Assignment 2
Cyber-Physical Systems Winter 2022

Brittany Taggart

February 5, 2022

Worked with: Jorian Bruslind and Andrey Kornilovich

Exercise 1 *This exercise has you prove some basic relations over polytopes and star sets. I recommed you review the notes PDF under Background module before attempting this exercise.*

1. *Show that the intersection of a polytope with a hyper-plane is a polytope.*

   The definitions of a polytope and hyperplane:

   $$P = \{x \in R^d \mid Ax \leq b\}$$

   $$P = \{x \in R^d \mid a_1^T x \leq b_1, a_2^T x \leq b_2, ..., a_N^T x \leq b_N\}$$

   $$H = \{x \in R^d \mid Yx = z\}$$

   Looking at the intersection between a polytope and hyperplane, if P is contained in one of the two closed half-spaces bounded by H, and the intersection is not an empty set, then H would be considered a "face", or a part, of P.

   So P can be rewritten as:

   $$P = \{x \in R^d \mid a_1^T x \leq b_1, a_2^T x \leq b_2, ..., a_N^T x \leq b_N, Yx \leq z\}$$

   Which means that H is a polytope.

2. *Show that a polytope is a special case of a star set. I.e., given a polytope Q, show that its points can be written as the points of a star. In other words, find the center c, matrix V and linear predicate P such that $Q = \{x \in R^d \mid x = c + \sum_i \alpha_i v_i,\ P(\alpha_1, \ldots, \alpha_m) = \top\}$.*

Supposed for some polytope Q, where:

$$Q = \{x = \lambda_1 v_1 + \lambda_2 v_2 + \ldots + \lambda_m v_m \mid \lambda_1 + \lambda_2 + \ldots + \lambda_m = 1, \lambda_i \geq 0\}$$

In order to convert this to star set form, the center c, matrix V and linear predicate P must be found. For c, this will be the origin of the polytope. Let $b_i \in R^n$ be some offset vectors, then

$$c = \lambda_i + b_i$$

Based on the equation for Q, the predicate P will be

$$P(\lambda_1, \ldots, \lambda_m) = T$$

And matrix V will remain the same, so Q can be rewritten as:

$$Q = \{x \in R^d \mid x = (\lambda_i + b_i) + \sum_i \lambda_i v_i,\ P(\lambda_1, \ldots, \lambda_m) = \top\}$$

3. *In general, the constraint in a star set need not be a linear inequality. What is the set described by the following star?*

$$c = [0; 0], v_1 = [1; 0], v_2 = [0; 1], \alpha_1^2 + \alpha_2^2 \leq 1$$

The set is represented by the inside of a circle with radius 1, centered at (0,0).

4. *What is the set described by the following star?*

$$c = [0; 0], v_1 = [2; 0], v_2 = [0; 1], \alpha_1/\alpha_2 = 3$$

The set is represented by a line with a slope of 1.5 going through the center (0,0).

Exercise 2 *This exercise has you use CORA to compute the reachsets of continuous-time linear systems. It refers to Girard's paper which is uploaded to Canvas under the 'Background and references module'.*

1. *Using CORA, compute the reachsets for the two linear systems at the beginning of Section 4 of Girard's paper: the first is a 2D system, the second is 5D. Use the parameters given in the paper (initial set, μ, input set, etc). You are provided with skeleton code for this assignment: use it as starting point. There are already place holders in the skeleton code for all the parameters. Make sure you are setting them correctly, that you understand which variable is mu, which variable is the initial set X_0, etc.*

   Note that the skeleton is such that if I execute
   `skeleton_cpsclass_reach(1)`
   it computes the reachset for the 2D System 1, and if I execute
   `skeleton_cpsclass_reach(2)`
   it computes the reachset for the 5D System 2.

   Run this code by putting either `skeleton_cpsclass_reach(1)`
   or `skeleton_cpsclass_reach(2)` *in the command line.*
   System 2 has a default timeStep value of 0.005.

