מעבדה במכשור

הנדסה ביורפואית

זרימה

מגישים:

נדב אמיתי

יובל כסיף

סול אמארה

תאריך:

05.04.2022

תוכן עניינים:

1 רקע תאורטי:

זרימה תמידית- זרימה תקרא תמידית אם בכל נקודה בזורם הזרימה אינה תלויה בזמן, כלומר נוכל להגדיר כל מאפיין של הזורם להיות תלוי רק במרחב $(f(\mathrm{x},\mathrm{y},\mathrm{z}))$, בנוסף נוכל לומר כי בכל רגע $\frac{\partial f}{\partial t} = 0$.

זרימה למינארית- זרימה למינארית הינה זרימה המאופיינת בשכבות, כלומר הנוזל זורם בשכבות מקבילות אחת לשנייה. תכונה נוספת של זרימה למינארית היא שחלקיקי הזורם נעים רק בכיוון הזרם בכל נקודה.

זרימה טורבולנטית- בזרימה טורבולנטית חלקיקי הזורם מבצעים ויברציות באקראיות לכל הכיוונים בתדר גבוה ואמפליטודה קטנה, אך נעים באופן כולל בכיוון הזרימה.

זרימה פועמת- את זרימה זו ניתן למצוא בגוף אדם בווריד‍ים ועורקים הנמצאים בקרבת הלב, זרימה זו מאופיינת בכך שמושפעת רבות מפעימות הלב, כלומר, זרימה בקצב שאינו אחיד, אך בעל תבנית דומה בכל פעימה של הלב.

מספר ריינולדס- מספר ריינולדס הינו מספר חסר יחידות, המתאר את היחס בין כוחות האינרציה לכוחות הויסקוזיים. כלומר:

$$Reynolds\ number = \frac{\rho U d}{\mu} = \frac{U d}{v}$$

כאשר U מייצג את מהירות הזורם, d את קוטר הצינור בו זורם הנוזל, ρ את הצפיפות, μ את הצמיגות הדינמית, ו-v את הצמיגות הקינמטית של הזורם ($v = \frac{\mu}{\rho}$). [1]

כאשר מספר ריינולדס קטן מ-2100, נקבל זרימה למינרית, כאשר הוא נמצא בין 2100-4200 אנו נמצאים בזרימה טרנזיאנטית, וכאשר מספר ריינלודס גדול מ4200 אנו נמצאים בזרימה טורבולנטית. [2]

זרימת פואזיי- זרימת פואזיי מאפיינת זורם ניוטוני, בלתי דחיס, בעל זרימה תמידית ולמינרית, בצינור קשיח ארוך ודק ($L \gg R$) ללא פיצולים. זרימה זו מושפעת מגרדיאנט הלחץ. ספיקתו של זורם המאופיין בזרימת פואזיי ניתנת לתיאור בעזרת נוסחאת האגן פואזיי:

$$(1)\ Q = \frac{\pi \Delta P R^4}{8\mu L^2}$$

כאשר Q היא הספיקה הנפחית ΔP- הפרש הלחצים, R- רדיוס הצינור, μ – צמיגות הזורם , L - אורך הצינור.[3]

ספיקה -ישנם שני סוגי ספיקה: נפחית ומסתית. ספיקה נפחית מתארת את כמות החומר שעוברת בשטח מסוים ליחידת זמן כלומר את השינוי הנפחי בזמן ומחושבת באמצעות אינטגרל משטחי על המהירות: $(2)\ Q = \frac{dV}{dt} = \iint_S v \cdot dS$.

ספיקה מסתית מתארת את מסת החומר שעוברת בשטח מסוים ליחידת זמן ומחושבת באמצעות הכפלת האינטגרל הנפחי בצפיפות החומר: $\dot{m} = \iint_S \rho \cdot v \cdot dS$ (3) [3]

התנגדות זורם- בעת זרימת הזורם בכלי ישנו חיכוך בינו לבין דפנות הכלי ולכן ישנה התנגדות של הזורם. הנוסחה עבור התנגדות הזורם בזרימת פואזיי הינה: $R = \frac{\Delta P}{Q}$ (4) . [1]

חוק בויל – חוק בויל מתאר את הקשר בין לחץ ונפח עבור גזים אידיאלים הנמצאים במצב של כמות חלקיקים קבועה וטמפרטורה קבועה. $PV = k$ (5) , כאשר K הינו קבוע התלוי בכמות החומר ובטמפרטורה. [4]

היענות – באורגניזם חי ניתן למדוד את הקשר בין לחץ ונפח, לדוגמא כאשר מדובר על כלי דם במערכת הקרדיווסקולרית, הגדלת הנפח גורמת להגדלת הלחץ ולהפך. [3]

$$(6)\ \ C = \frac{\Delta V}{\Delta P}$$

כאשר C הוא הענות החומר ו $\Delta P, \Delta V$ הם השינויים בלחץ ובנפח. בעזרת C נוכל להגדיר E (Elastance) המתאר את אחד חלקי C ($E = \frac{\Delta P}{\Delta V}$). הענות והאלסטנס כערכים כמותיים שימושיים למדידת זקנה ומחלה.

