

MECH 6313 - Homework 1

Jonas Wagner

2021, February 1

1 Problem 1 - Duffing's Equation

Duffings Equation exhibits chaotic behavior with certain parameters. It is described by the following:

$$\ddot{y} + \delta \dot{y} - y + y^3 = \alpha \cos(\omega_t t)$$

Problem: Simulate the equation for $\delta = 0.05$, $\alpha = 0.4$, and $\omega_t = 1.3$.

Solution: In matlab the nlsys class (something I have been developing to help in nonlin system simulation - <https://github.com/jonaswagner2826/nlsys>) was used to simulate the system and plot the following phase plots and time responses. The MATLAB code for this assignment is available in Appendix1

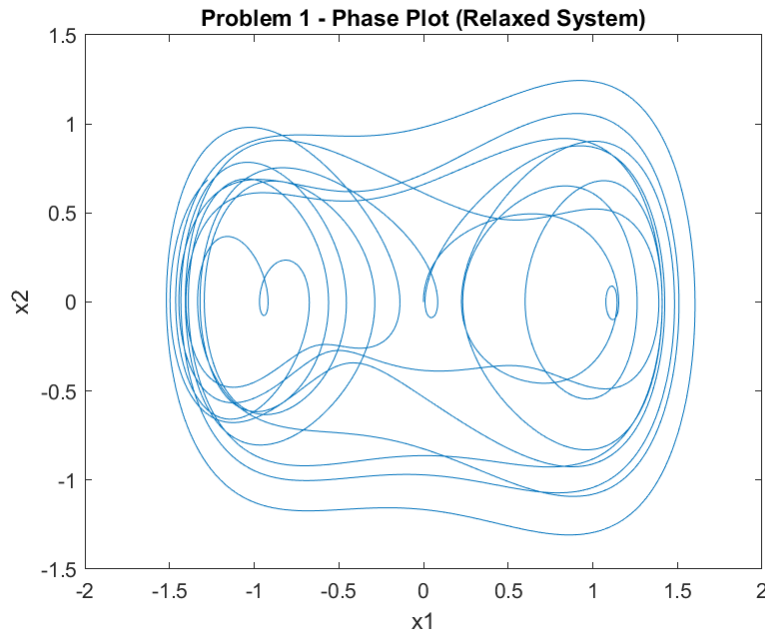


Figure 1: Phase Plot for the Relaxed System

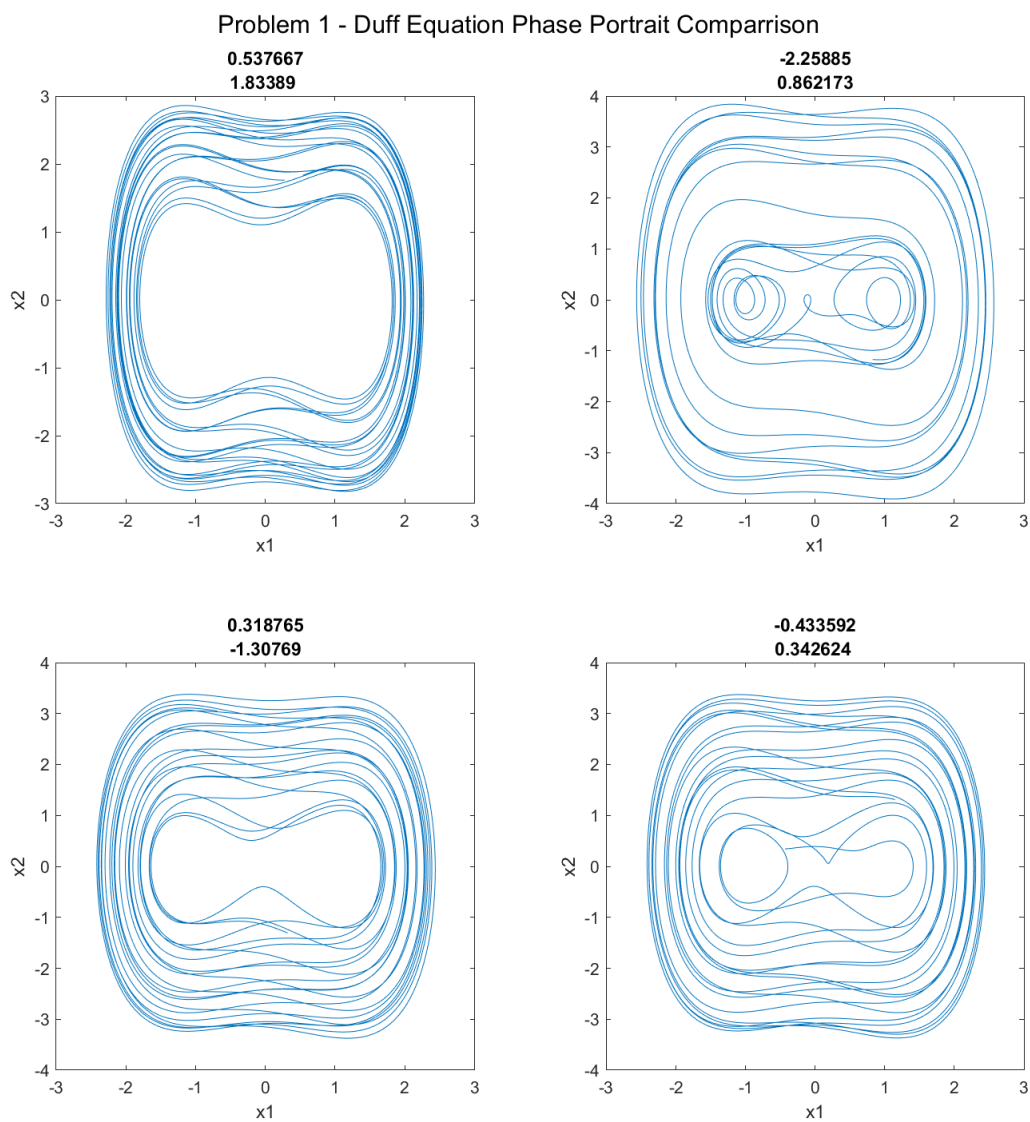


Figure 2: Phase Plot for multiple initial conditions

Problem 1 - Duff Equation Time Simulation

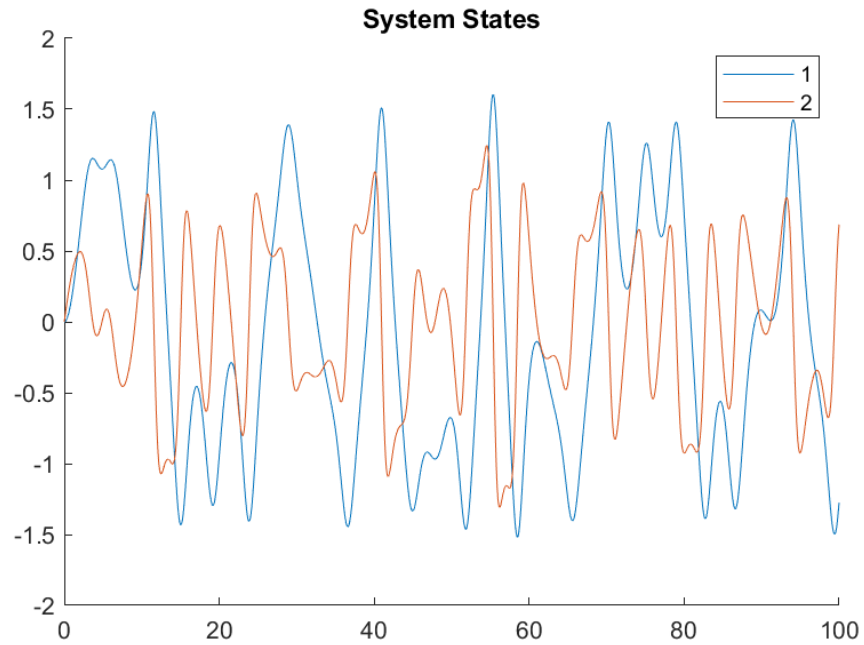


Figure 3: Plot of the relaxed system vs time

Discussion:

Both the phase portraits and time response of the Duffings equation indicate a fairly chaotic behavior, however it does appear to contentiously rotate around the origin in a periodic fashion (possibly due to the sinusoidal input). Either way, it is difficult to recognize a predictable behavior, and neither decays or explodes predictably.

