kan sol kulakcığa pulmener ven yardımıyla döner. Buradan kulakcıkların kasılmasıyla sol karıncığa geçer. Karıncıkların kasılmasıyla aort üzerinden tüm vücuda temiz kan pompalanır [5].



Şekil 2.1. Kalbin anatomik yapısı [5]

2.3. KALBİN ELEKTRİKSEL İLETİM SİSTEMİ

Kan pompalama işlemi, kalbin odacıkları etrafını çeviren kalp kaslarının kasılmasıyla olur. Kasların kasılması ise ancak elektriksel uyarıyla mümkündür. Kalp, kaslarının kasılma sıralarını ve periyotlarını düzene koyan özel bir elektriksel iletim sistemine sahiptir. Şekil 2.2'de kalbin elektriksel iletim sistemi gösterilmiştir. İletim sistemi sinoatrial düğüm (sinoatrial node-SA), his demeti, atrioventricular düğüm (AV), demet kolları (bundle branches) ve purkinje fiberlerinden oluşur [5].



Şekil 2.2. Kalbin Elektriksel İletim Sistemi [5]

SA düğümü kalbin pacemaker' ı olarak çalışır. Pacemaker, hareketi başlatan ve hareketin hızını, periyotunu belirleyen anlamına gelir. SA düğümünde kendi kendine oluşan aksiyon potansiyeli depolarizasyon dalgası halinde tüm kalbe yayılır. Kalp hücreleri arasındaki elektriksel geçiş, hücreler arası alçak direnç bölgelerini oluşturan geçit bölgeleri üzerinden olur. SA düğümünün oluşturduğu

aksiyon potansiyelinin frekansı, değişen koşulların gereksinimini karşılamak üzere aynı zamanda merkezi sinir sistemi tarafından da kontrol edilmektedir.

SA düğümünde oluşan aksiyon potansiyeli kulakcıklar üzerindeki iletim yolları üzerinden hızlı bir şekilde yayılarak kulakcıkların kasılmasını sağlar ve buradaki kan karıncıklara basılır. Kulakcıklardaki aksiyon potansiyeli hızı 30 cm/s kadardır. SA ve AV düğümleri arasındaki özel iletim hatlarında ise hız 45 cm/s kadardır. SA düğümünde oluşan aksiyon potansiyeli 30-50 ms sonra AV düğümüne ulaşır. Bu süre kulakcıklar içindeki kanı tümüyle karıncıklar içine doldurmaları için yeterli bir süre değildir. Bu nedenle karıncıkların kasılmasının geciktirilmesi gereklidir. Bu işlem bir geciktirme elamanı gibi çalışan AV düğümünde aksiyon potansiyelinin 110 ms geciktirilmesiyle yapılır. Kulakcıklarla karıncıklar arasında yağlı bir bölge elektriksel izalasyon sağlar. Bu nedenle kalbin bu iki iletim bölgesi arasındaki iletim sadece iletim sistemi üzerinden yapılabilir.

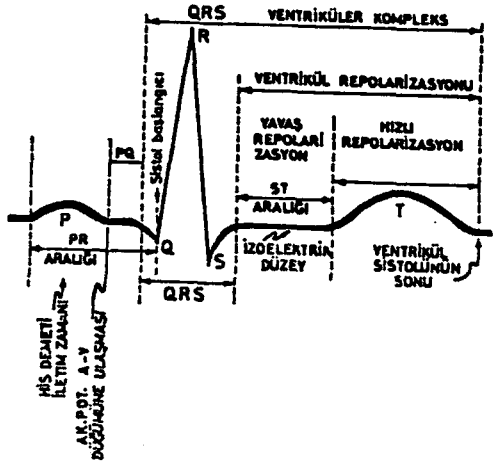
Karıncıkların uyarılması purkinje fiberleri ile olur. Bunlarda aksiyon potansiyelinin hızı 2-4 m/s kadardır. Bu fiberler ve tüm iletim mekanizması üzerindeki aksiyon potansiyelinin ulaşım hızı şekil 2.2' de gösterilmiştir. Purkinje fiberleri yardımıyla uyarılan myocardium kasılır ve buradaki kan arterlere basılır.

EKG işareti işte bu kalp kaslarının birlikte kasılmaları sonucu ortaya çıkan ve deri üzerinden elektrotlarla algılanabilen elektriksel bir işarettir.

2.4. ELEKTROKARDİOGRAMDAKİ ŞEKİLLERİN YORUMU

Şekil 2.3' te bir periyotluk bir EKG işaretinin zamana göre değişimi verilmektedir.

P Dalgası : SA düğümünden gelen uyarıyla kulakcık kaslarının kasılması P dalgasını meydana getirir. Süresi aksiyon potansiyelinin SA düğümünden AV düğümüne geçişini gösterir. Dakikada vurum hızı 75 olan sağlıklı bir kimsede P dalgasının süresi 0.1 ms dir. P dalgasının genliği, kulakcık kaslarının fonksiyonel aktivitesini belirtir. P dalgasının tepesi yuvarlak, sivri yada çentikli olabilir. Tepecikler arasındaki uzaklık 0.03 saniyeyi aşmadıkça çentiklenme normal bir görünümdür [1].



Şekil 2.3. Elektrokardiogram işareti [5]

PR Aralığı : SA düğümünden çıkan uyarının karıncıklara ulaşabilmesi için geçen süredir. P dalgasının başlangıcından Q dalgasının başlangıcına, Q dalgasının görülmediği durumlarda R dalgasının başlangıcına kadar olan uzaklık olarak ölçülür. Kalp hızının 70-90 vurum/dakika arasında olması koşulu ile, PR aralığının erişkinlerdeki normal değeri 0.12-0.2 saniyedir.

QRS kompleksi : QRS, karıncıkların depolarize olmasına karşılık gelir. Q ile gösterilen negatif bir dalga, R olarak adlandırılan pozitif bir dalga ve S adı ile bilinen ikinci bir negatif dalgadır. Karıncık kaslarının fonksiyonel aktivitesini gösterir. His demeti ve kollarındaki iletim bozuklukları da QRS' de değişikliklere neden olur. Karıncıkların kasılması ile R dalgasının yukarı çıkışı aynı anda olur. Erişkinlerde QRS bileşiğinin normal süresi 0.1 saniyeyi aşmaz.

ST Segmenti : QRS bileşiğinin sonlandığı nokta ile T dalgasının başlangıcını birleştiren aralık ST segmenti olarak adlandırılır. ST segmentinin elde edilmesi aşamasında kalp kası negatif olmakla beraber, görünürde ciddi bir elektrik akımı olmadığından izo elektrik çizgide gözükür.

T Dalgası : Karıncık kaslarının istirahat haline geçmesini, diğer bir deyimle karıncık kaslarının repolarizasyonunu yansıtır. Erişkinlerde normal süresi 0.1-0.25 saniyedir.

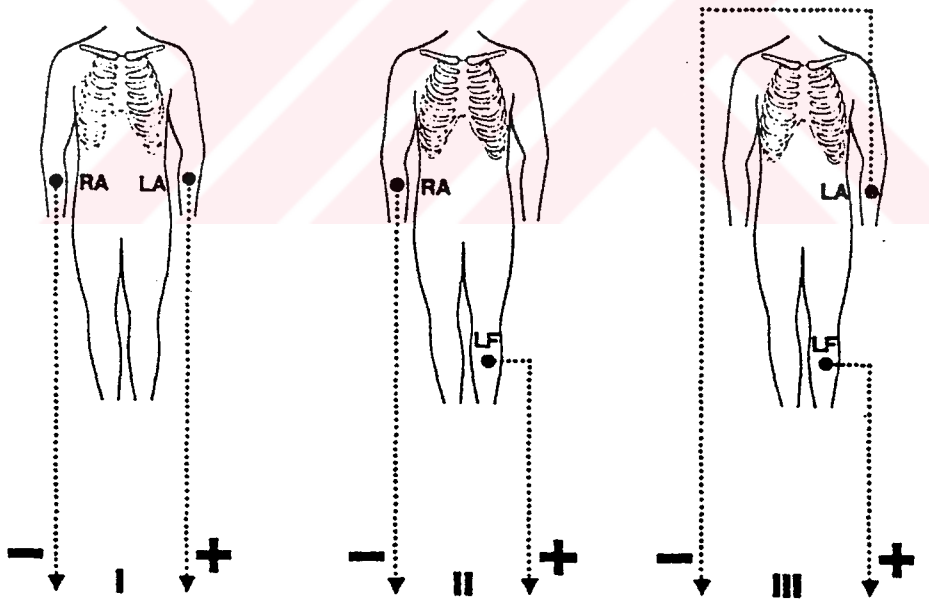
QT Aralığı : Karıncıkların depolarizasyon ve repolarizasyonu için geçen süreyi yansıtan QT aralığı, QRS kompleksinin başlangıcından T dalgasının bitimine kadar olan süreyi kapsar. Erişkinlerde normal olarak 0.35-0.44 s arasındadır [6].

2.5. EKG ÖLÇÜMÜNDE KULLANILAN DERİVASYONLAR

Günlük EKG uygulamalarında kullanılan 12 derivasyon taraf ve göğüs derivasyonları olarak iki ana kümeye ayrılırlar. Taraf derivasyonları bipolar ve unipolar olarak ikiye ayrılır.

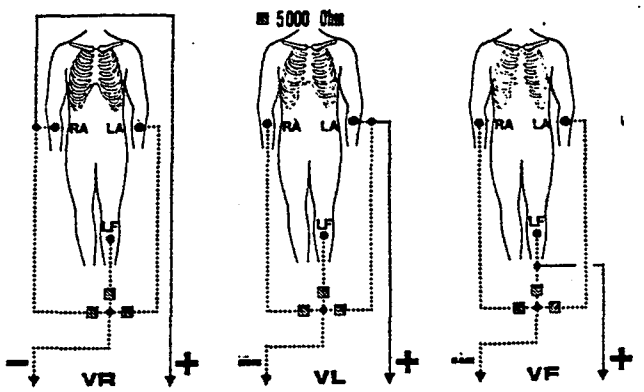
Bipolar derivasyonlar (şekil 2.4) :

1 nolu standart derivasyon : Sağ ve sol kollar arasında ölçüm yapılır.
2 nolu standart derivasyon : Sağ kol ve sol bacak arasında ölçüm yapılır.
3 nolu standart derivasyon : Sol kol ve sol bacak arasında ölçüm yapılır.



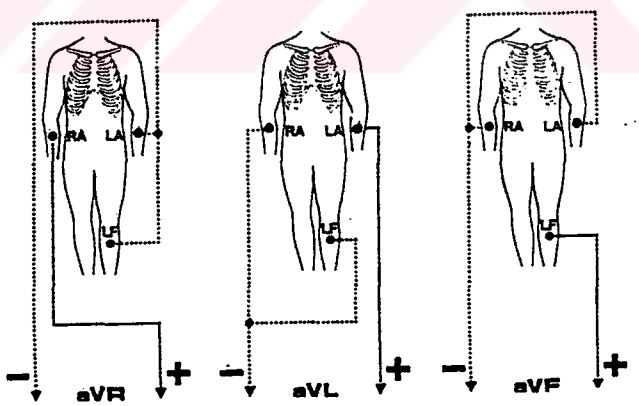
ŞEKİL 2-4: Bipolar Taraf Derivasyonları [6]

Unipolar Derivasyonlar : Elektrotların üçü eşit dirençler üzerinden birbirine bağlanarak merkezi terminal oluşturulur. Merkezi terminal ile üçüncü elektrot arasında ölçüm yapılırsa bu derivasyon unipolar derivasyon olarak bilinir. Bu ölçümler, üçüncü elektrotun konumuna göre VR,VL ve VF olarak isimlendirilir. Buna Wilson' un unipolar taraf derivasyonları adı verilir. (Şekil 2.5)



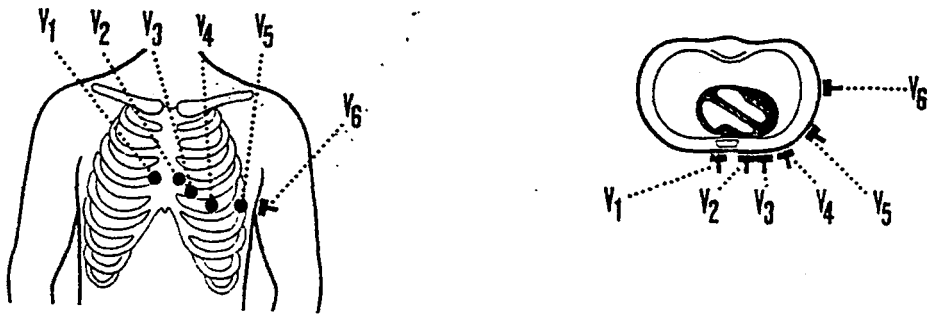
ŞEKİL 2-5: Wilson'un Unipolar Taraf Derivasyonları [6]

Bu taraf derivasyonlarından ölçülen işaretin genliği küçük olduğundan yerine, ölçülen potansiyel genliklerini 1.5 kat arttıran augmented derivasyon Goldenberg tarafından ortaya atılmıştır. Şekil 2.6' da aVR, aVL ve aVF şeklinde isimlendirilen derivasyonlar görülmektedir.



ŞEKİL 2-6: Goldenberg'in Unipolar Taraf Derivasyonları [6]

Unipolar Göğüs Derivasyonları : Merkezi terminal, elektrokardiografın negatif ucu ile birleştirilir ve aygıtın pozitif ucuna bağlı araştırıcı elektrot göğüs üzerinde belirli noktalarda gezdirilirse unipolar göğüs derivasyonları elde edilir.



ŞEKİL 2-7: Göğüs Derivasyonları [6]

2.6. ARİTMİ

Kalbin normal elektriksel aktivitesinde oluşan herhangi bir ritim yada şekil bozukluğuna aritmi denir. SA düğümünden kaynaklanan, hızı 60-100 vurum/dakika sınırları içerisinde yer alan belirli özellikler taşıyan düzenli ritme normal sinüs ritmi adı verilir.

Aritmiler temel olarak iki ana guruba ayrılabilir.

1. Ritim bozuklukları veya düzensizlikleri
2. Şekil bozuklukları

Ritim bozukluğu sinüs aritmi olarak bilinir. Normalde yetişkin insanın kalbi dakikada 70 defa atar. Bu yaşa ve fizyolojik efora, pisikolojik duruma göre değişiklik gösterir. Eğer patolojik bir bozukluk yoksa, ritimdeki bu bozuklukların bir kısmı geçicidir. Fakat patolojik bozukluk söz konusuysa ritimde devamlı bir düzensizlik görülür. Kalp normalin çok altında bir hızla çalışıyorsa bradikardi, çok üstünde bir hızda çalışıyorsa taşikardi söz konusudur. Bu iki aritmi türü iyi bilinen ve teşhis edilmesi kolay olan aritmilerdir.

Ritim bozukluklarından kaynaklanan aritmilerin deteksiyonu için R tepelerinin dedekte edilerek RR aralıklarının incelenmesi yeterli olmaktadır. Bu tip aritmilerin, mikroişlemcilerle kolay dedekte edilebilmeleri için RR aralıklarındaki

düzensizlikler matematiksel olarak ifade edilmişlerdir. Tablo 2.1' de bu matematiksel ifadeler verilmiştir. Bu ifadelerde arka arkaya gelen en son iki R dalgası arasında geçen zaman aralığı RR(t), son RR aralığından n önceki RR aralığı RR(t-n), arka arkaya gelen son sekiz RR aralığının aritmatik ortalaması ise AR(t) ile gösterilmektedir.

Tablo 2.1. Çeşitli aritmilerin matematiksel ifadeleri [4]

Taşikardi	: AR(t) < 0,5 s
Bradikardi	: AR(t) > 1,2 s
Asistol	: RR(t) > 1,6 s
Eksik atım	: RR(t) > 1,9·AR(t-1)
PVC (Ön karıncık kasılması)	: RR(t-1) < 0,9·AR(t-2) RR(t)+RR(t-1)=2·AR(t-2) hız > 10/dak.
T üzeri R	: RR(t-1) < 0,33·AR(t-2) RR(t)+RR(t-1)=2·AR(t-2)
Bigemini	: RR(t-3) < 0,9·AR(t-4) RR(t-3)+RR(t-2)=2·AR(t-4) RR(t-1) < 0,9·AR(t-4) RR(t)+RR(t-1)=2·AR(t-4)
Trigemini	: RR(t-2) < 0,9·AR(t-3) RR(t-1) < 0,9·AR(t-3) RR(t)+RR(t-1)+RR(t-2)=2·AR(t-3)
Ekstra sistol	: RR(t-1) < 0,9·AR(t-2) RR(t)+RR(t-1)=AR(t-2) hız > 10/dak.

Şekil bozuklukları, EKG işaretindeki bazı önemli dalgaların şeklindeki anormalliklerdir. En önemlileri, P dalgasının QRS kompleksine uzaklığı ve genliği, T dalgasının QRS kompleksine uzaklığı ve genliği, S-T segmentindeki izoelektrik hatta göre meydana gelen yükselme ve çökme halidir.

Şekil bozukluğu aritmilerini, ritim bozukluğu aritmileri gibi kolayca formalize etmek mümkün değildir. İleriki bölümlerde sözdizimi metodunun şekil bozukluğu aritmilerinin deteksiyonunda nasıl kullanılabileceği anlatılacaktır.

2.7. BAZI ÖNEMLİ ARİTMİLERİN İNCELENMESİ

SA düğümünden başka, AV düğümü, kalp kası ve iletim sisteminin diğer kısımları da ventrikül kasılmasıyla sonuçlanacak uyarıyı başlatabilirler. Bu patolojik durumlar, sinüs düğümünden gelen uyarıların iletilememesi nedeniyle oluşur. Bu darbeler periyotik olmalarına rağmen ritimleri AV düğümünden daha düşük, yaklaşık 50 vuru/dak. civarındadır. Bu ritim, iletim sistemi çevresinde daha da azalarak örneğin ventrikül duvarında yaklaşık 20 vuru/dak. kadar düşer. Eğer normal iletim sistemi çalışmıyorsa, vuru hızı normal sinüs ritminden daha düşük olacaktır. Bu durum "kalp tıkanması" adıyla bilinir. Kalp tıkanmasının çeşitli şekilleri vardır. İletim bozukluğunun tipine bağlı olarak şu isimleri alır.

- 1. dereceden AV tıkanması (AV blok) : Burada sadece iletim süresi uzamıştır. Bu nedenle EKG işaretindeki P-R aralığı uzar.

- 2. dereceden AV tıkanması : Bazı darbeler atriumdan iletilmezler.

- 3. dereceden AV tıkanması (veya total blok) : Bu durumda atrium ve ventrikül vurumları senkron değildir. Aniden meydana geldiğinde, hasta genellikle kendini kaybeder.

- Demet tıkanıklığı : Burada uyarılar, ulaştırılması gereken ventriküle iletilemez. Ventriküllerdeki kasılmanın yayılması normala göre daha uzun periyotda oluşur.

Bazı kalp tıkanmalarında kalbin fonksiyonu büyük ölçüde bozulur. Örneğin 3. derece AV bloğu gibi. Bu durumda dışarıdan uygulanacak bir pacemaker ile kalp kaslarının uyarılması sağlanır. Aritmilerin daha bir çok çeşitleri vardır. En çok rastlanılanı ekstra sistollerdir. Bu durumda kalp kaslarında erken depolarizasyonlar meydana gelir. Depolarizasyonun oluştuğu yere bağlı olarak "ventriküler", "nodal" ve "atrial ekstrasistol" adını alır. Atrial ekstra sistoller normal şahıslarda bile sıkça rastlanan bir durumdur. Atrial ekstra sistollerin karakteristik özelliği, fiziksel efor ile sayılarının artmamasıdır. Ventriküler ekstrasistoller, genelde erken ventriküler kasılma (PVC) olarak adlandırılırlar. Kalp hastalarının yoğun bakıma alındıkları zamanlarda; örneğin, bir dakikada

üçten fazla erken ventriküler vuru olduğunda alarm verdirtilerek gerekli tedbirlerin alınması sağlanabilir.

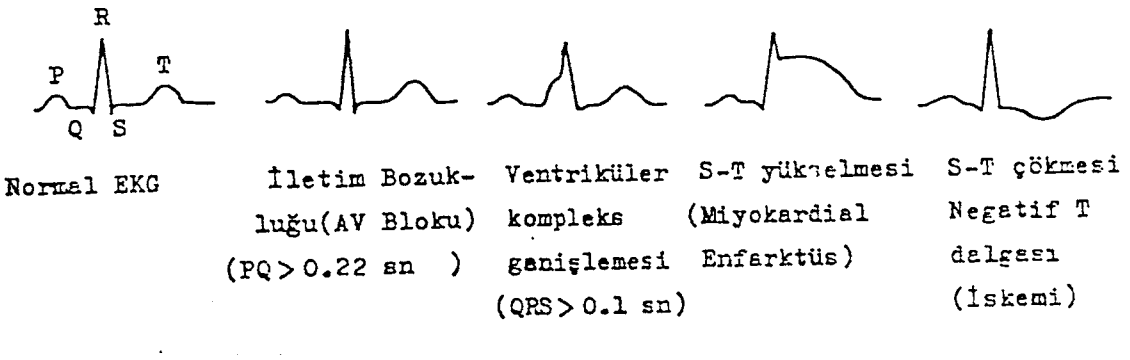
Atrial çarpıntıda, ritim düzenli fakat atrial hız büyüktür. (Dakikada 250-350 vuru civarında.)

Atrial fibrilasyonda ritim tamamen düzensizdir ve hız ise 300-500 vuru/dakika civarındadır. Ventriküller, bu yüksek frekansa uyum gösteremez, fakat vuru daha yavaş olarak sürer. Bu kan sirkülasyonunun hala devam edebilmesini sağlar.

Ventriküler fibrilasyon tehlikeli bir durumdur. Ventrikül kasları o kadar hızlı ve düzensiz kasılırlar ki, ventriküller dolmaya fırsat bulamazlar ve kan sirkülasyonu durur. Hasta 15-20 saniye içinde kendini kaybeder. Eğer acil tedbir alınmaz ise, hasta birkaç dakika içinde beyindeki oksijen yetersizliği nedeniyle ölür. Asistol, kalp kaslarının kasılmasındaki yetersizlikten doğan diğer bir tip sirkülasyon bozukluğudur. Her iki durumda da dışarıdan kalbe mesaj ve suni teneffüs acilen uygulanmalıdır. Ventriküller fibrilasyonda hemen her zaman defibrilasyon uygulamak gerekir. Bunun için büyük bir elektrik akımı kalp üzerinden geçirilir. Bu metodtan her defasında sonuç almak mümkün olmayabilir.

