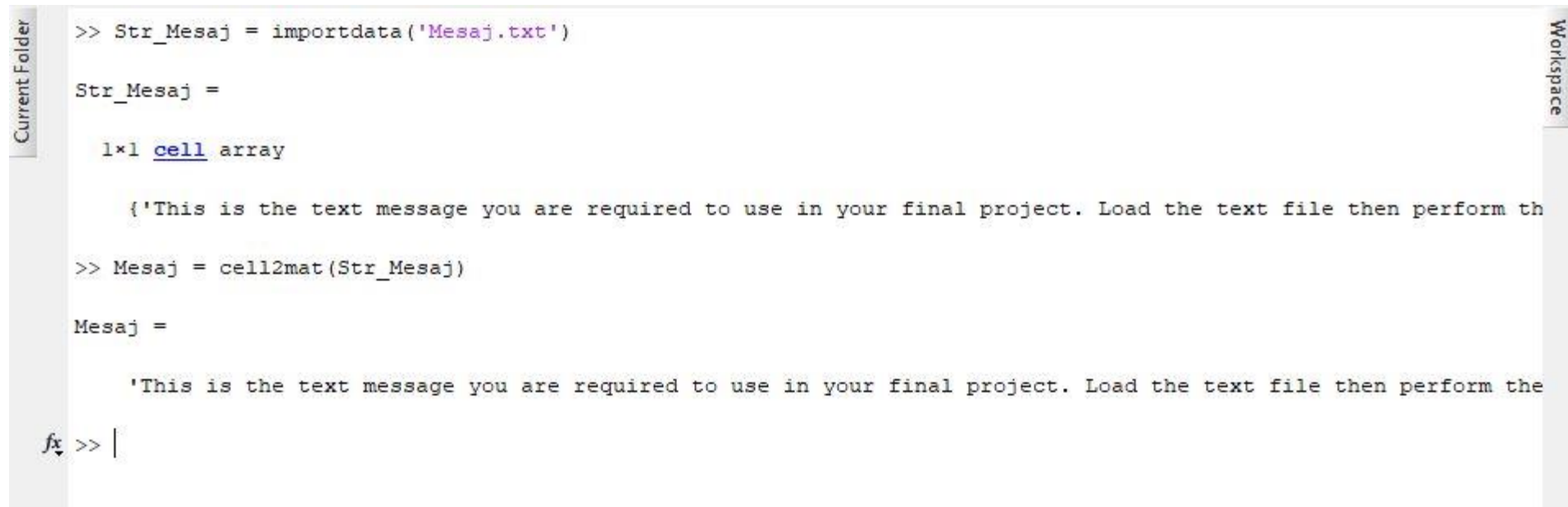


# Tutorial

# MATLAB'a bir metin dosyası nasıl yüklenir !!

- **Importdata (Dosya Adı)** komutunu kullanın.
- Değişkeni bir dizeye dönüştürmek için **cell2mat(içe aktarılan dosya içeriği)** kullanın.



```
>> Str_Mesaj = importdata('Mesaj.txt')

Str_Mesaj =

    1x1 cell array

    {'This is the text message you are required to use in your final project. Load the text file then perform th

>> Mesaj = cell2mat(Str_Mesaj)

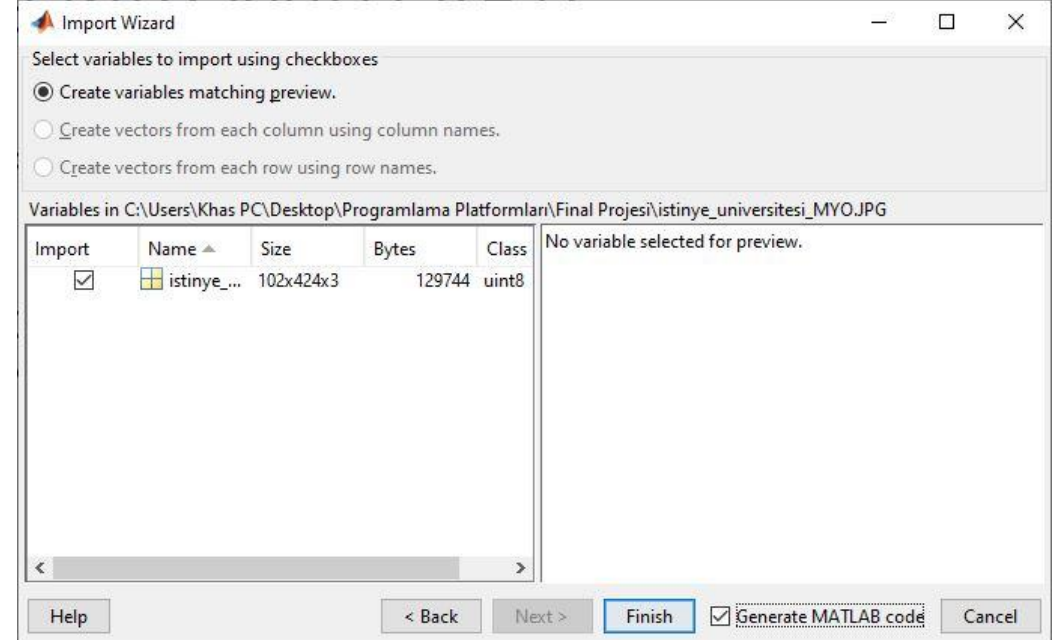
Mesaj =

    'This is the text message you are required to use in your final project. Load the text file then perform the

fx >> |
```

# MATLAB'a bir resim nasıl yüklenir !!

- Fotoğrafı sürükleyip komut penceresine bırakabilirsiniz!
- MATLAB, aşağıdaki resimdeki gibi yüklenen dosyanın ayrıntılarını gösteren yeni bir pencere açacaktır. "Generate MATLAB code" kutusunu işaretlemeyi unutmayın.



# Reshape

- Herhangi bir matrisin şeklini değiştirebilmek için **reshape(Matris, M, N)** fonksiyonunu kullanmanız gerekecektir.

```
Matrix =  
  
-0.1022    0.6277   -0.0068    0.0326   -1.0616  