2 Problem 2 - Van Der Pol Equations

Problem: The van der Pole equation is as follows:

$$\ddot{y} + (y^2 - 1) * \dot{y} + y = 0$$

Plot the phase portrait, time dependence and compare with the response of Duffing's equations.

Solution: In matlab the nlsys class (something I have been developing to help in nonlin system simulation - <https://github.com/jonaswagner2826/nlsys>) was used to simulate the system and plot the following phase plots and time responses. The MATLAB code for this assignment is available in Appendix1

2.1 Van Der Pol Simulation

2.1.1 Phase Portraits

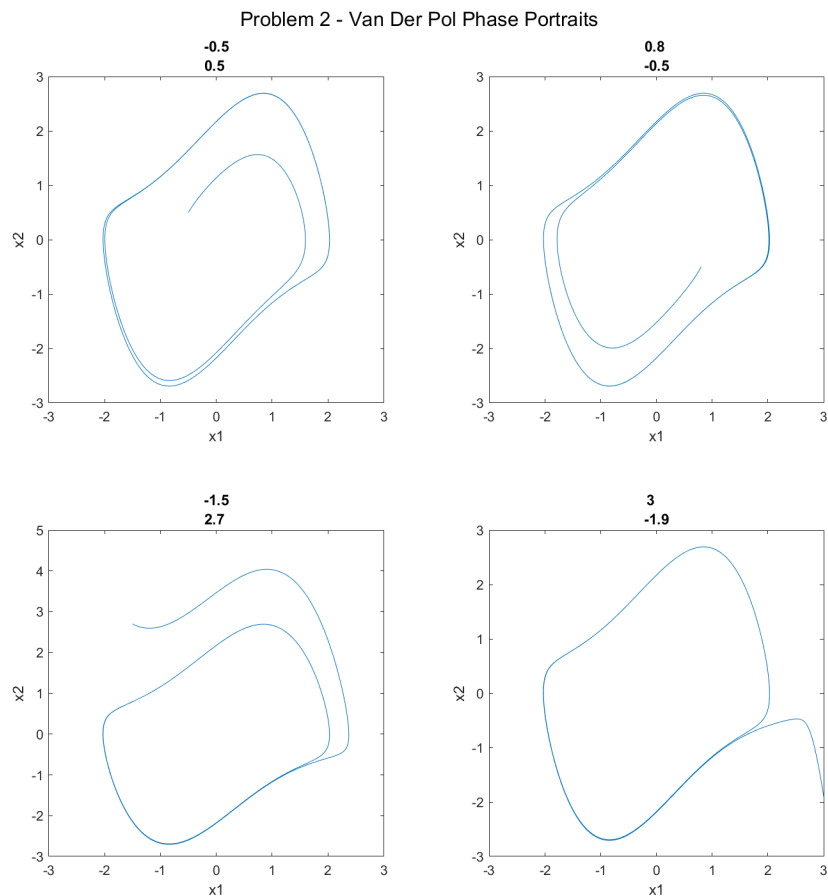


Figure 4: Phase Plot for multiple initial conditions

2.1.2 Time Response

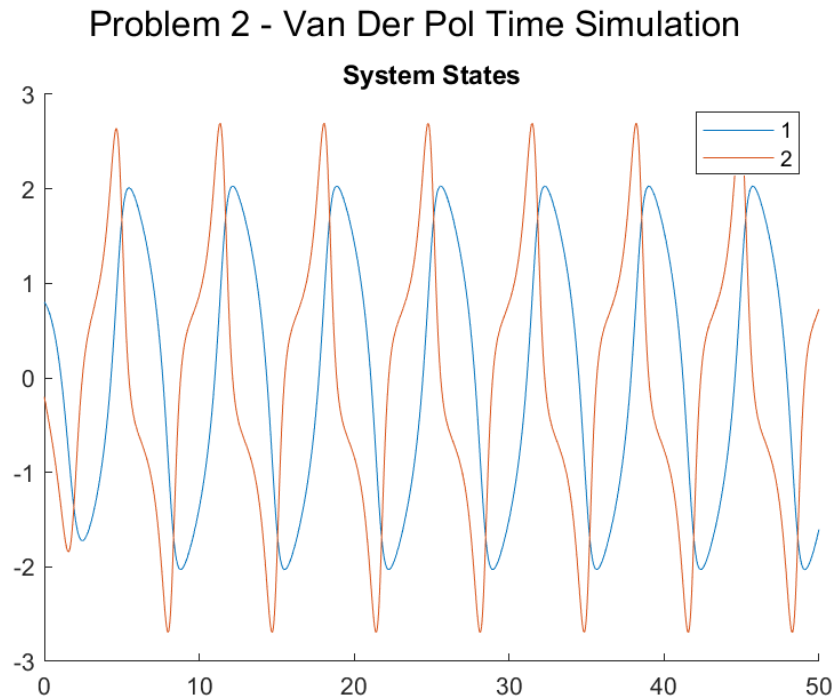


Figure 5: Phase Plot for multiple initial conditions

Discussion:

Unlike the results from the duffings equation, the Van Der Pol equations produce a very predictable response. Outside of the equilibrium point located at the origin, the unforced response from any initial equation appears to decay to the exact same periodic motion. Looking at the time response plot, the system states demonstrate a periodic response consistent with expectations from the phase portrait.