Elektrokardiyagramın yorumlanması, eğrinin genlik ve süre analizinin yapılması temeline dayanır. İletim zamanı, P dalgasının başlamasından ventriküler depolarizasyonun ilk başladığı an olan Q veya R dalgasına kadar geçen sürenin hesaplanmasıyla elde edilir. P-R aralığı normal olarak 0.12-0.22 saniye kadardır. Aralığın daha uzun olması, atriumdan ventriküle olan yayılmada bir engelleme (AV tıkanması) olduğunu gösterir. (Şekil 2.8)

QRS kompleksinin genişliği normal olarak 0.07-0.10 saniye kadardır. Bu sürenin artması, ventrikül duvarında uyarının yayılması sırasında iletim sisteminin bir kısmında oluşan kesintiyi gösterir. Bu durum demet tıkanması olarak bilinir ve demetin sağ veya sol başındaki parçasında olabilir.

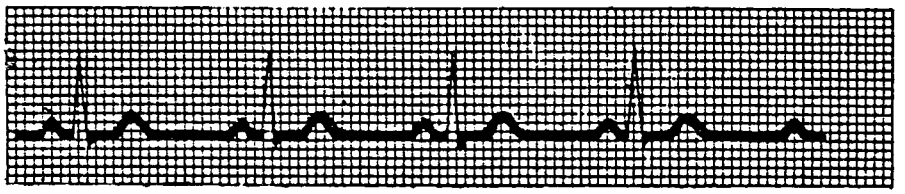


Şekil 2.8. Bazı önemli EKG dalga şekilleri [2]

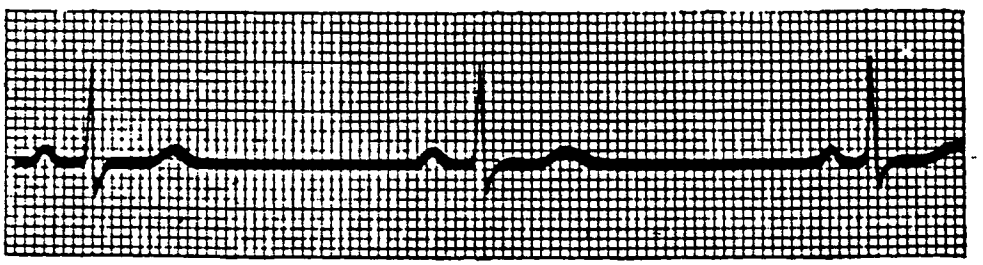
Miyokardial enfarktüs' te, EKG' de karakteristik değişiklikler vardır ki, bunlar teşhisi doğrularlar. Elektrokardiogram, enfarktüsü izleyen günlerde de tipik değişiklikler gösterir. Bunların incelenmesi faydalı bilgiler verir.

Kroner yetmezlikte, kroner arterlerindeki kan sirkülasyonu yetersizdir. Bunun EKG' ye yansıması, S-T aralığının kısalması ve bazen de T dalgasının ters dönmesi şeklindedir. Çünkü oksijen yetersizliği kalp kaslarının repolarizasyonunu engeller [2].

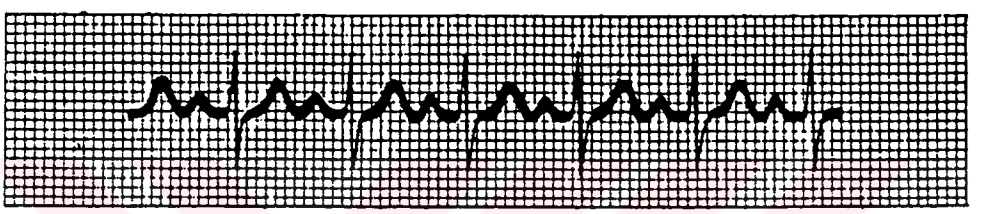
Şekil 2.9' da normal sinüs ritmi, şekil 2.10' da da çeşitli aritmileri içeren EKG işaretleri görülmektedir.



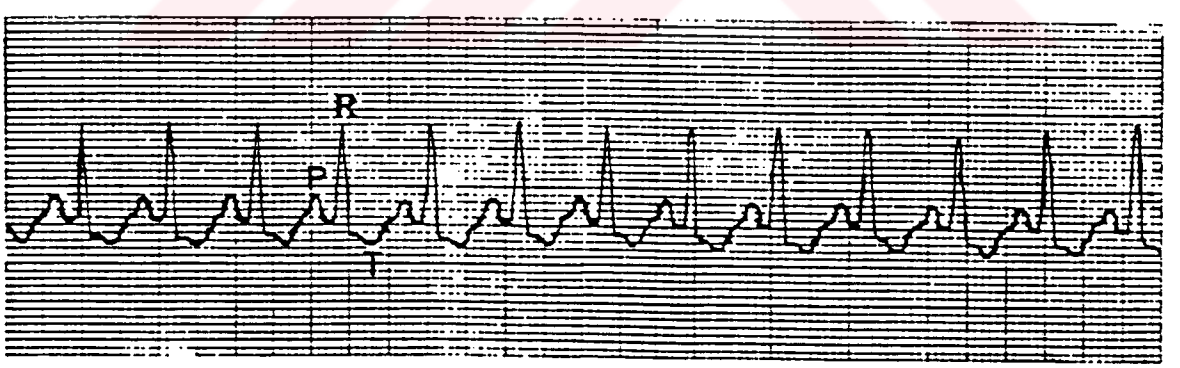
Şekil 2.9. Normal sinüs ritmi [6]



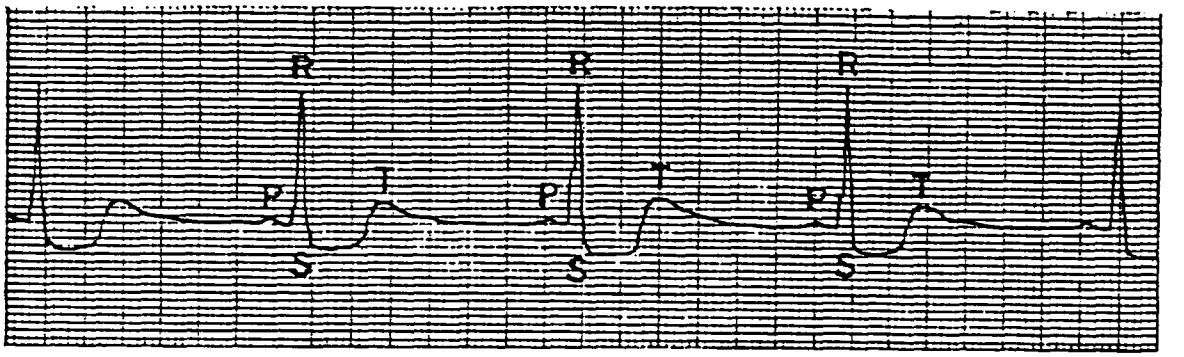
Şekil 2.10. A. Bradikardi [6]



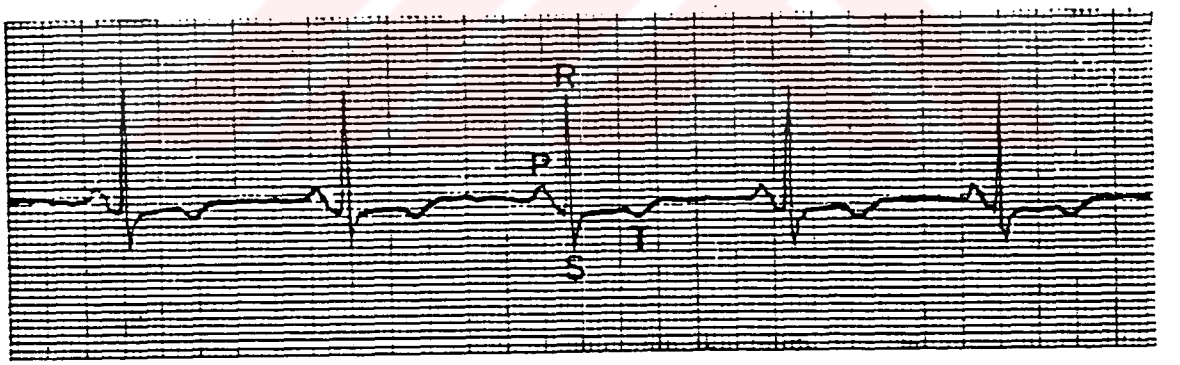
Şekil 2.10.B. Taşikardi [6]



Şekil 2.10.C. Taşikardi (Ritim düzenli fakat hızlı, S-T segmentinde çökme var -Miyokardial Enfarktüs-) [2]

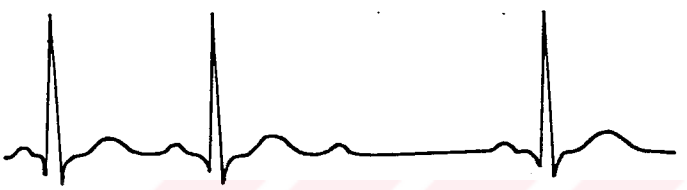


Şekil: 2.10.D - "Sinüs bradikardi.(Ritm düzenli fakat yavaş,S-T segmentinde çökme var-Miyokardial Enfarktüs)." [2]

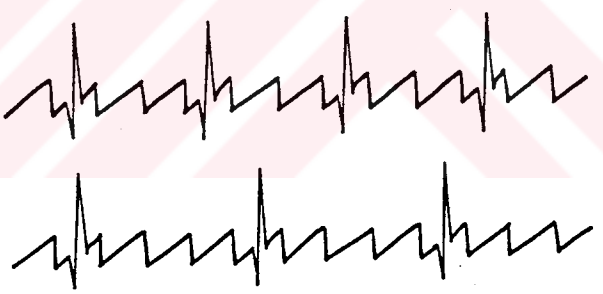


Şekil: 2.10.E - "Sinüs bradikardi.(Ritm düzenli fakat yavaş,T dalgası ters dönmüş-İskemi)." [2]

Şekil.-2.11 Sino-atrial blok.

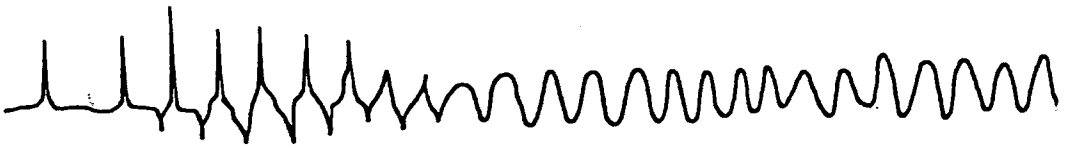


Şekil.-2.12 Atrio-ventriküler blok.



Şekil.-2.13 Atrial flatter.

Şekil.-2.14 Atrial fibrillasyon.

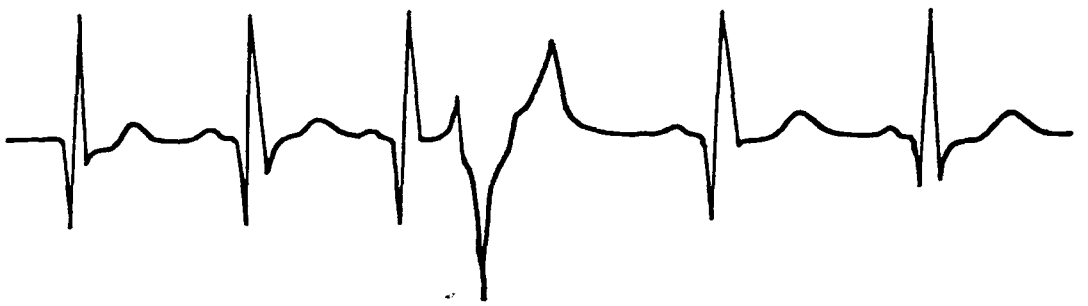


Şekil.- 2.15 Ventriküler fibrillasyon

Şekil.- 2.16 Atrio-ventriküler ritm.

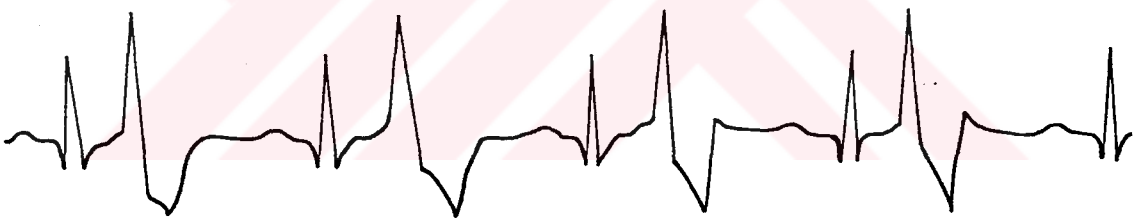
Şekil.- 2.17 Hasta sinüs sendromu (S.S.S.).

Şekil.- 2.18 Erken ventrikül kasılması (VPS).



Şekil.-2.19 T üzerinde R dalgası

Şekil.-2.20 Çok odaklı VPS.



Şekil.-2.21 Ventriküler Bigemini.

Şekil.-2.22 Çift VPS.

Şekil.-2.23 Ventriküler Taşikardi.

Şekil.-2.24 Kalp Tutulması.

BÖLÜM 3

SÖZDİZİMİ METODU (SYNTACTİC METHOD)

3.1. GENEL TANIM VE KAVRAMLAR

İşaret şekli tanıma probleminde iki genel yaklaşım vardır.

a) Karar teorisi veya ayırım yaklaşımı
b) Söz dizimi veya yapısal yaklaşım

Birinci yaklaşımda işaret kendisini temsil eden bir dizi öznitelik ile temsil edilir. Örneğin AR (Autoregressive) parametreleri gibi. Burada bu yaklaşım üzerinde durulmayacaktır.

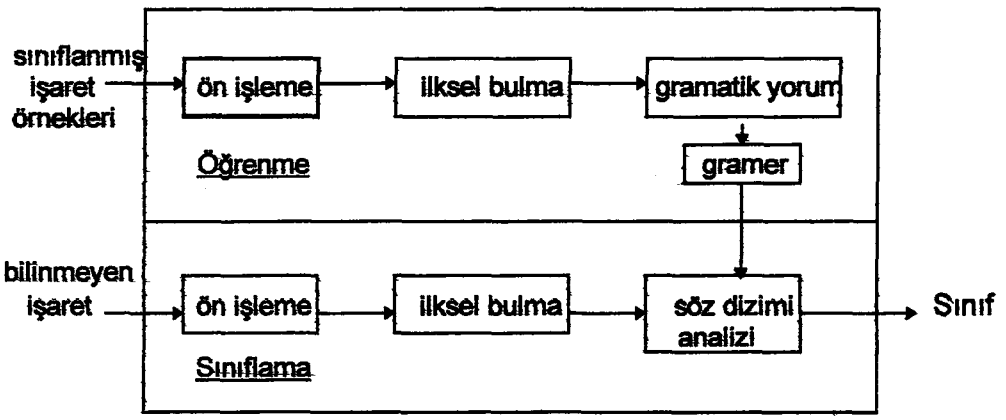
İkinci yaklaşım diğerlerinden daha yeni bir yöntemdir. İşaret şekillerini tanımak, belirlemek veya sınıflamak için yapısal bilgi kullanır. Bu metod bir dilin cümle yapısını oluşturmaya benzer olarak söz dizimi kullanır. Bu yöntemde, farklı farklı dillerin olması, farklı farklı sınıfların varlığına benzetilmştir. Nasıl her bir dilin cümle, kelime ve harflerinin dizimi ve kombinasyonu belli bir gramere göre ise, her bir sınıfa ait işaretten elde edilen ilksellerin dizimi ve kombinasyonu da o sınıfa ait gramere göredir. İlkselleri sembolize etmek için genelde alfabedeki harfler kullanılır. Bu harflerin ard arda dizilmesinden meydana gelen diziye sözdizimi denir.

3.2. SINIFLAMA İÇİN GENEL ALGORİTMA

Alınan bir işaretin hangi sınıftan olduğuna karar verebilmek için ilk önce her bir sınıf için bir dil oluşturmak gereklidir. Dil oluşturmak için, dilin ilkselleri belirlenmeli ve bu ilksellerden oluşan söz diziminin diziliş kurallarını veren gramer oluşturulmalıdır. Daha sonra, gelen işaretten çıkartılan söz diziminin bu gramere uyup uymadığına bakılarak, bu işaretinn bu sınıftan olup olmadığına karar verilir.

Bu açıklamalardan anlaşılacağı üzere, sınıflama için gerekli algoritmada şu iki ayrı modun olması gereklidir.

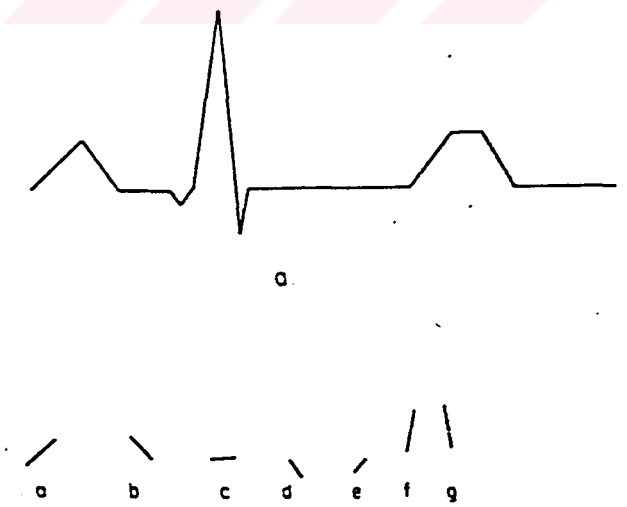
a) Öğrenme modu
b) Sınıflama modu



Şekil 3.1. Genel bir sözdizimi tanıma sistemi [7]

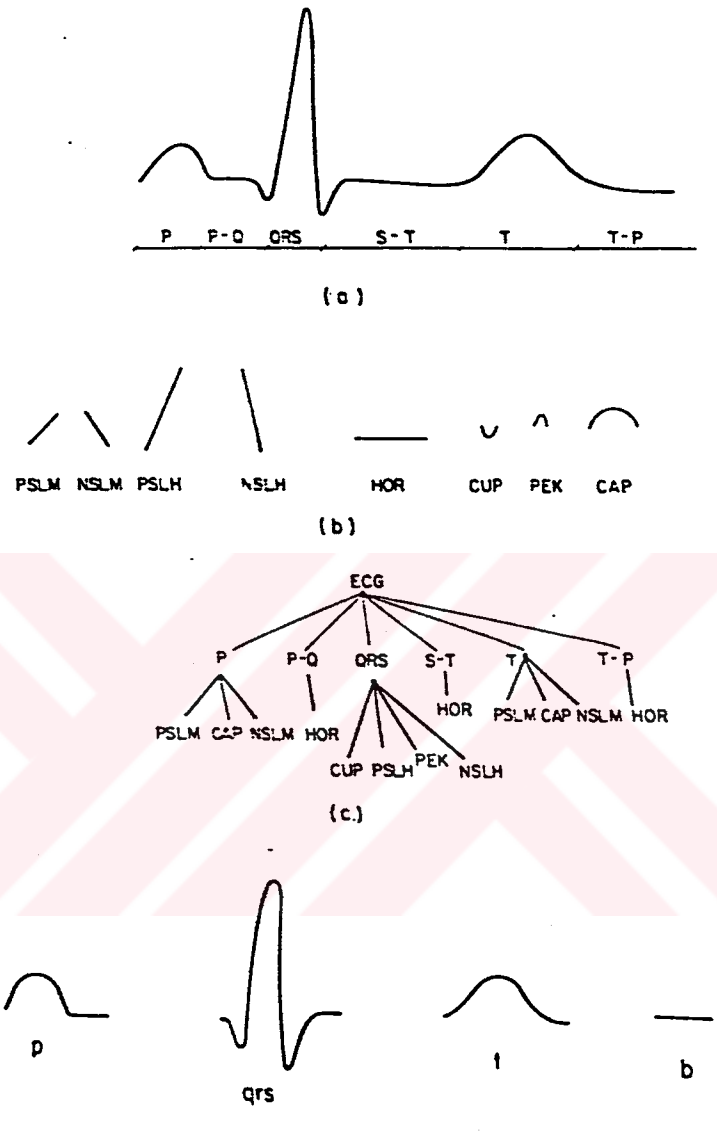
3.3. İLKSEL ÇIKARTMA

İlksel çıkartma sözdizimi metodunun ilk basamağıdır. İlksel çıkartmada genel yaklaşım, işaretin parçalara bölünerek her parçanın bir ilksel tarafından temsil edilmesidir.



Şekil 3.2. Bir EKG işareti ve ilkselleri [7]

Şekil 3.2' de her biri bir ilkseli temsil edecek şekilde değişik eğim ve boyda 7 farklı çizgiden bir EKG işaretinin nasıl oluşturulabileceği ve bu işaretin sözdizimi gösterilmektedir.

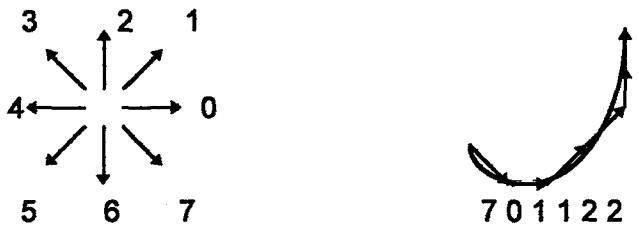


Şekil 3.3. Bir EKG' nin sözdizimi gösterimi. a. EKG modeli b. İlksel takımı c. İlişkisel ağaç d. Alternatif bir ilksel takımı [7]

Şekil 3.3.a ve b' de ise ilksel sayısı arttırılarak daha ideal bir EKG işareti oluşturulmaya çalışılmaktadır. Şekil 3.3.d' de ise EKG' deki P, QRS, ve T tepeleri ile yatay düz çizgi kalıpları birer ilksel olarak kabul edilmişlerdir. Alınan

bir EKG işaretinden bu ilkselleri bularak işaretin sözdizimini çıkartabilmek için, bu kalıplarla işaret karşılaştırılarak uygun kalıbın temsil ettiği harf söz dizisine eklenir [3] [7].

Şekil 3.4' te ise 8 farklı yönü ilksel olarak kullanan bir söz dizimi çıkartma yöntemi görülmektedir.