-0.2414    1.0933    1.5326    0.5525    2.3505  
 0.3192    1.1093   -0.7697    1.1006   -0.6156  
 0.3129   -0.8637    0.3714    1.5442    0.7481  
-0.8649    0.0774   -0.2256    0.0859   -0.1924  
-0.0301   -1.2141    1.1174   -1.4916    0.8886  
-0.1649   -1.1135   -1.0891   -0.7423   -0.7648  
  
>> [MatrisSatirSayisi, MatrisSutunSayisi] = size(Matrix)  
  
MatrisSatirSayisi =  
  
    7  
  
MatrisSutunSayisi =  
  
    5  
  
>> MatrixToArray = reshape(Matrix, 1, MatrisSatirSayisi*MatrisSutunSayisi)  
  
MatrixToArray =  
  
Columns 1 through 11  
  
-0.1022   -0.2414    0.3192    0.3129   -0.8649   -0.0301   -0.1649    0.6277    1.0933    1.1093   -0.8637  
  
Columns 12 through 22  
  
    0.0774   -1.2141   -1.1135   -0.0068    1.5326   -0.7697    0.3714   -0.2256    1.1174   -1.0891    0.0326  
  
Columns 23 through 33  
  
    0.5525    1.1006    1.5442    0.0859   -1.4916   -0.7423   -1.0616    2.3505   -0.6156    0.7481   -0.1924  
  
Columns 34 through 35  
  
    0.8886   -0.7648  
  
>> size(MatrixToArray)  
  
ans =  
  
    1    35
```