2.2 Negative Van Der Pole Equation

Modifying the nonlinear term of the van der pole equations results in the following differential equation:

$$\ddot{y} - (y^2 - 1) * \dot{y} + y = 0$$

2.2.1 System Stability at the Origin

The Van Der Pol equation can be linearized at the origin by using a Taylor's series expansion of the state-space model. The first term can be found by taking the jacobian and evaluating it with $x = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$.

Letting $x_1 = y$ and $x_2 = \dot{y}$, the following state-space model is derived:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} x_2 \\ -(x_1^2 - 1)x_2 - x_1 \end{bmatrix}$$

The jacobians for the A matrix can then be found as:

$$\begin{aligned} A &= \left[\begin{array}{cc} \frac{df_1}{dx_1} & \frac{df_1}{dx_2} \\ \frac{df_2}{dx_1} & \frac{df_2}{dx_2} \end{array} \right] \bigg|_{\mathbf{x}=\mathbf{x}_0} \\ &= \left[\begin{array}{cc} 0 & 1 \\ -2x_1x_2 - 1 & -x_1^2 + 1 \end{array} \right] \bigg|_{\mathbf{x}=\mathbf{x}_0} \\ &= \begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix} \end{aligned}$$

The stability of this matrix can then be determined by looking at the eigenvalues:

$$\Lambda\{A\} = 0.5 \pm j0.866$$

Since $\Re\{\Lambda\{A\}\} > 0$, the linearized system is said to be unstable at the origin.

