Lecture 7: The Fixed Point Theorem and its Consequences

My heart is fixed... Psalm 57:7

1. Fixed Points and Iteration

We are now going to solve the central mystery concerning fractals: why is it that the fractals generated both by recursive turtle programs and by iterated function systems are independent of the base case? The heart of the answer lies in the trivial fixed point theorem.

A *fixed point* of a function F is a point P such that $F(P) = P$. That is, P is a fixed point of F if P is unchanged by F. For example, if $f(x) = x^2$, then $f(0) = 0$ and $f(1) = 1$, so 0 and 1 are fixed points of f. We are interested in fixed points of transformations because fractals are fixed points of iterated function systems.

Fixed points are often generated by iteration. Consider the sequence,

$$P_1 = F(P_0)$$
$$P_2 = F^2(P_0) = F(P_1)$$
$$\vdots \qquad \vdots$$
$$P_\infty = F^\infty(P_0)$$

where $F^\infty(P_0)$ denotes the limit of $F^n(P_0) = F(P_{n-1})$ as n approaches infinity (assuming this limit exists). Suppose that we try to apply F to P_∞; informally,

$$F(P_\infty) = F\left(F^\infty(P_0)\right) = F^{\infty+1}(P_0) = F^\infty(P_0) = P_\infty .$$

Thus the limit point P_∞ is a fixed point of the function F. The point P_0 with which the iteration begins does not matter; as long as the iteration converges, iteration converges to a fixed point.

There are two important caveats here. First, there is no guarantee that the iteration must converge. For some starting values -- perhaps even for all starting values -- the iteration may diverge. Second, even if the iteration always converges, the fixed point need not be unique; different starting values may converge to different fixed points.

In the next section we are going to establish conditions which guarantee that these two problems never arise. That is, we shall provide conditions which guarantee that the iteration always converges and that the iteration converges to a unique fixed point independent of the initial value. If this result sounds a lot like fractals, this similarity is no accident. Indeed this general result about iteration is at the heart of our observations that fractals generated from iterated function systems are independent of the base case.

A word of caution before we proceed. Be forewarned: this lecture is by far the most mathematically challenging chapter in this text. We are going to prove a powerful mathematical theorem with applications to many different areas of Pure and Applied Mathematics, including root finding for transcendental functions, relaxation methods for solving large systems of linear equations, solution techniques for ordinary differential equations, and fractals. Therefore, we shall require some general mathematical machinery that we can apply in a wide range of diverse situations. *The power of Mathematics is the ability to reduce many seemingly different problems to their common mathematical essence*. The price we pay for this power is abstraction: a collection of mathematical terms, techniques, and theorems the significance of which may, at first, be difficult to understand. Cauchy sequences, complete spaces, compact sets, and Haussdorf metrics will all appear in this discussion. For each of these terms I will provide some informal intuition as well as a formal definition. For fractals, I will explain only the key ideas behind the proofs of the main theorems, providing references for those readers interested in all the fine details. Understanding this material may be difficult at first, and you may need to read this chapter more than once. It may be hard going, but hang in there; the final results will be well worth all the effort.

2. The Trivial Fixed Point Theorem

We begin with a rigorous formulation of our informal argument in Section 1 that if iteration converges, iteration necessarily converges to a fixed point.

Proposition 1: *Let T be a continuous function and let $P_{n+1} = T(P_n)$ for all $n \geq 0$. If $P_\infty = Lim_{n\to\infty} P_n$ exists, then P_∞ is a fixed point of T.*

Proof: Since, by assumption, T is continuous, we can push T past the limit sign. Therefore

$$T(P_\infty) = T\left(Lim_{n\to\infty} P_n\right) = Lim_{n\to\infty} T(P_n) = Lim_{n\to\infty} P_{n+1} = P_\infty.$$

For our applications, the most important continuous functions are *contractive transformations*. A transformation T is said to be *contractive* if there is a constant s, $0 < s < 1$, such that for all points P, Q

$$Dist\left(T(P), T(Q)\right) \leq s\, Dist(P, Q).$$

That is, contractive maps bring points closer together. The canonical example of a contractive transformation is uniform scaling, where the absolute value of the scale factor is less than one. It follows from Proposition 1 that if we iterate a contractive transformation and if the iteration converges, then the iteration necessarily converges to a fixed point. Our next result asserts that for contractive maps, this fixed point is unique.

Proposition 2: *If T is a contractive map, then T can have at most one fixed point.*

Proof: Suppose that P and Q are both fixed points of T. Then $T(P) = P$ and $T(Q) = Q$. Therefore

$$Dist\big(T(P),T(Q)\big) = Dist(P,Q).$$

Hence T is not a contractive map. Contradiction.

Proposition 2 guarantees the uniqueness, but not the existence, of fixed points for contractive maps. In order to establish existence, we need to introduce the notion of a *cauchy sequence*.

Informally, a sequence of points $\{P_n\}$ is a *cauchy sequence* if the points get closer and closer as n gets larger and larger. For example, the sequence $\{2^{-n}\}$ is cauchy, but the sequence $\{2^n\}$ is not cauchy. Formally, a sequence of points $\{P_n\}$ is said to be *cauchy* if for every $\varepsilon > 0$ there is an integer N such that

$$Dist(P_{n+m},P_n) < \varepsilon \text{ for all } n > N \text{ and all } m > 0.$$

Our next result shows that one way to generate a cauchy sequence is by iterating a contractive map.

Proposition 3: *Suppose that T is a contractive transformation, and that* $P_{n+1} = T(P_n)$ *for all* $n \geq 0$. *Then* $\{P_n\}$ *is a cauchy sequence for any choice of* P_0.

Proof: Since T is a contractive transformation, there is a constant s, $0 < s < 1$, such that

$$\begin{aligned} Dist\big(P_{n+1},P_n\big) &= Dist\big(T(P_n),T(P_{n-1})\big) \\ &\leq s\,Dist\big(P_n,P_{n-1}\big) = s\,Dist\big(T(P_{n-1}),T(P_{n-2})\big) \\ &\quad \vdots \\ &\leq s^n Dist\big(P_1,P_0\big). \end{aligned}$$

Therefore, by the triangular inequality, for n sufficiently large

$$\begin{aligned} Dist\big(P_{n+m+1},P_n\big) &\leq Dist\big(P_{n+m+1},P_{n+m}\big) + \cdots + Dist\big(P_{n+1},P_n\big) \\ &\leq (s^{n+m} + \cdots + s^n)\,Dist\big(P_1,P_0\big) \\ &\leq \frac{s^n}{1-s}\,Dist\big(P_1,P_0\big) \\ &< \varepsilon. \end{aligned}$$

The main fact about cauchy sequences is that for many well known spaces every cauchy sequences converges to some limiting value. A space in which every cauchy sequence converges is said to be *complete*. Intuitively, a space is complete if it has no holes because if the points in a sequence are getting closer and closer to each other, they should be getting closer and closer to some limiting value. The standard Euclidean spaces in *n*-dimensions are examples of complete spaces.