# Reshape

- Matrisi yeniden şekillendirmek için, matrisin içindeki toplam eleman sayısı M ve N'ye bölünebilir olmalıdır. Aksi takdirde, böyle bir hatayı alırsınız.
- Gördüğünüz gibi 35, 8'e bölünemez. Ne yapabiliriz?
- Sonuna sıfırlar ekleyebiliriz!!

```
>> MatrixToArray

MatrixToArray =

Columns 1 through 11

-0.1022 -0.2414 0.3192 0.3129 -0.8649 -0.0301 -0.1649 0.6277 1.0933 1.1093 -0.8637

Columns 12 through 22

0.0774 -1.2141 -1.1135 -0.0068 1.5326 -0.7697 0.3714 -0.2256 1.1174 -1.0891 0.0326

Columns 23 through 33

0.5525 1.1006 1.5442 0.0859 -1.4916 -0.7423 -1.0616 2.3505 -0.6156 0.7481 -0.1924

Columns 34 through 35

0.8886 -0.7648

>> reshape(MatrixToArray, 8, 35/8)
Error using reshape
Size arguments must be real integers.
```

```
>> MatrixToArray = [MatrixToArray, zeros(1,5)
]

MatrixToArray =

Columns 1 through 11

-0.1022 -0.2414 0.3192 0.3129 -0.8649 -0.0301 -0.1649 0.6277 1.0933 1.1093 -0.8637

Columns 12 through 22

0.0774 -1.2141 -1.1135 -0.0068 1.5326 -0.7697 0.3714 -0.2256 1.1174 -1.0891 0.0326

Columns 23 through 33

0.5525 1.1006 1.5442 0.0859 -1.4916 -0.7423 -1.0616 2.3505 -0.6156 0.7481 -0.1924

Columns 34 through 40

0.8886 -0.7648 0 0 0 0 0

>> reshape(MatrixToArray, 8, 40/8)

ans =

-0.1022 1.0933 -0.7697 1.5442 -0.1924
-0.2414 1.1093 0.3714 0.0859 0.8886
0.3192 -0.8637 -0.2256 -1.4916 -0.7648
0.3129 0.0774 1.1174 -0.7423 0
-0.8649 -1.2141 -1.0891 -1.0616 0
-0.0301 -1.1135 0.0326 2.3505 0
-0.1649 -0.0068 0.5525 -0.6156 0
0.6277 1.5326 1.1006 0.7481 0
```

# Huffman....

- İkinci ödevinizde yaptığınız gibi, veri dizisindeki benzersiz sembolleri bulmanız gerekiyor.
- Bunun için yalnızca bir for döngüsüne ve bir IF koşuluna ihtiyacınız var, ardından her benzersiz karakteri yeni bir dizi değişkeninde depolayın. symbols adlandırın.
- İkinci ev ödevinizde yapıldığı gibi her benzersiz karakterin olasılığını hesaplayın, ardından başka bir değişken dizisinde atın. prob olarak adlandırın.
- Son olarak, huffman ağacını oluşturmak için `huffmandict(symbols, prob)` komutunu kullanın. Değişkeni dict olarak adlandırın.
- Örnekler için, MATLAB komut penceresine `help huffmandict` yazın.

# Huffman....

- Mesajı kodlamak ve başka bir değişken dizisinde atamak için `huffmanenco(mesaj dizisi, dict)` kullanın.
- Kod çözme için `huffmandeco (kodlanmış mesaj, dict)` kullanın ve başka bir değişken dizide atın.
- Örnekler için:
  - `help huffmanenco`
  - `help huffmandeco`

