EE 5240 - Lecture 9 Friday Jan 30, 2015

Topics for Today:

- Questions?

Today - system data for computer studies

- MatLab - open and read CDF files.
- Data types - bits, bytes, characters, strings, integer, floating point. Formats used in Matlab.
- Loadflow Formulation
- Bus Data, Bus Types, Voltage controlled bus
- NR Algorithm implementation.

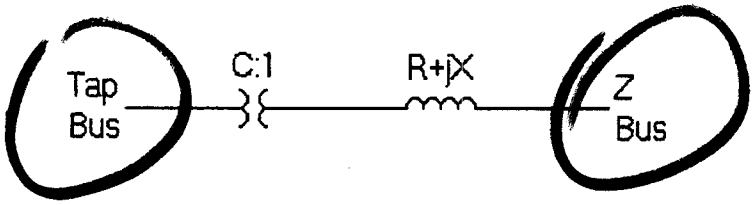
- Paralleling transformers
  - Unlike impedances, Unlike tap positions
- Coming up - keep studying Chapters 3 & 4.
  - Nonlinear systems of equations
  - Newton Iterative Method
  - Newton-Raphson Load Flow Formulation
  - Loadflow Setup, practical view

```
%
% MatLab Load Flow Program, to run with IEEE CDF input file format.
% Initial version, set to automatically open and run ieee14.cdf case
% kept in c:\matlab\work\ directory.
%
% Programmer:   Bruce A. Mork, Michigan Tech University
%                   26 Sep 2000 - CDF input sections written
%                   27 Sep 2000 - [Ybus] construction completed & debugged
%                   09 Feb 2003 - Changed I/O to open file as text file.

fid = fopen('ieee14.cdf','rt');

%Read in Title Data
line = fgetl(fid);
    date = sscanf(line,'%*1c%8c',1);
    orig = sscanf(line,'%*10c%20c',1);
    bmva = sscanf(line,'%*31c%f',1);
    year = sscanf(line,'%*38c%4i',1);
  season = sscanf(line,'%*43c%1c',1);
  caseid = sscanf(line,'%*45c%28c',1);
```

Transformer LTC's in the CDF File Format

Tap and impedance location specified in first two entries in branch data section.

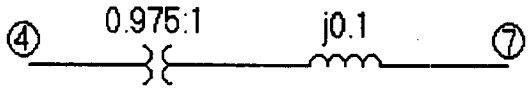
- **entry 1 is bus non-unity tap is connected to**
- **entry 2 is bus device impedance is connected to**



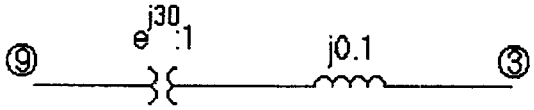
Complex turns ratio due to phase shifting transformer split to two entries

- **entry 15 is transformer final turns ratio**
- **entry 16 is transformer (phase shifter) final angle**