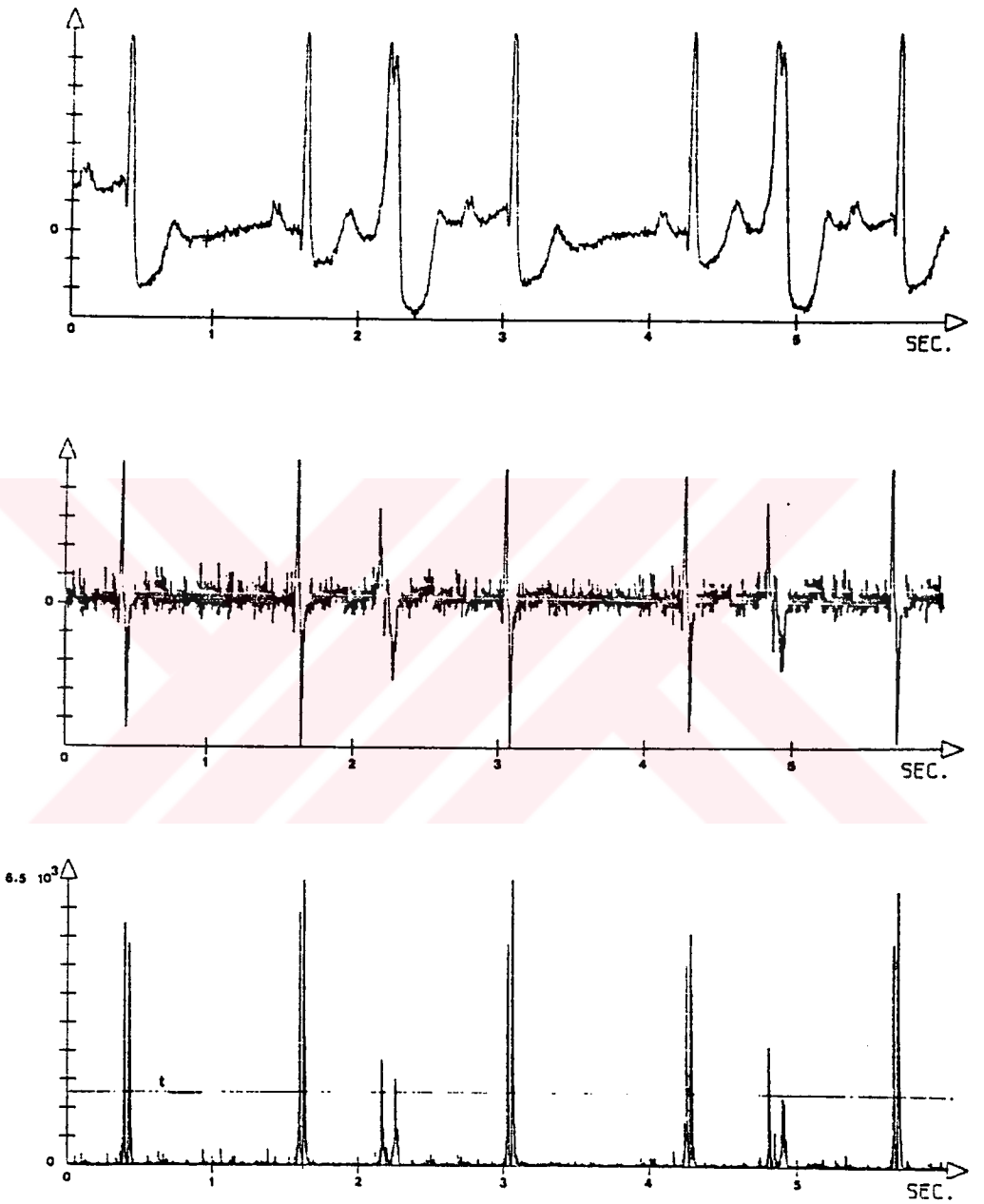


Şekil 3.4. 8 farklı yönü ilksel olarak kullanan bir sözdizimi yöntemi [3]

3.4. QRS DETEKSİYONU İÇİN İLKSEL ÇIKARTMA

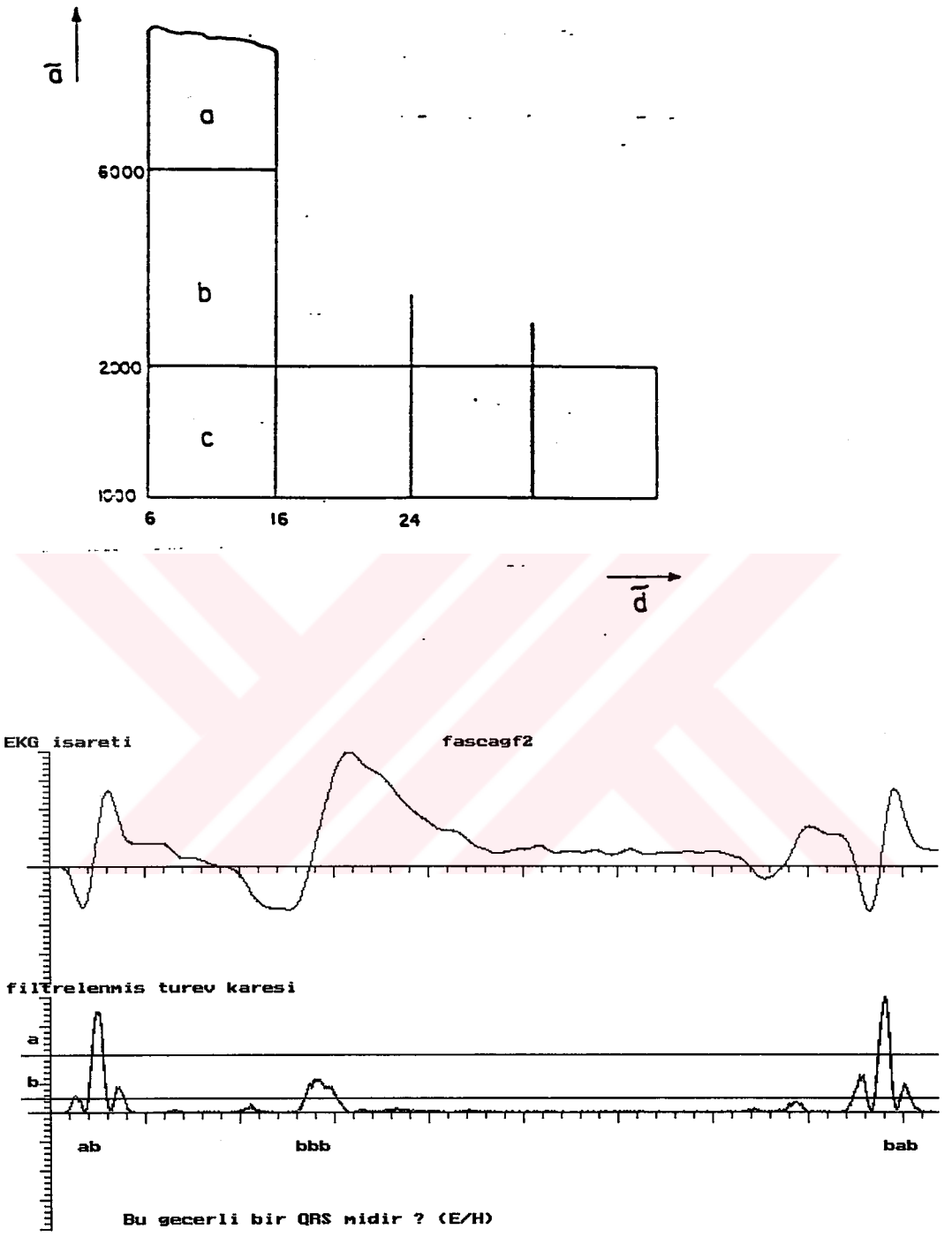
Bu konuda Gustavo Belforte tarafından ortaya atılmış bir algoritma vardır. Elde edilen EKG' nin türev karesi alınıp normalize edilir. Şekil 3.5 bir EKG' nin türevi ve karesi alındığında ortaya çıkan durumu göstermektedir. EKG' nin türev karesi değişik eşik seviyelerine bölünerek her seviye bir harf ile sembolize edilir. Tablo 3.1, bu harflerin zamanın fonksiyonu olarak nasıl atanması gerektiğini göstermektedir [8].

Şekil 3.6' da her bir QRS' nin söz diziminin nasıl çıkartıldığını gösteren bir uygulama verilmektedir. Bu şekil, bu tez çalışması çerçevesinde yazılan bir Turbo C programının çıktısıdır. Tablo 3.1' de dikkat edilmesi gereken bir nokta vardır. İşaret eşik seviyelerini geçer geçmez hemen o eşiğin harfini almazlar. Ancak 6 örnek o seviyede kaldıktan sonra alabilirler. Bu 6 örnekten sonra, eğer işaret aynı seviyede 16 örnek daha kalırsa, o seviyenin harfini ikinci kez alırlar.



Şekil 3.5. Bir EKG' nin türevi ve türev karesi [8]

Tablo 3.1. $\bar{a}$: genlik, $\bar{d}$: süre (örnek sayısı) [7]



Şekil 3.6. QRS komplekslerinin söz dizimini çıkaran "syntax.c" isimli Turbo C programın çıktısı

3.5. DİLLERİN VE GRAMERLERİN İFADE EDİLMESİ

G gramerine uyan sözdizimleri L(G) diline sınıflandırılır. G grameri 4 tane parametre ile verilir.

$$G = (V_N , V_T , R , \sigma) \qquad (3.1)$$

V_N : Ara sembollerin sonlu gurubu

V_T : ilksel sembollerinin sonlu gurubu

σ : Başlama sembolü

R : Gramer kurallarının sonlu gurubu

Özellikler :

1) $V_N \cup V_T = V$ (3.2)

2) $V_N \cap V_T = \varnothing$ (3.3)

3) $\sigma \in V_N$

Kullanılan Bazı Notasyonlar :

1) $\alpha \rightarrow \beta$

Gramer kurallarını ifade ederken kullanılır. "$\rightarrow$" sembolü türetme işlemi anlamına gelir. α ve β, V_N ve V_T elemanlarından oluşan dizilerdir.

$$\alpha \in (V_N \cup V_T) , \quad \beta \in (V_N \cup V_T)$$

Örnek 3.1

$$V_T = (a,b,c,d) , \quad V_N = (\sigma, A, B, C, D, E, F)$$

R : $\sigma \rightarrow aAb$ $\quad Dc \rightarrow Ec$ $\quad Dc \rightarrow cD$ $\quad cF \rightarrow Fc$

$A \rightarrow aAC$ $\quad bB \rightarrow bcd$ $\quad Dd \rightarrow dD$

$A \rightarrow D$ $\quad aF \rightarrow ab$ $\quad aFd \rightarrow Fdd$

2) x^n

x bir dizi iken x^n, x' in n kere yanyana yazılmasından meydana gelen diziyi belirtir.

Örnek 3.2

$$ababccc = (ab)^2 c^3$$

3) $|x|$

x dizisinin uzunluğunu gösterir.

4) V_T^*

V_T elemanlarından elde edilen tüm sonlu diziler gurubunu gösterir.

5) V^*

V_T ve V_N elemanlarından elde edilen tüm sonlu diziler gurubunu gösterir.

6) L

Dili ifade eder. V_T^* ' ın herhangi bir alt gurubudur.

3.6. GRAMER ÇEŞİTLERİ

İfade yapısı gramerleri 4 çeşittir.

1- Sınırlamasız gramerler :

Türetimlerde sınırlama yoktur. Çok genel bir gramer tipidir. Fazla uygulaması yoktur. Burada üzerinde durulmıyacaktır.

2- İlişki - Duyarlı (Context - sensitive) gramerler :

Gramer $\alpha \rightarrow \beta$ ifadelerinden oluşuyorsa ;

$\alpha \in V^*$, $\beta \in V^*$, $|\alpha| \leq |\beta|$ şartlarını sağlayan gramerdir.

Örnek 3.3

$$G = (V_N , V_T , R , \sigma) , \quad V_T = (a,b,c,d) , \quad V_N = (\sigma,A,B,C,D,E,F,G)$$

R :

$\sigma \rightarrow aAb$	$DC \rightarrow EC$	$dG \rightarrow Gd$	$cF \rightarrow Fc$
$A \rightarrow aAC$	$EC \rightarrow Ed$	$aG \rightarrow abcD$	$bF \rightarrow bbc$
$A \rightarrow D$	$DB \rightarrow FB$	$bG \rightarrow bbcD$	$aF \rightarrow ab$
$Dc \rightarrow cD$	$Ed \rightarrow Gd$	$dFB \rightarrow dFd$	$bB \rightarrow bcd$
$Dd \rightarrow dD$	$cG \rightarrow Gc$	$dFd \rightarrow Fdd$	

3 - İlişki - hür (Context - free) gramerler :

Gramer $\alpha \rightarrow \beta$ ifadelerinden oluşuyorsa ;

$$\alpha \in V_N , \quad \beta \in V^* , \quad |\alpha| \leq |\beta| \quad (3.4) , \quad |\alpha| = 1 \quad (3.5)$$

şartlarını sağlayan gramerdir [3].

Örnek 3.4

$$G_2 = (V_{N2} , V_{T2} , R_2 , \sigma) , V_{N2} = \{\sigma , A\} , V_{T2} = \{a , b\}$$

R_2 :

$\sigma \rightarrow Ab$

$A \rightarrow Aa$

$A \rightarrow a$

Bu gramer ifadelerinden nasıl bir söz dizimi çıkacağını araştırırsak şu sonucu elde ederiz.

$$\sigma \rightarrow Ab \rightarrow (Aa)b \rightarrow (Aa)ab \rightarrow ..... \rightarrow a^n b$$

$$L(G_2) = \{ a^n b \mid n = 1,2, ... \}$$

Örnek 3.5

$$G_3 = (V_{N3}, V_{T3}, R_3, \sigma)\ ,\ V_{N3} = \{\sigma, A, B\}\ ,\ V_{T3} = \{a, b\}$$

R_3 :

$\sigma \rightarrow aB$	$A \rightarrow a$
$\sigma \rightarrow bA$	$B \rightarrow bB$
$A \rightarrow a\sigma$	$B \rightarrow aBB$
$A \rightarrow bAA$	$B \rightarrow b$

4 - Sonlu hal (finite state) veya düzgün gramer :

$\alpha \in V_N$, $\beta \in V^*$, $1 \leq |\beta| \leq 2$, $|\alpha| = 1$ şartlarını sağlayan gramerdir[3].

Örnek 3.6

$$G_4 = \{V_{N4}, V_{T4}, R_4, \sigma\}\ ,\quad V_{N4} = \{\sigma, A, B\}\ ,\quad V_{T4} = \{a, b, c\}$$

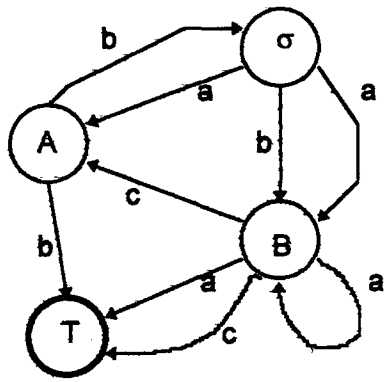
R_4 :

$\sigma \rightarrow aA$	$A \rightarrow b$	$B \rightarrow cA$
$\sigma \rightarrow bB$	$A \rightarrow b\sigma$	$B \rightarrow c$
$\sigma \rightarrow aB$	$B \rightarrow a$	$B \rightarrow aB$

3.7. GRAFİK GÖSTERİM

Graf, düğüm ve dallardan oluşur. Düğümler V_N' nin elemanlarından, dallar ise V_T' nin elemanlarından meydana gelir. Bu şekilde yukarıdaki örneklerde verilen gramer ifadeleri bir grafla ifade edilir. Örnek 3.6' da geçen G_4 gramerinin grafı şekil 3.7' de gösterilmektedir.

Şekil 3.7' deki T düğümü uç veya sonlanma düğümüdür. Bu şekildeki gösterime "durum geçiş diyagramı" da denir.



Şekil 3.7. G_4 gramerinin grafı

3.8. SÖZDİZİMİ TANIYICILAR

Sözdizimlerini tanımak için, her gramerin kendine özgü bir otomatosu vardır. Otomato işlemi, x sözdizimi içindeki elemanları sırası ile okuyan ve başlangıçta q_0' da bulunan bir cihazdır. Bu cihaz bir diğer duruma δ operatörü ile geçer. Eğer otomato verilen x dizisinin tümünü okuyabiliyor ve okuma işlemi sonunda sonlanma durumlarından birine geliyorsa, x dizisinin otomato tarafından kabul edildiği söylenir.

3.8.1. SONLU DURUM OTOMATOSU

Altbölüm 3.6' te bahsedilen sonlu hal grameri için kullanılan bir otomatodur.
A = (E , Q , δ , q_0 , F) şeklinde ifade edilir. Burada E, alfabe yada giriş sembolleri gurubu; Q, tüm durumların gurubu; δ, haritalama veya dönüşüm operatörü; q_0 , başlangıç durumu; F, sonlanma durumları gurubudur.

$$\delta(q_1 , \xi) = q_2 \qquad q_1 , q_2 , \xi \in E \qquad (3.6)$$

Otomato ξ giriş sembolü okununca q_1 durumundan q_2 durumuna geçer. Eğer x bir dizi ve $\delta(q_0 , x) = P$ (3.7), $P \in F$ şartı sağlanıyorsa; yani otomato verilen x dizisinin tümünü okuyabiliyor ve okuma işlemi sonunda sonlanma

durumlarından birine geliyorsa, x dizisinin A otomatosu tarafından tanındığı söylenir.

Örnek 3.7

$A_1 = (E_1, Q_1, \delta, q_0, F_1)$

$E_1 = \{a, b\}$, $Q_1 = \{q_0, q_1, q_2, q_3, q_4\}$, $F_1 = \{q_3, q_4\}$

A_1 ' in durum geçiş haritası :

$\delta(q_0, a) = \{q_1\}$

$\delta(q_0, b) = \{q_3\}$

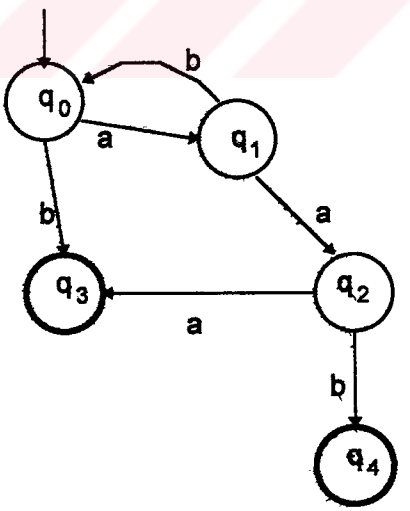
$\delta(q_1, a) = \{q_2\}$

$\delta(q_1, b) = \{q_0\}$

$\delta(q_2, a) = \{q_3\}$

$\delta(q_2, b) = \{q_4\}$

A_1 ' in durum geçiş diyagramı şekil 3.8' de verilmektedir.



Şekil 3.8. A_1 ' in durum geçiş diyagramı [7]

$x_1 = \{(ab)^n b\}$ A_1 tarafından tanınır. Çünkü,

$\delta(q_0, x_1) = \{q_3\}$ $q_3 \in F_1$ dir.

$x_2 = \{ (ab)^m a^3 \}$, A_1 tarafından tanınır. Çünkü,

$\delta(q_0, x_2) = \{q_3\}$ $\quad q_3 \in F_1$ dir.

$x_3 = \{ (ab)^n a^2 b \}$, A_1 tarafından tanınır. Çünkü,

$\delta(q_0, x_3) = \{q_4\}$ $\quad q_4 \in F_1$ dir.

Örnek 3.7' de verilen otomato deterministik bir sonlu durum otomatosudur. Yani, $\delta(q, \xi)$ dönüşümünde sadece bir tek konuma geçme söz konusudur. Eğer $\delta(q, \xi) = \{q_1, q_2, ..., q_n\}$ şeklinde birden fazla duruma geçebilen bir dönüşüm var ise otomato deterministik değildir. Örnek 3.8' de deterministik olmayan bir sonlu durum otomatosu verilmektedir.

Örnek 3.8

$A_2 = (E_2, Q_2, \delta, q_0, F_2)$

$E_2 = \{a, b\}$, $Q_2 = \{q_0, q_1, q_2, q_3\}$, $F_2 = \{q_3\}$

A_2 ' nin durum geçiş haritası :

$\delta(q_0, a) = \{q_1\}$

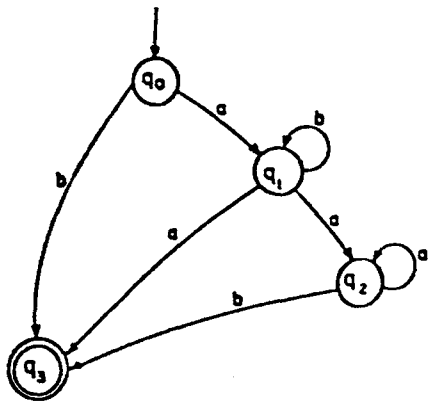
$\delta(q_0, b) = \{q_3\}$

$\delta(q_1, a) = \{q_2, q_3\}$ $\longrightarrow$ Otomatoyu deterministik olmaktan çıkartır.

$\delta(q_1, b) = \{q_1\}$

$\delta(q_2, a) = \{q_2\}$

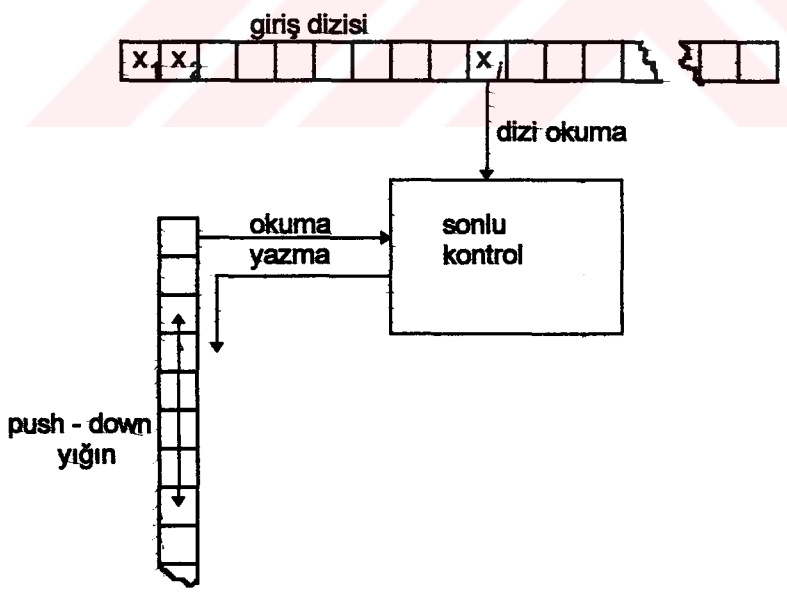
$\delta(q_2, b) = \{q_3\}$



Şekil 3.9. Örnek 3.8' deki otomatoya ait durum geçiş diyagramı [7]

3.8.2. İLİŞKİ - HÜR PUSH - DOWN OTOMATO (PDA)

İlişki - hür dilleri sonlu durum otomatosu tarafından tanınamazlar. Bu dilleri tanıyan otomato PDA' dır. PDA, sonlu durumlu otomatoya benzerdir, fakat push - down yığın kütüğü vardır. Yığın kütüğü, "ilk giren son çıkar" türünden dizi hafızasına sahiptir. En son giren en tepededir ve otomato her zaman en üsttekini okur. Yığın kütüğünün sonsuz kapasiteye sahip olduğu varsayılır.