מערכת הדם אחראית על הזרמת הדם מהלב אל חלקי הגוף השונים באמצעות מערכת של עורקים והחזרת הדם מהגוף מתבצעת באמצעות הורידים. הלב הוא משאבה בעזרתה מוזרם הדם, כאשר בעזרת כיווץ והרפיה, שולט בקצב העברת הדם ממנו. דיאסטולה- הרפיית שרירי הלב, סיסטולה- כיווץ שרירי הלב. בשל כך, העורקים והווורידים בקרבת הלב עומדים תחת לחץ רב וצריכים להתאים את מאפייניהם לעמידה בלחץ זה.

נפח סוף דיאסטולה *(EDV)* הוא הנפח המקסימלי בחדרים וערכו כ *120 ml*.

נפח סוף סיסטולה *(ESV)* הוא הנפח המינימאלי בחדרים וערכו כ *50 ml*.

כמו כן, **נפח פעימה *(SV)*** הוא הנפח של הדם היוצא מהלב בכל פעימה ולכן הוא שווה להפרש בין הנפח המקסימלי למינימלי $SV = EDV - ESV$ (7). וערכו כ *70 ml* . [3]

מודל Windkessel מתאר את מודל הזרימה הפועמת, לפי מודל זה פעימה של לחץ איננה זהה בין כל הווורידים והעורקים בגלל הגמישות השונה של כלי הדם ובשל כך בזמן הדיאסטולה הלחץ יורד בצורה אקספוננצאלית $P \propto e^{-\frac{t}{RC}}$. ניתן לבצע מידול חשמלי בעזרת קבל ונגד במקביל בעזרתו נקבל את המשוואה הדיפרנציאלית: $\dot{P} + \frac{P}{Rc} = \frac{Q_{in}}{c}$ (8). [1]

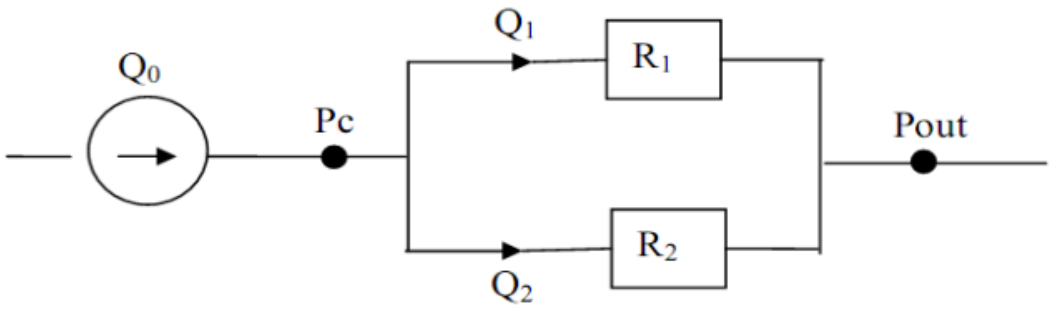
2 תשובות לשאלות הכנה:

2.1 שאלה 4.1:

בכדי להגיע לקשר לינארי בין הספיקה להפרש הלחצים, יש להניח כי מתקיימת זרימת פואזיי.
כלומר: זורם בלתי דחיס (צפיפות קבועה), זרימה תמידית, זרימה למינרית, זרימה במהירות קבועה, זורם ניוטוני, זרימה בצינור קשיח ללא פיצולים. [1]

2.2 שאלה 4.2:

סעיף א'



איור 1: אנלוגיה לזרימה רציפה [5]

הנגדים מחוברים במקביל ולכן המתח עליהם שווה. כמו כן נשתמש בKCL ובקשר הלינארי:

$$(2.2.0)\ Q_0 = Q_1 + Q_2$$

$$(2.2.1)\ P_c - P_{out} = Q_1 \cdot R_1 = Q_2 \cdot R_2 \rightarrow P_c = P_{out} + Q_1 \cdot R_1$$

$$(2.2.2)\ Q_2 = Q_1 \cdot \frac{R_1}{R_2}$$

נציב את 2.2.2 בנוסחה 2.2.0:

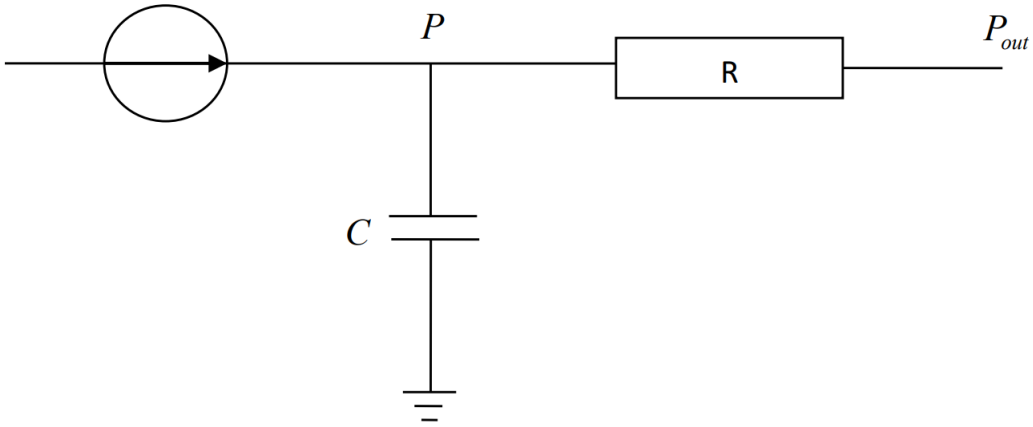
$$(2.2.3)\ Q_0 = Q_1\left(1 + \frac{R_1}{R_2}\right) \rightarrow \boldsymbol{Q_1 = \frac{Q_0 \cdot R_2}{R_1 + R_2}}$$

$$(2.2.4)\ Q_2 = \frac{Q_0 \cdot R_2}{R_1 + R_2} \cdot \frac{R_1}{R_2} \rightarrow \boldsymbol{Q_2 = \frac{Q_0 \cdot R_1}{R_1 + R_2}}$$

