

The Fibonacci Spiral and Pseudospirals**Cye H. Waldman**cye@att.net

Copyright 2013

Introduction

For the purpose of this paper we adopt the usual definition of a spiral, but with a caveat. The standard definition of a spiral is a curve on a plane that winds around a fixed center point at a continuously increasing or decreasing distance from the point. The caveat is an auxiliary stipulation that the curvature is monotonic.

We define pseudospirals as those composed of adjoining circular arcs. These may or may not meet the condition of continuously increasing distance from the center point, but they certainly violate the auxiliary condition.

Thus, by our definition, the classical Fibonacci spiral, shown in Figure 1, is a pseudospiral insofar as it is composed of quarter-circular segments with radii increasing in accordance with the Fibonacci sequence. It is typically drawn on “Fibonacci graph paper” in order to emphasize both the Fibonacci sequence and the quarter-circular structure, neither of which would be very apparent without it. Computer programs we have found for drawing the Fibonacci spiral use rather obtuse algorithms for determining where to place the arcs. We can do better than that.

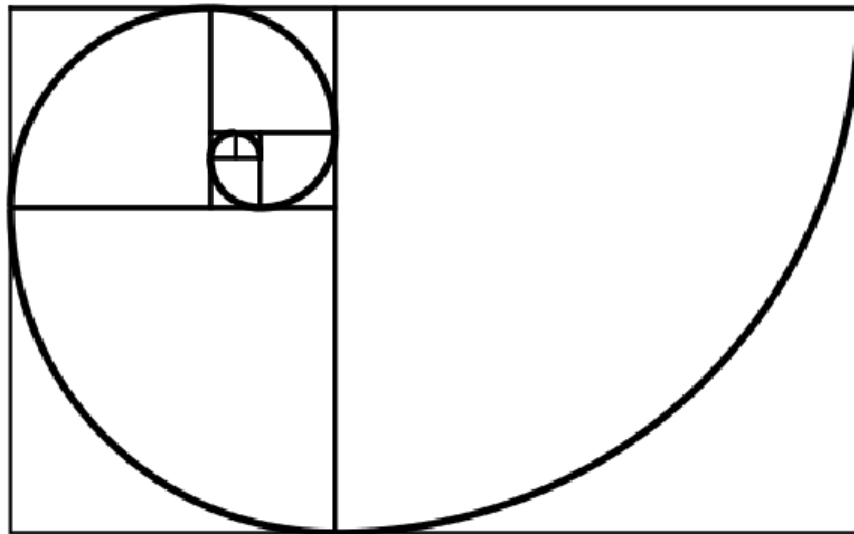


Figure 1: The classical Fibonacci spiral is composed of quarter-circular arcs.

In this paper we present a simplified algorithm for the Fibonacci spiral and a closed-form analytic solution (and by closed-form we mean specifically a finite number of well-known functions) for an arbitrary real number sequence (integers, rationals, irrationals, positives, and

negatives), in any order, as well as arbitrary circular arcs. However, for the present we shall limit the exercise to the same angular arc for all segments. The Fibonacci spiral will be seen to be just one of an infinitude of pseudospirals.

The Fibonacci spiral

The few algorithms for the Fibonacci spiral that we have found seem to try to emulate the mechanical drawing process and get bogged down in managing where they were last and where they are going next (up-down-left-right, or compass points to some). It boils down to where to put the origin for next circular arc and where its (angular) starting point is. Assuming the “center” point of spiral is at the origin and the spiral is unfolding in the counterclockwise direction (as in Figure 1), then we can easily write an algorithm for this in the complex plane. The pseudo-code for the operation is as follows:

```

z0 = 0+0i;
z = [ ];
Θ = [0 π/2];
θ = variable over Θ
r = Fibonacci(1)
repeat
    z = [z;r*exp(i* θ)+z0]
    r = Fibonacci(next)
    z0 = z(end) - r* exp[i* θ(end)]
    Θ = Θ + π/2
    θ = variable over Θ
until end

```

The crux of the algorithm is in the statement $z_0 = z(\text{end}) - r \exp[i \theta(\text{end})]$, which uses the complex variable to automatically compute the origin for the next arc; the exponential sets the direction from the last point on the current arc and the Fibonacci sequence gives the distance. Notice that we always move into the concave side of the curve for positive r . Later we will address the possibility of “negative” radius.