Şekil 3.10. Push - down otomaton [7]

Bir PDA aşağıdaki şekilde tanımlanır.

$$M = (E, Q, \Gamma, \delta, q_0, Z_0, F) \qquad (3.8)$$

Sonlu durum otomatosundan farklı olarak bu tanımda geçen sembollerin anlamları aşağıda verilmektedir.

Γ : Sonlu push - down sembol gurubu

$$\Gamma = V_T \cup V_N \qquad (3.9)$$

Z_0 : Push - down yığındaki ilk sembol

$$Z_0 \in \Gamma$$

Dönüşüm operatörü δ' nın formatı da şu şekildedir.

$$\delta(q, \xi, Z) = \{ (q_1, \varphi_1,), (q_2, \varphi_2), ...(q_m, \varphi_m) \} \qquad (3.10)$$

$q_1, q_2, ...q_m \in Q \longrightarrow$ durumlar

$\xi \in E \longrightarrow$ giriş sembolü

$Z \longrightarrow$ yığında en tepedeki sembol

$\varphi_1, \varphi_2, ... \varphi_m \in \Gamma$

Eşitlik 3.10' daki dönüşüm şu şekilde gerçekleşir. Kontrol q durumunda ve push - down yığının tepesinde Z sembolü varken giriş sembolü ξ, giriş dizisinden okunur. Kontrol, (q_i, φ_i) çiftlerinden birini seçer. Yığındaki sembol Z, φ_i dizisi ile değiştirilir. Otomaton durum q_i ' ye geçer. Sonra giriş dizisinin bir sonraki sembolü okunur.

Eğer giriş sembolü $\xi = \lambda$ (boş sembol) olursa, o zaman sembol Z yerine φ_i konur. Otomaton q durumunda kalır. $\varphi_i = \lambda$ olduğunda ise yığındaki Z sembolü silinir.

Eğer giriş dizisinden okunan ξ ve yığının en üstündeki değer olan Z' e karşılık gelen bir $\delta(q , \xi , Z)$ dönüşüm operatörü tanımlı değilse otomaton girişten değer okumayarak beklemeye geçer.
Push - down otomatonun bir diziyi kabul etmesi iki şekilde gerçekleşebilir.

1) Otomaton bekleme durumuna girmeden bütün giriş dizisini okur ve sonunda $q \in F$ sonlanma durumuna gelir.

2) Otomaton bekleme durumuna girmeden bütün giriş dizisini okur ve yığının boşaldığı ($Z = \lambda$ olduğu) $q \in Q$ (boş hafızalı kabul) durumuna girer ($F=\varnothing$).

Örnek 3.9

$$G = (V_N, V_T , R , \sigma) = (\{ \sigma , A , B , C , D \} , \{ a , b , c , d \} , R , \sigma)$$

R : (1) $\sigma \rightarrow aDAB$ (5) $A \rightarrow bCB$
(2) $\sigma \rightarrow aC$ (6) $C \rightarrow c$
(3) $D \rightarrow b$ (7) $B \rightarrow d$
(4) $A \rightarrow bAB$ (8) $A \rightarrow c$

Yukarıdaki şekilde tanımlanan G gramerinin x = { abc } dizisini tanıyıp tanıyamıyacağını bulalım. Bunun için ilk önce otomatonun tanımlanması gereklidir.

$$M = (E,Q,\Gamma,\delta,q_0 ,Z_0 ,F) = (\{ a,b,c,d \} , \{ q_0 \} , \{ \sigma,A,B,C,D \} ,\delta ,q_0 ,\sigma ,\varnothing)$$

1 - $\delta(q_0 , a , \sigma) = \{ (q_0 , DAB) , (q_0 ,C) \}$ (1)(2)
2 - $\delta(q_0 , b , D) = \{ (q_0 , \lambda) \}$ (3)
3 - $\delta(q_0 , b , A) = \{ (q_0 , AB) , (q_0 ,CB) \}$ (4)(5)
4 - $\delta(q_0 , c , C) = \{ (q_0 , \lambda) \}$ (6)
5 - $\delta(q_0 , d , B) = \{ (q_0 , \lambda) \}$ (7)
6 - $\delta(q_0 , c , A) = \{ (q_0 , \lambda) \}$ (8)

Başlangıçta yığın kütüğünde σ vardır ($Z_0 = \sigma$). Otomaton tek $q_0 \in Q$ durumuna ve F=∅ sonlanma durumuna sahiptir. x dizisine ait ilk sembol a geldiğinde 1 nolu dönüşüm operatörü devreye girer. Otomaton, operatörün gösterdiği iki durumdan birine gider. İlk durumu tercih ettiğini kabul edersek, yığın kütüğünden σ' yı silerek D en üstte olacak şekilde DAB dizisini yükler.
İkinci sembol olarak b okunur. Yığının en üstünde D olduğundan 2 nolu operatör devreye girer. Bu operatör yığından D' yi atarak yerine bir şey koymaz. Böylece yığının en üstünde A kalır.

Üçüncü sembol olarak c okunur. Yığının en üstünde de A olduğu için 6 nolu dönüşüm operatörü devreye girer. Bu operatör A' yı yığından atarak yerine bir şey koymaz. Böylece B yığında en üstte kalır.

Tüm x = { a,b,c } dizisi okunduğu halde yığın boşalmamıştır. F = ∅ olduğu için dizinin tanınabilmesi ancak tüm dizi bittiğinde yığının boş kalması ile mümkündür. Fakat bu aşamada, otomatonun giriş dizisini tanımadığı hükmünü vermek erken olur. Çünkü 1 nolu operatörde mümkün olan iki durumdan ilki alınmıştı. Diğer durum ise henüz denenmemişti.

1 nolu operatörde denenmeyen diğer durumu da denemek için, otomaton 1 nolu değişim operatörünün devrede olduğu andaki konumuna getirilir. Buna göre yığında σ vardır. Otomaton giriş sembolü olarak a' yı okur. 1 nolu değişim operatöründeki (q_0 , C) durumu seçilerek yığındaki σ yerine C konur.

Daha sonra giriş dizisinden b okunur. Bu durumda $\delta(q_0 ,b ,c)$ dönüşümü gerekir. Ancak böyle bir dönüşüm bu otomaton tarafından tanımsızdır. Bu, otomatonu bekleme durumuna düşürür. Başka seçenek de olmadığından x={abc} dizisi, M push -down otomatosu tarafından tanınamaz.

BÖLÜM 4

BİLGİSAYAR UYGULAMASI

4.1. GİRİŞ

EKG işaretlerinin sözdizimi metodu ile incelenebilmesi için, bu tez çalışması çerçevesinde Turbo C ile her biri diğerinin daha geliştirilmiş hali olan üç program yapılmıştır. İlki sözdizimi metodu ile QRS deteksiyonu yapmaya yönelik "syntax.c" isimli bir program, diğeri gene sözdizimi metodu ile QRS dedksiyonu yapmaya yönelik fakat QRS komplekslerinin yanısıra P ve T tepelerinin de sözdizimini çıkartabilecek hassasiyette "syntaxdd.c" isimli bir programdır. Ayrıca bu programda, ilk pogramdan farklı olarak, uzun kayıtların işlenip incelenebilmesi için monitörde işaretlerin soldan sağa yumuşak bir şekilde kaydırılması sağlanmıştır. Üçüncüsü ise öncekilere ilave olarak sözdizimi metodu ile bir periyotluk EKG işaretinin şeklini tanıyan "syntaxg.c" isimli bir programdır.

Bu bölümde EKG işaretlerinin sözdizimi metodu ile nasıl yorumlanabileceği, bu tez çalışması içinde bilgisayar uygulamasının nasıl yapıldığı, bunun için nasıl bir algoritma uygulandığı ve bu tezde ne tip yeni metodlar geliştirildiği anlatılacaktır.

Altbölüm 3.2' de de bahsedildiği üzere, tasarlanan bir algoritmanın, sözdizimi metodu ile bir işareti veya bir sınıfı tanıyabilmesi için, ilk önce öğrenme modunda o sınıfa ait örnek işaretler girilerek, bu sınıfın algoritmaya tanıtılması gereklidir. Bu tanıtma işlemi kabaca söylenecek olursa, örnek olarak girilen işaretlerin bir ön işleme tabi tutulup daha sonra ilksellerinin ve sözdiziminin bulunması ve gramatik bir yorumla bu sözdiziminden gramer kurallarının çıkartılması ile olur. (Şekil 3.1) Bu şekilde çıkartılmış olan gramer, öğrenme modunda örnekleri girilmiş olan sınıfa ait bir gramer olur.

Girilen bir işaretin, daha önce öğrenme modunda algoritmaya tanıtılmış olan sınıfa ait olup olmadığının algoritma tarafından tesbit edilmesi, algoritmanın

sınıflama modunda yapılır. Bunun için girilen işaret bir ön işaret işlemeye tabi tutulduktan sonra işaretin ilkselleri ve bu ilksellerin işaretten alınma sırasına göre sözdizimi bulunur. Daha sonra bir sözdizimi analizi ile gramerin bu sözdizisini tanıyıp tanımadığına bakılır. Eğer tanıyorsa algoritma işaretin bu sınıftan olduğuna karar verir. Tanıma işlemini gerçekleyen otomatonlar altbölüm 3.8' de anlatılmıştı.

4.2. QRS DETEKSİYONU İÇİN İLKSEL ÇIKARTMA ALGORİTMASI

Bu tez çalışması çerçevesinde yapılan, sözdizimi metodu ile QRS deteksiyonuna yönelik bilgisayar uygulamasında altbölüm 3.4' te anlatılan algoritma baz alınmıştır. Ancak ihtiyaca göre bu algoritma üzerinde bazı değişiklikler yapılmıştır. Gustavo Belforte tarafından ortaya atılan bu algoritmada EKG işaretinin türev karesi alınarak normalize edilir [8].

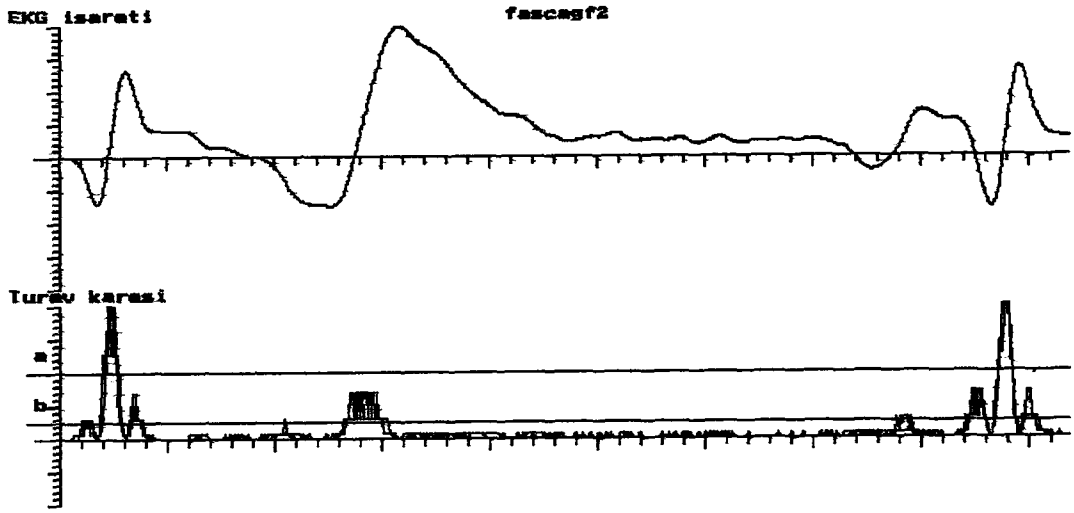
EKG işaretinin zaman domenindeki değişimine x(t) dersek, bunun T periyotlarıyla örneklenmiş hali x(k), (k=0,1,2,...,M) olur. Bu durumda türev karesi s(k) şu şekilde formülize edilir [7].

$$s(k) = \left(\frac{x(k) - x(k-1)}{T} \right)^2 \qquad (4.1) \quad [7]$$

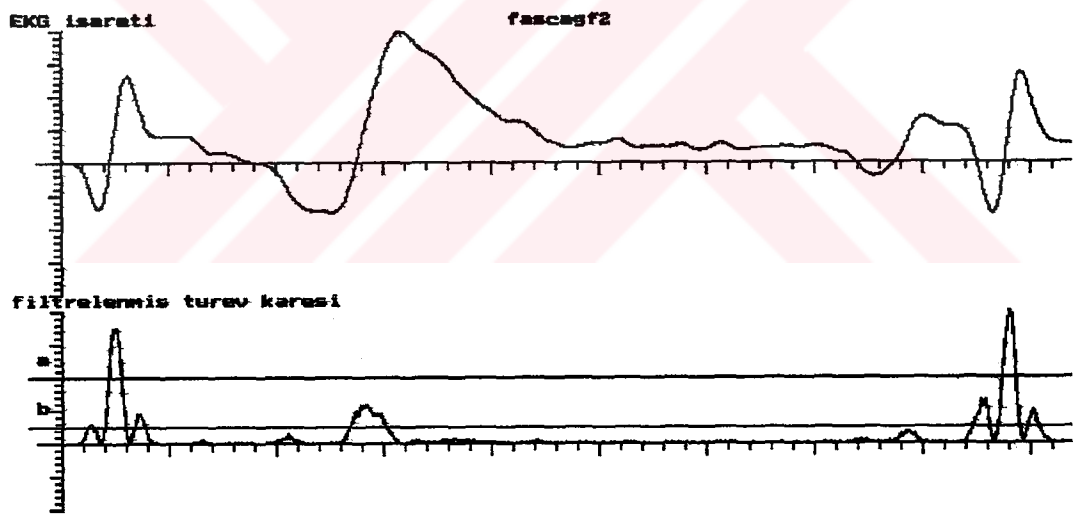
Fakat uygulamada bu şekilde yapıldığında, EKG işaretindeki küçük parazitler bile türev karesinde önemli bozukluklara neden olmaktadır. Şekil 4.1' de bilgsayarda filtrelenmiş ve "fascagf2" adı verilmiş bir EKG işareti ve bunun türev karesi görülmektedir. Görüldüğü gibi türev karesinde bozukluklar vardır.

Türev karesindeki bu bozukluğu gidermek için şekil 4.2' de görüldüğü gibi türev karesi alçak geçiren bir filtreden geçirilmiştir. Ayrıca hem EKG hem de türev karesi normalize edilip sabit bir katsayı ile çarpılmışlardır. İşareti normalize etmek için eşitlik 4.2 kullanılmıştır.

$$N[s(k)] = s_n(k) = \frac{s(k)}{\max(|s(k)|)} \qquad (4.2)$$



Şekil 4.1. Filtrelenmiş bir EKG işareti ve bunun eşitlik 4.1' e göre alınmış türev karesi ("syntax . c " programından elde edilen bir çıktı)



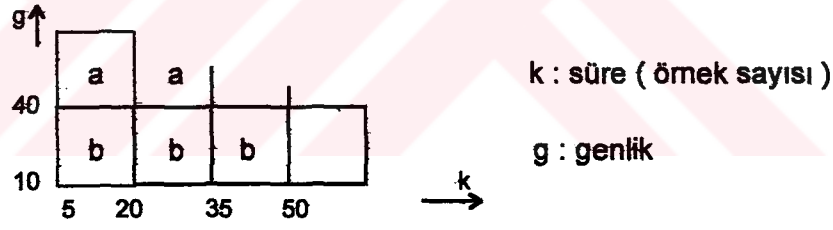
Şekil 4.2. Filtrelenmiş bir EKG işareti ve onun eşitlik 4.1' e göre alınmış türev karesinin alçak geçiren bir filtre ile filtrelenmiş hali ("syntax.c" programından elde edilen bir çıktı)

Algoritmanın daha sonraki aşamasında altbölüm 3.4' te bahsedildiği gibi türev karesi için belirli eşik seviyeleri tayin edilerek her bir eşik seviyesi bir harf ile sembolize edilmiştir. Bu harfler işaretin ilksel takımını oluşturur. "syntax.c"

programında "a" ve "b" olmak üzere iki ilksel, dolayısıyla da iki eşik seviyesi kullanılmıştır.

Tablo 4.1' de de gösterildiği gibi genlik, 10 birimlik eşik seviyenin üstünde ve 40 birimlik eşik seviyenin altında en az beş örnek süresince kalırsa sözdizisine b ilkseli atanır. Eğer en az 20 örneklik bir süre, aynı eşik seviyelerinin içinde kalırsa aynı ilkselden bir tane daha atanır. Bu durum 20 ile 5' in farkı olan 15' in katları için de devam eder. Benzer durum a ilkseli için de geçerlidir. Farklı olarak işaretin genliği 40 birimlik eşik seviyenin üstünde olması gereklidir. Eğer işaret 10 birimlik eşik seviyenin altında 50 örnekten fazla kalırsa çıkartılmakta olan söz dizimi bitirilerek diğerine başlanır. Şekil 3.6' da "syntax.c" programının, filtrelenmiş türev karesini eşik seviyelerine bölerek sözdizimlerini çıkartan bir çıktısı görülmektedir. Ek A' da "syntax.c" programında kullanılan sözdizimi çıkartma alt programının akış diyagramı verilmiştir.

Tablo 4.1. "syntax.c" programında a ve b ilksellerinin, eşik seviyelerine ve zamana göre nasıl atanacağını gösteren tablo



4.3. QRS' NİN YANINDA, P VE T TEPELERİNİN İLKSELLERİNİ DE ÇIKARTABİLMEK İÇİN ALGORİTMANIN GELİŞTİRİLMESİ

Normalize edilmiş EKG türev karesi ,P ve T tepelerine ait çok zayıf bir bilgi taşır. Bu zayıf bilgi de gürültülere karışır. P ve T tepelerinin ilksellerini çıkartabilmek için tablo 4.1' deki eşik seviyeler düşürülürse bu sefer gürültülerin yanlışlıkla P ve T tepeleri olarak algılanma ihtimali çoğalacaktır. P ve T tepelerinin algılanmasındaki hassasiyeti arttırabilmek için eşitlik 4.1' deki formül yerine bu tez çalışmasında geliştirilmiş olan eşitlik 4.3 ile verilen formül kullanılmıştır.

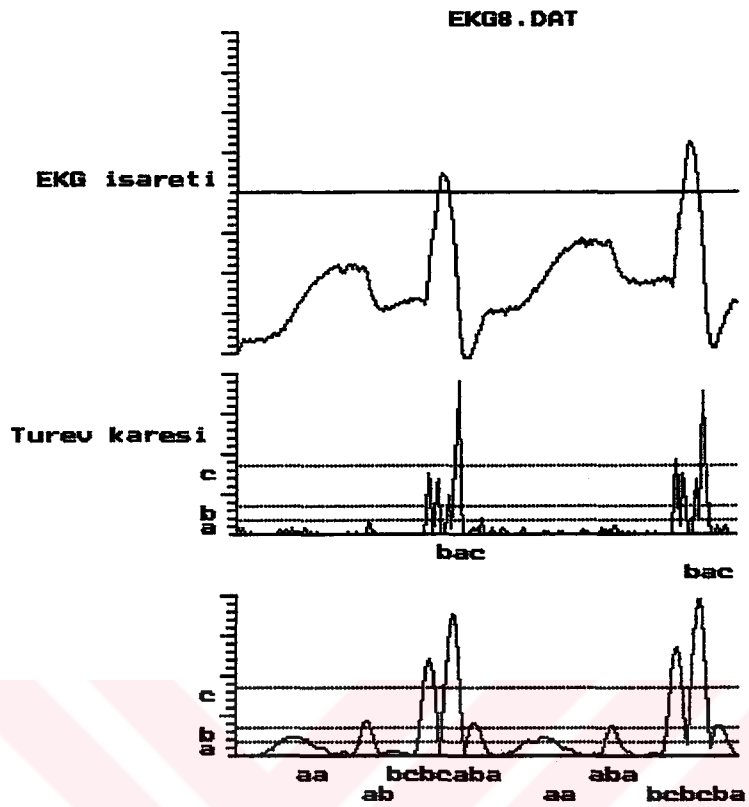
$$s(k) = \frac{\left| \frac{\sum_{i=0}^{9} x(k+i)}{10} - \frac{\sum_{i=0}^{9} x(k-i)}{10} \right|}{10T} \qquad (4.3)$$

Bu formül kısaca izah edilecek olursa; eşitlik 4.1' deki gibi k. örnek ile bir önceki örneğin farkını almak yerine, k. örnekten sonraki 10 örneğin ortalaması ile k. örnekten önceki 10 örneğin ortalamasının farkını almak suretiyle bir türev işlemi gerçekleştirilmektedir. Bu türev işlemi gürültülerin etkisini minimuma indirmektedir. Bunun için bu türev işleminden sonra filtreleme işlemine gerek yoktur. Ancak, eğer EKG işaretinde fazla miktarda gürültü var ise eşitlik 4.3 uygulanmadan önce bu gürültüleri azaltacak şekilde işaretin bir filtreden geçirilmesi gerekir. Eşitlik 4.3 dikkat edilecek olursa, eşitlik 4.1' deki kare alma işlemi yerine mutlak değer alınmıştır. Bu şekilde yapılmasının sebebi, kare alma işleminin büyük genlikli tepeleri katlayarak büyütmesine karşılık, küçük genlikli tepeleri küçük bırakmasındandır. Bu ise, türevleri QRS kompleksinin türevinden çok daha küçük kalan P ve T tepelerinin deteksiyonunu zorlaştıran bir faktördür. Gerçi kare alma yerine multak değer alındığında bu sefer de QRS komplekslerinin deteksiyonundaki hassasiyet azalmaktadır; fakat genelde QRS kompleksleri yeteri kadar büyük genlikli işaretler olduğu için bu, QRS komplekslerinin deteksiyonunda önemli bir dezavantaj değildir (şekil 4.3).

