Problem set 4

Jonas Lauri

December 2021

1 4.3a) - d)

```
%% Renyi dimension of Hénon map.
% Parameters.
clearvars
tic
a = 1.4; b = 0.3; nMax = 1000; sqrtInit = 100;
numberOfBoxes = [2*10^3, 10^3, 7*10^2, 5*10^2, 3*10^2, 10^2, 5*10^1];
Iqs = zeros(length(numberOfBoxes), 9);
epsilons = zeros(length(numberOfBoxes), 1);

for epsVal = 1:length(numberOfBoxes)

% Initialization.
x0 = linspace(-0.1, 0.1, sqrtInit); y0 = linspace(-0.1, 0.1, sqrtInit);
[x0s, y0s] = meshgrid(x0, y0);
xList = zeros(sqrtInit, sqrtInit, nMax);
yList = zeros(sqrtInit, sqrtInit, nMax);

% Main loop.
for i = 1:sqrtInit
    for j = 1:sqrtInit
        x = zeros(1, nMax);
        y = zeros(1, nMax);
        x(1) = x0s(i, j);
        y(1) = y0s(i, j);
        for n = 1:(nMax - 1)
            x(n + 1) = y(n) + 1 - a*x(n).^2;
            y(n + 1) = b*x(n);
        end
        xList(i, j, :) = x(:);
        yList(i, j, :) = y(:);
    end
end
```

```
% Make boxes.
xMin = min(min(min(xList))); xMax = max(max(max(xList)));
boxXs = linspace(xMin, xMax, numberOfBoxes(epsVal));
boxYs = linspace(xMin, xMax, numberOfBoxes(epsVal));
% Use xMin/xMax again as I want boxes to be squares.
boxesCounter = zeros(numberOfBoxes(epsVal), numberOfBoxes(epsVal));
for i = 1:sqrtInit
    for j = 1:sqrtInit
        for n = 25:nMax
         xIndex = floor((xList(i, j, n) - xMin)/(xMax - xMin) * numberOfBoxes(epsVal));
         if (xIndex == 0)
             xIndex = 1;
         end
         yIndex = floor((yList(i, j, n) - xMin)/(xMax - xMin) * numberOfBoxes(epsVal));
         boxesCounter(yIndex, xIndex) = boxesCounter(yIndex, xIndex) + 1;
        end
    end
end

% Calculation of Iq for q  [0, 4].
Iq = zeros(1, 9);
Ntot = sqrtInit^2*nMax;
epsilon = (xMax - xMin) / numberOfBoxes(epsVal);
% q = 0.
for j = 1:numberOfBoxes(epsVal)^2
    if (boxesCounter(j) ~= 0)
        Iq(1) = Iq(1) + 1;
    end
end

% q = 1.
for j = 1:numberOfBoxes(epsVal)^2
    if (boxesCounter(j) ~= 0)
        Iq(2) = Iq(2) + (boxesCounter(j)/Ntot)*log(Ntot/boxesCounter(j));
    end
end

% q = 2.
for j = 1:numberOfBoxes(epsVal)^2
    Iq(3) = Iq(3) + (boxesCounter(j)/Ntot)^2;
end

% q = [0.5, 1.5, 2.5, 3, 3.5, 4].
Q = [0.5, 1.5, 2.5, 3, 3.5, 4];
for q = 1:length(Q)
    for j = 1:numberOfBoxes(epsVal)^2
```

```
        Iq(q + 3) = Iq(q + 3) + (boxesCounter(j)/Ntot)^Q(q);
    end
end

Iqs(epsVal, :) = Iq;
epsilons(epsVal) = epsilon;

disp(epsVal)

end

% Fit lines to the different Iqs.
D0 = polyfit(log(1./epsilons), (1-0)^(-1) * log(Iqs(:, 1)), 1);
D1 = polyfit(log(1./epsilons), Iqs(:, 2), 1);
D2 = polyfit(log(1./epsilons), (1-2)^(-1) * log(Iqs(:, 3)), 1);
% c) The slopes are the values of [D0, D1, D2] which are [1.24, 1.21, 1.20].
Dqs = zeros(6, 1);
for q = 1:6
    [p, ~] = polyfit(log(1./epsilons), (1-Q(q))^(-1) * log(Iqs(:, q + 3)), 1);
    Dqs(q) = p(1);
end

toc

%% a) Plot of the points.
hold on
title('Approximation of the fractal attractor of the Hénon map.')
for i = 1:sqrtInit
    for j = 1:sqrtInit
        plot(squeeze(xList(i, j, 25:end)), squeeze(yList(i, j, 25:end)), '.')
    end
end
xlabel('x')
ylabel('y')
hold off

%% Plot of the boxes.
set(gca, 'YDir', 'normal')
colormap(flipud(gray))
imagesc(boxesCounter > 100)
pbaspect([1 1 1])
xlabel('x')
ylabel('y')
title("Plot of the boxes.")

%% b) Loglog-plot of Iqs.
```

```
hold on
plot(log(1./epsilons), (1-0)^(-1) * log(Iqs(:, 1)), '-o')
plot(log(1./epsilons), Iqs(:, 2), '-+')
plot(log(1./epsilons), (1-2)^(-1) * log(Iqs(:, 3)), '-*')
legend('q = 0', 'q = 1', 'q = 2', 'Location', 'northwest')
xlabel('ln(1 / \epsilon)')
title('(1 - q)^{-1} ln[I(q, \epsilon)] / \Sigma_1^{N_{boxes}} [p_k ln(1 / p_k)]')
hold off

%% Loglog-plot of Dqs.
hold on
plot(log(1./epsilons), (1-0)^(-1) * log(Iqs(:, 1)), '-o')
plot(log(1./epsilons), Iqs(:, 2), '-+')
plot(log(1./epsilons), (1-2)^(-1) * log(Iqs(:, 3)), '-*')
markers = {'>','pentagram','s','d','^','v'};
for q = 1:6
    plot(log(1./epsilons), (1-Q(q))^(-1) * log(Iqs(:, q + 3)), 'Marker', markers{q})
end
legend('q = 0', 'q = 1', 'q = 2','q = 0.5', 'q = 1.5', 'q = 2.5','q = 3', 'q = 3.5',
'q = 4', 'Location', 'northwest')
xlabel('ln(1 / \epsilon)')
title('(1 - q)^{-1} ln[I(q, \epsilon)] / \Sigma_1^{N_{boxes}} [p_k ln(1 / p_k)]
(The entries in the legend are NOT in the order of q value)')
hold off

%% d) Dqs vs q.
hold on
plot(0, D0(1), '-o')
plot(1, D1(1), '-+')
plot(2, D2(1), '-*')
markers = {'>','pentagram','s','d','^','v'};
for q = 1:6
    plot(Q(q), Dqs(q), 'Marker', markers{q})
end
legend('q = 0', 'q = 1', 'q = 2','q = 0.5', 'q = 1.5', 'q = 2.5','q = 3', 'q = 3.5',
'q = 4', 'Location', 'southwest')
xlabel('q')
title('D_q vs q (The entries in the legend are NOT in the order of q value)')
hold off
```

