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-Raphson Load Flow Formulation
- Newton-Seddieken Method
- Loadflow Setup, practical view
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:

\[
\begin{array}{cccc}
4 & 0.975:1 & j0.1 & 7 \\
\end{array}
\]

Entry: 1 2 15 16
4 7 .975 0

\[
\begin{array}{cccc}
3 & e^{j30.1} & j0.1 & 3 \\
\end{array}
\]

Entry: 1 2 15 16
9 3 1 30
Phase Shift
XFMR

A'
B'

C'
C
B
A

POE SEQ.

A B C t
\[ P = \frac{V_1 V_2}{X_{Ea}} \sin (\alpha - \beta) \]
Paralleling XFMRS:
- Unlike impedances, same voltage ratio

Base: 
\[
\frac{T1}{T2} = \frac{90 \text{ MVA}}{100 \text{ MVA}} = 0.9 \text{ p.u.}
\]

Constraints:
\[
\frac{I_{T1}}{I_{T2}} = \frac{4}{5} \quad (90 \text{ MVA})
\]
\[
\frac{I_{T1}}{I_{T2}} = \frac{4}{5} \quad (100 \text{ MVA})
\]

Key: Spec same \%Z on base of indiv XFMRS!
- i.e., $T2: j.09 @ 100 \text{ MVA BASE}$
- $T1: j.04 @ 90 \text{ MVA BASE}$

LOAD
- If $T1 @ 90 \text{ MVA}$, $T2 @ \frac{90.5}{4} \text{ MVA}$ (BAD)
- If $T2 @ 100 \text{ MVA}$, $T1 @ 80 \text{ MVA}$ (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);
<table>
<thead>
<tr>
<th>Character Code</th>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>Ctrl U</td>
<td>space</td>
</tr>
<tr>
<td>001</td>
<td>Alt S</td>
<td></td>
</tr>
<tr>
<td>002</td>
<td>Alt B</td>
<td></td>
</tr>
<tr>
<td>003</td>
<td>Alt E</td>
<td></td>
</tr>
<tr>
<td>004</td>
<td>Alt V</td>
<td></td>
</tr>
<tr>
<td>005</td>
<td>Alt P</td>
<td></td>
</tr>
<tr>
<td>006</td>
<td>Alt C</td>
<td></td>
</tr>
<tr>
<td>007</td>
<td>Alt M</td>
<td></td>
</tr>
<tr>
<td>008</td>
<td>Alt J</td>
<td></td>
</tr>
<tr>
<td>009</td>
<td>(tab)</td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>(line feed)</td>
<td>space</td>
</tr>
<tr>
<td>011</td>
<td>Alt G</td>
<td></td>
</tr>
<tr>
<td>012</td>
<td>Shf Alt T</td>
<td></td>
</tr>
<tr>
<td>013</td>
<td>(CR) Alt X</td>
<td></td>
</tr>
<tr>
<td>014</td>
<td>Align font</td>
<td></td>
</tr>
<tr>
<td>015</td>
<td>Alt Z</td>
<td></td>
</tr>
<tr>
<td>016</td>
<td>Alt D</td>
<td></td>
</tr>
<tr>
<td>017</td>
<td>Alt N</td>
<td></td>
</tr>
<tr>
<td>018</td>
<td>Alt W</td>
<td></td>
</tr>
<tr>
<td>019</td>
<td>Alt O</td>
<td></td>
</tr>
<tr>
<td>020</td>
<td>Alt K</td>
<td></td>
</tr>
<tr>
<td>021</td>
<td>Alt I</td>
<td></td>
</tr>
<tr>
<td>022</td>
<td>Alt G</td>
<td></td>
</tr>
<tr>
<td>023</td>
<td>Alt U</td>
<td></td>
</tr>
<tr>
<td>024</td>
<td>Alt H</td>
<td></td>
</tr>
<tr>
<td>025</td>
<td>Alt L</td>
<td></td>
</tr>
<tr>
<td>026</td>
<td>(end-file)</td>
<td></td>
</tr>
<tr>
<td>027</td>
<td>(escape)</td>
<td></td>
</tr>
<tr>
<td>028</td>
<td>Alt F</td>
<td></td>
</tr>
<tr>
<td>029</td>
<td>Shf Ctrl -</td>
<td></td>
</tr>
<tr>
<td>030</td>
<td>Alt R</td>
<td></td>
</tr>
<tr>
<td>031</td>
<td>Alt Y</td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**

- The table above lists various control characters and their corresponding codes in the ASCII character set. These codes range from 000 to 255.

**Character Set:**

- The chart shows how characters are represented using these codes, including control characters like Ctrl, Alt, Shf, and various operators such as space, tab, line feed, and more.

**Usage:**

- These codes are essential for understanding how data is encoded in text files, telecommunications, and other digital communication systems.
File Types

- Sequential Access
  - Direct Access

0 Seg →

Typical for ASCII (TEXT).

EOF
Data Types

Integers:
- 1-byte
- 2-byte
- 4-byte

Real:
- 4-byte - Single
- 8-byte - Double
- 16-byte - Double

Unsigned:
- 0-255

Signed:
- ±127
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  

\[
\text{bmva} = \text{sscanf}(\text{line}, "\%*31c\%f", 1);
\]

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