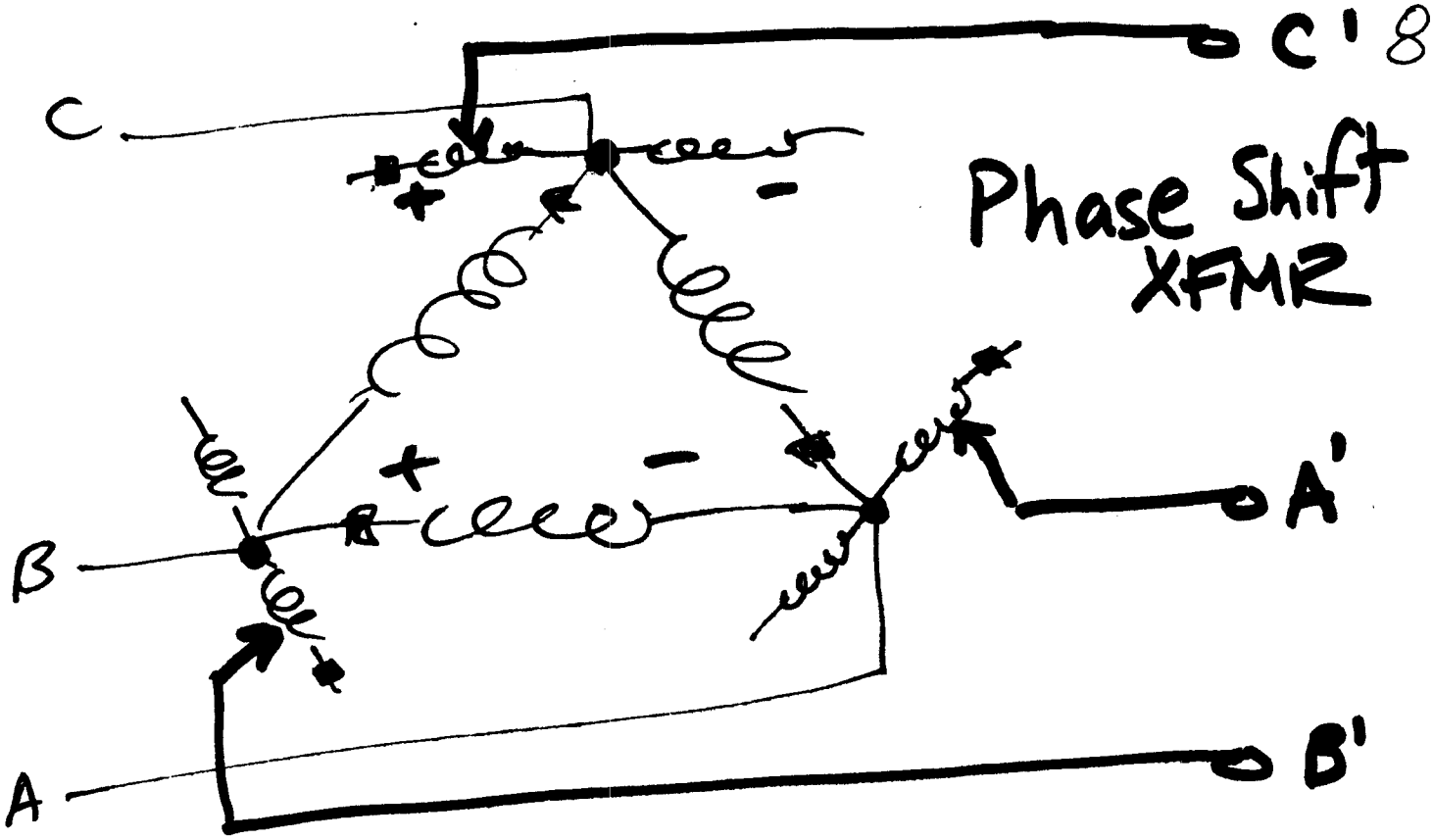
Examples:

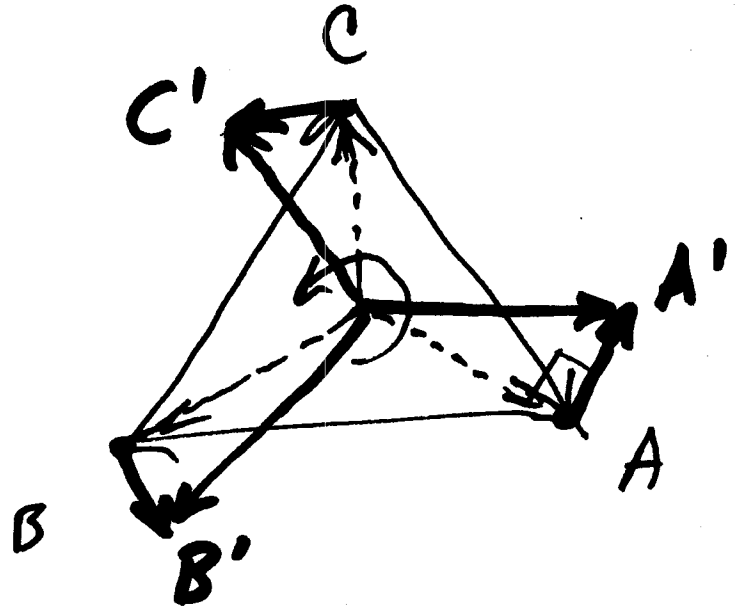


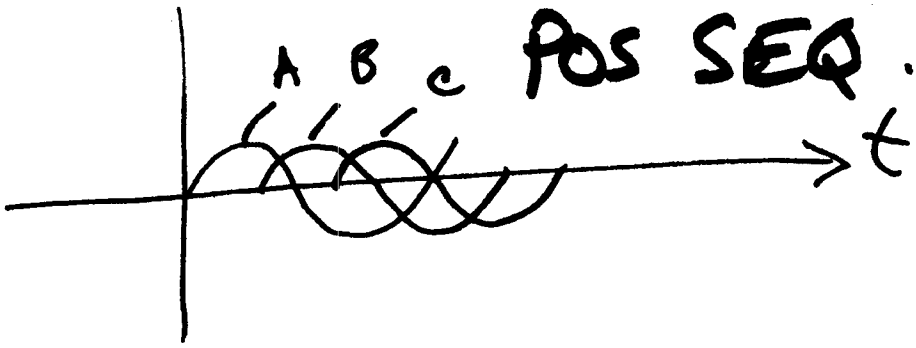
Entry:	1	2	15	16
	4	7	.975	0

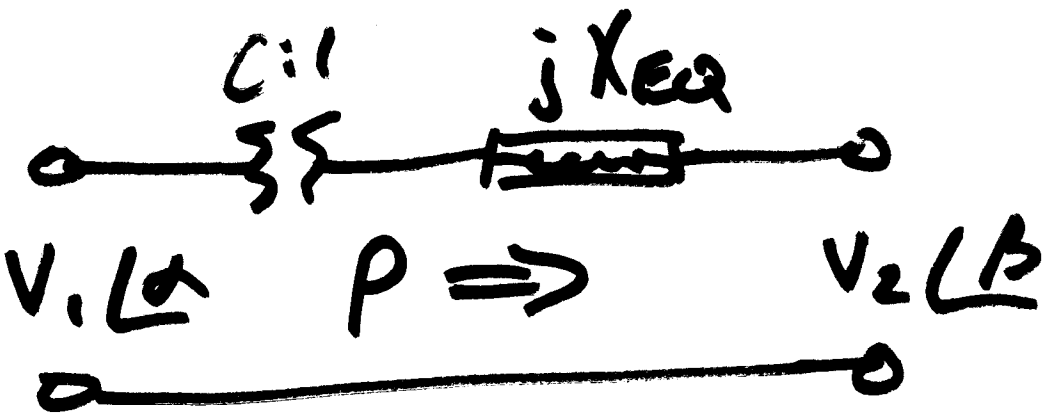


Entry:	1	2	15	16
	9	3	1	30


C'
C
Phase Shift
XFMR
+
-
+
-
A'
B
B'
A


C
C'
A'
A
B
B'

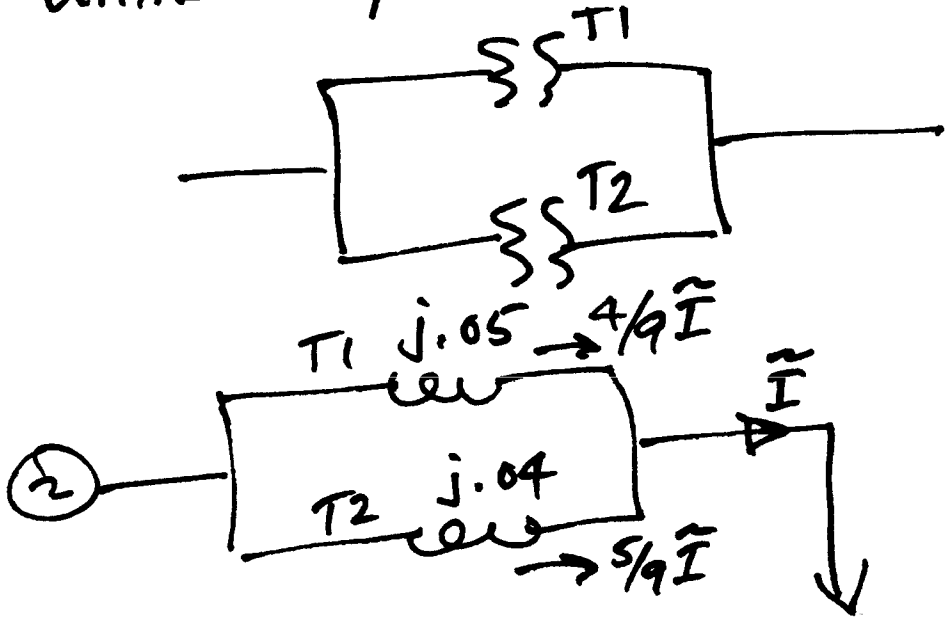

A B C
POS SEQ.
t



$$P = \frac{V_1 V_2}{X_{EQ}} \sin(\alpha - \beta)$$

Paralleling XFMRS:

- Unlike impedances, same voltage ratio



	Base: on 100 MVA
T1: 90 MVA	j.05 p.u.
T2: 100 MVA	j.04 p.u.