The associated animations shows the Fibonacci spiral evolving out of “square” Fibonacci spiral (along the appropriate edges of the graph paper) and further evolving totally into the graph paper itself. The second animation show all the component curves.

The Fibonacci spiral is frequently regarded as an *approximation* to the golden spiral, which is a logarithmic spiral whose growth factor is ϕ , the golden ratio. We find this amusing because an approximation should be easier than that which is being approximated. But calculation of the Fibonacci spiral is a cockamamie process as you have just seen, whereas the logarithmic spiral is a straightforward equation. A comparison of the two curves will be seen below.

We now turn our attention to development of a fully analytic solution for a general pseudospiral.

A closed-form analytic solution for pseudospirals

The above development of a rational algorithm for the Fibonacci spiral inspired some thinking that led to (a) an algorithm for arbitrary number sequences and (b) a path toward a true closed-form solution for any pseudospiral. In this section we present the derivation of that analytic solution based on the universal curve equation (a generalization of the polynomial spiral equation), which has been described elsewhere; the complete write-up is available at <http://curvebank.calstatela.edu/waldman4/waldman4.htm> .

The universal curve equation can be expressed in terms of the arc length, s or the tangent angle, θ as follows

$$\begin{aligned} z(s) &= \int e^{i \int \kappa(s) ds} ds \\ z(\theta) &= \int \rho(\theta) e^{i\theta} d\theta \end{aligned} \quad (1)$$

where the curvature and radius of curvature are given by

$$\kappa = \frac{1}{\rho} = \frac{d\theta}{ds} \quad (2)$$

If we limit ourselves to pseudospirals (as defined above, consisting of circular arcs), then the tangent angle θ is a polar angle relative to the local origin and $\rho(\theta)$ is the local radius given by any sequence of numbers, say $S = \{s_1, s_2, s_3, \dots, s_N\}$ with $S \in \mathbb{R}$. Thus,

$$\rho(\theta) = \sum_{k=1}^K (\rho_k - \rho_{k-1}) u(\theta - \theta_k) \quad (3)$$

where ρ_k are the elements of the sequence S and $\rho_0 = 0$, u is the Heaviside function, $\theta_k = (k-1)\Delta\theta$, and $\Delta\theta$ is the angular range of the circular arcs. Equation (3) can be substituted into Eq. (1) (bottom) and integrated to obtain

$$z(\theta) = i \cdot \sum_{n=1}^N (\rho_n - \rho_{n-1}) (e^{i\theta_n} - e^{i\theta}) u(\theta - \theta_n) \quad (4)$$

Without any loss of generality we can rotate ($e^{i\pi/2}$) and translate ($+\rho_1$) the curve so that it conforms to that in Figure 1. Thus, the curve will unfold in the counterclockwise direction from a point on the x -axis equal to the first number in the sequence. Then, Eq. (4) becomes

$$z(\theta) = \rho_1 - \sum_{n=1}^N (\rho_n - \rho_{n-1}) (e^{i\theta_n} - e^{i\theta}) u(\theta - \theta_n) \quad (5)$$

This equation is in perfect agreement with the “step-and-arc” algorithm and direct numerical simulation for well-known integer sequences, including those with negative integers, as well as sorted and unsorted random number sequences with positive and negative numbers. Figure 2-5 show some examples.

Figure 2 shows the Fibonacci spiral out to 16 terms along with the golden spiral. All is as expected. Figure 3 shows the Jacobsthal and Jacobsthal-Lucas spirals. Although not immediately obvious here, these two curves from apparently different sequences are, in fact, self-similar. [For information on these and many other integer sequences refer to the Wiki page “Category: Integer sequences” (http://en.wikipedia.org/wiki/Category:Integer_sequences) for a complete description.] We will have more to say about self-similarity below.

Figure 4-5 show results for a sorted and unsorted random 8-number sequence, respectively. The sequence values are in $S \in (-5:5)$. The sorted sequence exhibits a single cusp where this sequence changes sign. The unsorted sequence, however, exhibits a cusp whenever that sequence changes sign. Of course, a negative sequence number means a negative radius, or one whose center is oriented outward, i.e., on the concave side of the curve.

Some additional curves are shown in the Gallery below. A complete code for calculation and comparison of pseudospirals by various methods is presented in the Appendix.