# G matrisi nasıl oluşturulur?

- Örnek olarak aşağıdaki üç parti formülünü ele alalım:
  - $P1 = D1 + D2 + D4$
  - $P2 = D2 + D4$
  - $P3 = D1 + D2 + D3 + D4$
- Şimdi her bir eşlik formülünü MATLAB'da bir dizi olarak yazalım:

```
>> P1 = [1,1,0,1]
```

```
P1 =
```

```
1    1    0    1
```

```
>> P2 = [0,1,0,1]
```

```
P2 =
```

```
0    1    0    1
```

```
>> P3 = [1,1,1,1]
```

```
P3 =
```

```
1    1    1    1
```



# G matrisi nasıl oluşturulur?

- G matrisinin  $[I, A^T]$  kullanılarak oluşturulduğunu, I kimlik matrisi ve A'nın parite matrisi olduğunu da biliyoruz.
- A matrisiyle başlayalım. Oldukça basit; sadece her eşlik dizisini bir satır olarak eklememiz ve bir matris oluşturmamız gerekir.

```
>> A = [P1;P2;P3]
```

```
A =
```

```
1 1 0 1
0 1 0 1
1 1 1 1
```

# G matrisi nasıl oluşturulur?

- Kimlik matrisi nedir? Sadece köşegeninde birleri olan kare bir matristir.
- Kimlik matrisi oluşturmak için `eye(M)` kullanacağız.
- Bir kimliğin boyutu nedir?
- A'dan belirlenir.  $G = [I, A^T]$  olduğundan, yalnızca A'yı devrik etmemiz gerekir.

```
>> transpose(A)
```

```
ans =
```

1	0	1
1	1	1
0	0	1
1	1	1

# G matrisi nasıl oluşturulur?

- kimlik matrisi,  $A^T$  matrisindekiyle aynı sayıda satıra sahip olacaktır. Böylece, `eye(4)!!`

```
>> I = eye(4)

I =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

- Son olarak, G matrisini oluşturacağız.

```
>> G = [I,AT]

G =

     1     0     0     0     1     0     1
     0     1     0     0     1     1     1
     0     0     1     0     0     0     1
     0     0     0     1     1     1     1
```

# H matrisi nasıl oluşturulur

- G matrisi gibi, H matrisini oluşturabiliriz.
- H matrisi aşağıdaki gibi oluşturulur:  $[A, I]$
- Zaten A elmizde var. Bunun  $A^T$  olmadığını unutmayın.
- Kimlik matrisinin yine A matrisinin aynı sayıda satırıyla eşleşmesi gerekir.

```
>> A
```

```
A =
```

```
1 1 0 1
0 1 0 1
1 1 1 1
```

```
>> I = eye(3)
```

```
I =
```

```
1 0 0
0 1 0
0 0 1
```

```
>> H = [A, I]
```

```
H =
```

```
1 1 0 1 1 0 0
0 1 0 1 0 1 0
1 1 1 1 0 0 1
```

# Şimdi basit bir örnek yapmaya çalışalım

- Huffman kodlamasını yaptığımızı ve Binary dizisinin doğrusal bloklar için hazır olduğunu varsayacağız.
- Öncelikle ikili dizisini  $M \times N$  matrisine yeniden şekillendirmemiz gerekir, burada  $N$ ,  $G$  matrisindeki satır sayısına eşit olmalıdır.
- $G$  matrisimiz 4 satırlıdır; bu nedenle ikili matrisimiz  $M \times 4$  olarak yeniden şekillendirilmelidir.
- Dizideki ikili sayıların toplam sayısı 4'e bölünemezse, diziyi bölünebilir hale getirmek için dizinin sonuna sıfırlar ekleyeceğiz.

```
>> BinaryArray = [1,1,1,0,0,1,0,0,1,1,0,0,0,0,1,1,1,0,1,0,1,0,1,1,1,0,0,0,0,1,1,1,1,1]

BinaryArray =

Columns 1 through 18

    1    1    1    0    0    1    0    0    1    1    0    0    0    0    1    1    1    0

Columns 19 through 34

    1    0    1    0    1    1    1    0    0    0    0    1    1    1    1    1    1    1

>> BinaryMatrix = reshape(BinaryArray, length(BinaryArray)/4, 4)
Error using reshape
Size arguments must be real integers.