We are now ready to establish the main result of this section: the existence and uniqueness of fixed points for contractive transformations on complete spaces.

Trivial Fixed Point Theorem

Suppose that

T is a contractive transformation on a complete space

$P_{n+1} = T(P_n)$ *for all* $n \geq 0$.

Then $P_\infty = Lim_{n\to\infty} P_n$ *exists, and* P_∞ *is the unique fixed point of T for any choice of* P_0*!*

Proof: This result follows from our previous propositions because:

i. The sequence $\{P_n\}$ is cauchy. (Proposition 3)

ii. Therefore, since by assumption the space is complete, $Lim_{n\to\infty} P_n$ exists.

iii. Hence, since *T* is continuous, $P_\infty = Lim_{n\to\infty} P_n$ is a fixed point of *T*. (Proposition 1)

iv. But, since *T* is a contractive transformation, this fixed point is unique. (Proposition 2)

3. Consequences of the Trivial Fixed Point Theorem

The trivial fixed point theorem has many important applications in Mathematics and Computer Science. Here we will look at three of these applications: root finding for transcendental functions, relaxation methods for solving large systems of linear equations, and algorithms for generating fractals from iterated function systems.

3.1 Root Finding Methods. To find a root of a function $F(x)$ means to find a solution of the equation $F(x) = 0$. For low degree polynomials, such as quadratic equations, there are explicit formulas, such as the quadratic formula, for computing the roots. But for polynomials of degree greater than four and for transcendental functions such as trigonometric and exponential functions, simple explicit formulas for the roots do not exist. In these cases, iterative techniques are typically used to locate the roots.

One well-known iterative method that you probably encountered in calculus is Newton's method. To find a solution of the equation $F(x) = 0$ using Newton's method, we first make some initial guess, x_0. Once we have produced the *nth* guess, x_n, the $(n+1)^{st}$ guess is given by

$$x_{n+1} = x_n - \frac{F(x_n)}{F'(x_n)}.$$

The idea behind Newton's method is that the value of x_{n+1} is the location along the x-axis where the line tangent to the curve $y = F(x)$ at $x = x_n$ intersects the x-axis (see Exercise 6). Thus x_{n+1} is often closer to the root of $F(x) = 0$ than x_n (see Figure 1, left). If the initial guess x_0 is close to a root of $F(x)$ and if $F'(x_n)$ is not close to zero, then the sequence $x_0, x_1, \ldots$ typically converges rapidly to a solution of the equation $F(x) = 0$. Thus Newton's method can be summarized in the following fashion.

Newton's Method

1. Select any(!) initial guess x_0.
2. Compute $x_{n+1} = x_n - \dfrac{F(x_n)}{F'(x_n)}$ for $n = 1,2\ldots.$
3. Stop when $|x_{k+1} - x_k| < \varepsilon$.

Newton's method is actually a fixed point method. Let

$$T(x) = x - \frac{F(x)}{F'(x)}.$$

Then $F(x^*) = 0$ if and only if $T(x^*) = x^*$. Thus to find a root of $F(x)$, we seek a fixed point of $T(x)$. Hence we iterate $T(x)$ on some initial guess x_0. But iterating $T(x)$ is precisely the second step of Newton's method. We illustrate Newton's method in Figures 1 and 2. Notice that Newton's method is not guaranteed to converge (see Figure 2) because $T(x)$ need not be a contractive map; moreover, if $F(x)$ has more than one root, different initial guesses may converge to different roots.

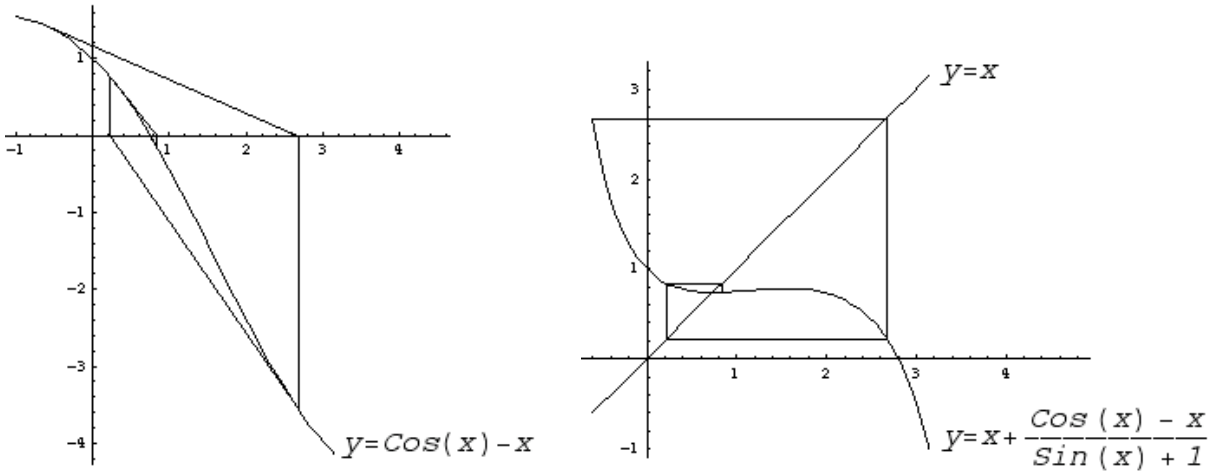

Figure 1: Finding a root of $F(x) = Cos(x) - x$ by using Newton iteration to find a fixed point of

$$T(x) \;=\; x - \frac{F(x)}{F'(x)} \;=\; x + \frac{Cos(x) - x}{Sin(x) + 1}.$$

Here the initial guess is at $x_0 = -0.6$. On the left is the traditional view of Newton's method: the next guess is where the line tangent to the curve $y = F(x) = Cos(x) - x$ at the current guess intersects the x-axis. On the right is the fixed point perspective for Newton's method: here the next guess is where the horizontal line $y = T(x_n)$ at the current guess intersects the line $y = x$. Notice how in both figures the guesses spiral into the root at $x \approx 0.739086$.