נציב את הספיקה שמצאנו במשוואה 2.2.1:

$$\boldsymbol{(2.2.5)\ P_c = P_{out} + \frac{Q_0 \cdot R_2 \cdot R_1}{R_1 + R_2}}$$

סעיף ב':



איור 2 - מודל לזרימה פועמת [5]

נניח:

$$Q_c = c\dot{P}$$

$$\Delta P = Q \cdot R$$

$$P(t=0) = P_{min}$$

נבצע KCL בצומת האמצעית:

$$Q_{in} + Q_c + Q_R \quad \rightarrow \quad Q_{in} = c\dot{P} + \frac{P - P_{out}}{R} \quad \rightarrow \quad \dot{P} + \frac{P}{Rc} = \frac{Q_{in}}{c} + \frac{P_{out}}{Rc}$$

$$\rightarrow P(t) = P_{min}e^{-\frac{t}{Rc}} + e^{-\frac{t}{Rc}} \cdot \left(\frac{Q_{in}}{c} + \frac{P_{out}}{Rc}\right) Rc \cdot \left(e^{\frac{t}{Rc}} - 1\right)$$

$$\rightarrow \quad P(t) = P_{min}e^{-\frac{t}{Rc}} + (Q_{in}R + P_{out}) \cdot (1 - e^{-\frac{t}{Rc}})$$

(-) ייצוג הפרמטרים במערכת ביולוגית:

P – מייצג את הלחץ שפועל על המערכת, Q- מייצג את הספיקה הנפחית, R – התנגדות, מייצג את ההתנגדות ההיקפית של הדם, C- התאמת כלי הדם (היחס בין שינוי הנפח לשינוי הלחץ).

(-) מודל Windkessel מתאר כי בקרוב הלחץ דועך אקספוננציאלית: $P \propto e^{-\frac{t}{RC}}$. ניתן לראות בסעיף ב' כי קיבלנו לחץ דועך בצורה אקספוננציאלית כאשר $\tau = RC$ כפי שמתאר מודל זה. כמו כן, התיאור איננו מדויק אלא רק מתאר קירוב ועל מנת לקבל התאמה גבוהה יותר נצטרך מודל מדויק יותר המתאר את הרכיבים השונים במחזור הדם המרכזי וכך גם הקבועים יהיו מדויקים.

2.3 שאלה 4.3:

א. כפי שציינו ברקע התאורטי, חוק בויל קובע כי $P \cdot V = k\ (const)$, בעוד ש-$C = \frac{\Delta V}{\Delta P}$. לכן נוכל לכתוב:

$$C = \frac{\Delta V}{\Delta P} = \frac{V_2 - V_1}{P_2 - P_1} = \frac{V_2 - V_1}{\frac{k}{V_2} - \frac{k}{V_1}} \quad or \quad \frac{\frac{k}{P_2} - \frac{k}{P_1}}{P_2 - P_1}$$

ב. נמצא את נפח האוויר בתא. ראשית, נפח הגליל הינו:

$$V_{cell} = H \cdot \pi R^2$$

נפח המים בתא הינו:

$$V_{water} = h \cdot \pi R^2$$

נפח האוויר בתא יתקבל ע"י:

$$V_{air} = V_{cell} - V_{water} = \pi R^2 (H - h)$$

ג. חוק בויל מתאר את הקשר בין הלחץ והנפח של גז אידאלי אשר נמצא בטמפ' קבועה וכמות חלקיקים קבועה, גם אם נצליח לשמור על טמפ' קבועה (מאפיינים פיזקליים תקינים) לא נוכל להצליח לחשב את הקבוע $k = P \cdot V$ מכיוון שלא נצליח לשמור במדויק על כמות החלקיקים בניסוי. לכן החישוב שלנו יהיה קירוב בלבד.

2.4 שאלה 4.4:

המושג RPM מתאר סיבובים לדקה (Revolutions per minute), כלומר כמה סיבובים מלאים נעשים בדקה אחת.