>> BinaryArray = [BinaryArray, zeros(1,2)]

BinaryArray =

Columns 1 through 18

    1    1    1    0    0    1    0    0    1    1    0    0    0    0    1    1    1    0

Columns 19 through 36

    1    0    1    0    1    1    1    0    0    0    0    1    1    1    1    1    0    0

>> BinaryMatrix = reshape(BinaryArray, length(BinaryArray)/4, 4)

BinaryMatrix =

    1    1    1    0
    1    0    0    0
    1    0    1    1
    0    0    0    1
    0    0    1    1
    1    1    1    1
    0    1    1    1
    0    1    0    0
    1    0    0    0
```

# Devam....

- İkili diziye  $M \times 4$  matrisine yeniden şekillendirebilmek için sadece 2 sıfıra ihtiyacımız olduğunu görebiliriz. KOD ÇÜZMDEKTEN SONRA SIFIRLAR KALDIRMAYI UNUTMAYIN!!
- Sonuç matrisi, bloklara bölündüğünde mesajı temsil eder. Her satır, mesajın gerçek bir bloğudur.
- Şimdi her mesaj bloğu için pariteyi hesaplamalı ve bunları her bloğa eklemeliyiz.
- İkili matrisi  $G$  matrisiyle çarpmanız yeterlidir....
- İşlemi İkili çarpma olarak gerçekleştirmek için mod işlevini kullanıyorum....

```
>> EncodedMessage = mod(BinaryMatrix*G,2)
```

```
EncodedMessage =
```

1	1	1	0	0	1	1
1	0	0	0	1	0	1
1	0	1	1	0	1	1
0	0	0	1	1	1	1
0	0	1	1	1	1	0
1	1	1	1	1	0	0
0	1	1	1	0	0	1
0	1	0	0	1	1	1
1	0	0	0	1	0	1

# Devam....

- Şimdi son üç sütunun ikili matrise eklendiğini görebilirsiniz. Bunlar, her mesaj bloğu için hesaplanan ve eklenen üç paritedir.
- İkili matris içeriğinin çarpmadan sonra değişmediğini de görebilirsiniz.
- Son olarak, serpiştirme yapacağız, bunun için kendi fonksiyonunuzu yazmanız gerekecek. Yalnızca kodlanmış mesajın her bir sütununu almanız ve tüm sütun içerikleriyle bir dizi oluşturmanız gerekir.
- Örneğimiz için, serpiştiren mesajı mesaj resimdeki gibi olmalıdır. Fonksiyonunuzu aynı kodlanmış matris üzerinde test edebilirsiniz, aynı sonucu almalısınız.
- Son olarak, frame eklemeniz gerekir.

```
InterleavedMessage =  
  
Columns 1 through 18  
1 1 1 0 0 1 0 0 1 1 0 0 0 0 1 1 1 0  
  
Columns 19 through 36  
1 0 1 0 1 1 1 0 0 0 0 1 1 1 1 1 0 0  
  
Columns 37 through 54  
0 1 0 1 1 1 0 1 1 1 0 1 1 1 0 0 1 0  
  
Columns 55 through 63  
1 1 1 1 0 0 1 1 1  
>> Frame = [0,1,1,1,1,1,1,1,0]  
  
Frame =  
0 1 1 1 1 1 1 1 1 0  
  
>> FinalMessage = [Frame,InterleavedMessage]  
  
FinalMessage =  
  
Columns 1 through 18  
0 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 0 0  
  
Columns 19 through 36  
1 1 0 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0  
  
Columns 37 through 54  
0 0 0 1 1 1 1 1 0 0 0 1 0 1 1 1 0 1  
  
Columns 55 through 72  
1 1 0 1 1 1 0 0 1 0 1 1 1 1 0 0 1 1  
  
Column 73  
1
```

# Şimdi kodu çözmelim

- Başlamadan önce tüm sendromları hesaplamamız gerekiyor.
- Bunun için ders notlarına bakmalısınız, ama bir sendromu hesaplayalım.
- Örnekte hesaplanan sendrom ilk bit içindir.
- Bu, tüm hata olasılıkları için yapılmalıdır.