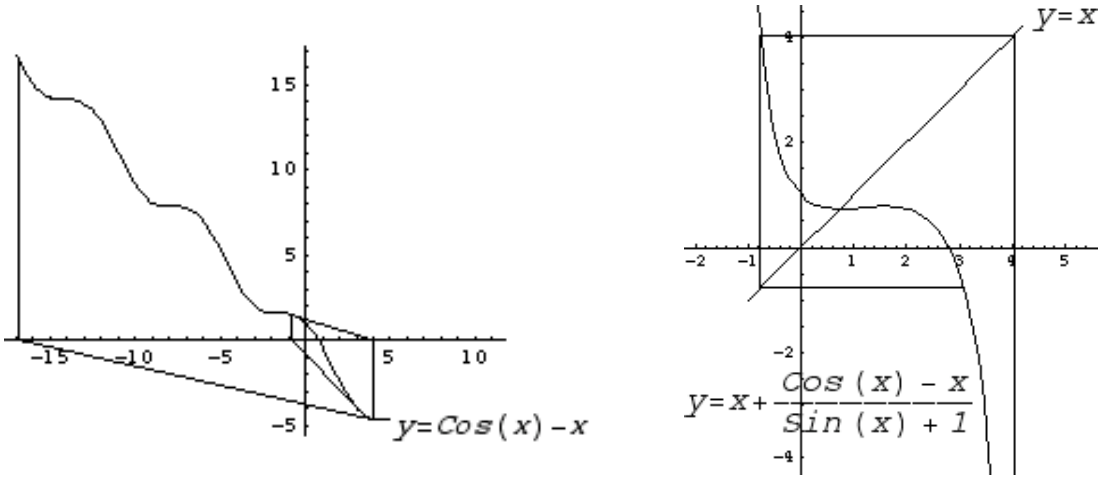

Figure 2: Again we look at Newton's method for the function $F(x) = Cos(x) - x$. Here the initial guess is at $x_0 = 3.08$. As in Figure 1, on the left is the traditional view of Newton's method; on the right is the fixed point perspective for Newton's method. Notice how in this case the guesses diverge, spiraling away from the root at $x \approx 0.739086$.

Newton's formula for the $(n+1)^{st}$ guess x_{n+1} is expensive; we need to compute both the value of the function and the value of the derivative at the current guess x_n. An easier approach that requires only evaluation and no differentiation is to use a simpler fixed point method.

Suppose that we want to find a solution of the equation $F(x)=0$. Let $G(x)=x+F(x)$. Then $F(x)$ has a root if and only if $G(x)$ has a fixed point -- that is, $F(x^*)=0$ if and only if $G(x^*)=x^*$. But by the trivial fixed point theorem, we can often find a fixed point by iteration. This observation leads to the following simple root finding algorithm.

Fixed Point Root Finding Algorithm

1. Replace $F(x)$ by $G(x)=x+F(x)$.
2. Select any(!) initial guess x_0.
3. Compute $x_{k+1}=G(x_k)$ for $k=1,\ldots,n$.
4. Stop when $|x_{k+1}-x_k|<\varepsilon$.

Notice that unlike Newton's method, this algorithm requires only evaluation; no differentiation is necessary. By the trivial fixed point theorem, the fixed point root finding algorithm is guaranteed to work whenever $G(x)$ is a contractive map. Moreover, any initial guess will do.

How can we tell if a map is contractive? One way is to invoke the mean value theorem:

$$G(b)-G(a)\le G'(c)(b-a) \qquad a\le c\le b.$$

By the mean value theorem, if $|G'(c)|<1$, then $G(x)$ is necessarily a contractive map. For example, $G(x)=Cos(x)$ is contractive in the interval $[0,a]$ for any $a<\pi/2$, and $G(x)=e^{-x}$ is contractive in the interval $[0,a]$ for any $a>0$.

Below we illustrate the fixed point algorithm for finding the roots of two transcendental equations. In Figure 3, we find a root of $F(x)=Cos(x)-x=0$ by finding a fixed point of $G(x)=x+F(x)=Cos(x)$; in Figure 4 we find a root of $F(x)=e^{-x}-x=0$ by finding a fixed point of $G(x)=x+F(x)=e^{-x}$. In both cases we show that the method converges to the same root, independent of the initial guess. Notice that this fixed point method often requires more iterations than Newton's method to get close to a root, but the reader should check that this fixed point method can converge to a root even in cases where Newton's method diverges (see Exercise 7).

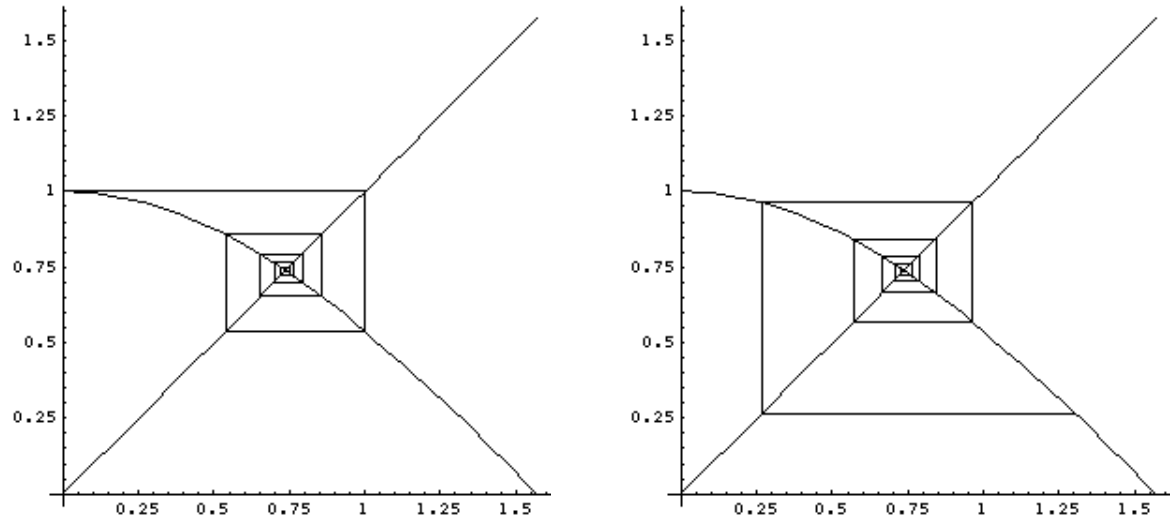

Figure 1: Finding a root of $F(x) = Cos(x) - x$ by finding a fixed point of $G(x) = F(x) + x = Cos(x)$ using iteration. On the left the initial guess is $x_0 = 0$; on the right the initial guess is $x_0 = 1.3$. In both cases the guesses converge to the intersection of the curves $y = Cos(x)$ and $y = x$ -- that is, to a fixed point of $Cos(x)$.

