```
classdef BurnerFlow < handle
    %BURNERFLOW Flow with fast kinetic burning, 1D Shapiro relations
    % Created by Thomas Satterly

    properties (SetAccess = private)
        flowElement;
        geometry;
        injectionFunc; % Function for dm_dot/dx
        h; % Heating value of the fuel
        %mdot; % Mass flow rate

        localInjection = 1; % Flag for using local x values for the
 injection function
        cea;
        waitFH;
        dx;
    end

    methods

        function obj = BurnerFlow(varargin)
            np = aeroBox.inputParser();
            np.addParameter('cea', @ishandle);
            np.parse(varargin{:});
            if ~isempty(np.results.cea)
                obj.cea = np.results.cea;
            else
                obj.cea = nasa.CEARunner();
            end
        end

        function setWaitFH(obj, fh, dx)
            obj.waitFH = fh;
            obj.dx =dx;
        end

        function setGlobalInjection(obj)
            obj.localInjection = 0;
        end

        function setLocalInjection(obj)
            obj.localInjection = 1;
        end

        function setHeatingValue(obj, h)
            obj.h = h;
        end

        function setGeometry(obj, geo)
            obj.geometry = geo;
        end
```

```
        function setInjectionFunc(obj, fh)
            assert(isa(fh, 'function_handle'), 'Must use function
handle for injection function!');
            obj.injectionFunc = fh;
        end

        function setMassFlowRate(obj, mdot)
            obj.flowElement.setMassFlow(mdot);
        end

        function setFlowElement(obj, flow)
            obj.flowElement = flow;
        end

        function [lastFlow, states] = solve(obj, numSteps, startX)
            if nargin < 2
                numSteps = 1;
            end

            maxX = obj.geometry.getLength();
            stepSize = maxX / numSteps;
            endFlow = obj.flowElement.getCopy();
            x = 0;
            genState = @(x, flow) struct('flow', flow.getCopy(), 'x',
x + startX);
            states = {};
            lastFlow = obj.flowElement.getCopy();

            if obj.localInjection
                injectionFH = obj.injectionFunc;
            else
                injectionFH = @(x) obj.injectionFunc(x + startX);
            end
            i = 0;
            lastWHUpdate = 0;
            while x < maxX
                i = i + 1;
                states{i} = genState(x, lastFlow);
                x = min(x + stepSize, maxX);
                if ~isempty(obj.waitFH)
                    if x > lastWHUpdate + obj.dx
                        obj.waitFH(x + startX);
                        lastWHUpdate = lastWHUpdate + obj.dx;
                    end
                end

                dTt_dx = lastFlow.Tt * (1 / lastFlow.mdot) *
injectionFH(x) * (obj.h / (lastFlow.cp * lastFlow.Tt) - 1);
                Tt = lastFlow.Tt + dTt_dx * stepSize;
                endFlow.setStagnationTemperature(Tt);


                dM_dx = lastFlow.M * ((1 + ((lastFlow.gamma - 1) / 2)
* lastFlow.M^2) / (1 - lastFlow.M^2)) * ...
```

```
                    ((-1 / obj.geometry.getArea(x - stepSize)) *
obj.geometry.getRateOfAreaChange + ...
                    ((1 + lastFlow.gamma * lastFlow.M^2) / 2) * (1 /
lastFlow.Tt) * dTt_dx);
                nextMach = max(1, lastFlow.M + stepSize * dM_dx);
                if nextMach == 1
                    %keyboard;
                end
                endFlow.setMach(lastFlow.M + stepSize * dM_dx);

                endFlow.setMassFlow(endFlow.mdot + injectionFH(x) *
stepSize);

                P = lastFlow.P * (obj.geometry.getArea(x -
stepSize) / obj.geometry.getArea(x)) * (lastFlow.M / endFlow.M) *
sqrt(endFlow.T / lastFlow.T);

                Pt = aeroBox.isoBox.calcStagPressure('mach',
endFlow.M, 'Ps', P, 'gamma', endFlow.gamma);
                endFlow.setStagnationPressure(Pt);
                try
                    params = obj.cea.run('prob', 'tp', 'p(bar)',
P/1e5, 't,k', endFlow.T(), 'reac', 'name', 'Air', 'wt%', 100, 'end');
                    endFlow.setGamma(params.output.gamma);
                    endFlow.setCp(params.output.cp * 1e3);
                catch
                    warning('CEA error....');
                end
                lastFlow = endFlow.getCopy();
            end
            states{end + 1} = genState(x, lastFlow);
            if ~isempty(obj.waitFH)
                obj.waitFH(x + startX);
            end
        end
    end

end
```

Published with MATLAB® R2015b