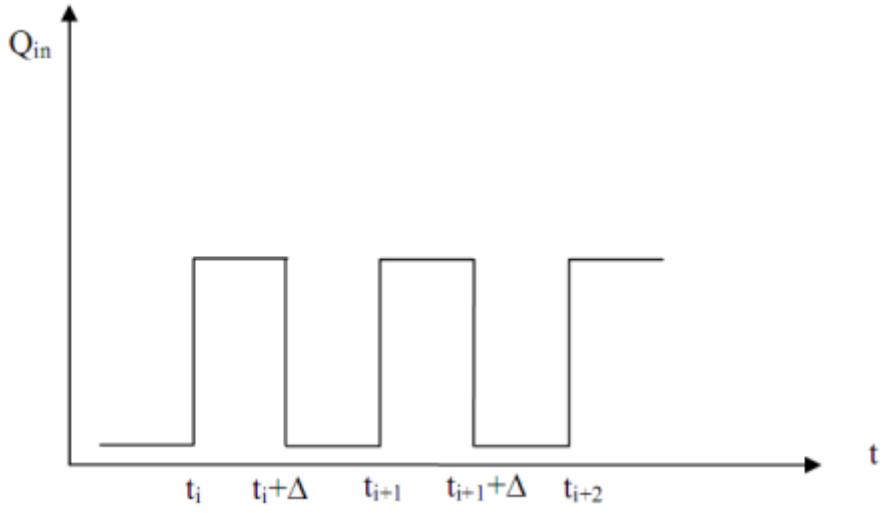
2.5 שאלה 4.5:

ספיקה נפחית מתארת את כמות החומר העוברת דרך שטח מסוים ביחידת זמן. RPM מתאר כמות סיבובים לדקה לכן נוכל לעשות המרת יחידות לכמות סיבובים לשנייה על ידי הכפלה ב60.

$$Q\left[\frac{m^3}{sec}\right] = V\,[m^3] \cdot RPM\left[\frac{1}{min}\right] \cdot 60\left[\frac{min}{sec}\right]$$

$$\boldsymbol{Q = 60 \cdot RPM \cdot V\ \left[\frac{m^3}{sec}\right]}$$

2.6 שאלה 4.6:



איור 3 : כניסה מלבנית

מודל לזרימה פועמת (מודל windkessel) מתואר ברקע התאורטי על ידי המשוואה הדיפרנציאלית :

$\dot{P} + \frac{P}{Rc} = \frac{Q_{in}}{c}$. נמצא פתרון עבור הכניסה המתוארת באיור 3 :

$$\dot{P} + \frac{P}{Rc} = \frac{Q_{in}}{c}$$

$$Q_{in} = \begin{matrix} A & t_j < t < t_{j+\Delta} \ ,j = i, i+1,.. \\ 0 & else \end{matrix}$$

הפתרון עבור המשוואה עם תנאי התחלה $P(t=0) = P_{min}$ הינו :

עבור $Q_{in} = 0$ נקבל משוואה הומוגנית: $\dot{P} + \frac{P}{Rc} = 0$ ולכן הפתרון שלה הינו : $P_{min}e^{-\frac{t}{Rc}}$.

עבור $Q_{in} = A$ נקבל משוואה: $\dot{P} + \frac{P}{Rc} = \frac{A}{c}$ ולכן הפתרון שלה הינו סכום של פתרון כללי: $Be^{-\frac{t}{Rc}}$

ופתרון פרטי : $A \cdot R$ ולכן : $Be^{-\frac{t}{Rc}} + A \cdot R$. נציב תנאי התחלה :

$$P(t=0) = P_{min} = B + A \cdot R \quad ,B = P_{min} - A \cdot R$$

לסיכום :

$$\boldsymbol{P(t) = \begin{matrix} (P_{min} - A \cdot R)e^{-\frac{t}{Rc}} + A \cdot R & t_j < t < t_{j+\Delta} \ ,j = i, i+1,.. \\ P_{min}e^{-\frac{t}{Rc}} & else \end{matrix}}$$

2.7 שאלה 4.7:

בכדי לפתור את השאלה השתמשנו בפונקציה lvm_import אשר מצאנו בMathWorks את פונקציה

זו נוסיף להגשה במודל.[6]

קוד ראשי:

```
[Mean,Data] = ReadData('File_Q6.lvm',195,'End',2);
```

קוד הפונקציה:

```
function [Mean,Data]=ReadData(File,StrInd,EndInd,NumCol)
Data_File=lvm_import(File);

if strcmp(EndInd,'End')%the case where End reading is END
    EndInd = length(Data_File.Segment1.data(:,1));
    Data = Data_File.Segment1.data(StrInd:EndInd,1:NumCol);
    Mean = mean(Data);
else
    Data = Data_File.Segment1.data(StrInd:EndInd,1:NumCol); %The case End
reading
    Mean = mean(Data);                                      %is a finite
number
end
```

3 בביליוגרפיה

[1] C. G. Caro, Ed., The mechanics of the circulation, 2nd ed. Cambridge, UK ; New York: Cambridge University Press, 2012.

[2] F. M. White, Fluid mechanics. 2017.

[3] A. Ostadfar, Biofluid mechanics: principles and applications. Amsterdam ; Boston: Elsevier/Academic Press, 2016.

[4] J. R. Welty, Fundamentals of momentum, heat and mass transfer, 6th edition. Hoboken, NJ: Wiley, 2015.

[5] "2022 זרימה.pdf."

[6] "LVM file import." https://www.mathworks.com/matlabcentral/fileexchange/19913-lvm-file-import (accessed Apr. 02, 2022).