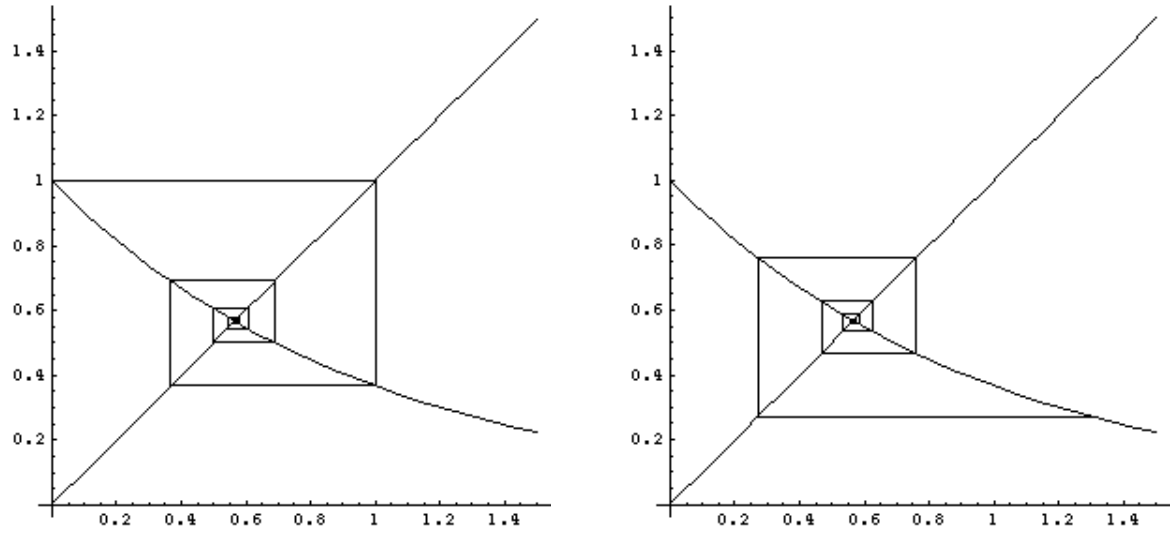

Figure 2: Finding a root of $F(x) = e^{-x} - x$ by finding a fixed point of $G(x) = F(x) + x = e^{-x}$ using iteration. On the left the initial guess is $x_0 = 0$; on the right the initial guess is $x_0 = 1.3$. In both cases the guesses converge to the intersection of the curves $y = e^{-x}$ and $y = x$ -- that is, to a fixed point of e^{-x}.

3.2 Relaxation Methods. Consider a large system of linear equations:

$$\begin{aligned} M_{11}x_1 + M_{12}x_2 + \cdots + M_{1n}x_n &= b_1 \\ M_{21}x_1 + M_{22}x_2 + \cdots + M_{2n}x_n &= b_2 \\ \vdots \qquad\qquad & \qquad \vdots \\ M_{n1}x_1 + M_{n2}x_2 + \cdots + M_{nn}x_n &= b_n \end{aligned} \tag{3.1}$$

We can rewrite these linear equations in matrix form as:

$$\underbrace{\begin{pmatrix} M_{11} & M_{12} & \cdots & M_{1n} \\ M_{21} & M_{22} & \cdots & M_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ M_{n1} & M_{n2} & \cdots & M_{nn} \end{pmatrix}}_{M} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}}_{X} = \underbrace{\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}}_{B} \quad \Leftrightarrow \quad M * X = B\,, \tag{3.2}$$

and we can solve this linear system for the unknowns $X = (x_1,\ldots,x_n)^T$ using either Gaussian elimination, or Cramer's rule, or simply by inverting the matrix M. But when n, the number of equations, is large, Cramer's rule and matrix inversion are numerically unstable, and Gaussian elimination is slow; iterative methods often work best. Iterative methods for solving systems of linear equations are called *relaxation methods*. Here we shall look at two such relaxation methods: one due to Jacobi and the other to Gauss-Seidel.

In Jacobi relaxation, the *pth* guess $X^p = (x_1^p,\ldots,x_n^p)^T$ is generated from the $(p-1)^{st}$ guess $X^{p-1} = (x_1^{p-1},\ldots,x_n^{p-1})^T$ by solving the equations

$$\begin{aligned} M_{11}x_1^p &= b_1 - M_{12}x_2^{p-1} - \cdots - M_{1n}x_n^{p-1} \\ M_{22}x_2^p &= b_2 - M_{21}x_1^{p-1} - \cdots - M_{2n}x_n^{p-1} \\ \vdots \qquad\qquad & \qquad \vdots \\ M_{nn}x_n^p &= b_n - M_{n1}x_1^{p-1} - \cdots - M_{n\,n-1}x_{n-1}^{p-1} \end{aligned} \tag{3.3}$$

which we can write in matrix form as

$$\underbrace{\begin{pmatrix} M_{11} & 0 & \cdots & 0 \\ 0 & M_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & M_{nn} \end{pmatrix}}_{D} \underbrace{\begin{pmatrix} x_1^p \\ x_2^p \\ \vdots \\ x_n^p \end{pmatrix}}_{X^p} = \underbrace{\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}}_{B} - \underbrace{\begin{pmatrix} 0 & M_{12} & \cdots & M_{1n} \\ M_{21} & 0 & \cdots & M_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ M_{n1} & M_{n2} & \cdots & 0 \end{pmatrix}}_{M-D} \underbrace{\begin{pmatrix} x_1^{p-1} \\ x_2^{p-1} \\ \vdots \\ x_n^{p-1} \end{pmatrix}}_{X^{p-1}}. \tag{3.4}$$

Thus from Equation (3.3) in Jacobi relaxation

$$x_i^p = \frac{b_i}{M_{ii}} - \sum_{j \neq i} \frac{M_{ij}}{M_{ii}} x_j^{p-1} \qquad 1 \le i \le n\,. \tag{3.5}$$

