Dissipative Solitons In A Generalized Coupled Cubic-Quintic Ginzburg-Landau Equations

Author: Gholam-Ali Zakeri Emmanuel Yomba Case 2A.

```
% Code
clc;
clear all;
close all;

% Figure 6
% parameter definition
gamma1 = -0.7i;
gamma2 = 0.75-1.1i;
delta1 = 2.4+0.6i;
delta2 = 2.75-i;
xi1 = 1.6+1.6i;
beta2 = -3-i;
% due to the eqivalence relation showsn
% in constrains for 2A for computing
% real xi we assign xi2 to be equal to
% beta 2
xi2 = beta2;
chi1 = 1.2;
b = 3;
v = 3;

% solve for alpha1 and alpha2 roots. Note:
% alpha1 and alpha2 must be less than 0
flag = 1;
p = [-4*real(delta1),8*imag(delta1), 3*real(delta1)];
alpha1_roots = roots(p);

p = [4*((-real(gamma2)*imag(delta2)) +
 (imag(gamma2)*real(delta2))), ...
    -8*((imag(gamma2)*imag(delta2))+(real(gamma2)*real(delta2))), ...
    3*real(gamma2)*imag(delta2)- 3*imag(gamma2)*real(delta2)];
alpha2_roots = roots(p);

alpha1 = alpha1_roots(alpha1_roots < 0);
alpha2 = alpha2_roots(alpha2_roots < 0);

A = @(x, t, n, r, w, k1, omega1, alpha1, b) ...
    (n * exp((r*x+w*t) + 1i*(k1 * x - omega1 * t)))/...
    ((1+b*exp(2*(r*x+w*t)))^(0.5 * i*alpha1));

B = @(x, t, mu, r, w, k2, omega2, alpha2, b) ...
    (mu * exp((r*x+w*t) + 1i*(k2 * x - omega2 * t)))/...
    ((1+b*exp(2*(r*x+w*t)))^(0.5 * i*alpha2));

omega1 = @(chi1, alpha1, r, v)...
```

```
        ((-chi1)/(4*alpha1))+alpha1 * (-r*v+chi1);
omega2 = @(chi1, alpha1, alpha2, gamma1, gamma2, r, v) ...
        (-4*r*alpha1*alpha2*imag(gamma1)*(v + r * real(gamma2)) + (-1
 + 4 * (alpha2^2))*imag(gamma2)*chi1)...
        /(4 * alpha1 * imag(gamma1))

w = @(r, v)...
    r * v;

rsq = @(chi1, alpha1, gamma1)...
    (chi1)/(alpha1 * imag(gamma1));

k2 = @(r, alpha2)...
    r * alpha2;

k1 = @(r, alpha1, gamma1, v)...
    r*alpha1 - (v/imag(gamma1));

nsq = @(W1)...
    2*sqrt(2) * W1;

musq = @(W1, xi2, beta2)...
    (-2 * sqrt(2) * W1 * imag(xi2))/(imag(beta2));

W1 = @(chi1, delta1, b)...
    sqrt((b*chi1)/real(delta1));

% constraints
% careful with this
xi2r = @(beta2, xi2i) ...
    ((real(beta2)*imag(xi2i))/imag(beta2));

beta1i = @(xi1,xi2,beta2)...
    ((imag(xi1) * imag(xi2))/(imag(beta2)))

beta1r = @(xi1,xi2,beta2)...
    ((real(xi1) * imag(xi2))/(imag(beta2)))

beta1 = @(xi1,xi2,beta2)...
    beta1r(xi1,xi2,beta2) + i*beta1i(xi1,xi2,beta2);

chi2 = @(alpha1, alpha2, gamma1, gamma2, chi1)...
    ((4*alpha2 * imag(gamma2) - real(gamma2) + 4*(alpha2^2)*
 real(gamma2))*chi1)/...
    (4 * alpha1 * imag(gamma1));

s8 = @(alpha2, gamma2)...
    8 * alpha2 * imag(gamma2) + (-3+4*(alpha2^2)) * real(gamma2)

gamma1_i = @(s8, alpha1, gamma2, beta2, delta1, delta2, chi2)...
    (s8(alpha2, gamma2) * (imag(beta2)^2) * real(delta1))/...