4 נספחים

קוד פונקציה lvm_import:

4.1 Contents

```
function data = lvm_import(filename,verbose)
%LVM_IMPORT Imports data from a LabView LVM file
% DATA = LVM_IMPORT(FILENAME,VERBOSE) returns the data from a LVM (.lvm)
%  ASCII text file created by LabView.
%
% FILENAME    The name of the .lvm file, with or without ".lvm" extension
%
% VERBOSE     How many messages to display. Default is 1 (few messages),
%              0 = silent, 2 = display file information and all messages
%
% DATA        The data found in the LVM file. DATA is a structure with
%              fields corresponding to the Segments in the file (see below)
%              and LVM file header information.
%
%
% This function imports data from a text-formatted LabView Measurement File
%  (LVM, extension ".lvm") into MATLAB. A LVM file can have multiple
%  Segments, so that multiple measurements can be combined in a single
%  file. The output variable DATA is a structure with fields named
%  'Segment1', 'Segment2', etc. Each Segment field is a structure with
%  details about the data in the Segment and the actual data in the field
%  named 'data'. The column labels and units are stored as cell arrays that
%  correspond to the columns in the array of data.
% The size of the data array depends on the type of x-axis data that is
%  stored in the LVM file and the number of channels (num_channels).
%  There are three cases:
%  1) No x-data is included in the file ('No')
%   The data array will have num_channels columns (one column per channel
%   of data).
%  2) One column of x-data is included in the file ('One')
%   The first column of the data array will be the x-values, and the data
%   array will have num_channels+1 columns.
%  3) Each channel has its own x-data ('Multi')
%   Each channel has two columns, one for x-values, and one for data. The
%   data array will have num_channels*2 columns, with the x-values and
%   corresponding data in alternating columns. For example, in a Segment
%   with 4 channels, columns 1,3,5,7 will be the x-values for the data in
%   columns 2,4,6,8.
%
% Note: because MATLAB only works with a "." decimal separator, importing
%  large LVM files that use a "," (or other character) will be noticeably
%  slower. Use a "." decimal separator to avoid this issue.
%
% The LVM file specification is available at:
%   http://zone.ni.com/devzone/cda/tut/p/id/4139
%
%
% Example:
%
%  Use the following command to read in the data from a file containing two
%   Segments:
%
% >> d=lvm_import('testfile.lvm');
%
% Importing testfile.lvm:
%
% Import complete. 2 Segments found.
%
```

```
% >> d
% d =
%       X_Columns: 'One'
%            user: 'hopcroft'
%     Description: 'Pressure, Flowrate, Heat, Power, Analog Voltage, Pump
on, Temp'
%            date: '2008/03/26'
%            time: '12:18:02.156616'
%           clock: [2008 3 26 12 18 2.156616]
%        Segment1: [1x1 struct]
%        Segment2: [1x1 struct]
%
% >> d.Segment1
% ans =
%            Notes: 'Some notes regarding this data set'
%     num_channels: 8
%          y_units: {8x1 cell}
%          x_units: {8x1 cell}
%               X0: [8x1 double]
%          Delta_X: [8x1 double]
%    column_labels: {9x1 cell}
%             data: [211x9 double]
%          Comment: 'This data rulz'
%
% >> d.Segment1.column_labels{2}
% ans =
% Thermocouple1
%
% >> plot(d.Segment1.data(:,1),d.Segment1.data(:,2));
% >> xlabel(d.Segment1.column_labels{1});
% >> ylabel(d.Segment1.column_labels{2});
%
%
%
% M.A. Hopcroft
%      < mhopeng at gmail.com >
%
% MH Sep2017
% v3.12 fix bug for importing data-only files
%                (thanks to Enrique Alvarez for bug reporting)
% MH Mar2017
% v3.1  use cellfun to vectorize processing of comma-delimited data
%       (thanks to Victor for suggestion)
% v3.0  use correct test for 'tab'
% MH Aug2016
% v3.0  (BETA) fixes for files that use comma as delimiter
%       improved robustness for files with missing columns
% MH Sep2013
% v2.2  fixes for case of comma separator in multi-segment files
%       use cell2mat for performance improvement
%       (thanks to <die-kenny@t-online.de> for bug report and testing)
% MH May2012
% v2.1  handle "no separator" bug
%       (thanks to <adnan.cheema@gmail.com> for bug report and testing)
%       code & comments cleanup
%       remove extraneous column labels (X_Value for "No X" files; Comment)
%       clean up verbose output
%       change some field names to NI names ("Delta_X","X_Columns","Date")
% MH Mar2012
% v2.0  fix "string bug" related to comma-separated decimals
%       handle multiple Special Headers correctly
%       fix help comments
%       increment version number to match LabView LVM writer
% MH Sep2011
% v1.3  handles LVM Writer version 2.0 (files with decimal separator)
%       Note: if you want to work with older files with a non-"." decimal
%       separator character, change the value of "data.Decimal_Separator"
% MH Sep2010
```

```
% v1.2  bugfixes for "Special" header in LVM files.
%        (Thanks to <bobbyjoe23928@gmail.com> for suggestions)
% MH Apr2010
% v1.1  use case-insensitive comparisons to maintain compatibility with
%        NI LVM Writer version 1.00
%
% MH MAY2009
% v1.02 Add filename input
% MH SEP2008
% v1.01 Fix comments, add Cells
% v1.00 Handle all three possibilities for X-columns (No,One,Multi)
%       Handle LVM files with no header
% MH AUG2008
% v0.92 extracts Comment for each Segment
% MH APR2008
% v0.9  initial version
%
%#ok<*ASGLU>
% message level
if nargin < 2, verbose = 1; end % use 1 for release and 2 for BETA
if verbose >= 1, fprintf(1,'\nlvm_import v3.1\n'); end
% ask for filename if not provided already
if nargin < 1
    filename=input(' Enter the name of the .lvm file: ','s');
    fprintf(1,'\n');
end
```