On the other hand, in Gauss-Seidel relaxation, the *pth* guess $X^p = (x_1^p,\ldots,x_n^p)^T$ is generated from the $(p-1)^{st}$ guess $X^{p-1} = (x_1^{p-1},\ldots,x_n^{p-1})^T$ by solving the diagonal system of equations

$$\begin{aligned} M_{11}x_1^p &= b_1 - M_{12}x_2^{p-1} - \cdots - M_{1n}x_n^{p-1} \\ M_{21}x_1^p + M_{22}x_2^p &= b_2 - M_{23}x_1^{p-1} - \cdots - M_{2n}x_n^{p-1} \\ &\vdots \qquad\qquad\qquad \vdots \\ M_{n1}x_1^p + M_{22}x_2^p + \cdots + M_{nn}x_n^p &= b_n \end{aligned} \tag{3.6}$$

which can be written in matrix form as

$$\underbrace{\begin{pmatrix} M_{11} & 0 & \cdots & 0 \\ M_{21} & M_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ M_{n1} & M_{n2} & \cdots & M_{nn} \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} x_1^p \\ x_2^p \\ \vdots \\ x_n^p \end{pmatrix}}_{X^p} = \underbrace{\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}}_{B} - \underbrace{\begin{pmatrix} 0 & M_{12} & \cdots & M_{1n} \\ 0 & 0 & \cdots & M_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}}_{M-L} \underbrace{\begin{pmatrix} x_1^{p-1} \\ x_2^{p-1} \\ \vdots \\ x_n^{p-1} \end{pmatrix}}_{X^{p-1}}. \tag{3.7}$$

Thus from Equation (3.6) in Gauss-Seidel relaxation

$$x_i^p = \frac{b_i}{M_{ii}} - \sum_{j=1}^{i-1} \frac{M_{ij}}{M_{ii}} x_j^p - \sum_{j=i+1}^{n} \frac{M_{ij}}{M_{ii}} x_j^{p-1} \qquad 1 \le i \le n \tag{3.8}$$

Notice from Equation (3.8) that in Gauss-Seidel relaxation, earlier values of the *pth* guess are used to generate later values of the *pth* guess. For this reason, Gauss-Seidel relaxation is usually somewhat faster than Jacobi relaxation.

Typically the initial guess for both methods is either $X^0 = \mathbf{0} = (0,\ldots,0)^T$ or $X^0 = (b_1,\ldots,b_n)^T$. Both relaxation methods converge for any initial guess, when *M* is *diagonally dominant* -- this is, when

$$|M_{ii}| \ge \sum_{j \ne i} |M_{ij}| \qquad 1 \le i \le n,$$

because both relaxation methods are really fixed point methods.

To understand why relaxation methods are fixed point methods, observe if we multiply Equation (3.4) by D^{-1}, then

$$X^p = D^{-1}B - (D^{-1}M - I) * X^{p-1} \qquad \text{(Jacobi Relaxation)} \tag{3.9}$$

Similarly if we multiply Equation (3.7) by L^{-1}, then

$$X^p = L^{-1}B - (L^{-1}M - I) * X^{p-1} \qquad \text{(Gauss-Seidel Relaxation)} \tag{3.10}$$

Let

$$T(X) = Q^{-1}B - (Q^{-1}M - I) * X. \tag{3.11}$$

where $Q = D$ for Jacobi relaxation and $Q = L$ for Gauss-Seidel relaxation. Now it is easy to see that the following statements are equivalent:

1. $T(X^*) = X^*$
2. $Q^{-1}B - (Q^{-1}M - I) * X^* = X^*$
3. $B - (M - Q) * X^* = Q * X^*$
4. $B - M * X^* = 0$
5. $M * X^* = B$

Therefore X^* is a solution of the original system of equations $M * X = B$ if and only if $T(X^*) = X^*$ -- that is, if and only if X^* is a fixed point of the transformation $T(X)$. Now it turns out that the transformation T is a contractive map whenever M is diagonally dominant and either

a. $Q = D$ is the diagonal part of M (*Jacobi Relaxation*)

or

b. $Q = T$ is the lower triangular part of M *(Gauss-Seidel Relaxation)*

We shall use these relaxation methods when we study radiosity in Lecture 23. The radiosity equations are a large system of linear equations, so relaxation methods are typically the fastest, most robust methods for solving the radiosity equations.

3.3 Fractals. Finally fractals. The trivial fixed point theorem asserts that iterating a contractive map on a complete space is guaranteed to converge to the unique fixed point of the transformation independent of the point at which we start the iteration. For fractals the contractive maps are iterated functions systems -- collections of contractive transformations on Euclidean space. But what exactly is the complete space that provides the necessary scaffolding for the trivial fixed point theorem?

To generate fractals, the *points* at which the iteration starts are actually sets of points; line segments, polygons, stars, almost any subset of the plane seems to work. However, we do not use unbounded subsets of the plane, since unbounded sets remain unbounded even after applying contractive affine transformations (e.g. uniform scaling), whereas the fractals we want to generate are typically bounded sets of points. Therefore we shall restrict our attention to compact sets.

A set is *compact* if it is both closed and bounded. *Bounded sets* are sets that lie within a circle of finite radius. Thus line segments, polygons, and stars are bounded; lines, infinite spirals, and half planes are unbounded. A set is *closed* if the set contains its boundary. A line segment is closed, but if we remove the end points of a line segment, the segment is no longer closed. Similarly, a disc is closed, but if we remove the bounding circle from the disc, then the disc is no

longer closed. We shall see shortly why we want to consider sets that are closed as well as bounded.

To study convergence, we need a notion of distance. We say that a sequence $s_0, s_1, \ldots$ converges to a limit s_∞ if and only if the distance between s_n and s_∞ gets smaller and smaller as n gets larger and larger. For fractals, we are dealing with sets. If we want to say that some sequence of sets converges to a fractal, we need some way to measure the distance between two sets. The standard notion of distance for compact sets is called the *Haussdorf metric*.

To define the Haussdorf metric, we first define the distance between a single point p and a compact set S by setting

$$Dist(p,S) = Min_{q \in S}\{Dist(p,q)\}.$$

Now we could try to define the distance between two sets R,S by setting

$$Dist(R,S) = Max_{p \in R}\{Dist(p,S)\}.$$

Unfortunately, with this definition, $Dist(R,S) \neq Dist(S,R)$. For example, if L_1, L_2 are two parallel lines of unequal length, then $Dist(L_1,L_2) \neq Dist(L_2,L_1)$ (see Figure 3, left). Similarly, if R is a proper subset of S, then $Dist(R,S) = 0$ but $Dist(S,R) > 0$ (see Figure 3, right). But clearly any reasonable notion of distance should be symmetric -- that is, for any reasonable notion of distance, we want $Dist(R,S) = Dist(S,R)$.