2.2.2 Phase Portraits

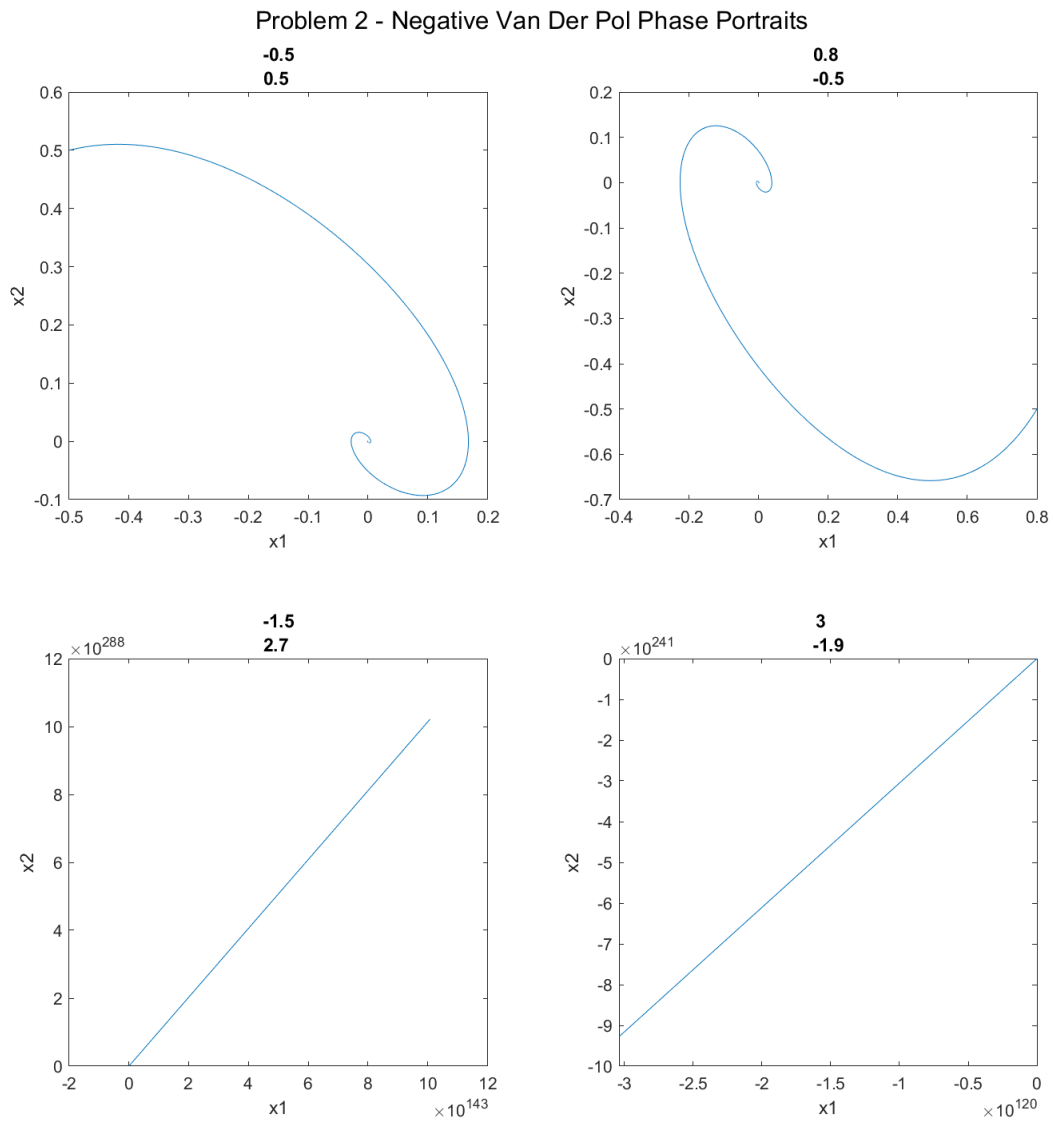


Figure 6: Phase Plot for multiple initial conditions

2.2.3 Time Response

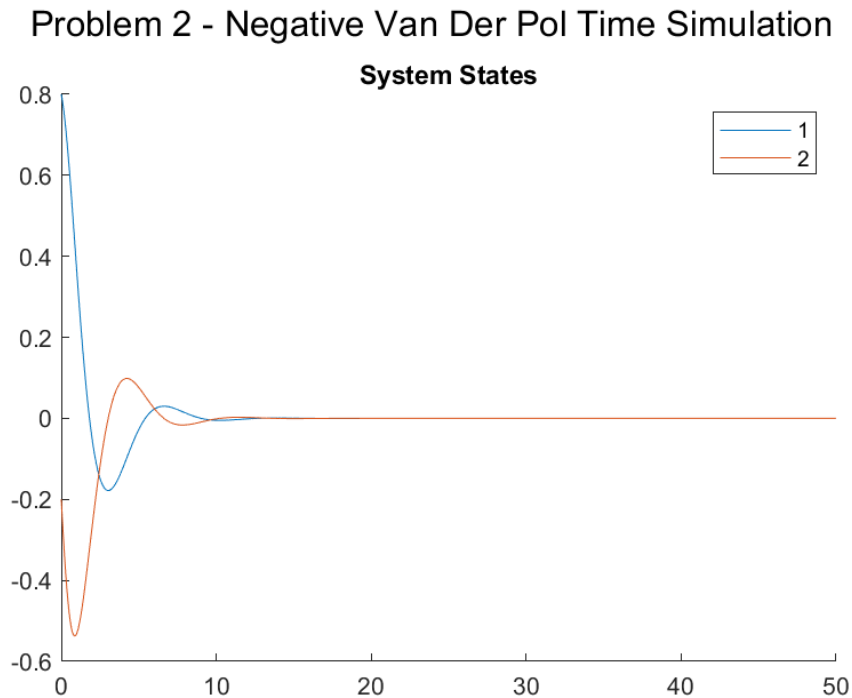


Figure 7: Phase Plot for multiple initial conditions

Discussion: Unlike in the positive van der pol equation, when the nonlinear term is set to be negative it reacts almost directly opposite of the positive one. As was discovered, the origin within the negative system is asymptotically stable, but this is not true in general. From the simulations it is apparent that the same boundary that is the steady-state response for the positive van der pol equation is the boundary of instability. The system is indeed locally stable within that boundary and decays to zero, but outside it blows up. It actually doesn't blow up exactly as expected either, depending on the initial condition it won't necessarily just explode out to the extremes of its own quadrant but instead it changes with the initial condition.

3 Problem 3 - Magnetic Suspension System

An electromagnet suspends a ball and is controlled by a feedback system based on the measured position. The equation of motion for the ball is given as:

$$m\ddot{y} = -k\dot{y} + mg + F(y, i) \quad (1)$$

where m is the mass of the ball, $y \geq 0$ is the vertical position of the ball relative to the electro magnet, k is the friction coefficient, g is the acceleration of gravity, and $F(y, i)$ is the force generated by the magnet dependent on the position and current (i).

The inductance of the electromagnet is given as a function of the ball's position as:

$$L(y) = L_1 + \frac{L_0}{1 + y/a} \quad (2)$$

where $L_1, L_0, a > 0$.

The energy stored within the electromagnet is given as a function of inductance:

$$E(y, i) = \frac{1}{2}L(y)i^2 = \frac{1}{2}\left(L_1 + \frac{L_0}{1 + y/a}\right)i^2 \quad (3)$$

The force on the ball is then calculated as the derivative of energy with respect to position:

$$F(y, i) = \frac{\partial E}{\partial y} = \frac{-L_0 i^2}{2a(1 + y/a)^2} \quad (4)$$

The electric circuit controlling the electro magnet is governed by KVL as:

$$v = \dot{\phi} + Ri \quad (5)$$

where v is the input voltage, R is the resistance and $\phi = L(y)i$.

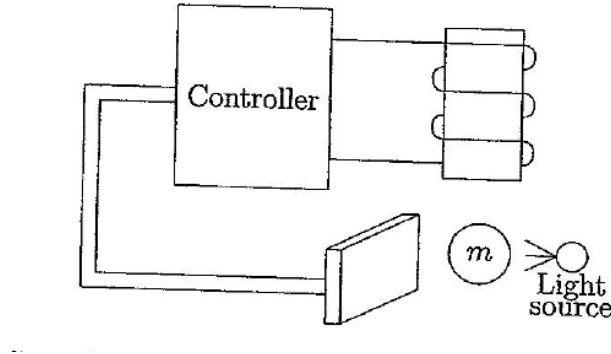


Figure 8: Magnetic suspension system diagram

3.1 State Space Model

Let the following state variables be defined:

$$x_1 = y \quad x_2 = \dot{y} \quad x_3 = i \quad u = v$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y \\ \dot{y} \\ i \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} v \end{bmatrix} \quad (6)$$

From the definition, the following state equation can be stated directly:

$$\dot{x}_1 = x_2 \quad (7)$$

A second equation of motion can be derived from (1) and (4):

$$\begin{aligned} \ddot{y} &= -\left(\frac{k}{m}\right) \dot{y} + g + \frac{1}{m} F(y, i) \\ &= -\left(\frac{k}{m}\right) \dot{y} + g + \frac{1}{m} \frac{-L_0(i)^2}{2a(1+y/a)^2} \\ \dot{x}_2 &= -\left(\frac{k}{m}\right) x_2 + g + \frac{-L_0(x_3)^2}{2am(1+\frac{x_1}{a})^2} \end{aligned} \quad (8)$$

From (5) the following can be derived:

$$\begin{aligned} v &= \frac{\partial}{\partial t} (L(y)i) + Ri \\ &= \left(\dot{L}(y)i + L(y)\dot{i} \right) + Ri \end{aligned} \quad (9)$$

$\dot{L}(y)$ can be calculated from (4):

$$\begin{aligned} \dot{L}(y) &= \frac{\partial}{\partial t} \left(L_1 + \frac{L_0}{1+y/a} \right) \\ &= -\frac{L_0 a \dot{y}}{(a+y)^2} \end{aligned} \quad (10)$$

Substituting (4) and (10) into (9), the following can be obtained:

$$\begin{aligned} v &= -\frac{L_0 a \dot{y} i}{(a+y)^2} + \left(L_1 + \frac{L_0}{1+y/a} \right) \dot{i} + Ri \\ \left(\frac{L_1(1+y/a) + L_0}{1+y/a} \right) \dot{i} &= \frac{L_0 a \dot{y} i}{(a+y)^2} - Ri + v \\ \dot{i} &= \left(\frac{1+y/a}{L_1(1+y/a) + L_0} \right) \left(\frac{L_0 a \dot{y} i}{(a+y)^2} - Ri + v \right) \\ \dot{x}_3 &= \left(\frac{1+x_1/a}{L_1(1+x_1/a) + L_0} \right) \left(\frac{L_0 a x_2 i}{(a+x_1)^2} - Rx_3 + u \right) \end{aligned} \quad (11)$$

The full state-space model is given as:

$$\dot{\mathbf{x}} = \begin{bmatrix} x_2 \\ -\left(\frac{k}{m}\right)x_2 + g + \frac{-L_0(x_3)^2}{2am\left(1 + \frac{x_1}{a}\right)^2} \\ \left(\frac{1 + x_1/a}{L_1(1 + x_1/a) + L_0}\right)\left(\frac{L_0 a x_2 i}{(a + x_1)^2} - Rx_3 + u\right) \end{bmatrix} \quad (12)$$

3.2 Steady-State Solution

The steady-state equation will occur when $\dot{x}_1 = \dot{x}_2 = 0$, so the state-space model can be used to find this condition at a certain position, $r > 0$.

