Table of Contents

USDataClass: Data class for computation of echo decorrelation

```
classdef USDataClass < handle
    % USDataClass: Data class for computation of echo decorrelation
    %   each set of data has is an object of type USDataClass
    %   contains the data parameters, raw spherical data, cartesian
 scan converted data, ibs,
    %   and decorrelation
    % Author: Peter Grimm 12/21/2019
    properties
        % data properties
        rawData;           % raw spherical volume data for one data
 set export
        rawData_cart;      % scan converted cartesian data from
 spherical data
        ibs;               % Integrated backscatter for every
 (cartesian) volume
        decorr;            % decorr (cartesian) between volume at t
 and t+tau
        autocorr01;        % autocorr between volume at t and t+tau
        decorrAvg;         % average of decorrelation in entire data
 set, either ensemble or running
        time;
        % bounds
        ROIBounds;         % bounds of region of interest [xMin xMax
 yMin yMax zMin zMax]
        ROIBounds_spherical; %TODO, minimum bounds in spherical
 coordinates

        % bounded data properties
        rawData_cart_ROI;
        ibs_ROI;
        decorr_ROI;
        autocorr01_ROI;
        decorrAvg_ROI;
        % parameters
        %The following would ideally be part of InfoFile in the future
        InfoFile;          % Info file provided by siemens SC2000
 scanner
```

```
        rmax,rmin,thetamax,thetamin,phimax,phimin; % bounds in cm of
 the spherical data
        xMin,xMax,yMin,yMax,zMin,zMax; % bounds in cm of cartesian
 data
        x_range,y_range,z_range; % range in cartesian plane
        windowSigma;       % sigma of gaussian smoothing kernel
        dPhi;              % angular difference between successive phi
        dTheta;            % angular difference between successive
 theta
        dr;                % distance in cm in the radius
 direction(cm/pixel)
        dx;                % distance in cm in the x direction (cm/
pixel)
        dy;                % distance in cm in the x direction (cm/
pixel)
        dz;                 % distance in cm in the x direction (cm/
pixel)
        interFrameTime;    % time between volume recordings (cm/pixel)
        cartScalingFactor; % Factor to scale cartesian distances by
 (dx/dr)
                           % e.g Given dr, to find dx take
                           % dr*cartScalingFactor = dx
                           % reduces resolution by a factor of
                           % cartScalingFactor for faster scan
 conversion
        rawData_cart_slicemethod;
        ibs_slicemethod;               % Integrated backscatter for
 every (cartesian) volume
        decorr_slicemethod;            % decorr (cartesian) between
 volume at t and t+tau
        autocorr01_slicemethod;        % autocorr between volume at t
 and t+tau
        decorrAvg_slicemethod;         % average of decorrelation in
 entire data set, either ensemble or running


    end

    methods
```

Constructor method

```
        function obj =
USDataClass(thisRawData,startTime,thisInfoFile,thisrmin,thisrmax,thisthetamin,thi
            obj.rawData = thisRawData;
            obj.InfoFile = thisInfoFile;
            obj.rmax = thisrmax;
            obj.rmin = thisrmin;
            obj.thetamax = thisthetamax;
            obj.thetamin = thisthetamin;
            obj.phimax = thisphimax;
            obj.phimin = thisphimin;
            obj.cartScalingFactor = thiscartScalingFactor;
```

```
            obj.windowSigma = thiswindowSigma;
            obj.interFrameTime = thisinterFrameTime;
            obj.time = startTime;
        end
```

Scan Conversion Method

Display functions

```
        function displayRawData_cart(obj)
            imagesc(squeeze(abs(obj.rawData_cart(:,:,30,1))))
        end
        function displayDecorr(obj)
            imagesc(squeeze(abs(obj.decorr(:,:,20,1))))
        end
```

compute ensemble average

```
        function computeDecorrelationAverage(obj,method)
            if strcmp(method,'ensemble') == 1
                obj.decorrAvg = sum(obj.decorr,4)/size(obj.decorr,4);

            elseif strcmp(metod,'running') == 1

            else
                error('please enter proper method name (ensemble or
 running)');
            end
        end
```

compute ensemble average in ROI

```
        function computeDecorrelationAverage_ROI(obj,method)
            if strcmp(method,'ensemble') == 1
                obj.decorrAvg =
 sum(subvolume(obj.decorr_slicemethod,obj.ROIBounds),4)/
size(obj.decorr_slicemethod,4);

            elseif strcmp(method,'running') == 1

            else
                error('please enter proper method name (ensemble or
 running)');
            end
        end
```

calculate minimum spherical coordinate bounds that encapsulates the cartesian ROI bounds

potentially useful for speeding up computation given small enough sigma to make windowing effect irrelevant

```
        function computeMinROIBoundsSpherical(obj)
            [minX maxX minY maxY minZ maxZ] = obj.ROIBounds;
            minTheta = arctan(minY/minX);
            maxTheta = arctan(minY/maxX);
            minPhi = arctan(sqrt(minX^2 + minY^2)/minZ);
            %todo
            maxPhi = minPhi;
            %minR not correct TODO
            minR = sqrt(minX^2 + minY^2 + minZ^2);
            maxR = sqrt(maxX^2 + maxY^2 + maxZ^2);
            obj.ROIBounds_spherical = [minR,
 maxR,minTheta,maxTheta,minPhi,maxPhi];
        end
        function setROI(obj,thisROIBounds)
            if(isempty(obj.decorr))
               error('execute computedecorr3d before getting
 decorrelation within bounds')
            end

            obj.ROIBounds = thisROIBounds;

            % convert cm to pixels
            minX = floor(obj.ROIBounds(1)/obj.dx)+1;
            maxX = floor(obj.ROIBounds(2)/obj.dx)+1;
            minY = floor(obj.ROIBounds(3)/obj.dy)+1;
            maxY = floor(obj.ROIBounds(4)/obj.dy)+1;
            minZ = floor(obj.ROIBounds(5)/obj.dz)+1;
            maxZ = floor(obj.ROIBounds(6)/obj.dz)+1;
            obj.rawData_cart_ROI =
 obj.rawData_cart(minX:maxX,minY:maxY,minZ:maxZ,:);
            obj.ibs_ROI = obj.ibs(minX:maxX,minY:maxY,minZ:maxZ,:);
            obj.decorr_ROI =
 obj.decorr(minX:maxX,minY:maxY,minZ:maxZ,:);
            obj.autocorr01_ROI =
 obj.autocorr01(minX:maxX,minY:maxY,minZ:maxZ,:);
            obj.decorrAvg_ROI =
 obj.decorrAvg(minX:maxX,minY:maxY,minZ:maxZ,:);
        end

    end


end
```

Published with MATLAB® R2016a