Constraints

$$\frac{I_{T_1}}{I_{T_2}} = \frac{4}{5}$$ (90 MVA) (100 MVA)

Key: Spec same % Z on Base of indiv XFMRS!

i.e, T2: j.04 @ 100 MVA BASE
T1: j.04 @ 90 MVA BASE

If T_1 @ 90 MVA, T_2 @ $90\frac{5}{4} = 112.5$ MVA BAD

If T_2 @ 100 MVA, T_1 @ 80 MVA GOOD/OK

However, can't use 10MVA of T1 capacity.

```
%
% MatLab Load Flow Program, to run with IEEE CDF input file format.
% Initial version, set to automatically open and run ieee14.cdf case
% kept in c:\matlab\work\ directory.
%
% Programmer:    Bruce A. Mork, Michigan Tech University
%                     26 Sep 2000 - CDF input sections written
%                     27 Sep 2000 - [Ybus] construction completed & debugged
%                     09 Feb 2003 - Changed I/O to open file as text file.

fid = fopen('ieee14.cdf','rt');

%Read in Title Data
line = fgetl(fid);
    date = sscanf(line,'%*1c%8c',1);
    orig = sscanf(line,'%*10c%20c',1);
    bmva = sscanf(line,'%*31c%f',1);
    year = sscanf(line,'%*38c%4i',1);
  season = sscanf(line,'%*43c%1c',1);
  caseid = sscanf(line,'%*45c%28c',1);
```