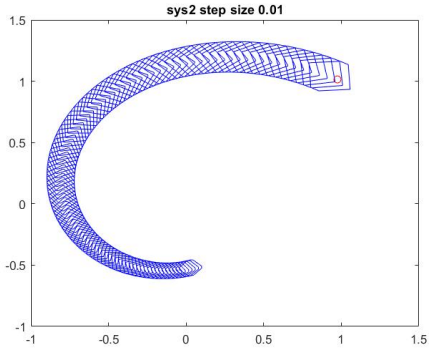
2. *Create a second matlab script. This one will compute the reachsets for System 2, for different values of the time step parameter. Specifically, it will set the timeStep parameter to the values 0.01, 0.02, 0.03, 0.04 and 0.05. For each value, it will compute the reach-set for System 2, as done in the preceding question. It will then plot the reachset in a new figure. Title every figure with the corresponding step size, using this command:*

`title(['step size', num2str(timeStep)])`

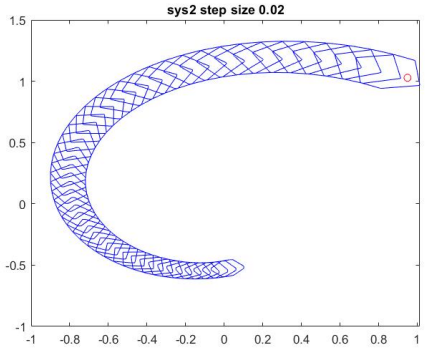
Make sure that the plot for every new reachset (for every new value of timeStep) is on a separate figure. You can create a new figure using the **`figure`** *command.*

Call this new script **`sweep_timestep.m`**.

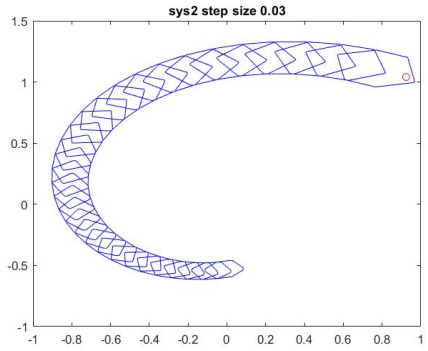
> *To run a time sweep, just enter* **`sweep_timestep`** *in the command line.*



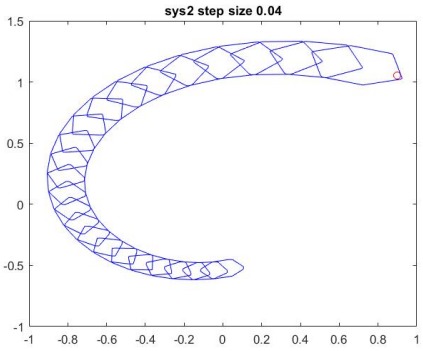
(a) Step Size 0.01



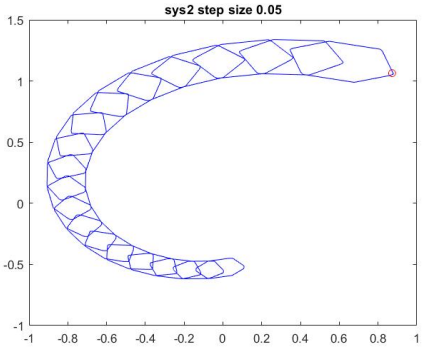
(b) Step Size 0.02



(c) Step Size 0.03



(d) Step Size 0.04



(e) Step Size 0.05

Figure 1: System 2 Time Sweep

Exercise 3 *Complete the skeleton code you are given for nnv, following the instructions in the comments*

of that file.