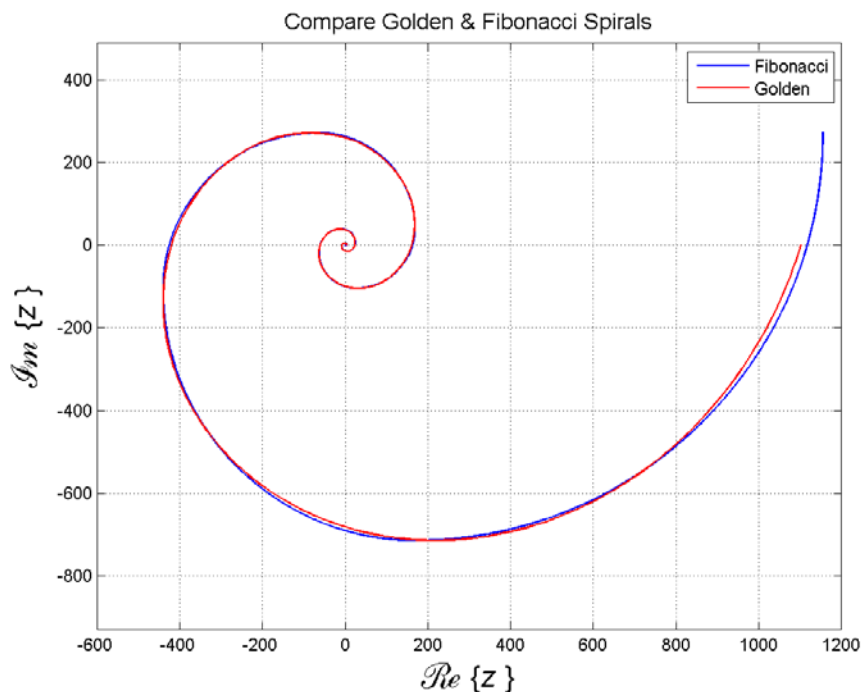


Figure 2: Fibonacci and golden spirals.

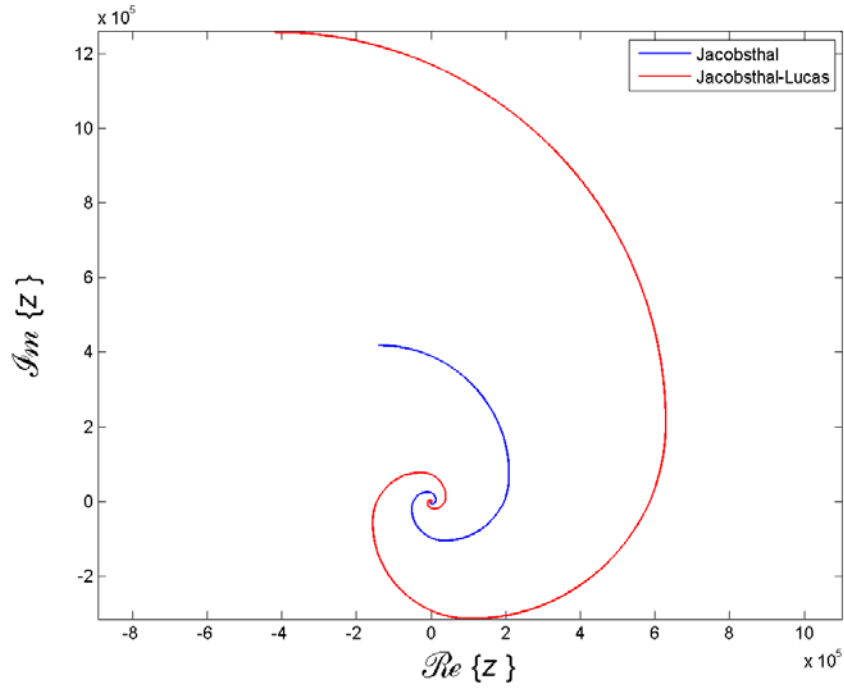


Figure 3: Jacobsthal and Jacobsthal-Lucas spirals.