2 4.3a) - d)

```
%% e) Calculation of the Lyapunov exponents.
% Parameters.
clearvars
```

```
tic
a = 1.4; b = 0.3;
nMax = 10^6;
ns = linspace(1, nMax, nMax);

% Initialization.
x0 = 0.1; y0 = 0.1;
x = zeros(1, nMax); y = zeros(1, nMax);
x(1) = x0; y(1) = y0;

% Calculation of the trajectories.
for n = 1:(nMax - 1)
    x(n + 1) = y(n) + 1 - a*x(n).^2;
    y(n + 1) = b*x(n);
end
Q = eye(2);
lambda = zeros(1, 2);
lambdaList = zeros(nMax, 2);

% Calculation of the eigenvalues.
for i = 1:nMax
    traj = [x(i), y(i)];
    J = [-2*a*traj(1), 1; b, 0];
    M = J*eye(2);
    [Q,R] = qr(M*Q);
    lambda(1) = lambda(1) + 1/nMax*log(abs(R(1,1)));
    lambda(2) = lambda(2) + 1/nMax*log(abs(R(2,2)));
    lambdaList(i, :) = nMax/i*lambda;
end

% f) Calculation of the Lyapunov exponents using the Kaplan-Yorke conjecture.
DL = 1 - lambda(1)/lambda(2);

toc

%% Plot of the development of the Lyapunov exponents.
plot(log(ns), lambdaList)
title('Approximation of the Lyapunov exponents \lambda_1 and \lambda_2.')
```