> *Code can be run as stated in the given file.*

1 Code

```
function skeleton_cpsclass_reach(sysnb, timeStep)
% sysnb selects which system to run: 1 or 2
% if no argument is given, runs system 1 by default
if nargin ==0
    sysnb = 1;
end
%------------- System 1 --------------
if (find(sysnb==1))
    % dimension of the system
    dim=2;
    % bound on input norm
    mu = 0.05;

    options.tStart=0; %start time
    %%%%%%%%%%%%%%%%%%
    options.tFinal=2; %final time
    %%%%%%%%%%%%%%%%%%
    options.x0=[1; 0]; %initial state for simulation
    options.R0=zonotope([options.x0,[0.1 0.1; 0 0]]); %initial state for reachability analysis
    %%%%%%%%%%%%%%%%%%
    options.timeStep=0.02; %time step size for reachable set computation
    %%%%%%%%%%%%%%%%%%
    options.taylorTerms=4; %number of taylor terms for reachable sets
    %%%%%%%%%%%%%%%%%%
    options.zonotopeOrder=10; %zonotope order
    %%%%%%%%%%%%%%%%%%
    options.originContained=0;
    options.reductionTechnique='girard';
    options.linAlg = 1;

    % center of input set
    options.uTrans=[0; 0];
    % input set is a zonotope with above center, and generator matrix
    % mu*identity matrix
    options.U=zonotope([zeros(dim,1),mu*eye(dim)]); %input for reachability analysis

    %specify continuous dynamics
    A=[-1 -4;
        4 -1];
    B=1;
    twoDimSys=linearSys('twoDimSys',A,B); %initialize system


    %compute reachable set using zonotopes
    Rcont = reach(twoDimSys, options);
```

```
    % safety check
    % obstacle set is a zonotope with given center and generators
    obstacle_center = [0;0];
    obstacle_generators = 0.1*[1 0.5; 0.5 1];
    % The intersection method, and, requires two zonotope bundles
    Obs  = zonotopeBundle([obstacle_center, obstacle_generators ]);
    % This variable is needed for plotting later.
    zObs = zonotope([obstacle_center, obstacle_generators ]);
    % Get the interesection and convert back to a zonotope object for
    % determining if it's empty or not.
    Zint = zonotope(and(Obs, Rcont));
    if Zint.isempty
        disp('Safe!')
    else
        disp('Unsafe!')
    end

    plotreachsets(Rcont, 'sys1');
    hold on
    plot(zObs);
end
%------------- System 2 --------------
if (find(sysnb==2))
    dim=5;
    mu = 0.01;

    options.tStart=0; %start time
    options.tFinal=1; %final time
    % The initial set is a zonotope centered on [1;1;1;1;1], and with
    % generators 0.1*identity matrix
    options.x0=ones(dim,1) ; %initial state for simulation
    options.R0=zonotope([options.x0,0.1*eye(dim)]); %initial state for reachability analysis
    %%%%%%%%%%%%%%%%%%%%%
    if nargin < 2
        timeStep = 0.005;
        options.timeStep=timeStep; %time step size for reachable set computation
    else
        options.timeStep = timeStep;
    end
    %%%%%%%%%%%%%%%%%%%%%
    options.taylorTerms=4; %number of taylor terms for reachable sets
    %%%%%%%%%%%%%%%%%%%%%
    options.zonotopeOrder=40; %zonotope order
    %%%%%%%%%%%%%%%%%%%%%
    options.originContained=0;
    options.reductionTechnique='girard';
    options.linAlg = 1;


    options.uTrans= zeros(dim,1); %input for simulation
    options.U=zonotope([zeros(dim,1),mu*eye(dim)]); %input for reachability analysis

    %specify continuous dynamics
    A=[-1 -4 0 0 0;
```

```
        4 -1 0 0 0;
        0 0 -3 1 0;
        0 0 -1 -3 0;
        0 0 0 0 -2];
    B=1;
    fiveDimSys=linearSys('fiveDimSys',A,B); %initialize system


    %compute reachable set using zonotopes
    Rcont = reach(fiveDimSys, options);


    plotreachsets(Rcont, 'sys2');
    title(['sys2 step size ', num2str(timeStep)])

end
end
function plotreachsets(reachset, figtitle)
% Plot a collection of zonotopes
figure;
Z=reachset{1};
center = Z.center;
plot(reachset{1});