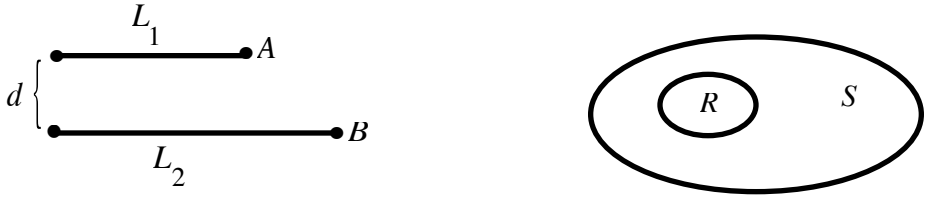

Figure 3: $Dist(R,S)$ is not symmetric. On the left two unequal parallel lines L_1, L_2: $Dist(L_1,L_2) = d$, but $Dist(L_2,L_1) = Dist(B,A) > d$. On the right two sets R,S, where R is a proper subset of S: $Dist(R,S) = 0$, but $Dist(S,R) > 0$.

To fix our definition of distance, we define the *Haussdorf metric* $H(R,S)$ between two compact sets R,S by setting

$$H(R,S) = Max\{Dist(R,S), Dist(S,R)\}.$$

Clearly, this definition is symmetric. Moreover the Haussdorf metric on compact sets has the following properties

Properties of Haussdorf Metric

i. $H(R,S) \geq 0$

ii. $H(R,S) = 0 \Leftrightarrow R = S$

iii. $H(R,S) = H(S,R)$

iv. $H(R,T) \leq H(R,S) + H(S,T)$

Properties *i, ii, iii* are immediate from the definition. To verify property *iv* is straightforward but somewhat tedious, so we shall not present the proof of property *iv* here. The interested reader can find the proof in Barnsley's book, *Fractals Everywhere,* on page 33.

Note that we need to restrict to closed sets, otherwise property *ii* would be false. For example, the distance between a line segment containing its end points and the same line segment without its end points would be zero, even though the two sets are clearly not identical. If we want the distance function to distinguish between different sets, we cannot have infinitely close sets; therefore, we are forced to restrict our attention to closed sets. Similarly, if we want to consider only finite distances, then we must restrict our attention to bounded sets. For these reasons we consider only compact sets -- that is, sets that are both closed and bounded.

Property *iv* is called the *triangular inequality*. We invoked the triangular inequality in the proof of Proposition 3 and we used Proposition 3 in the proof of the trivial fixed point theorem. Since we are going to invoke the trivial fixed point theorem to prove that fractals generated by iterated function systems are independent of the base case, we need the triangular inequality.

The space that provides the fractal framework for the trivial fixed point theorem is the space of compact sets where distance is measured by the Haussdorf metric. To invoke the trivial fixed point theorem, we need to know that this space is complete.

Recall that a space is *complete* if every cauchy sequence converges. Informally, a cauchy sequence of compact sets is a sequence of compact sets $\{S_n\}$ that get closer and closer as measured by the Haussdorf metric as n gets larger and larger. Formally, a sequence $\{S_n\}$ of compact sets is said to be cauchy if for every $\varepsilon > 0$ there is an integer N such that

$$H(S_{n+m}, S_n) < \varepsilon \text{ for all } n > N \text{ and all } m > 0.$$

For cauchy sequences of compact sets, we have the following theorem.

Completeness Theorem: *Every cauchy sequence of compact sets converges to a compact set.*

The main idea behind the proof of this theorem is that we can find the set S to which a cauchy sequence of compact sets $\{S_n\}$ converges by studying cauchy sequences of points $\{x_n\}$ where $x_n \in S_n$. In fact,

$$x \in S = Lim_{n\to\infty} S_n \quad \Leftrightarrow \quad x = Lim_{n\to\infty} x_n \quad x_n \in S_n$$

where $x = Lim_{n\to\infty} x_n$ is guaranteed to exist because ordinary Euclidean space is complete. We omit the gory details of this proof. The interested reader can find these details in Barnsley's book, *Fractals Everywhere*, on pages 35-37.

The trivial fixed point theorem is a result about contractive transformations on complete spaces. The space of compact sets with the Haussdorf metric is complete, but what are the contractive transformations on this space? Fractals are generated by iterated function systems, so if we are to invoke the trivial fixed point theorem, iterated function systems must be the contractive transformations that we seek. Let's investigate.

Suppose that w is a map of points in Euclidean space -- that is, w maps points to points. Then we can extend w to a map on sets of points S by setting

$$w(S) = \{w(x) \mid x \in S\}.$$

Now consider a collection of such maps $W = \{w_1,\ldots,w_l\}$. Again we can extend W to a map on sets S by setting

$$W(S) = w_1(S) \cup \cdots \cup w_l(S).$$

When the maps $\{w_1,\ldots,w_l\}$ are contractive, W is called an *iterated function system*. For the trivial fixed point theorem to apply, the map W must also be contractive.

Lemma: *If $w_1,\ldots,w_l$ are each contractive transformations on Euclidean space, then $W = \{w_1,\ldots,w_l\}$ is a contractive transformation on the space of compact sets.*

The essential idea behind the proof of this lemma is that since the transformations $w_1,\ldots,w_l$ bring points closer together, the map $W = \{w_1,\ldots,w_l\}$ must bring sets closer together since sets are collections of points. Again the details of the proof are straightforward but somewhat tedious, so we shall not present the proof here. The interested reader can find the proof in Barnsley's book, *Fractals Everywhere,* on pages *79–81*.

Now, finally, we have everything we need to invoke the trivial fixed point theorem. The complete space is the set of compact sets with the Haussdorf metric, and the contractive transformations are iterated functions systems -- collections of contractive transformations on Euclidean space.

Fractal Theorem

Suppose that

$W = \{w_1, \ldots, w_l\}$ *is an iterated function system.*

S_0 *is a compact set.*

$S_{n+1} = W(S_n)$ *for all* $n \geq 0$.

Then $S_\infty = Lim_{n \to \infty} S_n$ *exists, and* S_∞ *is the unique fixed point of W for any choice of* S_0.

Proof: This result follows immediately from the trivial fixed point theorem because W is a contractive map on a complete space.

Fractals are fixed points of iterated function systems. Therefore, given an iterated function system $W = \{w_1, \ldots, w_l\}$, we can generate the associated fractal A using the following algorithm.

Fractal Algorithm

- $A_0 = B$ (pick any compact set B)
- $A_{n+1} = W(A_n) = w_1(A_n) \cup \cdots \cup w_l(A_n)$
- A_n converges to the fractal (fixed point) A

The fractal algorithm is a consequence of the fractal theorem -- really then a consequence of the trivial fixed point theorem. Notice that this fractal algorithm can easily be modified to generate a fractal with a condensation set C simply by setting

$$A_{n+1} = W(A_n) \cup C = w_1(A_n) \cup \cdots \cup w_l(A_n) \cup C.$$

The fractal algorithm generates a fractal from an iterated function system. Conversely, if we know the fractal, but want to find the corresponding iterated function system so that we can generate the fractal, we can proceed in the following manner.

