```
classdef parfor_wait < handle
    %This class creates a waitbar or message when using for or parfor.
    %Required Input:
    %TotalMessage: N in "i = 1: N".
    %Optional Inputs:
    %'Waitbar': true of false (default). If true, this class creates a
    %               waitbar.
    %'FileName': 'screen' or a char array. If 'screen', print the message
    %               on screen; otherwise, save the message in the file
    %               named 'FileName'.
    %'ReportInterval': 1x1. Report at every i is costly. This number
    %                defines the interval for reporting.
    %%To use this class, one needs to call the class right before the loop:
    %N = 1000;
    %WaitMessage = parfor_wait(N);
    %%Call "Send" method in the loop.
    %for i = 1: N
    %   WaitMessage.Send;
    %   pause(0.5);
    %end
    %%Delete the obj after the loop.
    %WaitMessage.Destroy;
    %
    % Copyright (c) 2019, Yun Pu
    % All rights reserved.
    %
    % Redistribution and use in source and binary forms, with or without
    % modification, are permitted provided that the following conditions are met:
    %
    % * Redistributions of source code must retain the above copyright notice, this
    %   list of conditions and the following disclaimer.
    %
    % * Redistributions in binary form must reproduce the above copyright notice,
    %   this list of conditions and the following disclaimer in the documentation
    %   and/or other materials provided with the distribution
    % * Neither the name of  nor the names of its
    %   contributors may be used to endorse or promote products derived from this
    %   software without specific prior written permission.
    % THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
    % AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
    % IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
    % DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE
    % FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
    % DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
    % SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
    % CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
    % OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
    % OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
    properties (SetAccess = private)
        NumMessage; %Number of messages received from the workers.
        TotalMessage; %Number of total messages.
```

```
        Waitbar; %If waitbar is true, create a waitbar; otherwise, save the message in a file.
        FileName; %If FileName = 'screen', the current message does not save in a file.
        StartTime
        UsedTime_1; %Time at last step.
        WaitbarHandle;
        ReportInterval;
        FileID;
        DataQueueHandle;
    end

    methods
        function Obj = parfor_wait(TotalMessage, varargin)
            Obj.DataQueueHandle = parallel.pool.DataQueue;
            Obj.StartTime = tic;
            Obj.NumMessage = 0;
            Obj.UsedTime_1 = Obj.StartTime;
            Obj.TotalMessage = TotalMessage;
            InParser = inputParser;
            addParameter(InParser,'Waitbar',false,@islogical);
            addParameter(InParser,'FileName', 'screen', @ischar);
            addParameter(InParser,'ReportInterval', ceil(TotalMessage/100), @isnumeric);
            parse(InParser, varargin{:})
            Obj.Waitbar = InParser.Results.Waitbar;
            Obj.FileName = InParser.Results.FileName;
            Obj.ReportInterval = InParser.Results.ReportInterval;
            if Obj.Waitbar
                Obj.WaitbarHandle = waitbar(0, [num2str(0), '%'], 'Resize', true);
            end
            switch Obj.FileName
                case 'screen'
                otherwise
                    Obj.FileID = fopen(Obj.FileName, 'w');
            end
            afterEach(Obj.DataQueueHandle, @Obj.Update);
        end
        function Send(Obj)
            send(Obj.DataQueueHandle, 0);
        end
        function Destroy(Obj)
            if Obj.Waitbar
                delete(Obj.WaitbarHandle);
            end
            delete(Obj.DataQueueHandle);
            delete(Obj);
        end
    end

    methods (Access = private)
        function Obj = Update(Obj, ~)
            Obj.AddOne;
```

```
            if mod(Obj.NumMessage, Obj.ReportInterval)
                return
            end
            if Obj.Waitbar
                Obj.WaitbarUpdate;
            else
                Obj.FileUpdate;
            end
        end

        function WaitbarUpdate(Obj)
            UsedTime_now = toc(Obj.StartTime);
            EstimatedTimeNeeded = (UsedTime_now-Obj.UsedTime_1)/Obj.ReportInterval*(Obj.↙
TotalMessage-Obj.NumMessage);
            waitbar(Obj.NumMessage/Obj.TotalMessage, Obj.WaitbarHandle, [num2str(Obj.↙
NumMessage/Obj.TotalMessage*100, '%.2f'), '%; ', num2str(UsedTime_now, '%.2f'), 's used↙
and ', num2str(EstimatedTimeNeeded, '%.2f'), 's needed.']);
            Obj.UsedTime_1 = UsedTime_now;
        end

        function FileUpdate(Obj)
            UsedTime_now = toc(Obj.StartTime);
            EstimatedTimeNeeded = (UsedTime_now-Obj.UsedTime_1)/Obj.ReportInterval*(Obj.↙
TotalMessage-Obj.NumMessage);
            switch Obj.FileName
                case 'screen'
                    fprintf('%.2f%%; %.2fs used and %.2fs needed...\n', Obj.↙
NumMessage/Obj.TotalMessage*100, UsedTime_now, EstimatedTimeNeeded);
                otherwise
                    fprintf(Obj.FileID, '%.2f%%; %.2fs used and %.2fs needed...\n', Obj.↙
NumMessage/Obj.TotalMessage*100, UsedTime_now, EstimatedTimeNeeded);
            end
            Obj.UsedTime_1 = UsedTime_now;
        end
        function AddOne(Obj)
            Obj.NumMessage = Obj.NumMessage + 1;
        end
    end
end
```