r = read only

t = text/ASCII

CHARACTER CODES 000 - 255

000	Ctl U	space
001	Alt S	☻
002	Alt B	●
003	Alt E	♥
004	Alt V	♦
005	Alt P	♣
006	Alt C	♠
007	Alt M	•
008	Alt J	◘
009	(tab)	○
010	(line feed)	■
011	Alt G	♂
012	Shf Alt T	♀
013	(CR)	♪
014	Align font	♫
015	Alt Z	☼
016	Alt D	►
017	Alt N	◄
018	Alt W	↕
019	Alt O	‼
020	Alt K	¶
021	Alt I	§
022	Alt Q	▬
023	Alt U	↨
024	Alt H	↑
025	Alt L	↓
026	(end-file)	→
027	(escape)	←
028	Alt F	∟
029	Shf Ctl -	↔
030	Alt R	▲
031	Alt Y	▼
032		space
033		!
034		"
035		#
036		$
037		%
038		&
039		'
040		(
041		)
042		*
043		+
044		,
045		-
046		.
047		/
048		0
049		1
050		2
051		3
052		4
053		5
054		6
055		7
056		8
057		9
058		:
059		;
060		<
061		=
062		>
063		?
064		@
065		A
066		B
067		C
068		D
069		E
070		F
071		G
072		H
073		I
074		J
075		K
076		L
077		M
078		N
079		O
080		P
081		Q
082		R
083		S
084		T
085		U
086		V
087		W
088		X
089		Y
090		Z
091		[
092		\
093		]
094		^
095		
096		`
097		a
098		b
099		c
100		d
101		e
102		f
103		g
104		h
105		i
106		j
107		k
108		l
109		m
110		n
111		o
112		p
113		q
114		r
115		s
116		t
117		u
118		v
119		w
120		x
121		y
122		z
123		{
124		|
125		}
126		~
127		
128	C acc't ,	Ç
129	u acc't "	ü
130	e acc't '	é
131	a acc't ^	â
132	a acc't "	ä
133	a acc't `	à
134	o acc't a	å
135	c acc't ,	ç
136	e acc't ^	ê
137	e acc't "	ë
138	e acc't `	è
139	i acc't "	ï
140	i acc't ^	î
141	i acc't `	ì
142	A acc't "	Ä
143	A acc't o	Å
144	E acc't '	É
145	a acc't e	æ
146	A acc't E	Æ
147	o acc't ^	ô
148	o acc't "	ö
149	o acc't `	ò
150	u acc't ^	û
151	u acc't `	ù
152	y acc't "	ÿ
153	O acc't "	Ö
154	U acc't "	Ü
155	c acc't /	¢
156	- acc't L	£
157	= acc't Y	¥
158	t acc't P	R
159	- acc't f	ƒ
160	a acc't '	á
161	i acc't '	í
162	o acc't '	ó
163	u acc't '	ú
164	n acc't ~	ñ
165	N acc't ~	Ñ
166	a acc't _	ª
167	o acc't _	º
168	? acc't ?	¿
169	[acc't `	⌐
170	] acc't `	¬
171	1 acc't 2	½
172	1 acc't 4	¼
173	! acc't !	¡
174	< acc't <	«
175	> acc't >	»
176	1 acc't `	▒
177	2 acc't `	▒
178	3 acc't `	▒
179	Shf Alt R	│
180	Shf Alt D	┤
181	Shf Ctl K	╡
182	Shf Alt K	╢
183	Shf Alt O	╖
184	Shf Ctl O	╕
185	Shf Ctl D	╣
186	Shf Ctl R	║
187	Shf Ctl E	╗
188	Shf Ctl C	╝
189	Shf Alt M	╜
190	Shf Ctl M	╛
191	Shf Alt E	┐
192	Shf Alt Z	└
193	Shf Alt X	┴
194	Shf Alt W	┬
195	Shf Alt A	├
196	Shf Alt F	─
197	Shf Alt S	┼
198	Shf Ctl H	╞
199	Shf Alt H	╟
200	Shf Ctl Z	╚
201	Shf Ctl Q	╔
202	Shf Ctl X	╩
203	Shf Ctl W	╦
204	Shf Ctl A	╠
205	Shf Ctl F	═
206	Shf Ctl S	╬
207	Shf Ctl N	╧
208	Shf Alt N	╨
209	Shf Ctl I	╤
210	Shf Alt I	╥
211	Shf Alt B	╙
212	Shf Ctl B	╘
213	Shf Ctl U	╒
214	Shf Alt U	╓
215	Shf Alt J	╫
216	Shf Ctl J	╪
217	Shf Alt C	┘
218	Shf Alt Q	┌
219	4 acc't `	█
220	5 acc't `	▄
221	6 acc't `	▌
222	7 acc't `	▐
223	8 acc't `	▀
224	a acc't /	α
225	b acc't /	ß
226	g acc't /	Γ
227	p acc't /	π
228	S acc't /	Σ
229	s acc't /	σ
230	m acc't /	µ
231	t acc't /	τ
232	F acc't /	Φ
233	h acc't /	Θ
234	O acc't /	Ω
235	d acc't /	δ
236	$ acc't `	∞
237	f acc't /	φ
238	e acc't /	ε
239	^ acc't `	∩
240	= acc't `	≡
241	+ acc't `	±
242	> acc't `	≥
243	< acc't `	≤
244	(acc't `	⌠
245	) acc't `	⌡
246	: acc't `	÷
247	~ acc't `	≈
248	% acc't `	°
249	. acc't `	∙
250	& acc't `	·
251	/ acc't `	√
252	! acc't `	ⁿ
253	* acc't `	²
254	# acc't `	■
255	@ acc't `	