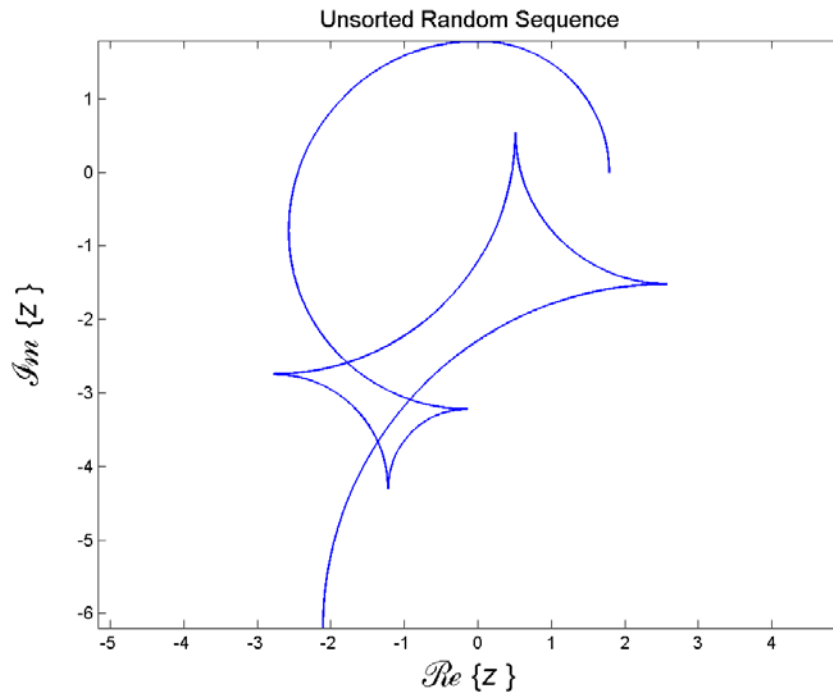


Figure 4: Unsorted random number sequence spiral, $S \in (-5:5)$.

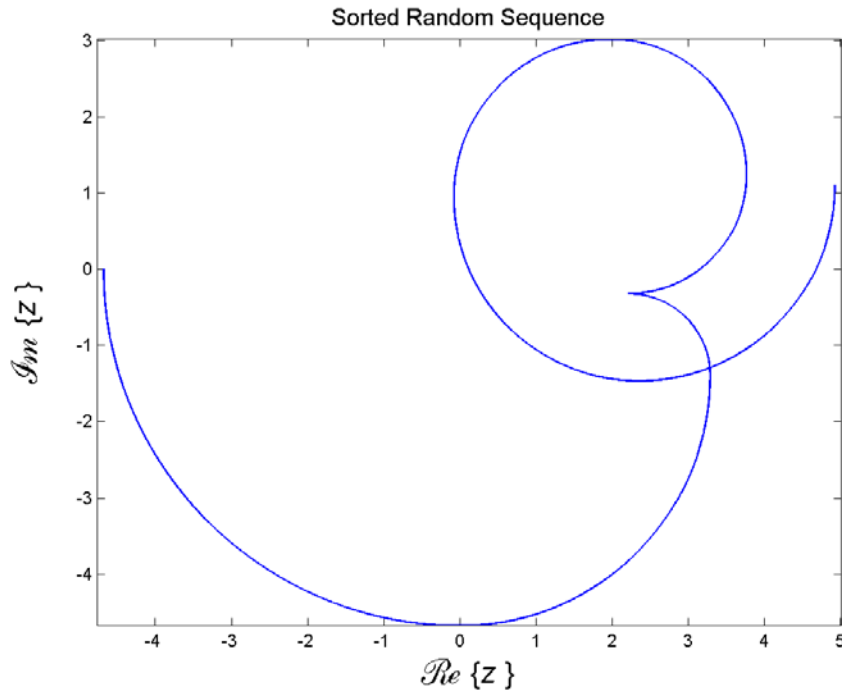


Figure 5: Sorted random number sequence spiral (same sequence as in Figure 4).

Self-similarity

Certain curves based on integer sequences exhibit self-similarity for sufficiently large n . Specifically, running curves out to various values of n will look identical (to within a rotation of axes). We have shown that curves that satisfy Binet-type functions meet this criterion. The Binet function has its origin in the well-known representation of the Fibonacci sequence in terms of the golden ratio

$$F_n = \frac{\varphi^n - \psi^n}{\varphi - \psi} = \frac{\varphi^n - (-1/\varphi)^n}{\varphi + 1/\varphi} \quad (6)$$

$$\varphi, \psi = \frac{1 \pm \sqrt{5}}{2}$$

Many authors have successfully extended the Binet formula to variations of the Fibonacci sequence as well as other sequences. For example, Maynard (2008) derives an expression for the following sequence

$$f_0 = 0, \quad f_1 = 1, \quad f_n = a f_{n-1} + b f_{n-2} \quad \text{for } n \geq 2 \quad (7)$$

Then, for $a, b \in \mathbb{R}^+$

$$f_n = \frac{\alpha^n - \beta^n}{\alpha - \beta} \quad (8)$$

$$\alpha, \beta = \left(a \pm \sqrt{a^2 + 4b} \right) / 2$$

This is in agreement with Eq. (6) for $a = b = 1$, which is the Fibonacci sequence. The Jacobsthal numbers are also of this type, with $a = 1$, $b = 2$.