Şekil 4.3.(b)' de görüldüğü gibi, Belforte' un uyguladığı EKG türev karesinde P tepesine ait türev karesi, sözdizimi çıkartılamıyacak kadar küçük kalmaktadır. Bu tezde buna alternatif olarak geliştirilmiş olan eşitlik 4.3' e göre alınmış türevde ise şekil 4.3.(c)' de de görüldüğü gibi P tepesinin çıkış ve iniş eğimlerine denk gelen iki tepecik oldukça belirgin hale gelmiştir.

Şekil 4.3' te görülen türevler, eşitlik 4.2' ye göre normalize edilmemişlerdir. Bunun yerine bu tez çalışmasında geliştirilmiş olan eşitlik 4.4 uygulanmıştır.

$$N[s(k)] = S_n(k) = \frac{s(k)}{E(s(k))}, \quad E(s(k)) = \frac{\sum_{k=0}^{M} s(k)}{M} \qquad (4.4)$$



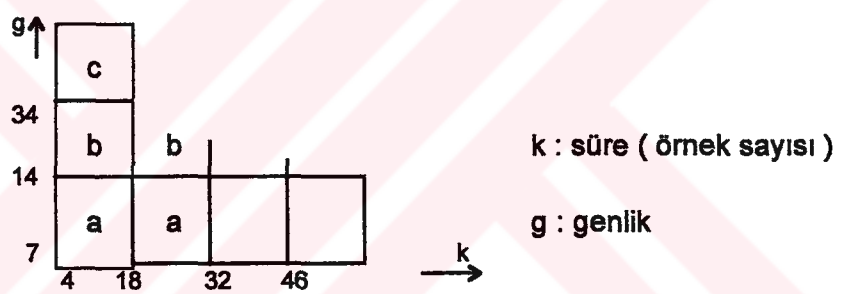
Şekil 4.3. **(a)** EKG işareti **(b)** Belforte tarafından geliştirilen EKG' nin eşitlik 4.1' e göre alınmış türev karesi ve sözdizimleri **(c)** Bu tez çalışmasında geliştirilen P ve T tepelerinin türevlerini belirginleştiren eşitlik 4.3' e göre alınmış yeni türev ve sözdizimleri ("syntaxdd.c" programından elde edilen çıktılar.)

Eşitlik 4.2' de verilen bir normalizasyon işlemi yerine bu tezde geliştirilen eşitlik 4.4' ün uygulanmasının nedeni, çeşitli denemeler sonucunda görülen şu sakıncayı ortadan kaldırmak içindir. Eşitlik 4.2 ile verilen formülde, tüm işaret türevinin maximumu bulunarak, bütün türev bu maximuma bölünür. Halbuki türevin maximumu, yeteri kadar kararlı ve güvenilir bir büyüklük değildir. Çünkü, dakikalarca uzunluktaki bir EKG kaydında, herhangi bir sebepten oluşabilecek olan, dikliği yüksek tek bir iniş veya çıkış bile, büyük bir türevin oluşmasına neden olabilecek, bu ise maximum türevin fazla büyük olmasına neden olabilecektir. Eşitlik 4.2' ye göre, bu bir anlık durum, dakikalarca uzunluktaki bütün bir EKG kaydının türevinin genliğini etkileyecektir. Halbuki, eşik seviyeleri sabit olduğu için, türev genliğindeki kaymalar, sözdizimlerini de değiştirecektir.

İşte bu kararsız durumdan kurtulabilmek için, eşitlik 4.4' te türev maksimumu yerine, daha kararlı bir büyüklük olan bütün bir kaydın türev ortalaması alınmıştır.

"syntax.c" programında a ve b olmak üzere iki olan ilksel sayısı, şekil 4.3' te de görüldüğü gibi "syntaxdd.c" programında a, b ve c olmak üzere üçe çıkartılarak türev üç tane eşik seviyesine bölünmüştür. Eşik seviyeleri, hem QRS' leri, hem de P ve T tepelerini de içine alacak şekilde yeniden düzenlenmiştir (Tablo 4.2). Bu, hem QRS koplekslerinin türevini, hem de, QRS' ninkinin yanında genliği çok daha küçük kalan P ve T tepelerinin türevlerini uygun bir şekilde eşik seviyelerine bölebilmek için gerekli bir değişikliktir.

Tablo 4.2. "syntaxdd.c" programında a, b ve c ilksellerinin, eşik seviyelerine ve zamana göre nasıl atanacağını gösteren tablo

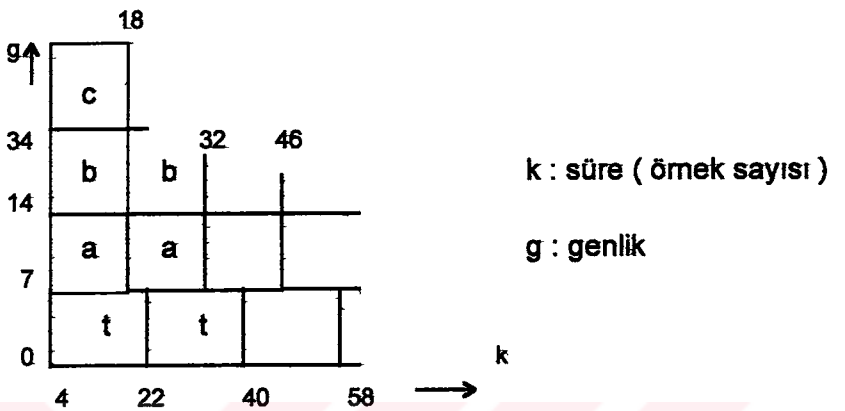


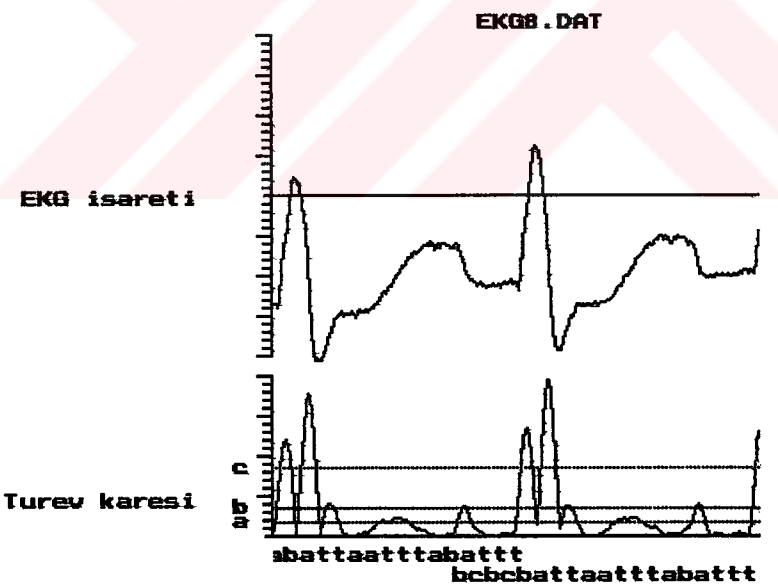
4.4. BİR PERİYODLUK EKG İŞARETİNİN İLKSELLERİNİ ÇIKARTMA

"syntaxg.c" programında, şekil tanımaya yönelik olarak, bir periyotluk EKG işaretinin sözdizimini çıkartmak için algoritma biraz daha geliştirilmiştir. Burada, belli büyüklükteki her tepecik için bir sözdizimi çıkartmak yerine, bir periyotluk Q-Q aralığı için bir sözdizimi çıkartılmıştır. Ayrıca eşik seviyenin altında kalan kısımlar için de t ilkseli atanmıştır. Böylece türev tepeleri arasındaki boşluklar t ile sembolize edilmişlerdir. Tablo 4.3' de bu ilksellerin nasıl atanacağı gösterilmiştir. Bu tabloda bir noktaya açıklık getirmek gereklidir. Dikey sütundaki sfır seviyesi, bir eşik seviye değildir. Tabloda görülen 7 genlikli eşik seviyesi, hem a ilkselinin hem de t ilkselinin eşik seviyesidir. Türev, bu eşik seviyesini 4 örneklik süreyle geçerse a ilkseli atanır; fakat 4 örneklik süre bu

eşiğin altında kalırsa t ilkseli atanır. Şekil 4.4' te, anlatılan bu algoritmaya göre sözdizimi çıkartan "syntaxg.c" programının bir çıktısı görülmektedir.

Tablo 4.3. "syntaxg.c" programında a, b, c ve t ilkssellerinin, eşik seviyelerine ve zamana göre nasıl atanacağını gösteren tablo

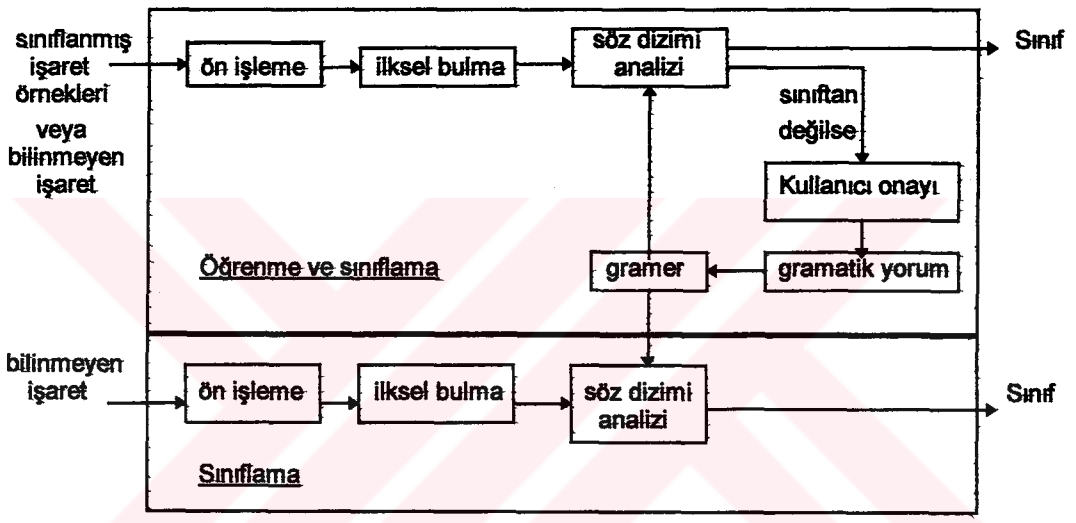





```
Bu normal bir EKG midir ? (E/H)
```

Şekil 4.4. Bir periyotluk EKG işaretinin sözdizimini çıkartan "syntaxg.c" isimli turbo C programının çıktısı

Bütün bu işlemler, genel çerçeve olarak şekil 3.1' de verilmiş olan şematik sistem çerçevesinde yapılmıştır. Daha ileriki aşamada bu sistem biraz daha geliştirilmiştir. Şekil 3.1' de verilen sistemde, öğrenme modunda, bir sınıfa ait belirli sayıda örnek EKG ard arda girilmek suretiyle çıkartılan gramer ile sınırlı kalınıyordu. İleride karşımıza çıkabilecek daha farklı örnekleri de girerek, grameri genişletme imkanı yoktu. Yeni sistemde, istenildiği an yeni EKG örnekleri girerek grameri bu yeni örnekleri de içine alacak şekilde genişletme imkanı sağlandı. Bu yeni sisteme ait şema şekil 4.5' te gösterilmektedir.



Şekil 4.5. Geliştirilmiş bir sözdizimi tanıma sistemi [7]

BÖLÜM 5

SONUÇ VE ÖNERİLER

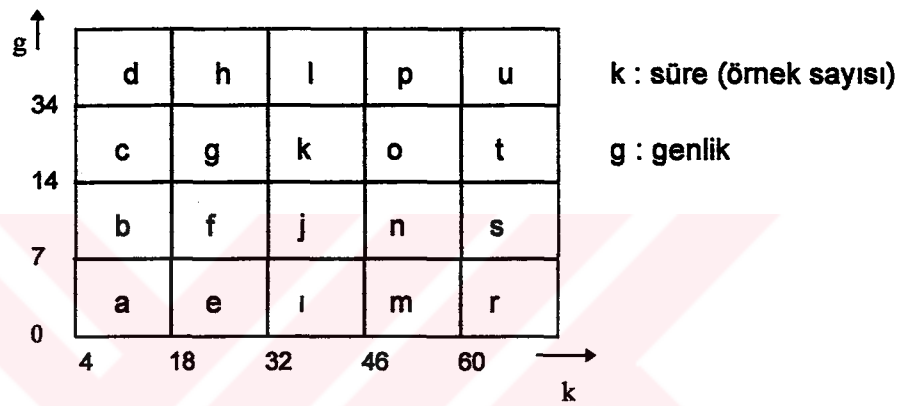
Bu tez çalışmasında, konuya bir ön bilgi olması açısından, kalbin yapısı ve çalışması, EKG işaretlerinin oluşumu ve aritmiler, genel hatlarıyla incelenmiştir. Bölüm 3' te sözdizimi metodunun teorisi incelenmiştir. Bölüm 4' te sözdizimi metoduyla EKG işaretlerini analiz ederek şekil tanıma ve sınıflama yapabilmek için algoritma geliştirilmiş ve bilgisayar uygulamaları ile denenerek olgunlaştırılmaya çalışılmıştır.

Bir periyotluk EKG işaretinin şeklinin tanınması probleminde, Şekil 4.5' te gösterilen şemadaki gramatik yorum işleminin bilgisayarda gerçekleştirilmesi sırasında bazı zorluklarla karşılaşılmıştır. Bu zorlukların nedeni, hem bir periyotluk EKG' nin sözdiziminin fazla uzun olması, hem de öğrenme modunda bir sınıfa ait fazla sayıda farklı EKG örnekleri girerek o sınıfı temsil eden grameri kapsamlı tutma isteğidir. Bu gibi faktörler, gramer ifadelerini çok fazla arttırarak karışmasına, ayrıca gramerde non-terminal sembolleri olarak kullanılan alfabedeki büyük harflerin yetmemesine neden olmaktadır. Halbuki QRS veya P ve T tepelerinin deteksiyonunda bu problemle karşılaşılmamıştı. Çünkü bunların sözdizimleri kısa ifadelerden oluşuyordu.

Bu probleme çözüm getirebilmek için algoritmada bazı değişiklik önerileri düşünülmüş, fakat bunları bizzat bilgisayarda deneyerek görme imkanı olmamıştır. Örneğin Şekil 4.4' te de görüldüğü gibi bir periyotluk bir EKG' nin sözdizimi "bcbcbattaatttabattt" şeklinde 19 harflik bir dizi şeklinde çıkmıştır. Görüldüğü üzere bazı harfler, içinde bulunulan eşik seviyesinde kalma süresini temsil etmek amacıyla ard arda birden fazla gelmiştir. Taşıdığı süre bilgisini kaybetmeden bunları bir harfe indirebilmek amacıyla her eşik seviyesindeki farklı kalma süreleri farklı harflerle ifade edilmiştir. Böylece terminal eleman sayısı artarken, sözdizimi ifadeleri kısalmıştır. Bu durumda, Tablo 4.3' te verilen ilkselleri atama tablosu yeniden düzenlenerek Tablo 4.4 elde edilmiştir. Aslında Gustove Belforte tarafından ortaya atılan algoritmada kullanılan tablo da benzer

şekildeydi. Ancak uygulamanın ilk aşamasında çıkartılan sözdizimleri zaten kısa olduğu için Belfortun kullandığı atama tablosundaki gibi farklı kalma süreleri için farklı terminal elemanları kullanmak yerine, aynı terminal elemanı ard arda tekrar ettirilmişti. Fakat uygulamada gelinmiş olan şu son aşamada, bu şekilde yapmanın hata olduğu anlaşılmıştır.

Tablo 4.4. Ard arda harf tekrarı yapmadan ilksellerin eşik seviyelerine ve zamana göre nasıl atanacağını gösteren tablo.



Bu durumda "bcbcbattaatttabattt" 19 harfli sözdizisi, yukarıdaki tabloya göre büyük ihtimal "cdcdcbıfmbcbm" 13 harfli sözdizisine dönüşecektir. Bu ise gramatik yorumu önemli ölçüde kolaylaştıracak ve bahsedilen problemlere çözüm getirecektir. Ancak gene de, gramer genişletildikçe, non-terminal elemanı olarak kullanılan alfabedeki büyük harflerin yetmeme ihtimali her zaman olacaktır. Bu durumda non-terminal elemanı olarak ilave semboller getirilerek çözüm bulunabilir. Aslında, bölüm 3.6' de anlatılan mevcut gramer çeşitlerinin yanında, bilgisayarda gramatik yorum yapmaya daha elverişli bir gramer çeşidinin ortaya atılması çok daha köklü bir çözüm olacaktır. Çünkü şu anki durumda, öğrenme modunda girilen çok sayıdaki sözdizimini tanıyabilecek ve bunların haricinde olanların tümünü reddedecek bir grameri bilgisayarda oluşturmak, çok karışık bir mantık örgüsü gerektirmektedir. Bunlara ilaveten bir de bölüm 4.4' te anlatıldığı gibi, gramer çıkartıldıktan sonra yeni örnek işaretler girerek buna göre gramere ilave yapabilme özelliği kazandırmak işlemi daha da karışık hale getirmektedir. Bu tez çalışması sırasında en çok zorlanılan ve

üzerinde en çok vakit kaybedilen konu, bilgisayara bu gramatik yorumu yaptırmak için gerekli alt programı yazmak olmuştur.

Şekil 3.1 ve Şekil 4.5' te sözdizimi analizi olarak belirtilen ve sınıfı bilinmeyen işaretlerin sözdizimlerinin mevcut gramere uyup uymadığını kontrol ederek sınıflama yapmaktan ibaret olan işlem, gramatik yoruma göre çok daha kolay ve sorunsuz olarak gerçeklenebilmiştir.

Buraya kadar olan şekliyle tasarlanmış olan algoritma, EKG' nin türevindeki negatif ve pozitif kısımları ayırd etme yeteneğine sahip değildir. Halbuki bazı aritmilerin deteksiyonunda bu özellik gerekli olmaktadır. Örneğin Şekil 2.10.E' de ters dönmüş T dalgası içeren bir aritmi görülmektedir. Algoritmaya, bu tip aritmileri de tanıyabilme yeteneğini kazandırmak için formül 4.3 ile verilen ifadede mutlak değer kaldırılmalıdır. Bu durumda, negatif türevler de ortaya çıkacağı için, Tablo 4.4' teki ilksel atama tablosunun benzerini negatif kısım için de yapmak gerekecektir. Bu durumda küçük harflerle ifade edilen terminal elemanı sayısı iki katına çıkacaktır.

KAYNAKLAR

[1] ALANYALI, G. , Otamatik Aritmi Deteksiyonu, İTÜ İstanbul 1993

[2] YAZGAN, E. , Biyomedikal Düzenler Ders Notları, İTÜ İstanbul 1992-1993

[3] KORÜREK, M, Biyomedikal Mühendisliğinde Özel Konular Ders Notları, İ.T.Ü. , İstanbul 1988

[4] DPT, Biyolojik İşaretlerin İşlenmesi Ve Değerlendirilmesi İstanbul 1993

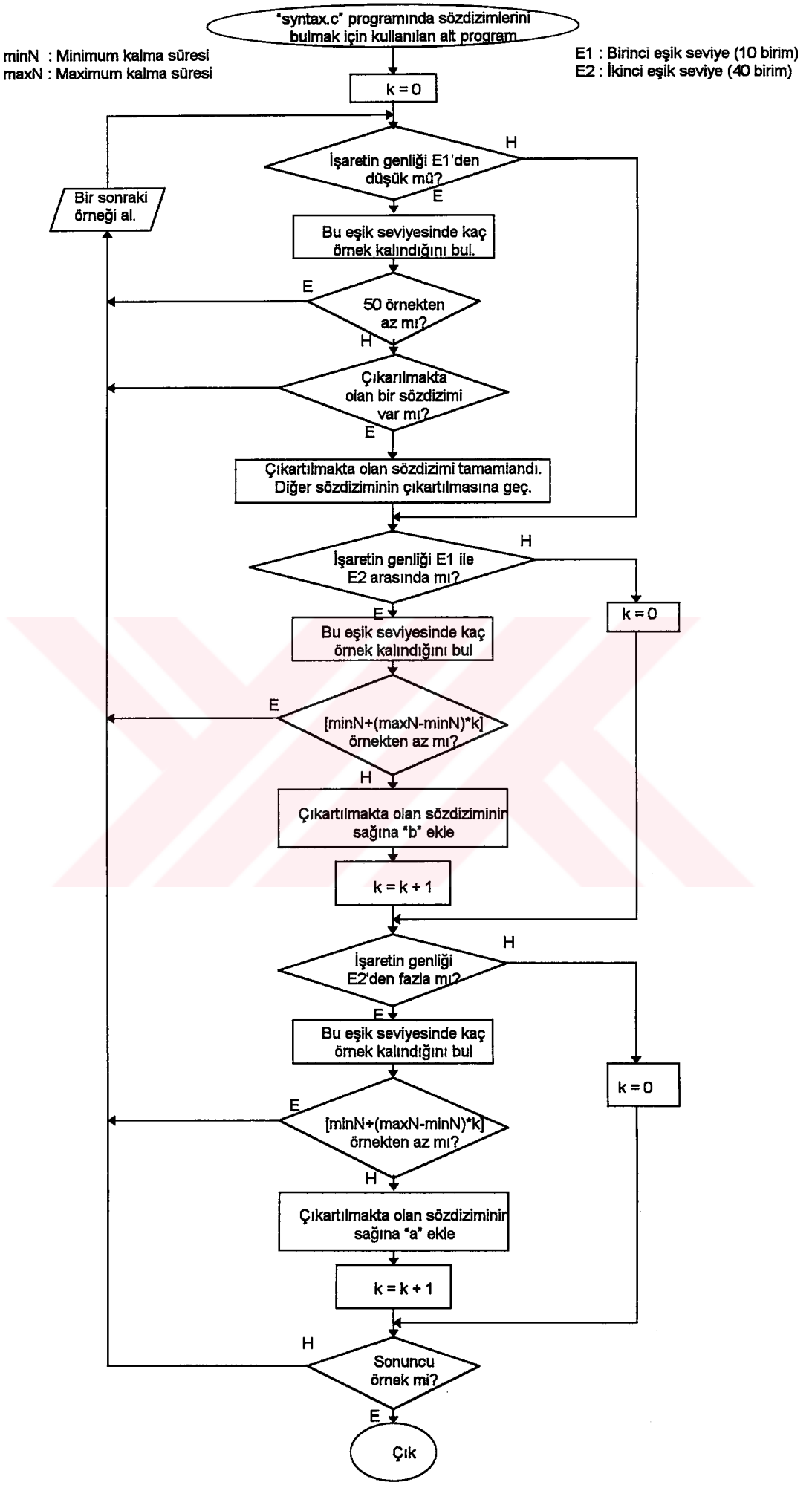
[5] YAZGAN, E. , Tıp Elektroniğine Giriş Ders Notları, İTÜ İstanbul 1990-1991