AC 7-BIT

VAX

VT-220

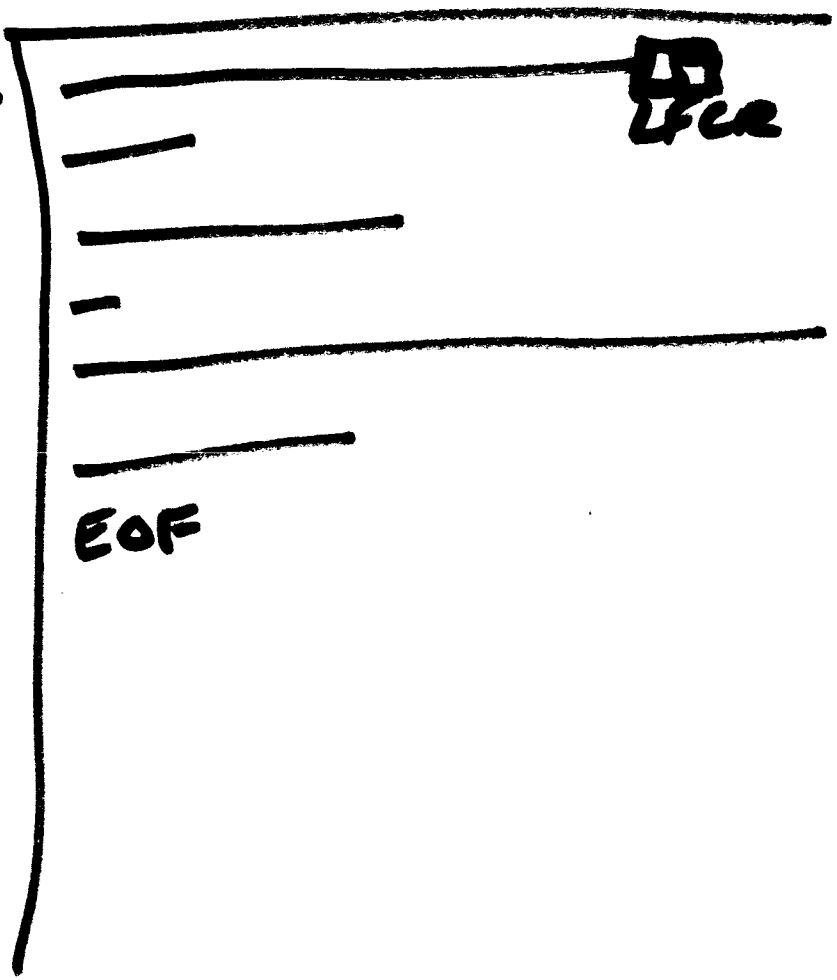
VT220

use char, ichar

File Types ① Sequential Access

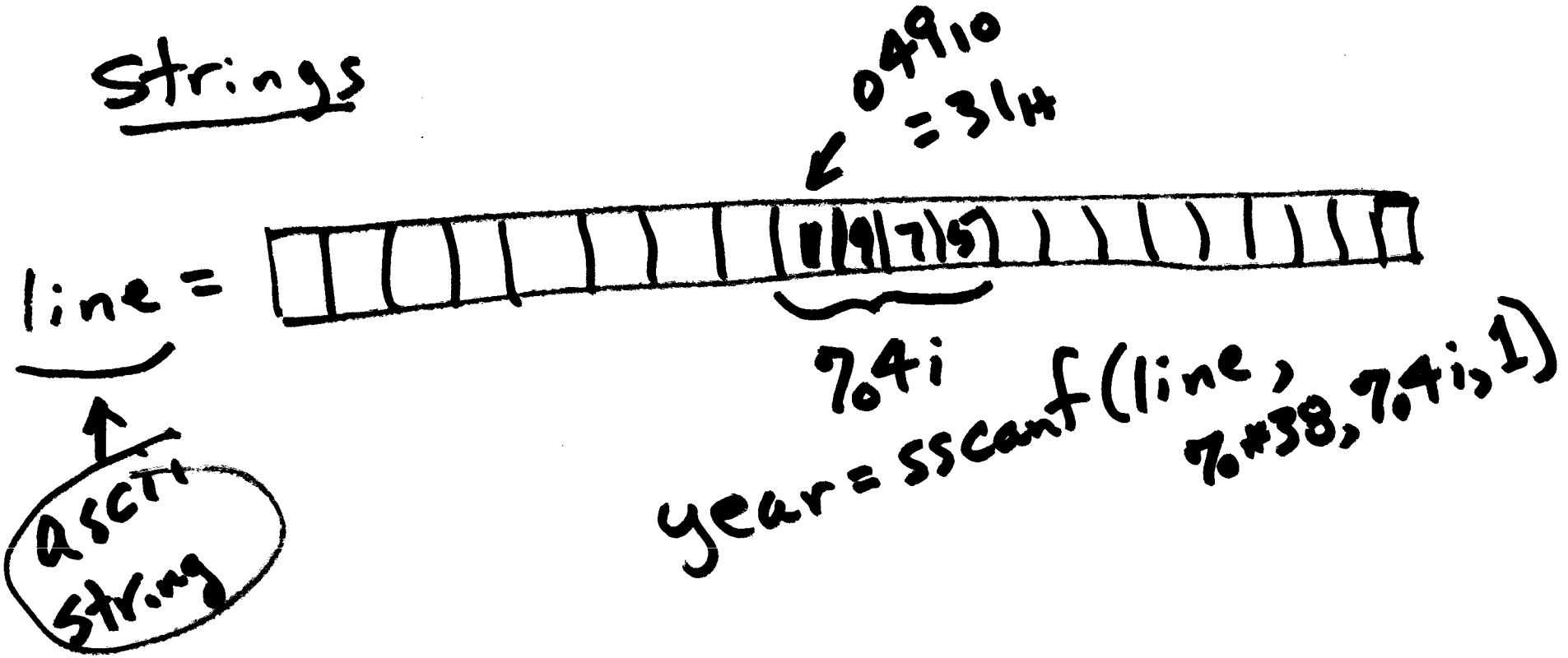
- Direct Access

① Seq →



Typical for ASCII (TEXT).

Strings



Data Types

		unsigned	signed
Integers:	1-byte	0-255	±127
	2-byte	0-65,535	
	4-byte	$0-(2^{32}-1)$	
Real	4-byte	- Single	
	8-byte	- Double	
Cmplx	8-byte	- Single	
	16-byte	- Double.	

To: ee5240-l@mtu.edu
From: Bruce Mork <bamork@mtu.edu>
Subject: data types, useful observations, info

From Matlab's help function (>> help datatypes) I obtained the following info and added a few comments of my own:

Data types (classes)
==========
Real (floating point):
single - Convert to single precision. (4 bytes)
double - Convert to double precision. (8 bytes)

Complex floating point:
complex - Convert to complex number stored as 2 real nos.
single and double functions can be applied to a complex number.
single ==> 2 single prec floating-pt nos, i.e. 8 bytes.
double ==> 2 double prec floating-pt now, i.e. 16 bytes.

Unsigned Integers:
uint8 - Convert to unsigned 8-bit integer. (1 byte)
uint16 - Convert to unsigned 16-bit integer. (2 bytes)
uint32 - Convert to unsigned 32-bit integer. (4 bytes) single precision
uint64 - Convert to unsigned 64-bit integer. (8 bytes) double-precision integer

Signed Integers:
int8 - Convert to signed 8-bit integer. (1 byte)
int16 - Convert to signed 16-bit integer. (2 bytes)
int32 - Convert to signed 32-bit integer. (4 bytes) single precision
int64 - Convert to signed 64-bit integer. (8 bytes) double-precision integer

Character or Character String:
char - Create character array (string). (1 byte per character)

Logic Variable:
logical - Convert numeric values to logical. 1 bit per logic variable, 8 bits/byte.