The Lucas numbers, i.e.,

$$f_0 = 2, \quad f_1 = 1, \quad f_n = f_{n-1} + f_{n-2} \quad \text{for } n \geq 2 \quad (9)$$

also have a well-known Binet solution that is given by

$$L_n = \varphi^n + \psi^n = \varphi^n + (-1/\varphi)^n \quad (10)$$

where φ and ψ are as given in Eq. (6). We next considered generalized Lucas numbers given by

$$f_0 = 2, \quad f_1 = a, \quad f_n = a f_{n-1} + b f_{n-2} \quad \text{for } n \geq 2 \quad (11)$$

Following Maynard's analysis, we found that

$$f_n = \alpha^n + \beta^n \quad (12)$$

where α and β are as given in Eq. (8). The Jacobsthal-Lucas numbers are of this type, again, with $a = 1$, $b = 2$. We verified Eq. (12) by direct comparison with the recursion relation and by using `RSolve` in `MATHEMATICA`®; the solutions are all in agreement.

The results of Eqs. (10) and (12) are in agreement with those of Kappraff and Adamson 2004, for which $b = 1$. In that case, we always have $\beta = -1/\alpha$. And of course, when $a = 1$, $\alpha = \varphi$.

We can now address the question of self-similarity. It comes down to this: for n sufficiently large, one or other of the terms α^n or β^n will dominate and the Binet function will devolve into a logarithmic spiral (for example, $\alpha^n = e^{n \ln \alpha}$), which is clearly self-similar. Figure 6 shows an example of self-similarity for the crab-claw curve (Fibonacci numbers from $[-n:n]$); on the left, $n = 16$ and on the right, $n = 32$. Notice the ~ 2200 -fold difference in the scale of these figures. The lower panels show details of the smaller- n region (~ 250 -fold zoom); these demonstrate that the self-similarity does not extend to the lower n -values.

Now, of course, the Binet functions can be analytically continued into the entire complex plane. But that is subject of another article. Stay tuned.