4.2 Open the data file

```
fid=fopen(filename);

if fid ~= -1, % then file exists
    fclose(fid);
else
    filename=strcat(filename,'.lvm');
    fid=fopen(filename);
    if fid ~= -1, % then file exists
        fclose(fid);
    else
        error(['File not found in current directory! (' pwd ')']);
    end
end
% open the validated file
fid=fopen(filename);
if verbose >= 1, fprintf(1,' Importing "%s"\n\n',filename); end
% is it really a LVM file?
linein=fgetl(fid);
if verbose >= 2, fprintf(1,'%s\n',linein); end
% Some LabView routines create an LVM file with no header; just a text file
%  with columns of numbers. We can try to import this kind of data.
if isempty(strfind(linein,'LabVIEW'))
    try
        data.Segment1.data = dlmread(filename);
        if verbose >= 1, fprintf(1,'This file appears to be an LVM file with
no header.\n'); end
        if verbose >= 1, fprintf(1,'Data was copied, but no other
information is available.\n'); end
        return
    catch fileEx
        error('This does not appear to be a text-format LVM file (no
recognizeable header or data).');
    end
end
```

4.3 Process file header

The file header contains several fields with useful information default values

```
data.Decimal_Separator = '.';
text_delimiter={',',' ','\t'};
data.X_Columns='One';
% File header contains date, time, etc.
% Also the file delimiter and decimal separator (LVM v2.0)
if verbose >= 2, fprintf(1,' File Header Contents:\n\n'); end
while 1

    % get a line from the file
    linein=fgetl(fid);
    % handle spurious carriage returns
    if isempty(linein), linein=fgetl(fid); end
    if verbose >= 3, fprintf(1,'%s\n',linein); end
    % what is the tag for this line?
    t_in = textscan(linein,'%s','Delimiter',text_delimiter);
    if isempty(t_in{1}{1})
        tag='notag';
    else
        tag = t_in{1}{1};
    end
    % exit when we reach the end of the header
    if strfind(tag,'***End_of_Header***')
        if verbose >= 2, fprintf(1,'\n'); end
        break
    end

    % get the value corresponding to the tag
%     if ~strcmp(tag,'notag')
%         v_in = textscan(linein,'%*s
%s','delimiter','\t','whitespace','','MultipleDelimsAsOne', 1);
        if size(t_in{1},1)>1 % only process a tag if it has a value
%             val = v_in{1}{1};
            val = t_in{1}{2};

            switch tag
                case 'Date'
                    data.Date = val;
                case 'Time'
                    data.Time = val;
                case 'Operator'
                    data.user = val;
                case 'Description'
                    data.Description = val;
                case 'Project'
                    data.Project = val;
                case 'Separator'
                    % v3 separator sanity check
                    if strcmpi(val,'Tab')
                        text_delimiter='\t';
                        if strfind(linein,',')
                            fprintf(1,'ERROR: File header reports "Tab" but
uses ",". Check the file and correct if necessary.\n');
                            return
                        end
                    elseif strcmpi(val,'Comma') || strcmpi(val,',')
                        text_delimiter=',';
                        if strfind(linein,sprintf('\t'))
                            fprintf(1,'ERROR: File header reports "Comma"
but uses "tab". Check the file and correct if necessary.\n');
                            return
                        end
                    end
```

```
                case 'X_Columns'
                    data.X_Columns = val;
                case 'Decimal_Separator'
                    data.Decimal_Separator = val;
            end
            if verbose >= 2, fprintf(1,'%s: %s\n',tag,val); end
        end
%     end

end
% create matlab-formatted date vector
if isfield(data,'time') && isfield(data,'date')
    dt = textscan(data.Date,'%d','Delimiter','/');
    tm = textscan(data.Time,'%d','Delimiter',':');
    if length(tm{1})==3
        data.clock=[dt{1}(1) dt{1}(2) dt{1}(3) tm{1}(1) tm{1}(2) tm{1}(3)];
    elseif length(tm{1})==2
        data.clock=[dt{1}(1) dt{1}(2) dt{1}(3) tm{1}(1) tm{1}(2) 0];
    else
        data.clock=[dt{1}(1) dt{1}(2) dt{1}(3) 0 0 0];
    end
end
if verbose >= 3, fprintf(1,' Text delimiter is "%s":\n\n',text_delimiter);
end
```

4.4 Process segments

process data segments in a loop until finished

```
segnum = 1;

val=[]; tag=[]; %#ok<NASGU>
while 1
    %segnum = segnum +1;
    fieldnm = ['Segment' num2str(segnum)];
```