[6] UÇAR, D. , Elektrokardiyografi, Aycan Yayıncılık, İstanbul 1989

[7] COHEN, A. , Biomedical Signal Processing, Vol. 2, CRC Press, INC. Florida 1986

[8] BELFORTE, G. , DE-MORİ, R. , and FERRARİS, F. , A Contribution to the Automatic Processing of Electrocardiograms Using Syntactic Methods, IEEE Trans. Biomed. Eng. ,26 ,125 ,1979

EK A


"syntax.c" programında sözdizimlerini bulmak için kullanılan alt program
minN : Minimum kalma süresi
maxN : Maximum kalma süresi
E1 : Birinci eşik seviye (10 birim)
E2 : İkinci eşik seviye (40 birim)
k = 0
İşaretin genliği E1'den düşük mü?
H
E
Bir sonraki örneği al.
Bu eşik seviyesinde kaç örnek kalındığını bul.
E
50 örnekten az mı?
H
Çıkarılmakta olan bir sözdizimi var mı?
E
Çıkartılmakta olan sözdizimi tamamlandı. Diğer sözdiziminin çıkartılmasına geç.
İşaretin genliği E1 ile E2 arasında mı?
H
k = 0
E
Bu eşik seviyesinde kaç örnek kalındığını bul
E
[minN+(maxN-minN)*k] örnekten az mı?
H
Çıkartılmakta olan sözdiziminin sağına "b" ekle
k = k + 1
İşaretin genliği E2'den fazla mı?
H
E
Bu eşik seviyesinde kaç örnek kalındığını bul
k = 0
E
[minN+(maxN-minN)*k] örnekten az mı?
H
Çıkartılmakta olan sözdiziminin sağına "a" ekle
k = k + 1
Sonuncu örnek mi?
H
E
Çık

EK B

SYNTAX.C

```
#include<stdio.h>
#include<ctype.h>
#include <graphics.h>
#include<values.h>
#include<dos.h>
#include<conio.h>
#include<fcntl.h>
#include<io.h>
#include<stat.h>
#include<float.h>
#include<string.h>

int pop,E1=10,E2=40,T1=5,T2=20;
int graphdriver,graphmode;
void discurv3(),wrt_graf(),syntax(int),gramer(),grm_yaz(int),dedect();
char *b1,*b2,*t1;
char *rd1(),*rd2(),*rd3();
char se,*mp1,*mp2;
char min,max,sec,scz,ss;
char sm1[1500],sm2[1500],s[1500];
char e2[30],e5[30],ff[60],grm[30][30],grm1[30][30],syn[30][30];
char  *gg1,*gg2;
char vnt[]="QABCDEFGHIJKLMNOPRSTUVWYZF";

float sd[1500];
float fd[1500],gd[1500],jd[1500];

float x,smin,smax;
int alc_gec(),exit(),write1();
int tur_kar();
int cizim();
int i,j,k,l,m=0,gp;
int sira[25];

main()
{
/*     char seciz;     */

  bas:  clrscr();

        l=0;

        printf("\n\n\n       BYOMEDKAL MHENDSLÎNDE ZEL KONULAR\n");
        printf("                  DERS DEV\n");
        printf("\n\n   KONU         : SZDZM (syntactic) METOD LE QRS DEDEKSYONU
        printf("\n   DEV VEREN : Do. Dr. Mehmet KORREK\n");
        printf("\n   HAZIRLAYAN  : Elektr. Mh. Sergin ZEN (490 Y.068)\n");
        printf("\n\n\n        kmak iin Q'ya devam iin herhangi bir tua basnz")

        sec=getch();
        if(sec == 'q')
               exit(0);

/*      printf("\n\n\n VERI DOSYALARINI CIZDIRMEK ISTERMISINIZ ? (E,H)");

        while ((seciz=getch())!='e' && seciz!='h');     */

        clrscr();
        printf("\n\n         KISIM 1 :  ÎRENME (training) MODU\n");
        printf("\n\n\n     1 : SARET DOSYASINI ZDREREK GRAMERN IKARTMA\n");
        printf("\n     2 : GRAMER DOSYASINDAN ALDIÎI HAZIR GRAMER KULLANMA\n")
```

```
        while((sec = getch()) != '1' && sec != '2');
        if(sec == '1')
        {
                for(i=0;i<30;i++)
                {strcpy(syn[i],"");strcpy(grm1[i],"");strcpy(grm[i],"");}
                m=0;i=0;j=0;k=0;l=0;gp=0;pop=0;
                tur_kar(1);
                gramer();
                grm_yaz(1);
        }

        else
        {
                clrscr();
grm:            printf("\n\n   Gramer DOSYASI : ");
                b1=rd3();
                if(b1==0L) goto grm;
                grm_yaz(0);
         }
                m=0;
                tur_kar(0);
                goto bas;

}

tur_kar(p)
{
      float smin,smax,a,b,c,d,e;
yen:            clrscr();
        if(p==1) printf("\n\n          KISIM 1 :  ÎRENME (training) MODU\n");
        else     printf("\n\n          KISIM 2 :  DEDEKSYON  MODU\n");


                    don:   printf("\n\n\nIsaret DOSYASI: ");

                           b1 = rd1();
                           if(b1 == 0L) goto don;


    smax=0;

    for(i=0;i<1019;i++)
    {
        a=(sm1[i+1]-sm1[i]);
        sd[i]=a*a;
        if(sd[i]>smax)   smax=sd[i];

    }
     sd[1020]=sd[1019];


     for(i=0;i<1020;i++) s[i]=sd[i]*80/smax;


         mp1=&sm1[0];
         mp2=&s[0];
         se='2';
         gg1="EKG isareti";
         gg2="Turev karesi";

         discurv3(mp1,mp2,se,gg1,gg2);

         while((ss=getch()) != 'b' && ss != 'f' && ss!='q');
```

```
        if(ss=='q')  goto q;
        if(ss=='b')
        {
               syntax(p);

               while((ss=getch()) != 'b');

        }

        else
        {
               alc_gec();
               syntax(p);
               while((ss=getch()) != 'b');
        }

q:      closegraph();
        clrscr();

        if(p==0) m=0;

        printf("\n\n       Yeni deneme yapmak istermisiniz ? (E/H) :\n");
        while((ss=getch())!='e' && ss!='h');
        if(ss=='e')  goto yen;

     return;

}


char *rd1()
{
    char *ptx1,*ptx2;
    int open(),read(),close(),handle;

    ptx1 = &sm1[0];
    ptx2 = ptx1;
         scanf("%10s",&e2[0]);
         handle = open(&e2[0],O_RDONLY|O_BINARY,S_IREAD);

         if(handle == -1)
         {
               printf("cannot open file");
               ptx2 = 0L;
         }

         if(read(handle,ptx1,1020) == -1)
         {
            printf("\n\nread error !");
            ptx2 =0L;
         }

         close(handle);
         return(ptx2);
}

char *rd2()
{

    char *ptx5,*ptx6;
    int open(),read(),close(),handle;

    ptx5 = &sm2[0];
    ptx6 = ptx5;
```

```
        scanf("%10s",&e5[0]);
        handle = open(&e5[0],O_RDONLY|O_BINARY,S_IREAD);

        if(handle == -1)
        {
               printf("cannot open file");
               ptx6 = 0L;
        }

        if(read(handle,ptx5,1020) == -1)
        {
           printf("\n\nread error !");
           ptx6 =0L;
        }

        close(handle);
        return(ptx6);
}

void discurv3()
{
       int p1,p2,p3,p4,dd;
       int graphdriver = DETECT, graphmode;
       int t;
       float b;


       b=1020./650.;
       dd=650/50;


      initgraph(&graphdriver,&graphmode,"c:\\tc\\bgi");
      cleardevice();

      setbkcolor(15);
      setcolor(8);
      line (30,10,30,170);
      line (15,90,715,90);


      outtext(gg1);

      outtextxy(300,0,&e2[0]);


        for(i=1;i<=20;i++)
        {
          line(30,90-i*4,26,90-4*i);
          if( (i/5)*5 == i )  line(30,90-i*4,23,90-i*4);
        }

        for(i=1;i<=20;i++)
        {
          line(30,90+i*4,26,90+i*4);
          if( (i/5)*5 == i )  line(30,90+i*4,23,90+i*4);
        }

      for(i=1;i<51;i++)
      {
          line(dd*i+30,90,dd*i+30,94);
          if((i/5)*5 == i)  line(dd*i+30,90,dd*i+30,97);
      }
```

```
for(t=0;t<1020;t++)
{
    p1= 30+t/b;
    p2=-(*mp1)+90;
    p3= 30+(t+1)/b;
    p4= -(*(mp1+1))+90;

    line(p1,p2,p3,p4);
    mp1++;
}

if(se=='1') return;

line (30,180,30,340);
line (15,260,715,260);

outtextxy(0,170,gg2);
/*
i=0;

while (e2[i]!=0)
{
  ff[i]=e2[i];
  i++;
  printf("\n%d %c",i,ff[i]);

}


j=0;


while(e5[j]!=0)
{
  ff[i+j]=e5[j];
  j++;
  printf("\nff[%d]=%c i=%d",j,ff[i+j],i);

}
  */

outtextxy(300,170,&e5[0]);
  outtextxy(15,205,"a");
  outtextxy(15,235,"b");

for(i=1;i<=20;i++)
{
    line(30,260-i*4,26,260-4*i);
    if( (i/5)*5 == i )  line(30,260-i*4,23,260-4*i);
}

for(i=1;i<=20;i++)
{
    line(30,260+i*4,26,260+4*i);
    if( (i/5)*5 == i )  line(30,260+i*4,23,260+4*i);
}

for(i=1;i<51;i++)
{
    line(dd*i+30,260,dd*i+30,264);
    if((i/5)*5 == i) line(dd*i+30,260,dd*i+30,267);
}

for(t=0;t<1020;t++)
```

```
        {
            p1= 30+t/b;
            p2=-(*mp2)+260;
            p3= 30+(t+2)/b;
            p4= -(*(mp2+2))+260;


            line(p1,p2,p3,p4);
            mp2++;
        }

          line(10,(260-E1),715,250);
          line(10,(260-E2),715,220);


}


write1(char *np)
{
          int open(),write(),close(),handle;
          int a;
          char e1[20];

          printf("\n\n\n File ismi: ");
          scanf("%10s",&e1[0]);

          handle = open(&e1[0],O_CREAT|O_TRUNC|O_RDWR|O_BINARY,S_IWRITE);

          if(handle == -1){

                  printf("cannot open file");
                  exit (0);
          }

           a = write(handle,np,1020);

            if(a == -1){

              printf("write error !");
            }

          close(handle);

          return;
}

alc_gec()

{
          float smin,smax,a,b,c,d,e;
          int f=500,fm=4;

          if(fm==fm/2*2) k=1000./f-0.5;
          else k=(1000./f-1)/2+0.5;
          printf("\n\nk=%d",k);
          smin=s[0];
          smax=smin;

          for(i=0;i<1020;i++)

          {
             if(i<4*k+4) a=0.;
             else {a=s[i-4*k-4];}
```

```
            if(i<3*k+3) b=0.;
            else {b=s[i-3*k-3];}

            if(i<2*k+2) c=0.;
            else {c=s[i-2*k-2];}

            if(i<k+1) d=0.;
            else {d=s[i-k-1];}

            e=s[i];

            sd[i]=-sd[i-4]+4.*sd[i-3]-6.*sd[i-2]+4.*sd[i-1];
            sd[i]=sd[i]+a-4.*b+6.*c-4.*d+e;

            if(sd[i]>smax) smax=sd[i];
            else if(sd[i]<smin) smin=sd[i];
          }


      if(smin<0)   smin=-smin;
      if(smax<0)   smax=-smax;

      if(smax<smin) smax=smin;

      for(i=0;i<1020;i++) s[i]=sd[i]*80/smax;


          clrscr();

          mp1=&sm1[0];
          mp2=&s[0];
          se='2';
          gg1="EKG isareti";
          gg2="filtrelenmis turev karesi";

          discurv3(mp1,mp2,se,gg1,gg2);

          while(getch() != 'b') ;

          return;

}

void syntax(p)

{
        char tb=0,ta=0,d=0; int k1=0;
        i=0;   k=0;   gp=0;

tur:    while(s[i]<E1 && i<1020)

          {
                ta=0;   tb=0;   i++;   k1++;
                if(k1==50 && d==1)

                {
                        d=0;
                        syn[m][gp]='\0';
                        gp=0; m++;
                        if(p==0) dedect();
                        else wrt_graf();
                }
          }
```

```
        k=0;

        while(s[i]>=E1 && s[i]<E2 && i<1020)

        {
              ta=0;   tb++;   i++;

              if(tb==(T1+(T2-T1)*k))

              {
                      syn[m][gp]='b';
                      k++;  gp++;  d=1;  k1=0;
              }
        }

        if(k>1) k--;
        if((tb-(T1+(T2-T1)*k))>=5)

        {
              syn[m][gp]='b';
              d=1;  gp++;
        }

        k=0;

        while(s[i]>=E2 && i<1020)

        {
              tb=0;   ta++;   i++;
              if(ta==(T1+(T2-T1)*k))

              {
                  syn[m][gp]='a';
                  k++;  d=1;  gp++;  k1=0;
              }
        }

        if(k>1) k--;
        if((ta-(T1+(T2-T1)*k))>=5)

        {
              syn[m][gp]='a';
              d=1;  gp++;
        }

        k=0;

        if(i<1020)  goto tur;

        if(d==1)
        {
              syn[m][gp]='\0';
              gp=0; m++;
              if(p==0) dedect();
              else wrt_graf();
        }
        syn[m][0]='\0';
}

void wrt_graf()

{
```

```
        int t;    j=m-1;

        outtextxy((-3+(i-50)/1.53),280,syn[j]);
        outtextxy(80,330,"Bu gecerli bir QRS midir ? (E/H)");
        while((sec=getch()) != 'e' && sec != 'h');

        if(sec=='e')
        {
                outtextxy((-18+(i-50)/1.53),300,"---QRS---");
                for(t=0;t<j;t++)
                {
                        if(strcmp(syn[j],syn[j-t-1])==0)
                        {
                                m--;
                                t=j;
                        }
                }
                return;
        }
        else  m--;

}


void gramer()

{
        clrscr(); gp=0;
        for(k=1;k<30;k++)
        {
                for(i=0;i<m;i++)
                {
                        if(strcmp(&syn[i][k],"")==0)  goto cik;
                        for(j=1;j<=i;j++)
                        {
                                if(i<j) { gp++;  strcpy(grm1[gp],&syn[0][k]);
                                if(strcmp(&syn[i][k],&syn[i-j][k])==0)  goto c

                        }
                        l=0;
                        while(strcmp(grm1[gp-l],"")!=0 )
                        {if(strcmp(grm1[gp-l],&syn[i][k])==0) goto cik;
                         if(strcmp(grm1[gp-l-1],grm1[gp])==0) goto cik;
                         l++; }

                        gp++;
                        strcpy(grm1[gp],&syn[i][k]);

cik:                    l++; l--;

                }
        }
        i=0; j=1;
        while(strcmp(syn[i],"")!=0)
        {
                j=1;
                grm[i][0]=vnt[0];
                grm[i][1]=syn[i][0];
                while(strcmp(grm1[j],&syn[i][1])!=0 && j<30) j++;
/*          if(j==0) { printf("\n\n  Gramer cikarilamiyor.\n"); getch(); retur
                grm[i][2]=vnt[j];
                i++;
        }
```

```
        k=1;
        while(k<=gp)
        {
                grm[i+k-1][0]=vnt[k];
                grm[i+k-1][1]=grm1[k][0];
                grm[i+k-1][2]='\0';
                if(strcmp(&grm1[k][1],"")==0) goto on;
                for(l=1;l<=gp;l++)
                {
                        if(l==k) l++;
                        if(strcmp(&grm1[k][1],&grm1[l][0])==0) goto in;
                }
                printf("\n\n  Gramer cikartilamiyor. \n");
                return;
in:             grm[i+k-1][2]=vnt[l];
on:             k++;

        }
}

void grm_yaz(a)

{

        clrscr();
        printf("\n---------------------------- G R A M E R ------------------
        l=0;
        while(grm[l][0]!='' && l<30)
        {
                if(l==(l/2)*2) printf("\n\n");
                printf("              %c : %s ",grm[l][0],&grm[l][1]);
                l++;
        }
        if(a==0)
        {
                while(getch()=='');
                return;
        }
        printf("\n\n\n        Gramer kaydedilsin mi ? (E/H) :");
        if((ss=getch())=='h') return;

        t1=grm[0];
        write1(t1);
        return;
}


char *rd3()
{
    char *ptx1,*ptx2;
    int open(),read(),close(),handle;

    ptx1 = grm[0];
    ptx2 = ptx1;
         scanf("%10s",&e2[0]);
         handle = open(&e2[0],O_RDONLY|O_BINARY,S_IREAD);

         if(handle == -1)
         {
                printf("cannot open file");
                ptx2 = 0L;
         }

         if(read(handle,ptx1,1020) == -1)
```

```
        {
           printf("\n\nread error !");
           ptx2 =0L;
        }

        close(handle);
        return(ptx2);
}

void dedect()

{

        int t=0,p,r=0,u=0;     j=m-1;  l=0;

        outtextxy((-3+(i-50)/1.6),280,syn[j]);

deg1:   while(syn[j][u]!=grm[t][1] && grm[t][0]=='Q' && grm[t][0]!='')  t++;
        if(grm[t][0]!='Q') return;
        u++; p=0; l=0;
        if(u==strlen(syn[j]) && grm[t][2]=='')  goto QRS;
        if(u>strlen(syn[j]) || grm[t][2]=='')  return;

deg2:           while(grm[t][2]!=grm[l][0] && grm[l][0]!='') l++;
                if(grm[l][0]=='' && p==0)  return;
                if(grm[l][0]=='' && p==1)  { l=0; u--; t++; goto deg1; }
                p=1;
                if(grm[l][1]!=syn[j][u]) { l++;  goto deg2; }
                u++; r=0; p=0;
                if(u==strlen(syn[j]) && grm[l][2]=='')  goto QRS;
                if(u>strlen(syn[j]) || grm[l][2]=='') return;

deg3:           while(grm[l][2]!=grm[r][0] && grm[r][0]!='') r++;
                if(grm[r][0]=='' && p==0) return;
                if(grm[r][0]=='' && p==1) { r=0; u--; l++; goto deg2; }
                p=1;
                if(grm[r][1]!=syn[j][u]) { r++;  goto deg3; }
                u++;
                if(u==strlen(syn[j]) && grm[r][2]=='')  goto QRS;
                if(u<=strlen(syn[j]) && grm[r][2]!='') { l=r; r=0; goto deg3
                return;

QRS:            outtextxy((-18+(i-50)/1.6),300,"---QRS---");

}
```