You can use the class function to find out what the class (data type) is of a given variable. Default class (data type) for all variables in matlab is double-precision real (8 bytes). You can change the default display precision (not the internal storage precision) with the "format" command. Check out >> help format, and >> help class to find out more.

In terms of reading and writing variables into an ascii text file, you need to apply the I/O formats like I described for sscanf function today. Recall

```
bmva = sscanf(line, '%*31c%f',1);
```

Pasting from Matlab help, we find out that these are the same as in "standard" C language.

FORMAT is a string containing C language conversion specifications.
Conversion specifications involve the character %, optional

assignment-suppressing asterisk and width field, and conversion characters d, i, o, u, x, e, f, g, s, c, and [. . .] (scanset). Complete ANSI C support for these conversion characters is provided consistent with 'expected' MATLAB behavior. For a complete conversion character specification, see a C manual.

It may help to think of this I/O formatting business as controlling how a variable stored in computer memory (in "raw" binary form) is converted to a string of ascii bytes/characters to be written into a .txt file. The same format must be used for both output (writing) and input (reading) in order to the information intact. (note that formatting is not used when data are stored in a "binary file" - we just read and write an exact copy of the variable's raw binary value.)

If you go to Matlab's nice help utility and read about sscanf, it gives a very good explanation of all of these I/O formats. From the interactive prompt, >> doc sscanf

Finally, note that there is another "format" that can be controlled, this is simply the display format for numbers displayed on the screen at the matlab prompt >>. Type help format for more information on this. Default display format is "short".

Enjoy. Hope this makes for a fun weekend... ?? 

Dr. Mork