Table of Contents

```
function v0Vals = M3_Algorithm_001_30(enzNum);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ENGR 132
% Program Description
% This program will use a method similar to Algorithm 1 of M2 in order
% to calculate v0 constants for a given enzyme at 10 different
 substrate
% concentrations.
%
% Note: changed or depreciated code is commented as such. New or
 unmodified
% code will remain uncommented.
%
% Function Call
% v0Vals = M3_Algorithm_001_30(enzNum);
%
% Input Arguments
% enzNum - the number of the enzyme being examined as integer 1 - 5.
 Other
% values will throw an error.
%
% Output Arguments
% v0vals - an array containing all 10 v0 values for a given enzyme at
 each
% substrate concentration. (Units: uM/minute)
%
% Assignment Information
%   Assignment:     Milestone 3, Algorithm
%   Team member:    Surya Manikhandan, smanikha@purdue.edu
%                   Julius Mesa, jmesa@purdue.edu
%                   Alex Norkus, anorkus@purdue.edu
%                   Luming Lin, lin971@purdue.edu
%   Team ID:        001-30
%   Academic Integrity:
%     [] We worked with one or more peers but our collaboration
%        maintained academic integrity.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

INPUT VALIDATION

```
% --- ALL CODE BELOW IN THIS SECTION IS DEPRECIATED ---
% CHANGE: Input validation has been removed as Algorithm is not meant
 to be used
% by other users and is only implemented in the executive function.
 Since
% the values for enzyme number parameter is set properly, we have no
 need
% for input validation

% Category 2 - By removing the unnecessary input validation step, the
% program runtime is shortened due to the reduction of CPU cycles
 spent on
% comparisons. However, this means we will need to be more
% careful and ensure proper parameter passes when using this function

% inval = 0; % this flag value will hold whether or not any of the
 params are invalid

% if((floor(enzNum) ~= enzNum) | (enzNum > 5) | (enzNum < 0)) % check
 if width is a positive integer
%    fprintf(2, "ERROR: enzNum parameter must be an integer between
 1-5 inclusive\n");
%    inval = 1; % toggle flag
% end

% if(inval) % quit if any parameter is invalid
%    return;
% end
```

INITIALIZATION

```
fileName = "Data_nextGen_KEtesting_allresults.csv"; % the name of the
 datafile

% import data vals crucial to the calculation
timeAxis = readmatrix(fileName, "range", "A4:A");
concentrationData = readmatrix(fileName, "range", "B5:CW7488");


% compute the starting column of the data given the enzyme number
origColumn = 1 + (20 * (enzNum - 1));
dupeColumn = 11 + (20 * (enzNum - 1));
```

CALCULATIONS OF v0 VALUES

```
passWidth = 17; % width for the moving average smoothing function

% v0ValsTest = []; % initialize v0s to void array to begin (for orig
 test)
% v0ValsDupe = []; % initialize v0s to void array to begin (for dupe
 test)

% [Category 2 Change - Previously, the array was not preallocated with
 zeros, which causes
% significant performance losses when adding elaments to the array.
 Therefore,
% we are preallocating the arays with zeros to increase performance]
v0ValsTest = zeros();
v0ValsDupe = zeros();

for product = 0:9

    % Code in next 5 lines will isolate test and dupe data without NAN
 vals
    % NOTE: BASED OFF CODE FOUND IN https://bit.ly/2OyqAQj
    testData = concentrationData(:, origColumn + product)';
    testData = testData(~isnan(testData))';

    dupeData = concentrationData(:, dupeColumn + product)';
    dupeData = dupeData(~isnan(dupeData))';

    % Find v0 values for original test and duplicate seperately. Then,
    % average those v0 values to get the final result.

    % __________ Original test __________
    [timeArray, dataArray] = M3_Smooth_001_30(testData, timeAxis,
 passWidth);

    % Category 1 Change - In M2, we used the first two points in the
    % reaction velocity plot and used the rise/run equation to finf
 the
    % vmax. However, we found that this does NOT create the best
 tangent
    % line. Instead of only considering the first two points, which
 leaves
    % room for error, we found that finding the linear regression of
 the
    % first 3 points created a much better tangent line and therefore
    % better vmax values. Comprehensive explanation located in M4
 docs.
    % v0 = (dataArray(2) - dataArray(1))/ (timeArray(2) -
 timeArray(1));
```

```
    timeArray = timeArray(1:3);
    dataArray = dataArray(1:3);
    v0 = timeArray(:) \ dataArray(:); % find slope, which is equal to
 v0 value

    % [Category 2 Change - Previously, we added the element to the end
 of the array using
    % concatenation, which is not effecient. Now using indexes instead
 for direct adding of v0 values where needed]
    %v0ValsTest = [v0ValsTest, v0];
    v0ValsTest(product + 1) = v0; % General change - fit new array
 scheme (use direct assignment)

    % __________ Duplicate test __________
    % NOTE: changes implemented here are identical to original test
 changes
    [timeArray, dataArray] = M3_Smooth_001_30(dupeData, timeAxis,
 passWidth);

    % [OLD] use rise/run to find the first slope value which is our v0
 val
    %v0 = (dataArray(2) - dataArray(1))/ (timeArray(2) -
 timeArray(1));

    timeArray = timeArray(1:3);
    dataArray = dataArray(1:3);
    v0 = timeArray(:) \ dataArray(:); % find slope, which is equal to
 v0 value

    %v0ValsDupe = [v0ValsDupe, v0];
    v0ValsDupe(product + 1) = v0; % fit new array scheme (use direct
 assignment)

end
```

FORMATTED TEXT/FIGURE DISPLAYS

COMMAND WINDOW OUTPUT

```
% final v0 array is the average of the test and dulplicate data
v0Vals = (v0ValsTest + v0ValsDupe) ./ 2;
```

ACADEMIC INTEGRITY STATEMENT

We have not used source code obtained from any other unauthorized source, either modified or unmodified. Neither have we provided access to my code to another. The function we are submitting is our own original work.

```
end
```

Published with MATLAB® R2019a