Computer Simulation and Applications

September 30
Period three window discussion; $a=3.8$; $f^3$
Period three window discussion; $a = 3.82$; $f^3$
Period three window discussion; $a=3.828$; $f^3$
Period three window discussion; $a = 3.8282; f^3$
Period three window discussion; $a=3.8282$; 

orbit of $x_0=0.25$

Quasi period-3 orbit = “ghost” 3 cycle
Period three window discussion; $a = 3.8282$; $f$; orbit of $x_0 = 0.25$

Time Series with Lines:

Quasi period-3 orbit = “ghost” 3 cycle
Period three window discussion; $a=3.8284$; $f^3$; $1+\sqrt{8} \approx 3.8284$ !!

Tangent Bifurcation Value of $a = 1+\sqrt{8}$
Period three window discussion; $a=3.83$; $f^3$
Period three window discussion; \(a = 3.835\); \(f^3\)

- Three open dots: unstable period-3 orbit
- Three closed dots: stable period-3 orbit
Period three window discussion; $a=3.84$; $f^3$
Period three window discussion; $a=3.86$; $f^3$

investigating period-3 window of the logistic map

$f^3$ for $a=3.86$, $f$ as usual $ax(1-x)$
Period three window discussion; $a=3.88$; $f^3$
chaos

How can we define it?
Chaos (χαος)

• Chaotic systems:
  – are *aperiodic*
  – Exhibit *sensitive dependence* on *initial conditions* (thus unpredictable in the long term)
  – Governed by one or more *control params*, a small change in which can cause chaos to appear or disappear
  – Their governing equations are *nonlinear*
Chaos (χάος)

- Logistic map system can exhibit aperiodic orbits for certain parameter values.
- Chaotic? A system should have sensitive dependence on initial conditions, in the sense that neighboring orbits separate exponentially fast on average.
- A chaotic orbit is a bounded non periodic orbit that displays sensitive dependence.
- Discussion is simplified by requiring exponential divergence.
Liapunov Exponent: intuition

Let $x_0$ be some initial condition, and let $x_0 + \delta_0$ be a nearby point. ($\delta_0$ is extremely small)

$\delta_n$ is the separation after $n$ iterates.

Intuitive definition: if $| \delta_n | \approx | \delta_0 | e^{n\lambda}$, then $\lambda$ is called the **Liapunov Exponent**.

$\lambda > 0 \rightarrow$ chaos
Liapunov Exponent: more precisely

\[ |\delta_n| \approx |\delta_0| e^{n\lambda} \quad \text{Get useful formula for } \lambda: \]

Taking logs and noting that \( \delta_n = f^n(x_0 + \delta_0) - f^n(x_0) \) we get:

\[ \lambda \approx \frac{1}{n} \ln \frac{\delta_n}{\delta_0} = \frac{1}{n} \ln \left| \frac{f^n(x_0 + \delta_0) - f^n(x_0)}{\delta_0} \right|, \]

where we take: limit \( \delta_0 \to 0 \) for the last equality.

\[ (f^n)'(x_0) = \prod_{i=0}^{n-1} (f)'(x_i), \text{ where } x_i = f^i(x_0). \text{ Thus} \]

\[ \lambda \approx \frac{1}{n} \ln |(f^n)'(x_0)| = \frac{1}{n} \ln \left| \prod_{i=0}^{n-1} f'(x_i) \right| = \frac{1}{n} \sum_{i=0}^{n-1} \ln |f'(x_i)|; \lambda = \lim_{n \to \infty} \left( \frac{1}{n} \sum_{i=0}^{n-1} \ln |f'(x_i)| \right), \]
Liapunov Exponent: more precisely

Notice that $\lambda$ depends on $x_0$. It is the same though for all $x_0$ in the basin of a given attractor. For stable (=sink) fixed points and periodic orbits (cycles) $\lambda$ is negative, for chaotic attractors $\lambda$ is positive.

Show that for a period-$p$ orbit which is a sink (i.e., a stable $p$-cycle) the Liapunov Exponent $\lambda$ is negative. If period-$p$ orbit is superstable, then $\lambda = -\infty$.
function lambda=liapunov(aBegin, aEnd, stepsInA)

    x0 = 0.25; % for now, need to randomize
    x=0.25;

    a = linspace(aBegin, aEnd, stepsInA);
    lambda = zeros(stepsInA);
    sizeOfSum = 10000;

    for i=1:stepsInA
        sum = 0;
        x=x0; % for now, need to randomize
        for k=1:sizeOfSum
            sum = sum + log(abs(a(i)-2*a(i)*x));
            x = a(i)*x*(1-x);
        endfor
        lambda(i) = sum/sizeOfSum;
    endfor

    plot(a,lambda);

endfunction

Approximation of Liapunov Exponent
Universality: qualitatively and quantitatively unimodal
Universality: qualitatively and quantitatively

- **Qualitative**: $x_{n+1} = f(x_n)$ where $f$ unimodal
- as you vary the parameter $a$ the order in which stable periodic solutions appear is *independent* of the unimodal map being iterated: the periodic attractors always occur in the same sequence, now called the U-sequence
- Algebraic form irrelevant! Only overall shape matters
- 1, 2, 2x2, 6, 5, 3, 2, 2x3, 5, 6, 4, 6, 5, 6, ....
Universality: qualitatively and quantatively

- Discussion of Quantitative universality:
  - $\delta \approx 4.669...$
  - $\alpha \approx -2.5029...$
- Period doubling has been measured in a variety of experimental systems
- Superstable sequence: $R_n$
Fractals
Many reasons they show up

\[(\sigma_{n+1}) = \Phi(\sigma_n)\]
// Appendix

// a C++ program computing
// the coordinates for the
// bifurcation diagram

#include <fstream>
#include <sstream>
#include <cstdlib>

#include <iostream>
using namespace std;

double logisticAXtimes1MinX(double a, double x)
{
    double result = a*x*(1.0-x);
    return result;
}

struct infoA {
    double beginA;
    double endA;
    int steps;
};

/* */ /* for the following it is better to package arguments: 1 param instead of 3 param etc */

void bifurcationDiagram(
    double beginA,
    double endA,
    int steps,
    int iterationsForAnOrbit,
    int cutOff,
    double x0,
    double (*pt2Func)(double, double),
    ostream & bifurcation
    // double x0=0.25 // has to be rightmost
)
{
    // in future will generate x0 randomly
    double x=x0; double a=beginA; double da=(endA-beginA)/(double)steps;

    for (int i=0; i<steps+1; i++){
        x = x0; // in future generate x0 randomly; x=x0;
        for (int i=0; i<cutOff; i++){
            x = pt2Func(a, x);
        }
        for (int k=cutOff; k < iterationsForAnOrbit; k++){
            x = pt2Func(a,x);
            bifurcation << a << " " << x << "\n";
        }
        a = a + da;
    }
}

int main () {
    ofstream coordinatesOfBifurcationPlot("bifurcation_plot", ios::out); //ios::app
    bifurcationDiagram(0.0, 4.0, 1000, 10000, 300, 0.25, &logisticAXtimes1MinX, coordinatesOfBifurcationPlot);
    //bifurcationDiagram(0.0, 2.0, 1000, 10000, 300, 0.25, &logisticAXtimes1MinX, cout);
    return 1;
}