Fractal Strategy

Given a fractal A

- Find a collection of contractive transformations $W = \{w_1, \ldots, w_l\}$ that map the fractal A <u>onto</u> itself -- that is, find W so that $W(A) = w_1(A) \cup \cdots \cup w_l(A) = A$.
- Then A is the fractal generated by the iterated function system W starting from any compact set S.

Recall that we used this fractal strategy in Lecture 6 to find iterated functions systems for several different fractals.

4.0 Summary

The theme of this lecture -- the one idea you should carry away for the rest of your life -- is that if you want to find a stable point of any process, iterate! If you are lucky -- if the process actually converges to some equilibrium -- then by iterating you will arrive at an equilibrium point -- a fixed point of the process. Moreover, it does not matter how you start the process, just keep iterating. If you are unlucky, the iteration might diverge or there might be more than one equilibrium point. The trivial fixed point theorem guarantees that if the space of possible states is complete and the transformation defined by the process is contractive, then the iteration must converge to a unique equilibrium point of the process independent of the initial state of the system.

We provided three examples of fixed point methods: root finding, relaxation techniques, and fractals. Since the theme of this lecture is that equilibrium values are generated by iteration, we have solved the mystery of why fractals generated by iterated function systems are independent of the base case: contractive transformation always converge to a unique fixed point independent of the starting value.

Concerning fractals, the main things to take away from this lecture are the following three items, which summarize what we know about fractals generated from iterated function systems.

Fractal Theorem

Suppose that

- $W = \{w_1,\ldots,w_l\}$ *is an iterated function system.*
- S_0 *is a compact set.*
- $S_{n+1} = W(S_n)$ *for all* $n \geq 0$.

Then $S_\infty = Lim_{n\to\infty} S_n$ *exists, and* S_∞ *is the unique fixed point of W for any choice of* S_0.

Fractal Algorithm

- $A_0 = B$ (pick any compact set B)
- $A_{n+1} = W(A_n) = w_1(A_n) \cup \cdots \cup w_l(A_n)$
- A_n converges to the fractal (fixed point) A determined by the transformations W

Fractal Strategy

Given a fractal A

- Find a collection of contractive transformations $W = \{w_1,\ldots,w_l\}$ that map the fractal A <u>onto</u> itself -- that is, find W so that $W(A) = w_1(A) \cup \cdots \cup w_l(A) = A$.
- Then A is the fractal generated by the iterated function system W starting from any compact set S.

For easy comparison, below we have assembled a table containing some analogies between ordinary Euclidean space and the fractal space of compact sets.

Euclidian Space	*Fractal Space*
Points	Compact Sets
Euclidean Distance	Haussdorf Metric
Completeness -- Cauchy sequences of points converge	Completeness -- Cauchy sequences of sets converge
Contractive Maps	Iterated Function Systems
Fixed Points	Fractals
Trivial Fixed Point Theorem	Fractal Theorem
Fixed Point Algorithm -- Start with any point and iterate	Fractal Algorithm -- Start with any set and iterate

Table 1: Some analogies between Euclidean space and the fractal space of compact sets.

The term *fractal* has two distinct definitions in this text:

- recursion made visible
- fixed point of an iterated function system.

Each of these definitions reflects a different aspect of fractal curves. The first captures the spirit of recursive turtle programs; the second captures the essence of contractive transformations. We shall make the connection between iterated function systems and recursive turtle programs more precise in our next lecture. When we understand this connection, we shall finally understand why fractals generated by recursive turtle programs are also independent of the base case of the recursion.

Exercises:

1. Let A be an affine transformation. Then there is a matrix M and a vector w such that
   $A(v) = v * M$
   $A(P) = P * M + w$.
   a. Show that A has a unique fixed point if and only if $\det(M - I) \neq 0$, where I is the identity matrix.
   b. Show that when the condition in part *a* is satisfied, the unique fixed point of A is given by $P = -w * (M - I)^{-1}$.

2. Let A,B be two transformations and let $A \circ B$ denote the composite transformation. That is, $(A \circ B)(P) = A\big(B(P)\big)$. Show that if A and B are contractive transformations, then $A \circ B$ is a contractive transformation.

3. A transformation A is said to be a *rigid motion* if for every pair of points P,Q
$Dist\big(A(P),A(Q)\big) = Dist(P,Q)$.
Let $A \circ B$ denote the composite of A and B (see Exercise 2). Show that:
   a. if A and B are rigid motions, then $A \circ B$ is also a rigid motion.
   b. if A is a rigid motion and B is a contractive transformation, then $A \circ B$ and $B \circ A$ are both contractive transformations.

4. For each of the affine transformations $Trans(w)$, $Rot(Q,\theta)$, $Scale(Q,s)$, $Scale(Q,w,s)$:
   a. State whether or not the transformation is a contractive transformation.
   b. Find all the fixed points of the transformation.

5. What is the result of iterating the function $f(x) = x^2$ on the three initial values: $x_0 = 2$, $x_0 = 1$, $x_0 = 1/2$.

6. Show that the line tangent to the curve $y = F(x)$ at $x = x_n$ intersects the x-axis at
$$x_{n+1} = x_n - \frac{F(x_n)}{F'(x_n)}.$$

7. Let $F(x) = Cos(x) - x$. In Figure 2 we observed that Newton's method applied to this function diverges for the initial guess $x_0 = 3.08$. Show that the fixed point method applied to this function converges for the initial value $x_0 = 3.08$.

8. For 2 linear equations in 2 unknowns prove that:
   a. When the coefficient matrix is diagonally dominant, the transformations matrices associated with the Jacobi and Gauss-Seidel relaxation methods are contractive maps.
   b. Conclude that both Jacobi and Gauss-Seidel relaxation always converges to a unique fixed point when the coefficient matrix is diagonally dominant.

9. Show that for Gauss-Seidel relaxation:
   a. For 2 linear equations in 2 unknowns all the points generated after the initial guess satisfy the second equation -- that is, all the points lie on the second line.
   b. Generalize the result in part a to n linear equations in n unknowns by showing that all the points generated after the initial guess satisfy the last equation -- that is, all the points lie on the last hyperplane.

10. Show that if an iterated function system W consists of a single contractive transformation w, then the fractal corresponding to W consists of a single point P.

11. Let $W = \{w_1,\ldots,w_l\}$ be an iterated function system, and let F be the fractal generated by W. Show that if P_k is a fixed point of w_k, then $P_k \in F$.

12. Show that a fixed point of a contractive affine transformation w is equivalent to an eigenvector of w corresponding to the eigenvalue 1.