plot(center(1), center(2),'ro');
hold on
for ii=2:length(reachset)
    plot(reachset{ii});
end
title([figtitle]);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



function sweep_timestep()

    for timeStep = 0.01:0.01:0.05
        skeleton_cpsclass_reach(2, timeStep);
    end

end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function cpsclass_nnv_skeleton(sysnb)
% cpsclass_nnv()
%   run all systems
% cpsclass_nnv(sysnb)
```

```
%   sysnb is a vector containing at most the numbers 1,2,3
supported_sys = [1,2,3];
if nargin ==0
    sysnb = supported_sys;
else
    temp = sort(unique(sysnb));
    for ii=1:length(sysnb)
        assert(~isempty(find(supported_sys==temp(ii),1)), ['testnb must be one of ', num2str(supported_s
    end
end
% If the parallel toolbox on your machine works fine, go for it and set
% this to 4 or 6. Otherwise, keep it at 1.
numcores = 1;
%------------------------------------------------------------------------------
% This first system is an example which demonstrates the basic use of the
% toolbox. No code modifications are required, you can run it.
if (find(sysnb==1))

    %% test 1: FFNNS falsify
    % Weights of NN layer. You can tell it takes in 2 inputs and has two
    % neurons
    W = [1 1; 0 1];
    % Bias of NN layer
    b = [0; 0.5];
    L = LayerS(W, b, 'poslin'); % poslin is ReLU
    Layers = [L];

    % Create the forward NN
    F = FFNNS(Layers);

    % Input star is simply the box given by the bounds below
    lb = [-1; -1];
    ub = [1; 1];
    I = Star(lb, ub);

    % Reach!
    [R, ~] = F.reach(I, 'exact-star');

    %This represents the unsafe set of states to be avoided
    % U = {x | Gx <= g}
    % So this is -x1 <= -1.5, i.e. x1 >= 1.5
    G = [-1 0];
    g = [-1.5];
    U = HalfSpace(G, g);

    % Plot the reachset and unsafe set
    R.plots(R);
    hold on
    U.plot();
    title('Reachset and Unsafe set')

    % Intersection with unsafe set?
    %figure
    nintersections = 0;
```

```
    for s=1:length(R) % in general, it's a union of stars
        SI = R(s).intersectHalfSpace(U.G, U.g);
        if (isa(SI, 'Star') && ~SI.isEmptySet())
            fprintf('Star %i intersects unsafe set\n', s)
            nintersections = nintersections+1;
        end
    end
    if nintersections > 0
        disp('Unsafe!! AAAAAAA!!!')
    end

    % Try to find points in the input set I which witness violation, i.e.
    % whose corresponding NN outputs are in the unsafe set
    n_samples = 1000;

    % Method falsify does a naive uniform random sampling. As such there is
    % no guarantee that it will find falsifying inputs even if they exist
    counter_inputs = F.falsify(I, U, n_samples);
    counter_outputs = F.sample(counter_inputs);

    figure;
    subplot(1, 2, 1);
    I.plot;
    hold on;
    plot(counter_inputs(1, :), counter_inputs(2, :), 'o');
    title('Input Set and counter inputs');
    subplot(1, 2, 2);
    Star.plots(R);
    hold on;
    plot(counter_outputs(1, :), counter_outputs(2, :), 'o');
    title('Output set and counter outputs');

end
%------------------------------------------------------------------------------

%tests below originally taken from test_FFNNS_verify_DFS.m
if (find(sysnb==2))

    % Layer 1 has the following parameters:
    % x1 = z1 - z2 - 1
    % x2 = 0.5z1 + 2z2 + 0.5
    % x3 = -z1 + z2
    % Activation function is poslin (i.e., ReLU)
    W1 = [1 -1; 0.5 2; -1 1];
    b1 = [-1; 0.5; 0];
    L1 = LayerS(W1, b1, 'poslin');

    % Layer 2 has following parameters:
    % y1 = -2x1 + x2 + x3 - 0.5
    % y2 = 0.5x1 + x2 + x3 -0.5
    % Activation function is purelin
    W2 = [-2 1 1; 0.5 1 1];
    b2 = [-0.5; -0.5];
    L2 = LayerS(W2, b2, 'purelin');
```

```
F = FFNNS([L1 L2]); % construct Feedforward neural network