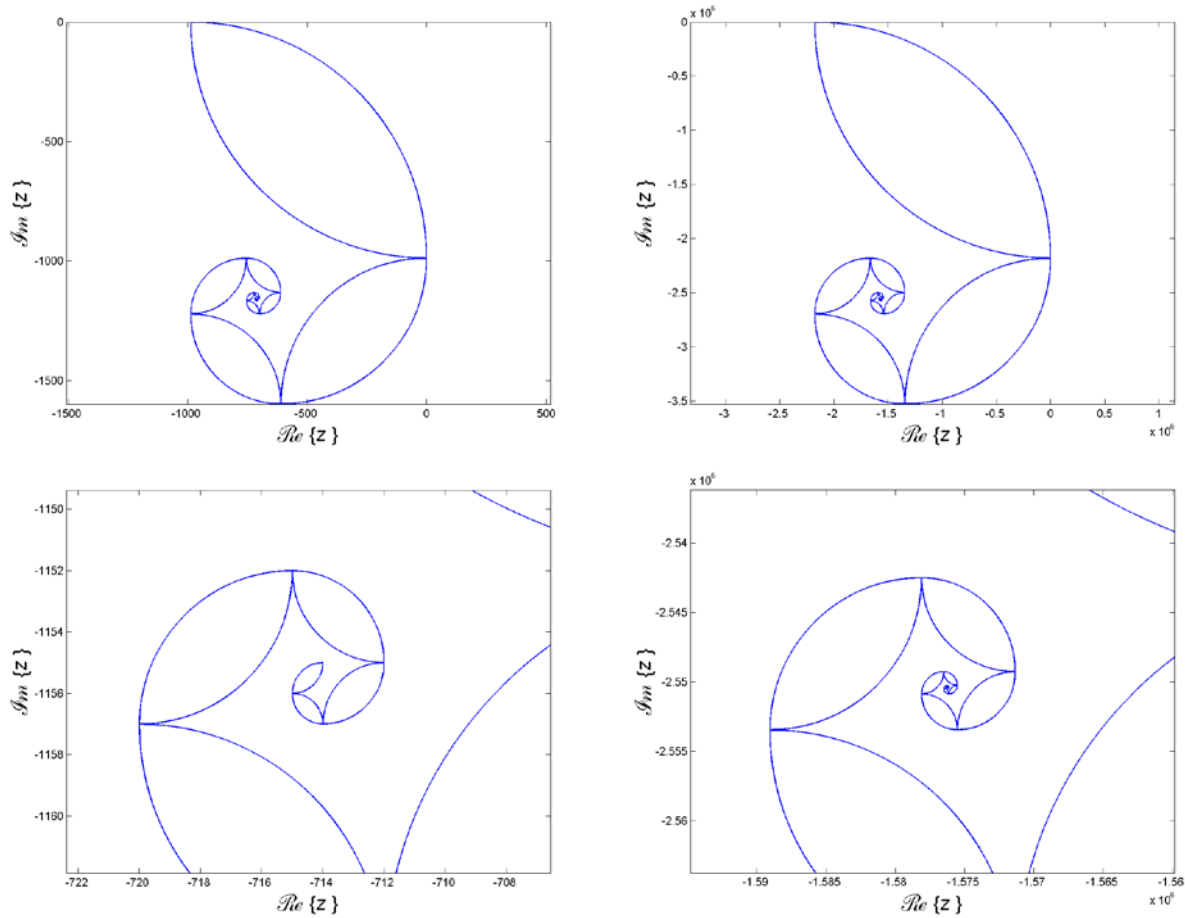


Figure 6: Crab-claw spiral similarity demonstration. Top panels show full figure for 16 and 32 terms on the left and right, respectively. Lower panels show smaller- n region, where self-similarity breaks down.

Summary and conclusions

We have demonstrated both a rational “step-and-arc” algorithm and a closed-form analytic solution for the Fibonacci spiral (good riddance to graph paper). The concept was extended to an entire class of pseudospirals based on arbitrary real number sequences. We have also found a Binet-type formula for generalized Lucas numbers. It is not known if this has been done previously since there is a rather large body of literature on generalized Binet functions.

References

Kappraff, J. and Adamson, G.W. (2004). “Generalized Binet Formulas, Lucas Polynomials, and Cyclic Constants,” *Forma*, **19**, 355-366. (<http://www.scipress.org/journals/forma/pdf/1904/19040355.pdf>)

Maynard, P. (2008), “Generalised Binet Formulae,” *Applied Probability Trust*; available at <http://ms.appliedprobability.org/data/files/Articles%2040/40-3-2.pdf>.