    (8 * alpha1 * real(delta2) * (imag(chi2)^2));

% Define Required Variables
```

```
t_xi2i = 0;
t_r = sqrt(rsq(chi1, alpha1, gamma1));
t_omega1 = omega1(chi1, alpha1, t_r, v)
t_omega2 = omega2(chi1, alpha1, alpha2, gamma1, gamma2, t_r, v)
t_W1 = W1(chi1, delta1, b)
t_n = sqrt(nsq(t_W1))
t_k1 = k1(t_r, alpha1, gamma1, v)
t_k2 = k2(t_r, alpha2)
t_w = w(t_r, v)
t_xi2r = xi2r(beta2, imag(t_xi2i))
t_xi2 = t_xi2r + i*t_xi2i
t_mu = sqrt(musq(t_W1, xi2, beta2))

% Run
t = 0:1/8:5;
[X,Y] = meshgrid(-20:1:20, -0:1/40:1);
t = meshgrid(t);
amplitude = [];
amplitude_b = [];


for i = 1:length(X)
   for j = 1:length(t)
        if t(1, i) == 0 || X(1, i) == 0
            t_x = X(1, i);
            t_t = t(1, i);
            amplitude(i, j) =  A(t_x, t_t, t_n, t_r, t_w, t_k1,
 t_omega1, alpha1, b) + ...
                A(t_x, t_t, t_n, t_r, t_w, t_k1, t_omega1, alpha1, b)
 * ((0.1 *rand(1)));

            amplitude_b(i, j) =  B(t_x, t_t, t_mu, t_r, t_w, t_k2,
 t_omega2, alpha2, b) + ...
                B(t_x, t_t, t_mu, t_r, t_w, t_k2, t_omega2, alpha2, b)
 * ((0.1 * rand(1)));
        else
            amplitude(i, j) =  A(t_x, t_t, t_n, t_r, t_w, t_k1,
 t_omega1, alpha1, b);
            amplitude_b(i, j) =  B(t_x, t_t, t_mu, t_r, t_w, t_k2,
 t_omega2, alpha2, b);
        end
   end
end

% A = amplitude;
% B = amplitude_b;
% [Ax, Ay] = gradient(A);
% [Axx, Axy] = gradient(Ax);
%
% [Bx, By] = gradient(B);
% [Bxx, Bxy] = gradient(Bx);
%
% amplitude = chi1 .* A + gamma1 .* Axx - beta1(xi1,xi2,beta2) *
 (abs(A).^2) * A - delta1 * (abs(A).^4) * A - xi1 * (abs(B).^2) * A;
```

```
% amplitude_b = chi2(alpha1, alpha2, gamma1, gamma2, chi1) .* B +
 gamma2 .* Bxx - beta2 * (abs(B).^2) * B - delta2 * (abs(B).^4) * B -
 xi2 * (abs(A).^2) * B;

% Draw
amplitude = (abs(amplitude));
amplitude_b = (abs(amplitude_b));

figure(1);
subplot(2,1,1)
colormap(cool);
s1 = surf(t,Y,amplitude,'FaceAlpha',1);
% s1.EdgeColor = 'none';
%title(sprintf('2D wave equation at t = %1.2f, con sigma = %1.2f y
 gamma = %1.2f',t(j),sigma, gamma),'Fontsize',11);
title("dark-dark solitary waves for eq(1) (Perturbed) A(x, t)");
xlabel('time'); ylabel('x'); zlabel("|A|^2");
%zlabel(sprintf('u(x,y,t = %1.2f)',t(j)),'Fontsize',11);
%axis ([0 1 0 1 -1 1]);

subplot(2,1,2)
colormap(winter);
s2 = surf(t,Y,amplitude_b,'FaceAlpha',1);

%title(sprintf('2D wave equation at t = %1.2f, con sigma = %1.2f y
 gamma = %1.2f',t(j),sigma, gamma),'Fontsize',11);
title("dark-dark solitary waves for eq(2) (Perturbed) B(x, t)");
xlabel('time'); ylabel('x'); zlabel("|A|^2");
```

```
Error using evalin
Undefined function or variable 'GinzburgLandau'.
```

Published with MATLAB® R2018b