% Input set is the box -2 <= z1 <= 2, -1 <= z2 <= 2
lb = [-2; -1];
ub = [2; 2];
I = Star(lb, ub); % construct input set

% Compute the reachset using the exact-star method
[R, t] = F.reach(I, 'exact-star', numcores, []); % compute the exact reachable set

% Also compute it using the approx-star method
[Rapx, t] = F.reach(I, 'approx-star', numcores, []);

% plot reachable set
fig = figure;
subplot(1, 2, 1);
I.plot;
title('Input Set', 'FontSize', 20);
xlabel('x_1', 'FontSize', 16);
ylabel('x_2', 'FontSize', 16);

subplot(1, 2, 2)
Star.plots(Rapx, 'g')
hold on
Star.plots(R)

title('Output Set', 'FontSize', 20);
xlabel('y_1', 'FontSize', 16);
ylabel('y_2', 'FontSize', 16);

% unsafe region: y[1] >= 5
G = [-1 0];
g = [-5];
U = HalfSpace(G, g);
subplot(1,2,2);
hold on
U.plot();

% Check whether the reachset intersects the unsafe set

nintersections = 0;
for s=1:length(R) % in general, it's a union of stars
    SI = R(s).intersectHalfSpace(U.G, U.g);
    if (isa(SI, 'Star') && ~SI.isEmptySet())
        fprintf('Star %i intersects unsafe set\n', s)
        nintersections = nintersections+1;
    end
if nintersections > 0
    disp('Exact Unsafe!! AAAAAAA!!!')
end
end

nintersections = 0;
```

```
    for s=1:length(Rapx) % in general, it's a union of stars
        SI = Rapx(s).intersectHalfSpace(U.G, U.g);
        if (isa(SI, 'Star') && ~SI.isEmptySet())
            fprintf('Star %i intersects unsafe set\n', s)
            nintersections = nintersections+1;
        end
    end
    if nintersections > 0
        disp('Approx Unsafe!! AAAAAAA!!!')
    end
end
% %_____________________________________________________________________________________

if (find(sysnb==3))
    %% Test 6: ACAS Xu
    load ACASXU_run2a_2_1_batch_2000.mat;
    Layers = [];
    n = length(b);
    % Instantiate the first n-1 layers
    for i=1:n - 1
        bi = cell2mat(b(i));
        Wi = cell2mat(W(i));
        Li = LayerS(Wi, bi, 'poslin');
        Layers = [Layers Li];
    end
    % Instantiate the last layer, which has the purelin activation
    bn = cell2mat(b(n));
    Wn = cell2mat(W(n));
    Ln = LayerS(Wn, bn, 'purelin');

    Layers = [Layers Ln];
    F = FFNNS(Layers);

    % Input Constraints
    % 1500 <= i1(\rho) <= 1800,
    % -0.06 <= i2 (\theta) <= 0.06,
    % 3.1 <= i3 (\shi) <= 3.14
    % 980 <= i4 (\v_own) <= 1200,
    % 960 <= i5 (\v_in) <= 1000

    lb = [1500; -0.06; 3.1; 980; 960];
    ub = [1800; 0.06; 3.14; 1200; 1000];
    % normalize input
    for i=1:5
        lb(i) = (lb(i) - means_for_scaling(i))/range_for_scaling(i);
        ub(i) = (ub(i) - means_for_scaling(i))/range_for_scaling(i);
    end
    I = Star(lb, ub);

    % Output: [x1 = COC; x2 = Weak Left; x3 = Weak Right; x4 = Strong Left; x5 = Strong Right]

    % [R, ~] = F.reach(I, 'exact-star', numCores); % exact reach set using polyhdedron
    % F.print('F_exact_star.info'); % print all information to a file
```

```
    % Compute the reach-set using approx-star and the specified numcores
    [R, ~] = F.reach(I, 'approx-star', numcores, []);
    F.print('F_approx_star.info'); % print all information to a file
    %R2.plot('r', [1 0 0 0 0 ; 0 1 0 0 0]);
    %title('Projection of reachset on first 2 dimensions')

    % unsafe region: COC is the minimal score: x1 <= x2; x1 <= x3; x1 <= x4, x1<= x5
    G = [1 -1 0 0 0;
        1 0 -1 0 0;
        1 0 0 -1 0;
        1 0 0 0 -1;
        0 0 0 0 0];
    g = [0; 0; 0; 0; 0];


    t = tic;
    fprintf('\nVerifying reach set...\n');
    unsafe = 0;
    n = length(R);
    for i=1:n
        S = R(i).intersectHalfSpace(G, g);
        if ~isempty(S)
            fprintf('\nThe %d^th star output set reaches the unsafe region', i);
            unsafe = unsafe + 1;
        end
    end

end
```