Extra Credit

The sequence of curves used in the animation was based on substituting the circular arc routine (see the function `ComplexCircularArc` in the Appendix) by one which yields a concave or convex rounded rectangular arc. The function is given here:

```
function z=ComplexRoundedRect(R,z0,Dtheta,p,npts)
% calculate a quasi-rectangular arc of power p, radius R at z=z0 from
% Dtheta=[Dtheta(1) Dtheta(2)] using npts points in the complex plane;
% this program has been rewritten into degrees to avoid problems with
% cos(pi/2) and sin(pi) raised to small powers
% Cye H. Waldman, 2013

Dtheta=linspace(Dtheta(1),Dtheta(2),npts)';
x=R*sign(cosd(Dtheta)).*(abs(cosd(Dtheta))).^p+real(z0);
y=R*sign(sind(Dtheta)).*(abs(sind(Dtheta))).^p+imag(z0);
z=complex(x,y);

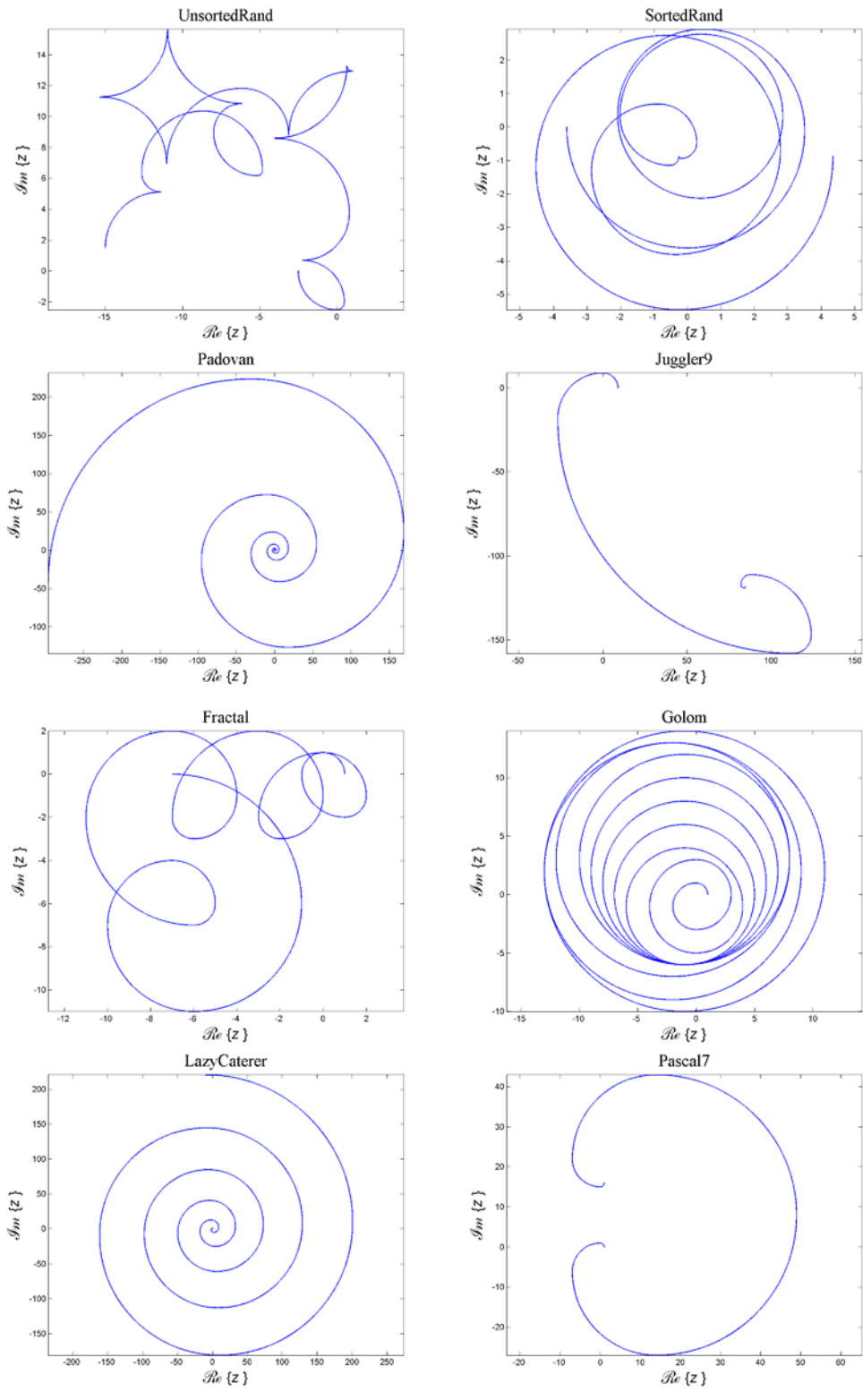
return
```

The parameter p can basically vary from zero to infinity. There are a few noteworthy things to mention about this code. Consider a full circle, if we let $p = 2$, the result is a square inscribed in the circle at a 45° angle. For values of $p < 2$, the square bulges outward (convex) until at $p = 0$ you have the circle inscribed in a normal square. For values of $p > 2$, the square bulges inward (concave) and tends toward a plus sign as $p \rightarrow \infty$.

The code was written in (x,y) coordinates because we cannot express $z = \cos^p \theta + i \sin^p \theta$ very conveniently in the complex plane. Moreover, we have to resort to the Matlab functions `cosd` and `sind` (the degrees equivalent of `cos` and `sin`) in order to avoid the numerical round-off errors that accrue with $\cos(\pi/2)$ and $\sin(\pi)$. For example, in Matlab, $\cos^{0.1}(\pi/2) = 0.6884$, while $\cos^{0.1}(90) = 0$, as it should be. In addition, note that you don't want to exponentiate negative values, hence $\cos^p \theta$ becomes $\text{sign}(\cos \theta) \cdot |\cos \theta|^p$, and so on.