13. Let F_i be the fractal generated by the iterated function system W_i, $i = 1,2$. Show that if $W_1 \subset W_2$, then $F_1 \subset F_2$.

14. Let F be the fractal generated by the iterated function system $W = \{w_1,\ldots,w_l\}$, and let T be a nonsingular affine transformation.

a. Show that $T(F)$ is the fractal generated by the set of transformations

$$T^{-1} * W * T = \left\{T^{-1} * w_1 * T,\ldots,T^{-1} * w_l * T\right\}.$$

b. Conclude that an affine transformation applied to a fractal generates another fractal.

15. Consider two iterated functions systems $F = \{f_1,\ldots,f_m\}$ and $G = \{g_1,\ldots,g_m\}$.

a. Show that

$$H(t) = (1-t)F + tG = \{(1-t)f_1 + t\,g_1,\ldots,(1-t)\,f_m + t\,g_m\}$$

is an iterated function system for all $0 \le t \le 1$.

b. Using the result in part *a*, construct a collection of iterated function systems that morph
   a. the Sierpinski gasket into the fractal hangman.
   b. the Koch curve into a fractal bush.

16. Let f,g be continuous functions on the interval $[0,1]$, Define

$$dist(f,g) = \max_{0\le x\le 1}\left|f(x) - g(x)\right|$$

$$T(f)(x) = 1 + \int_0^x \int_0^t -f(s)\,ds\,dt$$

a. Show that

i. $dist(f,g) \ge 0$

ii. $dist(f,g) = 0 \Leftrightarrow f = g$

iii. $dist(f,g) = dist(g,f)$

iv. $dis(f,h) \le dist(f,g) + dist(g,h)$

b. Show that $T(f)$ is a contractive transformation. That is, show that

v. $dist\big(T(f),T(g)\big) \le dist\big(f,g\big)$

vi. $T(f)(0) = 1$ and $T(f)'(0) = 0$

c. Verify that $f(x) = \cos(x)$ is a fixed point of $T(f)$.

d. Verify that $f(x) = \cos(x)$ is the unique solution of the ordinary differential equation $f''(x) = -f(x)$ satisfying the initial conditions $f(0) = 1,\ f'(0) = 0$.

17. In this exercise we shall show how to use iterative fixed point methods to solve the second order ordinary differential equation

$$y'' = a_1(x)\,y' + a_0(x)\,y + b(x) \qquad (*)$$

subject to the initial conditions

$$y(0) = c_0 \text{ and } y'(0) = c_1. \qquad (**)$$

Let f,g be continuous functions on the interval $[0,1]$, and define

$$dist(f,g) = \max_{0\le x\le 1}\big|f(x) - g(x)\big|$$

$$T(f)(x) = c_0 + c_1 x + \int_0^x \int_0^t \big(a_1(s)\,f'(s) + a_0(s)\,f(s) + b(s)\big)\,ds\,dt$$

a. Show that

i. $dist(f,g) \ge 0$

ii. $dist(f,g) = 0 \Leftrightarrow f = g$

iii. $dist(f,g) = dist(g,f)$

iv. $dis(f,h) \le dist(f,g) + dist(g,h)$

b. Show that

v. $dist\big(T(f),T(g)\big) \le dist\big(f,g\big)$

vi. $T(f)(0) = c_0$ and $T(f)'(0) = c_1$

vii. $T(f)(x) = f \ \Leftrightarrow\ f''(x) = a_1(x)\,f'(x) + a_0(x)\,f(x) + b(x)$

c. Consider the sequence of continuous functions defined by setting

- $f_0 = g$
- $f_{n+1} = T(f_n)$

By invoking the Trivial Fixed Point Theorem, explain why $\{f_n\}$ converges to a solution of the differential equation (*) that satisfies the initial conditions (**) independent of the choice of *g*.

d. Use the result of part *c* to develop an iterative algorithm for solving second order ordinary differential equations.

e. Use the method in part *d* to solve the ordinary differential equation $y'' = -y$ subject to the initial conditions $y(0) = 1$ and $y'(0) = 0$, starting with the initial guess $y = 0$.

Programming Projects:

1. *Fractal Tennis*

In this project we provide an alternative approach to rendering fractals F corresponding to iterated function systems $W = \{w_1,\ldots,w_l\}$.

Fractal Tennis

i. Select any point P_0 in F. {see below}

ii. Choose a point $P_{n+1} \in \{w_1(P_n),\ldots,w_l(P_n)\}$ -- pick a point at random by assigning probabilities to the maps $\{w_1,\ldots,w_l\}$.

iii. Display the points $P = \{P_0,P_1,\ldots\}$ -- these points form a dense subset of the fractal F generated by the IFS W.

There are two ways to select a point P_0 in the fractal F:

- Find an eigenvector corresponding to the eigenvalue 1 for one of the transformations w_k (see Exercises 11 and 12).
- Pick any point Q in the plane and apply step *ii n* times for n sufficiently large.

Note that *Fractal Tennis* is more efficient than the standard *Fractal Algorithm* because only one additional point is computed and stored at each stage of the algorithm.

a. Implement *Fractal Tennis* in your favorite programming language using your favorite API.

b. Use *Fractal Tennis* to generate the following fractals:
   i. The Sierpinski gasket and Koch curve.
   ii. The fractals in Lecture 2, Figure 8 left and center.
   iii. The fractals in Lecture 2, Figure 11.
   iv The fractals in Lecture 2, Figure 16.
   v. Some novel fractals of your own design.

2. *Root Finding*
   a. Implement Newton's method and the Fixed Point Method in your favorite programming language, using your favorite API to make pictures like those in Figures 1-3 of the resulting iterations.
   b. Compare the speed and accuracy of these two root finding methods for different functions and different initial guesses.

3. *Relaxation Methods*
   a. Implement the Jacob and Gauss-Seidel relaxation methods in your favorite programming language.
   b. Compare the speed and accuracy of these two methods to standard methods such as Gaussian elimination for solving large systems of linear equations.

4. *Second Order Ordinary Differential Equations*
   a. Implement in your favorite programming language the iterative fixed point method described in Exercise 16 for solving second order ordinary differential equations

      $$y'' = a_1(x)\,y' + a_0(x)\,y + b(x) \qquad (*)$$

      subject to the initial conditions

      $$y(0) = c_0 \quad \text{and} \quad y'(0) = c_1 . \qquad (**)$$

   b. Using your favorite API, make pictures of the functions generated by successive iterations of this algorithm.
   c. Whenever closed form solutions exist, compare your solutions to these closed form expressions.