Since $\dot{x}_1 = 0$ and $y = r$, the following can be defined:

$$x_1 = y = r \quad (13)$$

$$x_2 = \dot{x}_1 = 0 \quad (14)$$

$L_{ss}(r)$ is then calculated from (2):

$$L_{ss}(r) = L_1 + \frac{L_0}{1 + r/a} \quad (15)$$

Similarly, $\dot{L}_{ss}(r)$ is then calculated from (10):

$$\dot{L}_{ss}(r) = -\frac{L_0 a (0)}{(a + r)^2} = 0 \quad (16)$$

Referring back to (9), v and i can be related by the following:

$$\begin{aligned} v &= \left(\dot{L}(y)i + L(y)\dot{i}\right) + Ri \\ &= \left((0)i + \left(L_1 + \frac{L_0}{1 + r/a}\right)\dot{i}\right) + Ri \\ &= \left(L_1 + \frac{L_0}{1 + r/a}\right)\dot{i} + Ri \end{aligned} \quad (17)$$

If you make the assumption that I_{ss} is a constant (something I don't believe),

$$V_{ss} = RI_{ss} \quad (18)$$

$$I_{ss} = \frac{V_{ss}}{R} \quad (19)$$

In this case, F_{ss} can be calculated from (4) and (19):

$$\begin{aligned} F_{ss} &= \frac{-L_0\left(\frac{V_{ss}}{R}\right)^2}{2a(1 + r/a)^2} \\ &= \frac{-L_0V_{ss}^2}{2aR^2(1 + r/a)^2} \end{aligned} \quad (20)$$

In order to maintain static conditions, $F_{ss} = -mg$, so V_{ss} can be found as:

$$F_{ss} = mg = \frac{-L_0 V_{ss}^2}{2aR^2(1+r/a)^2} \quad (21)$$

$$V_{ss}^2 = \frac{-2amgR^2(1+r/a)^2}{L_0} \quad (22)$$

$$V_{ss} = \sqrt{\left(\frac{-2a(mg)(R^2)}{L_0}\right)(1+r/a)^2} \quad (23)$$

I_{ss} is then calculated from (19) and (23):

$$\begin{aligned} I_{ss} &= \frac{1}{R} \sqrt{\left(\frac{-2a(mg)(R^2)}{L_0}\right)(1+r/a)^2} \\ &= \sqrt{\left(\frac{-2a(mg)}{L_0}\right)(1+r/a)^2} \end{aligned} \quad (24)$$

4 Problem 4 - Bifurcation Examples

Additional functionality was added to the nlsys class in order to introduce bifurcation plots.

4.1 Pblm 3.4.2

$$\dot{x} = rx - \sinh(x)$$

As is evident in the bifurcation diagram, the pitchfork diagram is supercritical. r_c was found by differentiating $f(x, r)$, solving for x_c , substituting back into $f(x, r)$ and then solving for $r_c = 1$.

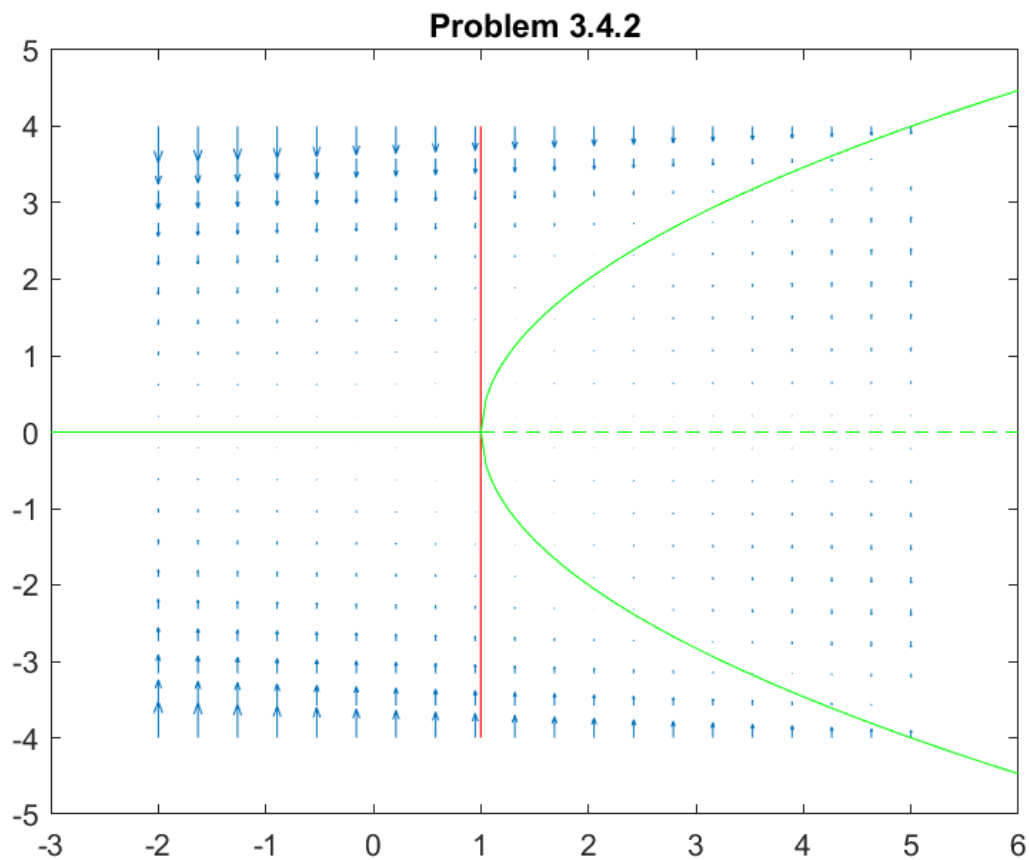


Figure 9: Bifurcation Diagram for problem 3.4.2

4.2 Pblm 3.4.4

$$\dot{x} = x + \frac{rx}{1+x^2}$$

As is evident in the bifurcation diagram, the pitchfork diagram is subcritical. r_c was found by differentiating $f(x, r)$, solving for x_c , substituting back into $f(x, r)$ and then solving for $r_c = -1$.

