Table of Contents


```
function [truncatedTime, smoothedData] = M3_Smooth_001_30(dataArray,
 timeArray, segmentWidth);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ENGR 132
% Program Description
% This user-defined functin will use the moving average method to
 smooth
% an array of data and return the smoothed array back to the calling
% function. Averages are done sequentially with the width given by
 user.
%
% Note: changed or depreciated code is commented as such. New or
 unmodified
% code will remain uncommented.
%
% Function Call
% [truncatedTime, smoothedData] = M3_Smooth_001_30(dataArray,
 timeArray, passWidth);
%
% Input Arguments
% dataArray -    a one dimensional array containing the data for the
 product
%                conc. data of an enzyme at a given substrate conc.
 value.
% timeArray -    the time data array. Will be returned at an
 appropriate length.
% segmentWidth - the width the function will use to calculate the
 moving
%                average.
%
% Output Arguments
% truncatedTime - the array of the time elements corresponding to
 smoothed
```

```
%                 data values.
% smoothedData - the array of smoothed data determined through the
 moving
%               average method.
%
% Assignment Information
%   Assignment:     Milestone 3
%   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 Smooth is not meant to
 be used
% by other users and is only implemented in the algorithm. Since
% the values for width parameter is set properly, we have no need for
 this.

% 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

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

% if((floor(segmentWidth) ~= segmentWidth) | (segmentWidth <= 0)) %
 check if width is a positive integer
%    fprintf(2, "ERROR: passWidth parameter must be an integer greater
 than zero\n");
%    inval = 1; % toggle flag
% end

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

INITIALIZATION

CALCULATIONS

```
% general change - parameters do not need to be initialized
% void anymore as we will do that later with the zeros function
% smoothedData = [];
% truncatedTime = [];

% [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]
smoothedData = zeros();
truncatedTime = zeros();

arrayindex = 1; % will keep track of the array index

for index = 1:ceil(segmentWidth/2):(length(dataArray) - segmentWidth)

    % isolate segment of width from dataset
    dataSegment = dataArray(index : index+segmentWidth);
    timeSegment = timeArray(index: index+segmentWidth);

    % sum all elements in the segment
    sumDataSegment = sum(dataSegment);
    sunTimeSegment = sum(timeSegment);

    % take average of elements in that array
    avgDataSegment = sumDataSegment / (segmentWidth + 1);
    avgTimeSegment = sunTimeSegment / (segmentWidth + 1);

    % [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 elements]

    % add the averaged value to the smoothed array
    %smoothedData = [smoothedData, avgDataSegment];
    %truncatedTime = [truncatedTime, avgTimeSegment];

    % general change - fit new array scheme (direct assignment)
    smoothedData(arrayindex) = avgDataSegment;
    truncatedTime(arrayindex) = avgTimeSegment;
```

```
        arrayindex = arrayindex + 1; % increment the index of the array
    end
```

FORMATTED TEXT/FIGURE DISPLAYS

COMMAND WINDOW OUTPUT

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