These are the curves that compose the second animation.

Gallery



Appendix: Source code for Pseudospiral

```

function Pseudospiral
% this function computes random or prescribed pseudospirals and compares the
% solutions from step-and-arc, direct numerical simulation, and closed-form
% analytic methods
% Copyright 2013, Cye H. Waldman
rot=pi/2;           % angular rotation for each arc
z0=0+0i;           % starting point
Theta=[0 rot];     % initial theta range
npts =1001;        % number of points per circular arc

% VARIOUS SEQUENCES PREVIOUSLT TRIED
UnsortedRand=-5+10*rand(20,1);
SortedRand=sort(-5+10*rand(20,1));
Padovan=[1,1,1,2,2,3,4,5,7,9,12,16,21,28,37,49,65,86,114,151,200,265]';
;
Juggler3=[3, 5, 11, 36, 6, 2, 1]';
Juggler9=[9, 27, 140, 11, 36, 6, 2, 1]';
Fractal=[1,1,2,1,2,3,1,2,3,4,1,2,3,4,5,1,2,3,4,5,6]';
Golom=[1,2,2,3,3,4,4,4,5,5,5,6,6,6,6,7,7,7,7,8,8,8,8,9,9,9,9,...
    10,10,10,10,10,11,11,11,11,11,12,12,12,12,12,12]';
LazyCaterer=[1,2,4,7,11,16,22,29,37,46,56,67,79,92,106,121,137,154,172,
    ,191,211]';
Jacobsthal=[0,1,1,3,5,11,21,43,85,171,341,683,...
    1365,2731,5461,10923,21845,43691,87381,174763,349525]';
JacobsthalLucas=[2,1,5,7,17,31,65,127,257,511,1025,...
    2047,4097,8191,16385,32767,65537,131071,262145,524287,1048577]';
Pascal5=[1 5 10 10 5 1]';
Pascal6=[1 6 15 20 15 6 1]';
Pascal7=[1 7 21 35 35 21 7 1]';
Fibonacci=[1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,41
    81,6765]';

% CHOOSE YOUR SEQUENCE AND NUMBER OF TERMS, OR ADD YOUR OWN
n=20;
Sequence=Fibonacci(1:n)';

% STEP-AND-ARC
zstep=[];
for k=1:n
    r=Sequence(k);
    c4th=ComplexCircularArc(r,z0,Theta,npts);
    zstep=[zstep;c4th(2:end)];
    if k<n
        rnext=Sequence(k+1);
        Theta=Theta+rot;
        z0=zstep(end)-rnext*exp(i*Theta(1));
    else
        zstep=[zstep;c4th(end)];
    end
end

```

```

% DIRECT NUMERICAL SIMULATION
theta=linspace(0,n*rot,n*(npts-1)+1)';
dtheta=theta(2)-theta(1);
rho=Sequence; % must be row vector
[r,c]=size(rho);
if r>c; rho=rho.'; end
rho=repmat(rho,(npts-1),1);
rho=reshape(rho,n*(npts-1),1);
rho=[rho;rho(end)];
I=find(diff(rho)~=0);
for k=1:length(I)
    rho(I(k)+1)=(rho(I(k))+rho(I(k)+2))/2;
end

zdns=exp(i*pi/2)*cumtrapz(exp(i*theta).*rho)*dtheta;
zdns=zdns+max(real(zstep))-max(real(zdns)); % align w/ step-arc soln

% CLOSED-FORM ANALYTIC SOLUTION
zanal=Sequence(1);
for k=1:n
    try
        delSequence=Sequence(k)-Sequence(k-1);
    catch
        delSequence=Sequence(k);
    end
    zanal=zanal-delSequence*...
        (exp(i*(k-1)*rot)-exp(i*theta)).*heaviside(theta-(k-1)*rot);
end

figure
hold on
plot(zstep,'r')
plot(zdns,'Color',[0 .5 0])
plot(zanal,'b','LineWidth',1)
box
axis equal

return

function z=ComplexCircularArc(R,z0,Theta,npts)
% calculate a circular arc of radius R at z=z0 from Theta=[Theta(1)
% Theta(2)] using npts points in the complex plane

theta=linspace(Theta(1),Theta(2),npts)';
z=R*exp(i*theta)+z0;

return

```