4.5 - Segment header

```
    if verbose >= 1, fprintf(1,' Segment %d:\n\n',segnum); end
    % loop to read segment header
    while 1
        % get a line from the file
        linein=fgetl(fid);
        % handle spurious carriage returns/blank lines/end of file
        while isempty(linein), linein=fgetl(fid); end
        if feof(fid), break; end
        if verbose >= 3, fprintf(1,'%s\n',linein); end

        % Ignore "special segments"
        % "special segments" can hold other types of data. The type tag is
        % the first line after the Start tag. As of version 2.0,
        % LabView defines three types:
        %  Binary_Data
        %  Packet_Notes
        %  Wfm_Sclr_Meas
        % In theory, users can define their own types as well. LVM_IMPORT
        %  ignores any "special segments" it finds.
        % If special segments are handled in future versions, recommend
        %  moving the handler outside the segment read loop.
        if strfind(linein,'***Start_Special***')
            special_seg = 1;
            while special_seg
                while 1 % process lines until we find the end of the special
segment
```

```
                    % get a line from the file
                    linein=fgetl(fid);
                    % handle spurious carriage returns
                    if isempty(linein), linein=fgetl(fid); end
                    % test for end of file
                    if linein==-1, break; end
                    if verbose >= 2, fprintf(1,'%s\n',linein); end
                    if strfind(linein,'***End_Special***')
                        if verbose >= 2, fprintf(1,'\n'); end
                        break
                    end
                end

                % get the next line and proceed with file
                %  (there may be additional Special Segments)
                linein=fgetl(fid);
                % handle spurious carriage returns/blank lines/end of file
                while isempty(linein), linein=fgetl(fid); end
                if feof(fid), break; end
                if isempty(strfind(linein,'***Start_Special***'))
                    special_seg = 0;
                    if verbose >= 1, fprintf(1,' [Special Segment
ignored]\n\n'); end
                end
            end
        end % end special segment handler


        % what is the tag for this line?
        t_in = textscan(linein,'%s','Delimiter',text_delimiter);
        if isempty(t_in{1}{1})
            tag='notag';
        else
            tag = t_in{1}{1};
            %disp(t_in{1})
        end
        if verbose >= 3, fprintf(1,'%s\n',linein); end
        % exit when we reach the end of the header
        if strfind(tag,'***End_of_Header***')
            if verbose >= 3, fprintf(1,'\n'); end
            break
        end

        % get the value corresponding to the tag
        % v3 assignments use dynamic field names
        if size(t_in{1},1)>1 % only process a tag if it has a value
            switch tag
                case 'Notes'
    %                %d_in = textscan(linein,'%*s
%s','delimiter','\t','whitespace','');
    %                d_in = linein;
                    data.(fieldnm).Notes = t_in{1}{2:end};
                case 'Test_Name'
    %                %d_in = textscan(linein,'%*s
%s','delimiter','\t','whitespace','');
    %                d_in = linein;
                    data.(fieldnm).Test_Name = t_in{1}{2:end};  %d_in{1}{1};
                case 'Channels'
    %                numchan =
textscan(linein,sprintf('%%*s%s%%d',text_delimiter),1)
    %                data.(fieldnm).num_channels = numchan{1};
                    data.(fieldnm).num_channels = str2num(t_in{1}{2});
                case 'Samples'
    %                numsamp =
textscan(linein,'%s','delimiter',text_delimiter);
    %                numsamp1 = numsamp{1};
                    numsamp1 = t_in{1}(2:end);
    %                numsamp1(1)=[]; % remove tag "Samples"
```

```
                    num_samples=[];
                    for k=1:length(numsamp1)
                        num_samples = [num_samples
sscanf(numsamp1{k},'%f')]; %#ok<AGROW>
                    end
                    %numsamp2=str2num(cell2mat(numsamp1));
%#ok<ST2NM>
                    data.(fieldnm).num_samples = num_samples;
                case 'Y_Unit_Label'
    %                 Y_units =
textscan(linein,'%s','delimiter',text_delimiter);
    %                 data.(fieldnm).y_units=Y_units{1}';
                    data.(fieldnm).y_units=t_in{1}';
                    data.(fieldnm).y_units(1)=[]; % remove tag
                case 'Y_Dimension'
    %                 Y_Dim =
textscan(linein,'%s','delimiter',text_delimiter);
    %                 data.(fieldnm).y_type=Y_Dim{1}';
                    data.(fieldnm).y_type=t_in{1}';
                    data.(fieldnm).y_type(1)=[]; % remove tag
                case 'X_Unit_Label'
    %                 X_units =
textscan(linein,'%s','delimiter',text_delimiter);
    %                 data.(fieldnm).x_units=X_units{1}';
                    data.(fieldnm).x_units=t_in{1}';
                    data.(fieldnm).x_units(1)=[];
                case 'X_Dimension'
    %                 X_Dim =
textscan(linein,'%s','delimiter',text_delimiter);
    %                 data.(fieldnm).x_type=X_Dim{1}';
                    data.(fieldnm).x_type=t_in{1}';
                    data.(fieldnm).x_type(1)=[]; % remove tag
                case 'X0'
                    %[Xnought, val]=strtok(linein);
                    val=t_in{1}(2:end);
                    if ~strcmp(data.Decimal_Separator,'.')
                        val = strrep(val,data.Decimal_Separator,'.');
                    end
                    X0=[];
                    for k=1:length(val)
                        X0 = [X0 sscanf(val{k},'%e')]; %#ok<AGROW>
                    end
                    data.(fieldnm).X0 = X0;
                    %data.(fieldnm).X0 = textscan(val,'%e');
                case 'Delta_X' %,
                    %[Delta_X, val]=strtok(linein);
                    val=t_in{1}(2:end);
                    if ~strcmp(data.Decimal_Separator,'.')
                        val = strrep(val,data.Decimal_Separator,'.');
                    end
                    Delta_X=[];
                    for k=1:length(val)
                        Delta_X = [Delta_X sscanf(val{k},'%e')]; %#ok<AGROW>
                    end
                    data.(fieldnm).Delta_X = Delta_X;
            end
        end

    end % end reading segment header loop
    % Done reading segment header

    % after each segment header is the row of column labels
    linein=fgetl(fid);
    Y_labels = textscan(linein,'%s','delimiter',text_delimiter);
    data.(fieldnm).column_labels=Y_labels{1}';
    % The X-column always exists, even if it is empty. Remove if not used.
    if strcmpi(data.X_Columns,'No')
        data.(fieldnm).column_labels(1)=[];
```

```
    end
    % remove empty entries and "Comment" label
    if any(strcmpi(data.(fieldnm).column_labels,'Comment'))

data.(fieldnm).column_labels=data.(fieldnm).column_labels(1:find(strcmpi(dat
a.(fieldnm).column_labels,'Comment'))-1);
    end
    % display column labels
    if verbose >= 1
        fprintf(1,' %d Data Columns:\n |
',length(data.(fieldnm).column_labels));
        for i=1:length(data.(fieldnm).column_labels)
            fprintf(1,'%s | ',data.(fieldnm).column_labels{i});
        end
        fprintf(1,'\n\n');
    end
```