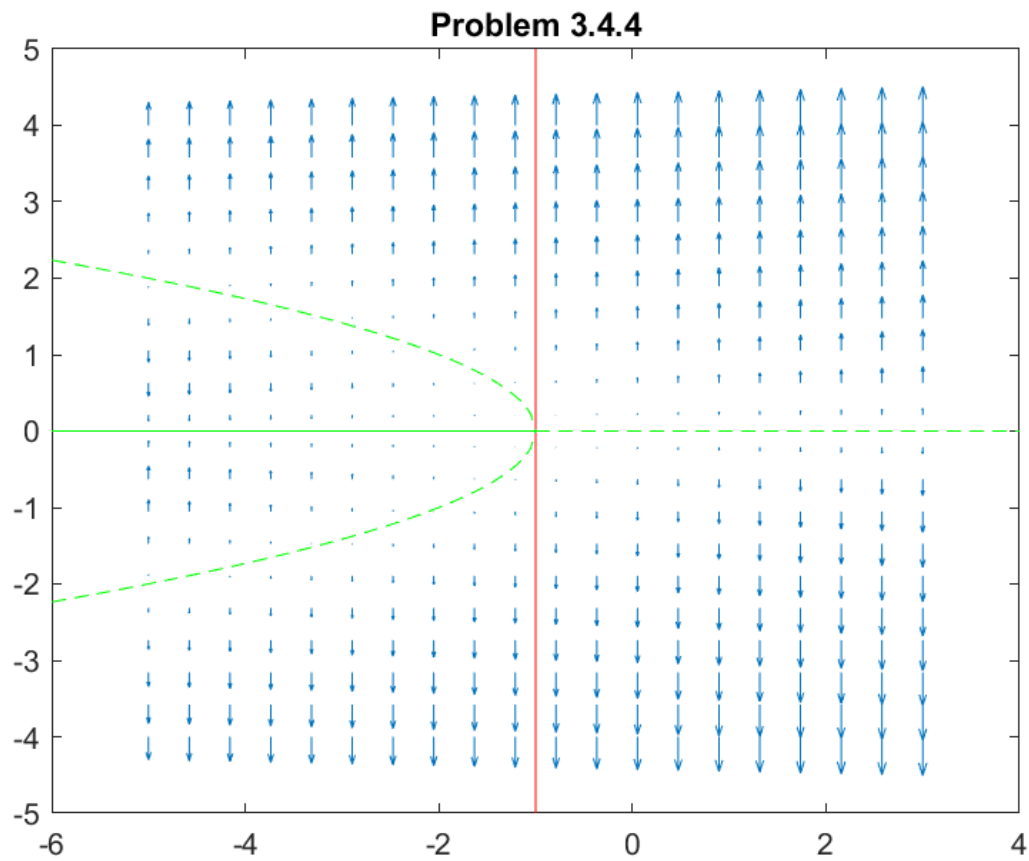


Figure 10: Bifurcation Diagram for problem 3.4.4

4.3 Pblm 3.4.7

$$\dot{x} = 5 - re^{-x^2}$$

As is evident in the bifurcation diagram, is a saddle node bifurcation with $r_c = 1$.

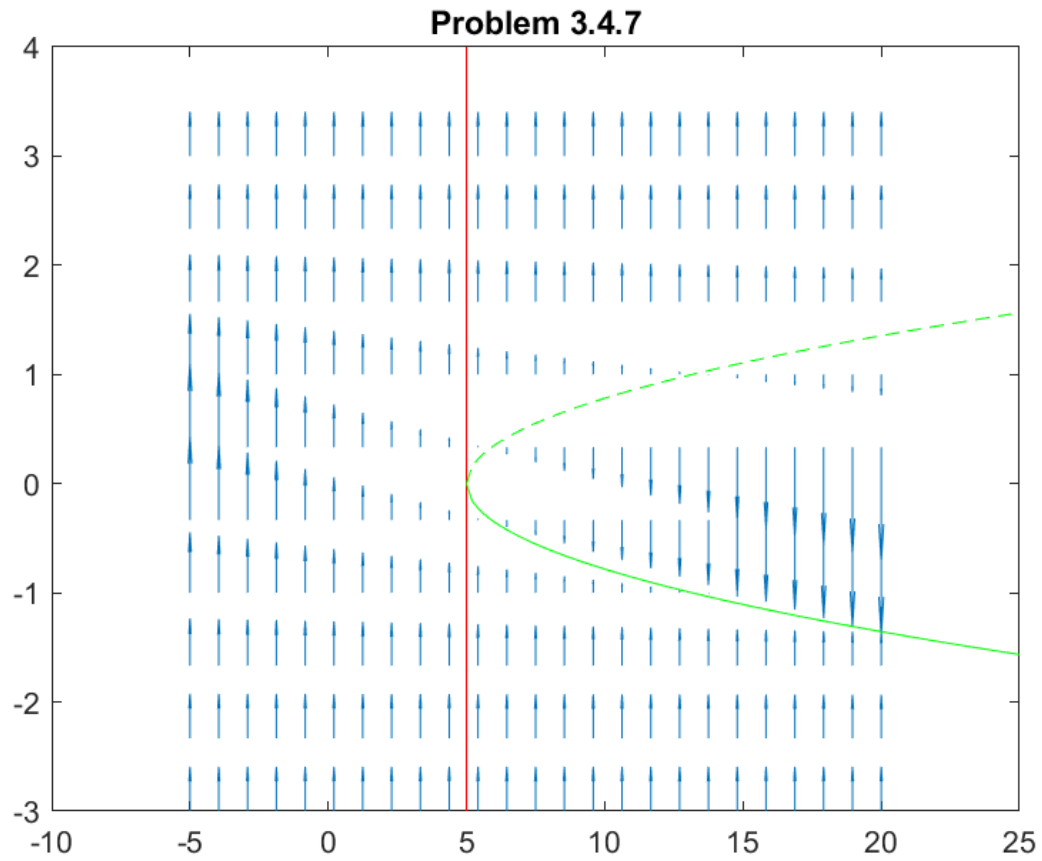


Figure 11: Bifurcation Diagram for problem 3.4.7

4.4 Pblm 3.4.9

$$\dot{x} = x + \tanh(rx)$$

As is evident in the bifurcation diagram, the pitchfork diagram is subcritical. r_c was found by differentiating $f(x, r)$, solving for x_c , substituting back into $f(x, r)$ and then solving for $r_c = -1$.