```
>> FirstBitError = [1,0,0,0,0,0,0,0]
```

```
FirstBitError =
```

```
1    0    0    0    0    0    0    0
```

```
>> Syndrome_1 = H*transpose(FirstBitError)
```

```
Syndrome_1 =
```

```
1  
0  
1
```



# Devam....

- Tüm sendromlar hesaplandıktan sonra, artık mesajda hata olup olmadığını kontrol edebiliriz.
- Son adımda eklenen frame arayan bir işlev yazmanız ve ardından kaldırmanız gerekir..
- Frame kaldırdıktan sonra, serpiştirleyen mesaj dizisini orijinal  $M \times N$  şekline çevirmek için başka bir işlev yazmanız gerekir.

# Devam....

- Son olarak, her mesaj bloğu için sendromları hesaplamamız ve hataları kontrol etmeniz gerekecektir.
- Sendrom matrisi tüm sıfırlarla tutarlı olduğundan, mesajımızda hata olmadığı anlamına gelir.
- Bu nedenle, matrisin yalnızca her bloğun paritelerine karşılık gelen son üç sütununu kaldırmamız gerekir.

```
>> ReceivedMessage
```

```
ReceivedMessage =
```

1	1	1	0	0	1	1
1	0	0	0	1	0	1
1	0	1	1	0	1	1
0	0	0	1	1	1	1
0	0	1	1	1	1	0
1	1	1	1	1	0	0
0	1	1	1	0	0	1
0	1	0	0	1	1	1
1	0	0	0	1	0	1

```
>> MessageSyndromes = mod(H*transpose(ReceivedMessage),2)
```

```
MessageSyndromes =
```

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

```
>> RemovedParitesMessage = ReceivedMessage(:,1:4)
```

```
RemovedParitesMessage =
```

1	1	1	0
1	0	0	0
1	0	1	1
0	0	0	1
0	0	1	1
1	1	1	1
0	1	1	1
0	1	0	0
1	0	0	0

# Devam....

- Ya mesajda hatalar varsa?
- Alınan mesajın ilk beş mesaj bloğunun ilk bitine hatalar ekleyeceğiz.
- Ardından sendrom matrisini hesaplayacağız.
- Sendrom matrisinin sıfır olmayan bir matris olduğunu görebiliriz.

```
>> ReceivedMessage(1:5,1) = mod(ReceivedMessage(1:5,1) + 1,2)
```

```
ReceivedMessage =
```

0	1	1	0	0	1	1
0	0	0	0	1	0	1
0	0	1	1	0	1	1
1	0	0	1	1	1	1
1	0	1	1	1	1	0
1	1	1	1	1	0	0
0	1	1	1	0	0	1
0	1	0	0	1	1	1
1	0	0	0	1	0	1

```
>> MessageSyndromes = mod(H*transpose(ReceivedMessage),2)
```

```
MessageSyndromes =
```

1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0



# Devam....

- Son olarak daha önce yaptığımız gibi parite sütunlarını kaldıracağız.
- Ardından mesajı  $1 \times M$  dizisinde yeniden şekillendireceğiz, böylece Huffman kod çözme işlemini gerçekleştirebiliriz..
- Mesajın kodunu çözmeden önce, Huffman kodlamasından sonra eklenen tüm sıfırları kaldırın.
- Kod çözme işleminden sonra, kodu çözülen mesajı orijinal şekline göre yeniden şekillendirin ve kodu çözülen mesaj ile orijinal mesajın eşit olup olmadığını kontrol edin.