EK C

SYNTAXG.C

```
#include<stdio.h>
#include<ctype.h>
#include<graphics.h>
#include<values.h>
#include<dos.h>
#include<conio.h>
#include<fcntl.h>
#include<io.h>
#include<stat.h>
#include<float.h>
#include<string.h>
#include<stdlib.h>
#include<dir.h>
#include<math.h>

FILE *isr;
int pop,E1=7,E2=14,E3=34,T1=4,T2=18,T11=4,T22=22,say,bas,esit=0;
int graphdriver,graphmode,color,bkcolor;
void discurv3(),discurv4(),sor(),syntax(int),gramer(),grm_yaz(int),dedect(int)
int bul();
char *b1,*b2,*t1;
char *rd1(),*rd2(),*rd3();
int *mp1,*mp2;
char min,max,sec,scz,ss,se,at=0;
int sm1[15],sm2[15],s[15];
int *is,*tkn;
double far *tk;
char e2[30],e5[30],ff[60],grm[50][30],grm1[50][30],syn[2][50];
char  *gg1,*gg2;
char vnt[]="QABCDEFGHIJKLMNOPRSTUVWYZ";
char *image1,*vt='*';

int p1,p2,p3,p4,dd,maxx,maxy,u,t,gpe,i,j,k,l,m=0,gp,r,umax,umax1,umax2,tgec,ze
unsigned N,say1=0;

float sd[1500];
/*float fd[1500],gd[1500],jd[1500];*/

float x; double smin,smax;
int alc_gec(),write1();
int tur_kar();
int cizim();
int sira[25];
unsigned im_size;

main()
{

  bas:  clrscr();

        l=0;

        printf("\n\n\n        BYOMEDKAL MHENDSLÎNDE ZEL KONULAR\n");
        printf("                  DERS DEV\n");
        printf("\n\n   KONU         : SZDZM (syntactic) METOD LE QRS DEDEKSYONU
        printf("\n   DEV VEREN : Do. Dr. Mehmet KORREK\n");
        printf("\n   HAZIRLAYAN  : Elektr. Mh. Sergin ZEN (490 Y.068)\n");
        printf("\n\n\n         kmak iin Q'ya devam iin herhangi bir tua basnz")

        sec=getch();
        if(sec == 'q')
               exit(0);
```

```
        sec='';
        clrscr();
        printf("\n\n          KISIM 1 :  ÎRENME (training) MODU\n");
        printf("\n\n\n     1 : SARET DOSYASINI ZDREREK GRAMERN IKARTMA\n");
        printf("\n     2 : GRAMER DOSYASINDAN ALDIÎI HAZIR GRAMER KULLANMA\n")

        while((sec = getch()) != '1' && sec != '2');
        if(sec == '1')
        {
                for(i=0;i<50;i++)
                { strcpy(grm1[i],"\0"); strcpy(grm[i],"\0");}
                m=0;i=0;j=0;k=0;l=0;gp=0;pop=0;
                tur_kar(1);
                gramer();
                grm_yaz(1);
        }

        else
        {
                clrscr();
grm:            printf("\n\n   Gramer DOSYASI : ");
                b1=rd3();
                if(b1==0L) goto grm;
                strcpy(&grm[18][2],"*");
                grm_yaz(0);
         }
                m=0;
                tur_kar(0);
                goto bas;

}

tur_kar(p)
{
      float a,b,c,d,a1,a2; int j;
      double e;
yen:  e=0.; a1=0.; a2=0.;
                clrscr();
        if(p==1) printf("\n\n          KISIM 1 :  ÎRENME (training) MODU\n");
        else     printf("\n\n          KISIM 2 :  DEDEKSYON  MODU\n");


                    don:   b1 = rd1();
                           if(b1 == 0L) goto don;


    smax=0;
/*       mp1=is;
         alc_gec();*/
/*     for(i=0;i<N-2;i++) e=e+(*(tk+i)/(N-2));*/
/*     for(i=0;i<(N);i++) *(is+i)=(*(is+i)*1000.)/smax;*/


    for(i=9;i<(N-10);i++)
    {   a1=0; a2=0;
        for(j=0;j<10;j++)  { a1=a1+(*(is+i+j)); a2=a2+(*(is+i-j)); }
/*      a=(*(is+i+10))-(*(is+i));*/
        a1=a1/10; a2=a2/10;
        a=a1-a2;
/*      *(tk+i)=a*a;*/
        *(tk+i)=fabs(a);/*   *(tk+i)=sqrt(*(tk+i));*/
        if(*(tk+i)>smax)  smax = *(tk+i);

    }
```

```
      *(tk+N)= *(tk+N-2);
      *(tk+N-1)= *(tk+N-2);
      for(i=9;i<N-10;i++) e=e+(*(tk+i)/(N-20));
      for(i=0;i<(N);i++) *(tk+i)=(*(tk+i)*12)/e;
      for(i=0;i<(N);i++) { *(tkn+i)=(*(tk+i)); *(tk+i)=0.; }
/*        mp1=tkn;
          alc_gec();*/



/*      for(i=0;i<N;i++) *(tkn+i)=abs(*(tkn+i));*/
          mp2=&s[0];
          se='2';
          gg1="EKG isareti";
          gg2="Turev karesi";


                  discurv3();
                  syntax(p);
                  free(is); free(tk); free(tkn);
                  free(image1);
                  getch();
q:        closegraph();
          clrscr();

          if(p==0) m=0;

          printf("\n\n       Yeni deneme yapmak istermisiniz ? (E/H) :\n");
          while((ss=getch())!='e' && ss!='h');
          if(ss=='e')  goto yen;

      return;

}


char *rd1()
{
    int *ptx1,*ptx2;
    char *uzan=".dat";
    ptx1 = &sm1[0];
    ptx2 = ptx1;
    bul();
         scanf(" %8s",&e2[0]);
         clrscr();
         strcat(&e2[0],uzan);
         strupr(&e2[0]);
        if ((isr=fopen(e2,"rb"))==NULL )
        {
         puts("Dosya amada hata var.");getch();
         ptx2 = 0L;
         return(ptx2);
        }

        fseek(isr,0,SEEK_END);
        N=ftell(isr);
        fseek(isr,0,SEEK_SET);
        N=N/2;
/*      if(at==1) goto ge;*/
        if((is=calloc(N+1,sizeof(int)))==NULL || (tk=calloc(N+1,sizeof(double
          {
            printf("Bellek Hatas...");  getch();
            exit(0);
          }
```

```
ge:     fread(is,sizeof(int),N,isr);
         at=1;
         fclose(isr);

         return(ptx2);
}

char *rd2()
{

    char *ptx5,*ptx6;
    int open(),read(),close(),handle;

    ptx5 = &sm2[0];
    ptx6 = ptx5;
         scanf("%10s",&e5[0]);
         handle = open(&e5[0],O_RDONLY|O_BINARY,S_IREAD);

         if(handle == -1)
         {
                printf("cannot open file");
                ptx6 = 0L;
         }

         if(read(handle,ptx5,1020) == -1)
         {
            printf("\n\nread error !");
            ptx6 =0L;
         }

         close(handle);
         return(ptx6);
}

void discurv3()
{
       int graphdriver = DETECT, graphmode;

       dd=650/50;

      initgraph(&graphdriver,&graphmode,"c:\\tc\\bgi");
      cleardevice();
       maxx=getmaxx(); maxy=getmaxy();
      setbkcolor(15);
      setcolor(8);
      line (197,10,197,170);
      line (194,90,439,90);

      outtextxy(100,80,gg1);
      outtextxy(300,0,&e2[0]);

        for(i=1;i<=20;i++)
        {
          line(197,90-i*4,193,90-4*i);
          if( (i/5)*5 == i )  line(197,90-i*4,190,90-i*4);
        }

        for(i=1;i<=20;i++)
        {
          line(197,90+i*4,193,90+i*4);
          if( (i/5)*5 == i )  line(197,90+i*4,190,90+i*4);
        }
```

```
/*      for(i=1;i<51;i++)
      {
         line(dd*i+197,90,dd*i+197,94);
         if((i/5)*5 == i)  line(dd*i+197,90,dd*i+197,97);
      }
*/
      line(197,180,197,260);
      line(194,260,439,260);

      outtextxy(100,270,gg2);
      outtextxy(300,170,&e5[0]);
        outtextxy(180,260-E2,"b");
        outtextxy(180,260-E1,"a");
        outtextxy(180,260-E3,"c");
     for(i=1;i<=20;i++)
      {
         line(197,260-i*4,193,260-4*i);
         if( (i/5)*5 == i )  line(197,260-i*4,190,260-4*i);
      }


/*      for(i=1;i<51;i++)
      {
         line(dd*i+197,260,dd*i+197,264);
         if((i/5)*5 == i) line(dd*i+197,260,dd*i+197,267);
      }
*/
        line(197,(260-E1),439,(260-E1));
        line(197,(260-E2),439,(260-E2));
        line(197,(260-E3),439,(260-E3));
        color=getcolor(); bkcolor=getbkcolor();
        im_size=imagesize(200,10,439,310);
        if((image1=malloc(imagesize(200,10,439,310)))==NULL)
        { closegraph();
          printf("Bellek Hatas...."); getch();
          exit(0);}
        else
        i=0; pop=0;
        settextjustify(RIGHT_TEXT,CENTER_TEXT);
}

void discurv4()

{      int t;

           t=238; if(pop<3) { pop++; goto gec; }
           pop=0;
           getimage(200,10,439,310,image1);
           putimage(198,10,image1,COPY_PUT);
           setcolor(bkcolor);
           line(439,10,439,310); line(438,10,438,310);/* line(437,10,437,310);
           setcolor(color);
           putpixel(439,(260-E1),color);
           putpixel(439,(260-E2),color);
           putpixel(439,(260-E3),color);
    for(say=0;say<4;say+=2)
    {
           putpixel(439-say/2,90,color);
           putpixel(439-say/2,260,color);

          t=t-say/2;
          p1= 200+t;
          p2= 90-*(is+i-say)/3;
          p3= 200+(t+1);
```

```
            p4= 90-*(is+i+2-say)/3;

            line(p1,p2,p3,p4);


            p1= 200+t;
            p2= 260-*(tkn+i-say);
            p3= 200+(t+1);
            p4= 260-*(tkn+i+2-say);

            line(p1,p2,p3,p4);
      }
gec:     sec='';
         if( kbhit() ) { while((sec=getch())=='s');}

}


write1(char *np)
{
         int open(),write(),close(),handle;
         int a;
         char e1[20];

         printf("\n\n\n File ismi: ");
         scanf("%10s",&e1[0]);

         handle = open(&e1[0],O_CREAT|O_TRUNC|O_RDWR|O_BINARY,S_IWRITE);

         if(handle == -1){

                  printf("cannot open file");
                  exit (0);
         }

          a = write(handle,np,1020);

           if(a == -1){

             printf("write error !");
           }

         close(handle);

         return;
}

alc_gec()
{
       double a,b,c,d,e;
       int f,fm;
       double g=0.;
                  clrscr();
                  printf("\n\nfiltre mertebesi(1,2,3 veya 4) =");
                  scanf("%d",&fm);

         printf("\n\n\nkose frekansi=");
         scanf("%d",&f);

         if(fm==fm/2*2) k=1000./f-0.5;
         else  k=(1000./f-1)/2+0.5;
         printf("\n\nk=%d",k);
```

```
smin=*mp1;
smax=smin;

for(i=4;i<N-4;i++)
{
  if(fm==4)
  {
    if(i<4*k+4) a=0.;
    else {a=*(mp1+i-4*k-4);}

    if(i<3*k+3) b=0.;
    else {b=*(mp1+i-3*k-3);}

    if(i<2*k+2) c=0.;
    else {c=*(mp1+i-2*k-2);}

    if(i<k+1) d=0.;
    else {d=*(mp1+i-k-1);}

    e=*(mp1+i);

    *(tk+i)=-*(tk+i-4)+4.*(*(tk+i-3))-6.*(*(tk+i-2))+4.*(*(tk+i-1));
    *(tk+i)=*(tk+i)+a-4.*b+6.*c-4.*d+e;
  }


  else if(fm==3)
  {
    if(i<6*k+3) a=0.;
    else a=*(mp1+i-6*k-3);

    if(i<4*k+2) b=0.;
    else b=*(mp1+i-4*k-2);

    if(i<2*k+1) c=0.;
    else c=*(mp1+i-2*k-1);

    d=*(mp1+i);

    *(tk+i)=*(tk+i-3)-3.*(*(tk+i-2))+3.*(*(tk+i-1))-a+3.*b-3.*c+d;

  }

  else if(fm==2)
  {
    if(i<2*k+2) a=0.;
    else a=*(mp1+i-2*k-2);

    if(i<k+1) b=0.;
    else b=*(mp1+i-k-1);

    c=*(mp1+i);

    *(tk+i)=-*(tk+i-2)+2.*(*(tk+i-1))+a-2.*b+c;
  }

  else if(fm==1)
  {
    if(i<2*k+1) a=0.;
    else a=*(mp1+i-2*k-1);

    b=*(mp1+i);

    *(tk+i)=*(tk+i-1)-a+b;
```

```
      }


        if(*(tk+i)>smax)   smax=*(tk+i);
        else if(*(tk+i)<smin)   smin=*(tk+i);

    }


     if(smin<0)   smin=-smin;
     if(smax<0)   smax=-smax;

     if(smax<smin) smax=smin;
/*     for(i=0;i<N-2;i++) g=g+(*(tk+i)/(N-2));
     for(i=0;i<(N);i++) *(mp1+i)=(*(tk+i)*200)/smax; */
     return;

}


void syntax(p)

{
        char tb=0,ta=0,tc=0,d=0; int k1=0;
        i=0;   gp=0;

tur:      k=0;
          while(*(tkn+i)<E1 && i<(N+1))

          {
                i++; k1++; discurv4();
                if(sec=='q') goto son;
                if(k1==(T11+(T22-T11)*k))

                {
                        d=0;
                        syn[0][gp]='t';
                        gp++; ta=0; tb=0; tc=0; k++; d=i;
                }
          }

          k=0;

          while(*(tkn+i)>=E1 && *(tkn+i)<E2 && i<(N+1))

          {
                   ta++;   i++;  discurv4();

                if(ta==(T1+(T2-T1)*k))

                {
                        syn[0][gp]='a';
                        k++; gp++; k1=0; tb=0; tc=0;
                }
          }

/*        if(k>0) { k--; k1=0; tb=0; tc=0;}
          if((ta-(T1+(T2-T1)*k))>=T1)

          {
                syn[m][gp]='a';
                gp++; ta=0;
          }
*/
```

```
            k=0;

            while(*(tkn+i)>=E2 && *(tkn+i)<=E3 && i<(N+1))

            {
                   tb++;   i++;  discurv4();
                  if(tb==(T1+(T2-T1)*k))

                  {
                      syn[0][gp]='b';
                      k++;  gp++; k1=0; ta=0; tc=0;
                  }
            }

/*          if(k>0) { k--; k1=0; ta=0; tc=0; }
            if((tb-(T1+(T2-T1)*k))>=T1)

            {
                  syn[m][gp]='b';
                    gp++;  tb=0;
            }
*/
            k=0;
            while(*(tkn+i)>=E3 && i<(N+1))

            {
                tc++;   i++;  discurv4();
                if(tc==(T1+(T2-T1)*k))

                { syn[0][gp]='\0';
                  if(d!=0)
                  { gpe=gp; while(syn[0][gpe]!='t') gpe--;
                    strcpy(syn[1],syn[0]); syn[1][gpe+1]='\0';
                    dedect(p); strcpy(syn[0],&syn[0][gpe+1]); gp=gp-gpe-1;
                  }
                  syn[0][gp]='c';
                  k++;  d=0;  gp++; k1=0; ta=0; tb=0;
                }
            }

/*          if(k>0) { k--; k1=0; ta=0; tb=0; }
            if((tc-(T1+(T2-T1)*k))>=T1)

            {
                  syn[m][gp]='c';
                  gp++; tc=0;
            }
*/
            k=0;

            if(i<(N+1))  goto tur;

son:        syn[0][0]='\0';
            sec='';
}

void sor()

{


/*        outtextxy(439,(270+10*(say1-2*(say1/2))),syn[1]);*/
          outtextxy(336,330,"Bu normal bir EKG midir ? (E/H)");
          while((sec=getch()) != 'e' && sec != 'h');
```

```
        if(sec=='e')
        {
                outtextxy(410,300,vt);
        }

}


void gramer()

{
/*      clrscr(); gp=0;
        for(k=1;k<30;k++)
        {
                for(i=0;i<m;i++)
                {
                        if(strlen(syn[i])<=k)  goto cik;
                        for(j=1;j<=i;j++)
                        {
                                if(i<j) { gp++;  strcpy(grm1[gp],&syn[0][k]);
                                if(strcmp(&syn[i][k],&syn[i-j][k])==0)  goto c

                        }
                        l=0;
                        while(strcmp(grm1[gp-l],"")!=0 )
                        {if(strcmp(grm1[gp-l],&syn[i][k])==0) goto cik;
                         if(strcmp(grm1[gp-l-1],grm1[gp])==0) goto cik;
                         l++; }

                        gp++;
                        strcpy(grm1[gp],&syn[i][k]);

cik:                    l++; l--;

                }

        }
        i=0; j=1;
        while(strcmp(syn[i],"")!=0)
        {
                j=1;
                grm[i][0]=vnt[0];
                grm[i][1]=syn[i][0];
                if(strcmp(&syn[i][1],"")==0) { grm[i][2]='\0'; goto atla; }
                while(strcmp(grm1[j],&syn[i][1])!=0 && j<30) j++;
/*          if(j==0) { printf("\n\n  Gramer cikarilamiyor.\n"); getch(); retur
                grm[i][2]=vnt[j];  grm[i][3]='\0';
atla:           i++;
        }

        k=1;
        while(k<=gp)
        {
                grm[i+k-1][0]=vnt[k];
                grm[i+k-1][1]=grm1[k][0];
                grm[i+k-1][2]='\0';
                if(strcmp(&grm1[k][1],"")==0) goto on;
                for(l=1;l<=gp;l++)
                {
                        if(l==k) l++;
                        if(strcmp(&grm1[k][1],&grm1[l][0])==0) goto in;
                }
                printf("\n\n  Gramer cikartilamiyor. \n");
```

```
                return;
in:             grm[i+k-1][2]=vnt[l];  grm[i+k-1][3]='\0';
on:             k++;

        }*/
}

void grm_yaz(a)

{

        clrscr();
        printf("\n---------------------------- G R A M E R -------------------
        l=0;
        while(grm[l][0]!='' && l<30)
        {
                if(l==(l/2)*2) printf("\n\n");
                printf("               %c : %s ",grm[l][0],&grm[l][1]);
                l++;
        }
        if(a==0)
        {
                while(getch()=='');
                return;
        }
        printf("\n\n\n         Gramer kaydedilsin mi ? (E/H) :");
        if((ss=getch())=='h') return;

        t1=grm[0];
        write1(t1);
        return;
}


char *rd3()
{
    char *ptx1,*ptx2;
    int open(),read(),close(),handle;

    ptx1 = grm[0];
    ptx2 = ptx1;
         scanf("%10s",&e2[0]);
         handle = open(&e2[0],O_RDONLY|O_BINARY,S_IREAD);

         if(handle == -1)
         {
                printf("cannot open file");
                ptx2 = 0L;
         }

         if(read(handle,ptx1,1020) == -1)
         {
            printf("\n\nread error !");
            ptx2 =0L;
         }

         close(handle);
         return(ptx2);
}

void dedect(deg)

{
```

```
        int p; r=0; u=0; l=0; say1++; ilk=0; t=0; deg=1; umax1=0;

        outtextxy(439,270+10*(say1-2*(say1/2)),syn[1]);

deg1:   while(syn[1][u]!=grm[t][1] && grm[t][0]!='')  t++;
        if(grm[t][0]!='Q' && grm[t][0]!='')
        {/* if(u>=umax1) { umax1=u; tgec=t; }*/
          t++; goto deg1;
        }
        if(grm[t][0]=='') { if(deg==0) return;
                            else { gramer_ek(); if(sec=='h') return; }
                          }
        if(u+1<strlen(syn[1]) &&  grm[t][2]=='*') { t++; u=0; goto deg1; }
        if(u>umax1) { umax1=u; tgec=t; }
        u++; p=0; l=0;
        if(u==strlen(syn[1]) && grm[t][2]=='*') { bas=t; goto QRS;}
        if(u==strlen(syn[1])) { if(deg==0) return;
                                else { esit=1;gramer_ek();if(sec=='h')return;
                                esit=0; }
                                goto QRS;
                              }


deg2:   while(grm[t][2]!=grm[l][0] && grm[l][0]!='') l++;
        if(grm[l][0]=='' && p==0) { if(deg==0) return;
                                    else { gramer_ek();if(sec=='h') return;}
                                  }
        if(grm[l][0]=='' && p==1)  {/* if(u>umax1) { umax1=u; tgec=l; }*/
                                     l=0; u=0; t++; goto deg1;
                                   }
        p=1;
        if(grm[l][1]!=syn[1][u] ) {l++;goto deg2; }
        if(u+1<strlen(syn[1]) && grm[l][2]=='*') { l++; goto deg2; }
        if(u>umax1) { umax1=u; tgec=l; }
        u++; r=0; p=0;
        if(u==strlen(syn[1]) && grm[l][2]=='*') { bas=l; goto QRS; }
        if(u==strlen(syn[1])) { if(deg==0) return;
                                else {esit=1;gramer_ek();if(sec=='h')return;
                                      esit=0; }
                                goto QRS;
                              }


deg3:   while(grm[l][2]!=grm[r][0] && grm[r][0]!='') r++;
        if(grm[r][0]=='' && p==0) { if(deg==0) return;
                                    else { gramer_ek();if(sec=='h') return;}
                                  }
        if(grm[r][0]=='' && p==1) {/* if(u>umax1) { umax1=u; tgec=r; }*/
                                    r=0; u--; l++; goto deg2;
                                  }
        p=1;
        if(grm[r][1]!=syn[1][u]) { r++; goto deg3; }

        if(u+1<strlen(syn[1]) && grm[r][2]=='*') { r++; goto deg3; }

        if(u>umax1) { umax1=u; tgec=r; }
        u++; p=0;
        if(u==strlen(syn[1]) && grm[r][2]=='*') { bas=r; goto QRS; }
        if(u==strlen(syn[1])) { if(deg==0) return;
                                else { esit=1;gramer_ek();if(sec=='h')return;
                                       esit=0; }
                                goto QRS;
                              }
```

```
        l=r; r=0; goto deg3;

QRS:    ret=0;  outtextxy(439,300,&grm[bas][2]);

}

int bul()

{
 struct ffblk f;
 int err;
 clrscr();
 textcolor(BLACK);
 textbackground(LIGHTGRAY);
 cprintf("\r\n Aktif directory'deki dosyalar :\r\n");
 findfirst("*.dat",&f,FA_ARCH);
print:  printf("   %s",f.ff_name);
        if((err=findnext(&f))!=-1) goto print;
 cprintf("\r\n\r\n\r\n Okumak istediÏiniz dosyann adn yaznz : (    .dat)");
 printf("\n\n ");
 textbackground(BLACK);
}


void gramer_ek()
{
    static int set,ret1,tgec1; char car; int gr;
    if(ilk==0)
    { ilk=1; sor(); if(sec=='h') return; }
    if(ret==0)
    {
       umax1++; umax=umax1; umax2=0; zet=0;
       while(grm[zet][0]!='\0') zet++;
       grm[zet][0]=grm[tgec][2];
       grm[zet][1]=syn[1][umax1];
       grm[zet+1][0]='\0';
       set=24; gr=zet;
don:   while(vnt[set]!=grm[gr][0] && gr>0) gr--;
       if(gr==0 && set!=0) {set--; gr=zet; goto don;}

    }

    if(esit==1) { grm[zet+1][0]=grm[r][0]; grm[zet+1][1]=syn[1][umax1];
              grm[zet+1][2]='*'; grm[zet+1][3]='\0'; gr=0;
              while( strcmp(grm[zet+1],grm[gr])!=0 ) gr++;
              if( gr<zet+1 ) { bas=gr; grm[zet+1][0]='\0'; }
              else bas=zet+1;
              ret=0; return;
            }

    if(umax1>umax2) { umax2=umax1; ret1=ret-1; tgec1=tgec; }
    if(ret<=set && u!=strlen(syn[1]))
    {
       if(vnt[ret]==grm[zet][0]) ret++;
       grm[zet][2]=vnt[ret]; u=umax;
       grm[zet][3]='\0';
        t=zet; l=zet; r=zet;
       ret++; return;
    }

    else
    {
       ret=0;
       if(umax2>umax)
```

```
            { grm[zet][2]=vnt[ret1]; grm[zet][3]='\0';
              t=tgec1; l=tgec1; r=tgec1; u=umax2; umax1=umax2;
            }

        else
            { grm[zet][2]=vnt[set+1];
              t=zet; l=zet; r=zet; u=umax1; tgec=zet;
            }
    }

}
```