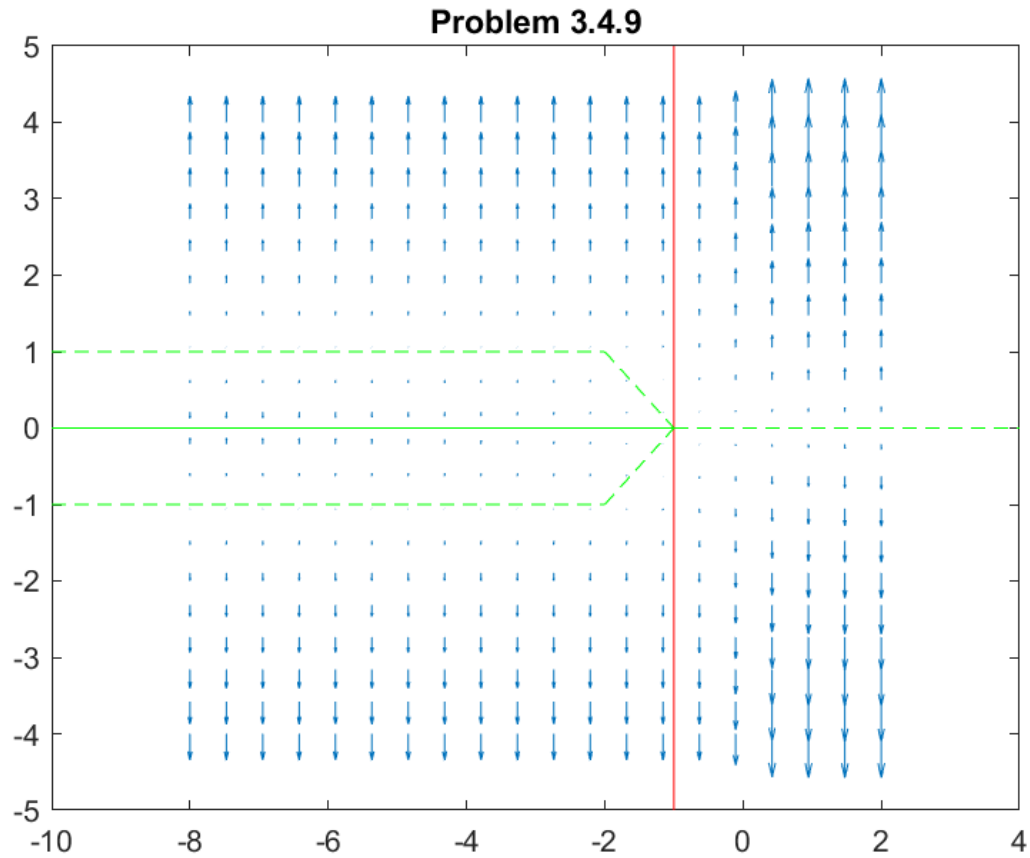


Figure 12: Bifurcation Diagram for problem 3.4.9

A MATLAB Code:

All code I write in this course can be found on my GitHub repository:

<https://github.com/jonaswagner2826/MECH6313>

Script 1: MECH6313_HW1

```
1 %% MECH6313 - HW 1
2 clear
3 close all
4
5 pblm1 = false;
6 pblm2 = false;
7 pblm4 = true;
8
9 if pblm1
10 %% Problem 1
11 % System Def
12 delta = 0.05;
13 alpha = 0.4;
14 omega_t = 1.3;
15 sys1 = nlsys(@duff_eq);
16
17 % Simulation Setup
18 x_0 = [0;0];
19
20 N = 1e4;
21 t_step = 0.01;
22 t_max = N * t_step - t_step;
23 T = reshape(0:t_step:t_max,N,1);
24 U = alpha * cos(omega_t * T);
25 SYS1 = nlsim(sys1,U,T,x_0);
26
27 % Phase Plot
28 fig = SYS1.phasePlot(1,2,'Problem 1 - Phase Plot (Relaxed System)');
29 saveas(fig,fullfile([pwd '\\' 'HW1' '\\' 'fig'],'pblm1_phase.png'))
30
31 % Phase Plot Comparison
32 N = 5e3;
33 fig = figure('position',[0,0,1000,1000]);
34 for i = 1:4
35     x_0 = randn(2,1);
36     SYS(i) = nlsim(sys1,U,T,x_0);
37     ax = subplot(2,2,i);
38     SYS(i).phasePlot(1,2,x_0,fig,ax);
```

```

39 end
40 sgtitle('Problem 1 - Duff Equation Phase Portrait Comparrision')
41 saveas(fig,fullfile([pwd '\\' 'HW1' '\\' 'fig'],'pblm1_phase_comparrision.png'))
42
43 % Vs Time Plot
44 fig = SYS1.plot(-1,0,0);
45 sgtitle('Problem 1 - Duff Equation Time Simulation')
46 saveas(fig,fullfile([pwd '\\' 'HW1' '\\' 'fig'],'pblm1_vs_time.png'))
47
48 end
49
50 if pblm2
51 %% Problem 2
52 % System Def
53 sys2 = nlsys(@van_der_pol);
54
55 % Simulation Setup
56 x_0 = [0.8; -0.2];
57
58 N = 5e3;
59 t_step = 0.01;
60 t_max = N * t_step - t_step;
61 T = reshape(0:t_step:t_max,N,1);
62 U = 0 * T;
63 SYS2 = nlsim(sys2,U,T,x_0);
64
65 % Phase Plot
66 fig = SYS2.phasePlot(1,2,'Problem 2 - Phase Plot');
67 saveas(fig,fullfile([pwd '\\' 'HW1' '\\' 'fig'],'pblm2_phase.png'))
68
69 % Phase Plot Comparrision
70 fig = figure('position',[0,0,1000,1000]);
71 X_0 = [ -0.5,0.8, -1.5, 3;
72        0.5, -0.5, 2.7, -1.9];
73 for i = 1:4
74     SYS(i) = nlsim(sys2,U,T,X_0(:,i));
75     ax = subplot(2,2,i);
76     SYS(i).phasePlot(1,2,X_0(:,i),fig,ax);
77 end
78 sgtitle('Problem 2 - Van Der Pol Phase Portraits')
79 saveas(fig,fullfile([pwd '\\' 'HW1' '\\' 'fig'],'pblm2_phase_comparrision.png'))
80
81 % Vs Time Plot

```

```

82 x_0 = X_0(1);
83 fig = SYS2.plot(-1,0,0);
84 sgttitle('Problem 2 - Van Der Pol Time Simulation')
85 saveas(fig,fullfile([pwd '\\ 'HW1' '\\ 'fig'],'pblm2_vs_time.png'))
86
87
88 % Negative Equivelent
89 sys2_neg= nlsys(@van_der_pol,'empty',0,-1,0,0,-1);
90
91 % Negative Stability
92 x_0 = [0;0];
93 u_0 = 0;
94 sys2_neg_lin = sys2_neg.ss(x_0,u_0)
95 eig_A = eig(sys2_neg_lin.A)
96 is_stable = isstable(sys2_neg_lin)
97
98
99
100 %Negative Sim
101 x_0 = [0.8; -0.2];
102 SYS2_neg = nlsim(sys2_neg,U,T,x_0)
103
104
105 % Phase Plot Comparrison
106 fig = figure('position',[0,0,1000,1000]);
107 %X_0(i) is from last set of plot
108 for i = 1:4
109     SYS_neg(i) = nlsim(sys2_neg,U,T,X_0(:,i));
110     ax = subplot(2,2,i);
111     SYS_neg(i).phasePlot(1,2,X_0(:,i),fig,ax);
112 end
113 sgttitle('Problem 2 - Negative Van Der Pol Phase Portraits')
114 saveas(fig,fullfile([pwd '\\ 'HW1' '\\ 'fig'],'pblm2_phase_comparrision_neg.png'))
115
116 % Vs Time Plot
117 x_0 = X_0(1);
118 fig = SYS2_neg.plot(-1,0,0);
119 sgttitle('Problem 2 - Negative Van Der Pol Time Simulation')
120 saveas(fig,fullfile([pwd '\\ 'HW1' '\\ 'fig'],'pblm2_vs_time_neg.png'))
121
122 end
123
124