4.6 - Segment Data

Create a format string for textscan depending on the number/type of channels. If there are additional segments, texscan will quit when it comes to a text line which does not fit the format, and the loop will repeat.

```
    if verbose >= 1, fprintf(1,' Importing data from Segment %d...',segnum);
end

    % How many data columns do we have? (including X data)
    switch data.X_Columns
        case 'No'
            % an empty X column exists in the file
            numdatacols = data.(fieldnm).num_channels+1;
            xColPlural='no X-Columns';
        case 'One'
            numdatacols = data.(fieldnm).num_channels+1;
            xColPlural='one X-Column';
        case 'Multi'
            numdatacols = data.(fieldnm).num_channels*2;
            xColPlural='multiple X-Columns';
    end

    % handle case of not using periods (aka "dot" or ".") for decimal point
separators
    %  (LVM version 2.0+)
    if ~strcmp(data.Decimal_Separator,'.')
        if verbose >= 2, fprintf(1,'\n  (using decimal separator
"%s")\n',data.Decimal_Separator); end

        % create a format string for reading data as numbers
        fs = '%s'; for i=2:numdatacols, fs = [fs ' %s']; end
%#ok<AGROW>
        % add one more column for the comment field
        fs = [fs ' %s'];
%#ok<AGROW>
        % v3.1 - use cellfun to process data
        % Read columns as strings
        rawdata = textscan(fid,fs,'delimiter',text_delimiter);
        % Convert ',' decimal separator to '.' decimal separator
        rawdata = cellfun(@(x) strrep(x,data.Decimal_Separator,'.'),
rawdata, 'UniformOutput', false);
                % save first row comment as The Comment for this segment
        data.(fieldnm).Comment = rawdata{size(rawdata,2)}{1};
        % Transform strings back to numbers
        rawdata = cellfun(@(x) str2double(x), rawdata, 'UniformOutput',
false);

    % else is the typical case, with a '.' decimal separator
```

```
    else
        % create a format string for reading data as numbers
        fs = '%f'; for i=2:numdatacols, fs = [fs ' %f']; end
%#ok<AGROW>
        % add one more column for the comment field
        fs = [fs ' %s'];
%#ok<AGROW>
        % read the data from file
        rawdata = textscan(fid,fs,'delimiter',text_delimiter);
        % save first row comment as The Comment for this segment
        data.(fieldnm).Comment = rawdata{size(rawdata,2)}{1};
    end

    % v2.2 use cell2mat here instead of a loop for better performance
    % consolidate data into a simple array, ignore comments
    data.(fieldnm).data=cell2mat(rawdata(:,1:numdatacols));

    % If we have a "No X data" file, remove the first column (it is
empty/NaN)
    if strcmpi(data.X_Columns,'No')
        data.(fieldnm).data=data.(fieldnm).data(:,2:end);
    end

    if verbose >= 1, fprintf(1,' complete (%g data points
(rows)).\n\n',length(data.(fieldnm).data)); end

    % test for end of file
    if feof(fid)
        if verbose >= 2, fprintf(1,' [End of File]\n\n'); end
        break;
    else
        segnum = segnum+1;
    end
end % end process segment
if verbose >= 1
    if segnum > 1, segplural='Segments';
    else segplural='Segment'; end
    fprintf(1,'\n Import complete. File has %s and %d Data
%s.\n\n',xColPlural,segnum,segplural);
end
% close the file
fclose(fid);
return
```

Published with MATLAB® R2019a