EK D

```
#include<stdio.h>
#include<ctype.h>
#include<graphics.h>
#include<values.h>
#include<dos.h>
#include<conio.h>
#include<fcntl.h>
#include<io.h>
#include<stat.h>
#include<float.h>
#include<string.h>
#include<stdlib.h>
#include<dir.h>
#include<math.h>

FILE *isr;
int pop,E1=7,E2=14,E3=34,T1=4,T2=18,say;
int graphdriver,graphmode,color,bkcolor;
void discurv3(),discurv4(),wrt_graf(),syntax(int),gramer(),grm_yaz(int),dedect
int bul();
char *b1,*b2,*t1;
char *rd1(),*rd2(),*rd3();
int *mp1,*mp2;
char min,max,sec,scz,ss,se,at=0;
int sm1[15],sm2[15],s[15];
int *is,*tkn;
double far *tk;
char e2[30],e5[30],ff[60],grm[30][30],grm1[30][30],syn[30][30];
char  *gg1,*gg2;
char vnt[]="QABCDEFGHIJKLMNOPRSTUVWYZF";
char *image1;

int p1,p2,p3,p4,dd,maxx,maxy,t;
unsigned N,say1=0;

float sd[1500];
/*float fd[1500],gd[1500],jd[1500];*/

float x; double smin,smax;
int alc_gec(),write1();
int tur_kar();
int cizim();
int i,j,k,l,m=0,gp;
int sira[25];
unsigned im_size;

main()
{

  bas:  clrscr();

        l=0;

        printf("\n\n\n        BYOMEDKAL MHENDSLÎNDE ZEL KONULAR\n");
        printf("                DERS DEV\n");
        printf("\n\n    KONU          : SZDZM (syntactic) METOD LE QRS DEDEKSYONU
        printf("\n    DEV VEREN : Do. Dr. Mehmet KORREK\n");
        printf("\n    HAZIRLAYAN  : Elektr. Mh. Sergin ZEN (490 Y.068)\n");
        printf("\n\n\n         kmak iin Q'ya devam iin herhangi bir tua basnz")

        sec=getch();
        if(sec == 'q')
               exit(0);
```

```
        sec='';
        clrscr();
        printf("\n\n          KISIM 1 :  ÎRENME (training) MODU\n");
        printf("\n\n\n     1 : SARET DOSYASINI ZDREREK GRAMERN IKARTMA\n");
        printf("\n     2 : GRAMER DOSYASINDAN ALDIÎI HAZIR GRAMER KULLANMA\n")

        while((sec = getch()) != '1' && sec != '2');
        if(sec == '1')
        {
                for(i=0;i<30;i++)
                {strcpy(syn[i],"");strcpy(grm1[i],"");strcpy(grm[i],"");}
                m=0;i=0;j=0;k=0;l=0;gp=0;pop=0;
                tur_kar(1);
                gramer();
                grm_yaz(1);
        }

        else
        {
                clrscr();
grm:            printf("\n\n   Gramer DOSYASI : ");
                b1=rd3();
                if(b1==0L) goto grm;
                grm_yaz(0);
         }
                m=0;
                tur_kar(0);
                goto bas;

}

tur_kar(p)
{
      float a,b,c,d,a1,a2; int j;
      double e;
yen:  e=0.; a1=0.; a2=0.;
                clrscr();
        if(p==1) printf("\n\n          KISIM 1 :  ÎRENME (training) MODU\n");
        else     printf("\n\n          KISIM 2 :  DEDEKSYON  MODU\n");


                    don:   b1 = rd1();
                           if(b1 == 0L) goto don;


    smax=0;
/*       mp1=is;
         alc_gec();*/
/*     for(i=0;i<N-2;i++) e=e+(*(tk+i)/(N-2));*/
/*     for(i=0;i<(N);i++) *(is+i)=(*(is+i)*1000.)/smax;*/


    for(i=9;i<(N-10);i++)
    {   a1=0; a2=0;
        for(j=0;j<10;j++)  { a1=a1+(*(is+i+j)); a2=a2+(*(is+i-j)); }
/*      a=(*(is+i+10))-(*(is+i));*/
        a1=a1/10; a2=a2/10;
        a=a1-a2;
/*      *(tk+i)=a*a;*/
        *(tk+i)=fabs(a);/*    *(tk+i)=sqrt(*(tk+i));*/
        if(*(tk+i)>smax)  smax = *(tk+i);

    }
```

```
     *(tk+N)= *(tk+N-2);
     *(tk+N-1)= *(tk+N-2);
     for(i=9;i<N-10;i++) e=e+(*(tk+i)/(N-20));
     for(i=0;i<(N);i++) *(tk+i)=(*(tk+i)*12)/e;
     for(i=0;i<(N);i++) { *(tkn+i)=(*(tk+i)); *(tk+i)=0.; }
/*        mp1=tkn;
          alc_gec();*/



/*      for(i=0;i<N;i++) *(tkn+i)=abs(*(tkn+i));*/
          mp2=&s[0];
          se='2';
          gg1="EKG isareti";
          gg2="Turev karesi";


                  discurv3();
                  syntax(p);
                  free(is); free(tk); free(tkn);
                  free(image1);
                  getch();
q:        closegraph();
          clrscr();

          if(p==0) m=0;

          printf("\n\n       Yeni deneme yapmak istermisiniz ? (E/H) :\n");
          while((ss=getch())!='e' && ss!='h');
          if(ss=='e')  goto yen;

     return;

}


char *rd1()
{
    int *ptx1,*ptx2;
    char *uzan=".dat";
    ptx1 = &sm1[0];
    ptx2 = ptx1;
    bul();
         scanf(" %8s",&e2[0]);
         clrscr();
         strcat(&e2[0],uzan);
         strupr(&e2[0]);
        if ((isr=fopen(e2,"rb"))==NULL )
        {
         puts("Dosya amada hata var.");getch();
         ptx2 = 0L;
         return(ptx2);
        }

        fseek(isr,0,SEEK_END);
        N=ftell(isr);
        fseek(isr,0,SEEK_SET);
        N=N/2;
/*      if(at==1) goto ge;*/
        if((is=calloc(N+1,sizeof(int)))==NULL || (tk=calloc(N+1,sizeof(double
          {
            printf("Bellek Hatas...."); getch();
            exit(0);
          }
```

```
ge:     fread(is,sizeof(int),N,isr);
         at=1;
         fclose(isr);

         return(ptx2);
}

char *rd2()
{

    char *ptx5,*ptx6;
    int open(),read(),close(),handle;

    ptx5 = &sm2[0];
    ptx6 = ptx5;
         scanf("%10s",&e5[0]);
         handle = open(&e5[0],O_RDONLY|O_BINARY,S_IREAD);

         if(handle == -1)
         {
                printf("cannot open file");
                ptx6 = 0L;
         }

         if(read(handle,ptx5,1020) == -1)
         {
            printf("\n\nread error !");
            ptx6 =0L;
         }

         close(handle);
         return(ptx6);
}

void discurv3()
{
        int graphdriver = DETECT, graphmode;

        dd=650/50;

       initgraph(&graphdriver,&graphmode,"c:\\tc\\bgi");
       cleardevice();
        maxx=getmaxx(); maxy=getmaxy();

       setcolor(8);  setbkcolor(15);
       line (197,10,197,170);
       line (194,90,439,90);

       outtextxy(100,80,gg1);
       outtextxy(300,0,&e2[0]);

         for(i=1;i<=20;i++)
         {
           line(197,90-i*4,193,90-4*i);
           if( (i/5)*5 == i )  line(197,90-i*4,190,90-i*4);
         }

         for(i=1;i<=20;i++)
         {
           line(197,90+i*4,193,90+i*4);
           if( (i/5)*5 == i )  line(197,90+i*4,190,90+i*4);
         }
```

```
/*      for(i=1;i<51;i++)
      {
         line(dd*i+197,90,dd*i+197,94);
         if((i/5)*5 == i)  line(dd*i+197,90,dd*i+197,97);
      }
*/
      line(197,180,197,260);
      line(194,260,439,260);

      outtextxy(100,270,gg2);
      outtextxy(300,170,&e5[0]);
        outtextxy(180,260-E2,"b");
        outtextxy(180,260-E1,"a");
        outtextxy(180,260-E3,"c");
     for(i=1;i<=20;i++)
      {
         line(197,260-i*4,193,260-4*i);
         if( (i/5)*5 == i )  line(197,260-i*4,190,260-4*i);
      }


/*      for(i=1;i<51;i++)
      {
         line(dd*i+197,260,dd*i+197,264);
         if((i/5)*5 == i) line(dd*i+197,260,dd*i+197,267);
      }
*/
        line(197,(260-E1),439,(260-E1));
        line(197,(260-E2),439,(260-E2));
        line(197,(260-E3),439,(260-E3));
        color=getcolor(); bkcolor=getbkcolor();
        im_size=imagesize(200,10,439,310);
        if((image1=malloc(imagesize(200,10,439,310)))==NULL)
        { closegraph();
          printf("Bellek Hatas..."); getch();
          exit(0);}
        else
        i=0; pop=0;
        settextjustify(RIGHT_TEXT,CENTER_TEXT);
}

void discurv4()

{ int t;

          t=238; if(pop<3) { pop++; goto gec; }
          pop=0;
          getimage(200,10,439,310,image1);
          putimage(198,10,image1,COPY_PUT);
          setcolor(bkcolor);
          line(439,10,439,310); line(438,10,438,310);/* line(437,10,437,310);
          setcolor(color);
          putpixel(439,(260-E1),color);
          putpixel(439,(260-E2),color);
          putpixel(439,(260-E3),color);
    for(say=0;say<4;say+=2)
    {
          putpixel(439-say/2,90,color);
          putpixel(439-say/2,260,color);

         t=t-say/2;
         p1= 200+t;
         p2= 90-*(is+i-say)/3;
         p3= 200+(t+1);
```

```
            p4= 90-*(is+i+2-say)/3;

            line(p1,p2,p3,p4);


            p1= 200+t;
            p2= 260-*(tkn+i-say);
            p3= 200+(t+1);
            p4= 260-*(tkn+i+2-say);

            line(p1,p2,p3,p4);
      }
gec:     sec='';
         if( kbhit() ) { while((sec=getch())=='s');}

}


write1(char *np)
{
         int open(),write(),close(),handle;
         int a;
         char e1[20];

         printf("\n\n\n File ismi: ");
         scanf("%10s",&e1[0]);

         handle = open(&e1[0],O_CREAT|O_TRUNC|O_RDWR|O_BINARY,S_IWRITE);

         if(handle == -1){

                  printf("cannot open file");
                  exit (0);
         }

          a = write(handle,np,1020);

           if(a == -1){

             printf("write error !");
           }

         close(handle);

         return;
}

alc_gec()
{
       double a,b,c,d,e;
       int f,fm;
       double g=0.;
                  clrscr();
                  printf("\n\nfiltre mertebesi(1,2,3 veya 4) =");
                  scanf("%d",&fm);

         printf("\n\n\nkose frekansi=");
         scanf("%d",&f);

         if(fm==fm/2*2) k=1000./f-0.5;
         else  k=(1000./f-1)/2+0.5;
         printf("\n\nk=%d",k);
```

```
smin=*mp1;
smax=smin;

for(i=4;i<N-4;i++)
{
  if(fm==4)
  {
    if(i<4*k+4) a=0.;
    else {a=*(mp1+i-4*k-4);}

    if(i<3*k+3) b=0.;
    else {b=*(mp1+i-3*k-3);}

    if(i<2*k+2) c=0.;
    else {c=*(mp1+i-2*k-2);}

    if(i<k+1) d=0.;
    else {d=*(mp1+i-k-1);}

    e=*(mp1+i);

    *(tk+i)=-*(tk+i-4)+4.*(*(tk+i-3))-6.*(*(tk+i-2))+4.*(*(tk+i-1));
    *(tk+i)=*(tk+i)+a-4.*b+6.*c-4.*d+e;
  }


  else if(fm==3)
  {
    if(i<6*k+3) a=0.;
    else a=*(mp1+i-6*k-3);

    if(i<4*k+2) b=0.;
    else b=*(mp1+i-4*k-2);

    if(i<2*k+1) c=0.;
    else c=*(mp1+i-2*k-1);

    d=*(mp1+i);

    *(tk+i)=*(tk+i-3)-3.*(*(tk+i-2))+3.*(*(tk+i-1))-a+3.*b-3.*c+d;

  }

  else if(fm==2)
  {
    if(i<2*k+2) a=0.;
    else a=*(mp1+i-2*k-2);

    if(i<k+1) b=0.;
    else b=*(mp1+i-k-1);

    c=*(mp1+i);

    *(tk+i)=-*(tk+i-2)+2.*(*(tk+i-1))+a-2.*b+c;
  }

  else if(fm==1)
  {
    if(i<2*k+1) a=0.;
    else a=*(mp1+i-2*k-1);

    b=*(mp1+i);

    *(tk+i)=*(tk+i-1)-a+b;
```

```
      }


         if(*(tk+i)>smax)   smax=*(tk+i);
         else if(*(tk+i)<smin)   smin=*(tk+i);

     }


      if(smin<0)   smin=-smin;
      if(smax<0)   smax=-smax;

      if(smax<smin) smax=smin;
/*      for(i=0;i<N-2;i++) g=g+(*(tk+i)/(N-2));
      for(i=0;i<(N);i++)  *(mp1+i)=(*(tk+i)*200)/smax; */
      return;

}


void syntax(p)

{
         char tb=0,ta=0,tc=0,d=0; int k1=0;
         i=0;    k=0;    gp=0;

tur:     while(*(tkn+i)<E1 && i<(N+1))

           {
                 i++; k1++; discurv4();
                 if(sec=='q') goto son;
                 if(k1==25 && d==1)

                 {
                         d=0;
                         syn[m][gp]='\0';
                         gp=0; m++; ta=0; tb=0; tc=0;
                         if(p==0) dedect();
                         else wrt_graf();
                 }
           }

           k=0;

           while(*(tkn+i)>=E1 && *(tkn+i)<E2 && i<(N+1))

           {
                    ta++;    i++;   discurv4();

                 if(ta==(T1+(T2-T1)*k))

                 {
                         syn[m][gp]='a';
                         k++; gp++; d=1; k1=0; tb=0; tc=0;
                 }
           }

/*         if(k>0) { k--; k1=0; tb=0; tc=0;}
           if((ta-(T1+(T2-T1)*k))>=T1)

           {
                 syn[m][gp]='a';
                 d=1;  gp++; ta=0;
           }
```

```
*/
            k=0;

            while(*(tkn+i)>=E2 && *(tkn+i)<=E3 && i<(N+1))

            {
                    tb++;   i++;  discurv4();
                  if(tb==(T1+(T2-T1)*k))

                  {
                      syn[m][gp]='b';
                      k++;  d=1;  gp++; k1=0; ta=0; tc=0;
                  }
            }

/*          if(k>0) { k--; k1=0; ta=0; tc=0; }
            if((tb-(T1+(T2-T1)*k))>=T1)

            {
                  syn[m][gp]='b';
                  d=1;  gp++;  tb=0;
            }
*/
            k=0;
            while(*(tkn+i)>=E3 && i<(N+1))

            {
                  tc++;   i++;  discurv4();
                  if(tc==(T1+(T2-T1)*k))

                  {
                      syn[m][gp]='c';
                      k++;  d=1;  gp++; k1=0; ta=0; tb=0;
                  }
            }

/*          if(k>0) { k--; k1=0; ta=0; tb=0; }
            if((tc-(T1+(T2-T1)*k))>=T1)

            {
                  syn[m][gp]='c';
                  d=1;  gp++; tc=0;
            }
*/
            k=0;

            if(i<(N+1))  goto tur;

            if(d==1)
            {
                  syn[m][gp]='\0';
                  gp=0; m++;
                  if(p==0) dedect();
                  else wrt_graf();
            }
son:        syn[m][0]='\0';
            sec='';
}

void wrt_graf()

{

        int t; j=m-1; say1++;
```

```
        outtextxy(439,(270+10*(say1-2*(say1/2))),syn[j]);
        outtextxy(336,330,"Bu gecerli bir QRS midir ? (E/H)");
        while((sec=getch()) != 'e' && sec != 'h');

        if(sec=='e')
        {
                outtextxy(439,300,"QRS");
                for(t=0;t<j;t++)
                {
                        if(strcmp(syn[j],syn[j-t-1])==0)
                        {
                                m--;
                                t=j; return;
                        }
                }
                return;
        }
        else  m--;

}


void gramer()

{
        clrscr(); gp=0;
        for(k=1;k<30;k++)
        {
                for(i=0;i<m;i++)
                {
                        if(strlen(syn[i])<=k)  goto cik;
                        for(j=1;j<=i;j++)
                        {
                                if(i<j) { gp++;  strcpy(grm1[gp],&syn[0][k]);
                                if(strcmp(&syn[i][k],&syn[i-j][k])==0)  goto c

                        }
                        l=0;
                        while(strcmp(grm1[gp-l],"")!=0 )
                        {if(strcmp(grm1[gp-l],&syn[i][k])==0) goto cik;
                         if(strcmp(grm1[gp-l-1],grm1[gp])==0) goto cik;
                         l++; }

                        gp++;
                        strcpy(grm1[gp],&syn[i][k]);

cik:            l++; l--;

                }

        }
        i=0; j=1;
        while(strcmp(syn[i],"")!=0)
        {
                j=1;
                grm[i][0]=vnt[0];
                grm[i][1]=syn[i][0];
                if(strcmp(&syn[i][1],"")==0) { grm[i][2]='\0'; goto atla; }
                while(strcmp(grm1[j],&syn[i][1])!=0 && j<30) j++;
/*          if(j==0) { printf("\n\n  Gramer cikarilamiyor.\n"); getch(); retur
                grm[i][2]=vnt[j];  grm[i][3]='\0';
atla:           i++;
        }
```

```
        k=1;
        while(k<=gp)
        {
                grm[i+k-1][0]=vnt[k];
                grm[i+k-1][1]=grm1[k][0];
                grm[i+k-1][2]='\0';
                if(strcmp(&grm1[k][1],"")==0) goto on;
                for(l=1;l<=gp;l++)
                {
                        if(l==k) l++;
                        if(strcmp(&grm1[k][1],&grm1[l][0])==0) goto in;
                }
                printf("\n\n  Gramer cikartilamiyor. \n");
                return;
in:             grm[i+k-1][2]=vnt[l];  grm[i+k-1][3]='\0';
on:             k++;

        }
}

void grm_yaz(a)

{

        clrscr();
        printf("\n---------------------------- G R A M E R ------------------
        l=0;
        while(grm[l][0]!='' && l<30)
        {
                if(l==(l/2)*2) printf("\n\n");
                printf("                %c : %s ",grm[l][0],&grm[l][1]);
                l++;
        }
        if(a==0)
        {
                while(getch()=='');
                return;
        }
        printf("\n\n\n         Gramer kaydedilsin mi ? (E/H) :");
        if((ss=getch())=='h') return;

        t1=grm[0];
        write1(t1);
        return;
}


char *rd3()
{
    char *ptx1,*ptx2;
    int open(),read(),close(),handle;

    ptx1 = grm[0];
    ptx2 = ptx1;
         scanf("%10s",&e2[0]);
         handle = open(&e2[0],O_RDONLY|O_BINARY,S_IREAD);

         if(handle == -1)
         {
                printf("cannot open file");
                ptx2 = 0L;
         }
```

```
          if(read(handle,ptx1,1020) == -1)
          {
             printf("\n\nread error !");
             ptx2 =0L;
          }

          close(handle);
          return(ptx2);
}

void dedect()

{

        int p,r=0,u=0;     j=m-1;  l=0; say1++; t=0;

        outtextxy(439,270+10*(say1-2*(say1/2)),syn[j]);

deg1:   while(syn[j][u]!=grm[t][1] && grm[t][0]!='')  t++;
        if(grm[t][0]!='Q' && grm[t][0]!='') { t++; goto deg1; }
        if(grm[t][0]=='') return;
        u++; p=0; l=0;
        if(u==strlen(syn[j]) && grm[t][2]=='')  goto QRS;
        if(u<strlen(syn[j]) &&  grm[t][2]=='') { t++; u=0; goto deg1; }
/*      if(u>strlen(syn[j]) || grm[t][2]=='')  return;*/

deg2:           while(grm[t][2]!=grm[l][0] && grm[l][0]!='') l++;
                if(grm[l][0]=='' && p==0)  return;
                if(grm[l][0]=='' && p==1)  { l=0; u=0; t++; goto deg1; }
                p=1;
                if(grm[l][1]!=syn[j][u]) { l++;  goto deg2; }
                u++; r=0; p=0;
                if(u==strlen(syn[j]) && grm[l][2]=='')  goto QRS;
                if(u<strlen(syn[j]) && grm[l][2]=='') {l++; u--; p=1; goto deg
/*              if(u>strlen(syn[j]) || grm[l][2]=='') return;*/

deg3:           while(grm[l][2]!=grm[r][0] && grm[r][0]!='') r++;
                if(grm[r][0]=='' && p==0) return;
                if(grm[r][0]=='' && p==1) { r=0; u--; l++; goto deg2; }
                p=1;
                if(grm[r][1]!=syn[j][u]) { r++; goto deg3; }
                u++;
                if(u==strlen(syn[j]) && grm[r][2]=='')  goto QRS;
                if(u<strlen(syn[j]) && grm[r][2]=='') { l=r; r=0; goto deg3; }
                if(grm[r][2]!='') { l=r; r=0; goto deg3; }
        /*      return;*/  l=r; r=0; goto deg3;

QRS:            outtextxy(439,300,"QRS");

}

int bul()

{
 struct ffblk f;
 int err;
 clrscr();
 textcolor(BLACK);
 textbackground(LIGHTGRAY);
 cprintf("\r\n Aktif directory'deki dosyalar :\r\n");
 findfirst("*.dat",&f,FA_ARCH);
print:  printf("   %s",f.ff_name);
        if((err=findnext(&f))!=-1) goto print;
 cprintf("\r\n\r\n\r\n Okumak istediÏiniz dosyann adn yaznz : (    .dat)");
```

```
 printf("\n\n ");
 textbackground(BLACK);
}
```

ÖZGEÇMİŞ

1969 yılında Aydın' ın Germencik ilçesinde doğdu. İlkokul ve ortaokul tahsilini İzmir' de tamamladı. 1983' te İzmir Atatürk Lisesinde başladığı lise öğrenimini 1986' da tamamladı. Aynı yıl Yıldız Üniversitesi Kocaeli Mühendislik Fakültesi Elektronik ve Haberleşme Mühendisliği bölümünde öğrenime başladı. 1990 yılında mezun oldu. Aynı yıl İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik Bölümü Biyomedikal Programına girmeye hak kazandı.