```

```

125 if pblm4
126 %% Problem 4
127
128 syms f(x,r) g(x,r)
129
130 % Problem 3.4.2
131 sys342 = nlsys(@pblm342);
132
133
134 r = linspace(-2,5,20);
135 x = linspace(-4,4,20);
136 [r_c,fig] = sys342.bifurcationPlot(r,x);
137
138
139 xlimit = xlim;
140 plot([xlimit(1),r_c],[0,0],'g');
141 plot([r_c,xlimit(2)],[0,0],'g--');
142
143 x = linspace(r_c,xlimit(2));
144 y = 2 * sqrt(x-r_c);
145 plot(x,y,'g-')
146 plot(x,-y,'g-')
147 hold off
148
149 title('Problem 3.4.2')
150
151 saveas(fig,fullfile([pwd '\\ ' 'HW1' '\\ ' 'fig'], 'pblm4_342.png'))
152
153
154 % Problem 3.4.4
155 sys344 = nlsys(@pblm344);
156 r = linspace(-5,3,20);
157 x = linspace(-4,4,20);
158 [r_c,fig] = sys344.bifurcationPlot(r,x);
159
160
161 xlimit = xlim;
162 plot([xlimit(1),r_c],[0,0],'g')
163 plot([r_c,xlimit(2)],[0,0],'g--')
164
165 x = linspace(xlimit(1),r_c);
166 y = 1 * sqrt(abs(x-r_c));
167 plot(x,y,'g--')

```

```

168 plot(x,-y,'g--')
169 hold off
170
171 title('Problem 3.4.4')
172
173 saveas(fig,fullfile([pwd '\\ 'HW1' '\\ 'fig'],'pblm4_344.png'))
174
175 % Problem 3.4.7
176 sys347 = nlsys(@pblm347);
177 r = linspace(-5,20,25);
178 x = linspace(-3,3,10);
179 [r_c,fig] = sys347.bifurcationPlot(r,x);
180
181
182 xlimit = xlim;
183 x = linspace(r_c,xlimit(2));
184 y = 0.35 * sqrt(x-r_c);
185 plot(x,y,'g--')
186 plot(x,-y,'g-')
187 hold off
188
189 title('Problem 3.4.7')
190
191 saveas(fig,fullfile([pwd '\\ 'HW1' '\\ 'fig'],'pblm4_347.png'))
192
193 % Problem 4.4.9
194 sys349 = nlsys(@pblm349);
195 r = linspace(-8,2,20);
196 x = linspace(-4,4,20);
197 [r_c,fig] = sys349.bifurcationPlot(r,x);
198
199
200 xlimit = xlim;
201 x = linspace(xlimit(1),-2);
202 y = 0 * x;
203 plot(x,y-1,'g--');
204 plot(x,y+1,'g--');
205 plot(x,y,'g-');
206
207 x = linspace(-2,r_c);
208 y = 0 * x;
209 plot(x,y-x-1,'g--');
210 plot(x,y+x+1,'g--');

```

```

211 plot(x,y,'g-');
212
213 x = linspace(r_c,xlimit(2));
214 y = 0*x;
215 plot(x,y,'g--');
216
217 hold off
218
219 title('Problem 3.4.9')
220
221 saveas(fig,fullfile([pwd '\\ 'HW1' '\\ 'fig'],'pblm4_349.png'))
222
223 end
224
225
226
227
228
229
230
231 %% Local Functions
232 function y = duff_eq(x,u,parms)
233     % DUFF_EQ nonlin function with caotic behavior
234     arguments
235         x (2,1) = [0; 0];
236         u (1,1) = 0;
237         parms = false
238     end
239
240     if parms == false
241         delta = 0.05;
242     else
243         delta = parms(1);
244     end
245
246     % Array sizes
247     n = 2; % Number of states
248     p = 1; % Number of inputs
249
250     % State Update Equations
251     y(1,1) = x(2);
252     y(2,1) = - delta * x(2) + x(1) - x(1)^3 + u;
253

```

```

254
255     if nargin ==0
256         y = [n;p];
257     end
258 end
259
260 function y = van_der_pol(x,u,parms)
261     % VAD_DER_POL nonlin function
262     arguments
263         x (2,1) = [0; 0];
264         u (1,1) = 0;
265         parms = false
266     end
267
268     if parms == false
269         a = 1;
270     else
271         a = parms(1);
272     end
273
274     % Array sizes
275     n = 2; % Number of states
276     p = 1; % Number of inputs
277
278     % State Update Equations
279     y(1,1) = x(2);
280     y(2,1) = - a * (x(1)^2 -1) * x(2) - x(1) + u;
281
282
283     if nargin ==0
284         y = [n;p];
285     end
286 end
287
288
289 function y = pblm342(x,u,parms)
290     % VAD_DER_POL nonlin function
291     arguments
292         x (1,1) = 0;
293         u (1,1) = 0;
294         parms = false
295     end
296

```

```

297     if parms == false
298         r = 1;
299     else
300         r = parms(1);
301     end
302
303     % Array sizes
304     n = 1; % Number of states
305     p = 1; % Number of inputs
306
307     % State Update Equations
308     y(1,1) = r * x(1) - sinh(x(1));
309
310
311     if nargin ==0
312         y = [n;p];
313     end
314 end
315
316
317 function y = pblm344(x,u,parms)
318     % VAD_DER_POL nonlin function
319     arguments
320         x (1,1) = 0;
321         u (1,1) = 0;
322         parms = false
323     end
324
325     if parms == false
326         r = 1;
327     else
328         r = parms(1);
329     end
330
331     % Array sizes
332     n = 1; % Number of states
333     p = 1; % Number of inputs
334
335     % State Update Equations
336     y(1,1) = x(1) + (r*x(1))/(1+x(1)^2);
337
338
339     if nargin ==0

```



```

340     y = [n;p];
341 end
342 end
343
344
345 function y = pblm347(x,u,parms)
346     % VAD_DER_POL nonlin function
347     arguments
348         x (1,1) = 0;
349         u (1,1) = 0;
350         parms = false
351     end
352
353     if parms == false
354         r = 1;
355     else
356         r = parms(1);
357     end
358
359     % Array sizes
360     n = 1; % Number of states
361     p = 1; % Number of inputs
362
363     % State Update Equations
364     y(1,1) = 5 - r * exp(-x(1)^2);
365
366
367     if nargin ==0
368         y = [n;p];
369     end
370 end
371
372 function y = pblm349(x,u,parms)
373     % VAD_DER_POL nonlin function
374     arguments
375         x (1,1) = 0;
376         u (1,1) = 0;
377         parms = false
378     end
379
380     if parms == false
381         r = 1;
382     else

```

```

383     r = parms(1);
384 end
385
386 % Array sizes
387 n = 1; % Number of states
388 p = 1; % Number of inputs
389
390 % State Update Equations
391 y(1,1) = x + tanh(r*x(1));
392
393
394 if nargin ==0
395     y = [n;p];
396 end
397 end

```