Maplesoft
Mathematics • Modeling • Simulation
Application Demonstration
www.maplesoft.com

# Classroom Tips and Techniques:

# Fixed-Point Iteration

Robert J. Lopez
Emeritus Professor of Mathematics and Maple Fellow
Maplesoft

## Introduction

Fixed-point iteration, also called Picard iteration, linear iteration, and repeated substitution, is easy to investigate in Maple for the scalar case.  The syntax for the vector case is a bit more complex, so we show how to define a vector-valued function of a vector argument.

*Note*: In this article, our implementation of the Maple calculations is command-based and uses Maple syntax for mathematical expressions.  It is worthing noting that standard math notation is also available, and many of these calculations could also have been implemented interactively, using Clickable Math tools such as palettes and context-sensitive menus.

## Initializations

```
> restart;
> with(plots):
```

## Fixed-Point Iteration: Scalar Case

Starting from $x=x_0$, fixed-point iteration for the scalar function $g(x)$ generates the sequence $\{x_k\}$ by computing

$$x_{k+1}=g(x_k)$$

Under the right conditions on $g(x)$, this sequence converges to a fixed point defined by the equation $x=g(x)$.  This process is easy to demonstrate in the scalar case.  For example, the quadratic equation

```
> q := x^2-6*x+5 = 0;
```

$$q := x^2-6\,x+5=0$$

has the two solutions

```
> solve(q,x);
```

$$5, 1$$

Rearranging the equation to the form $x = g(x)$ by the steps

```
> q1 := q + 6*x;
  q2 := q1/6;
```

$$q1 := x^2 + 5 = 6\,x$$
$$q2 := \frac{1}{6}\,x^2 + \frac{5}{6} = x$$

allows us to define the function $g(x)$ via

```
> g := unapply(lhs(q2),x):
  'g'(x) = g(x);
```

$$g(x) = \frac{1}{6}\,x^2 + \frac{5}{6}$$

The solutions of the quadratic equation are fixed points of $g(x)$, as we see from

```
> 1 = g(1);
  5 = g(5);
```

$$1 = 1$$
$$5 = 5$$

The sequence defined by $x_{k+1} = g\left(x_k\right)$ will converge to $x = 1$ if $x_0$ is sufficiently close to 1 because $|g'(1)| < 1$, as shown by

```
> `g'`(1) = D(g)(1);
```

$$g'(1) = \frac{1}{3}$$

To demonstrate that fixed-point iteration starting at $x = 4$ converges to $x = 1$, we work in floating-point arithmetic and write

```
> X := 4.0;
  for k from 1 to 10 do
  X := g(X);
  print(x[k] = X);
  od:
```

$$X := 4.0$$
$$x_1 = 3.500000000$$
$$x_2 = 2.875000000$$
$$x_3 = 2.210937500$$
$$x_4 = 1.648040772$$
$$x_5 = 1.286006398$$
$$x_6 = 1.108968743$$

$$x_7 = 1.038301946$$

$$x_8 = 1.013011822$$

$$x_9 = 1.004365492$$

$$x_{10} = 1.001458340$$

Maple's ability to implement the scalar function $g(x)$ makes fixed-point calculations natural in the scalar case.

## Fixed-Point Iteration: Vector Case

If the vector $\mathbf{x}_0$ satisfies the equation $\mathbf{x} = \mathbf{g}(\mathbf{x})$, where $\mathbf{g}(\mathbf{x})$ is a vector-valued function of the vector argument $\mathbf{x}$, then $\mathbf{x}_0$ is a fixed-point of $\mathbf{g}(\mathbf{x})$. As in the scalar case, fixed-point iteration can be used to find solutions to equations of the form $\mathbf{F}(\mathbf{x}) = \mathbf{0}$, where $\mathbf{F}(\mathbf{x})$ is itself a vector-valued function of the vector argument $\mathbf{x}$.

For example, if $\mathbf{x}$ is the vector

```
> X := <x, y>;
```

$$X := \begin{bmatrix} x \\ y \end{bmatrix}$$

and the function $\mathbf{F}(\mathbf{x})$ is the vector

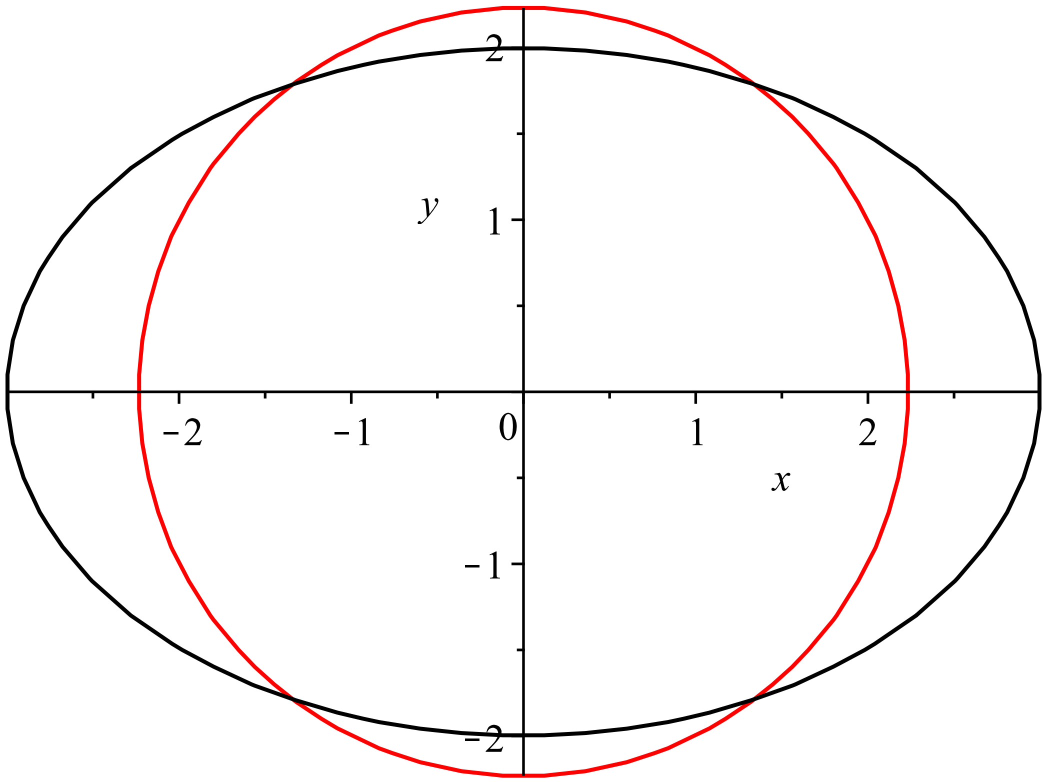
```
> F := <x^2 + y^2 - 5, 4*x^2 + 9*y^2 - 36>;
```

$$F := \begin{bmatrix} x^2 + y^2 - 5 \\ 4\,x^2 + 9\,y^2 - 36 \end{bmatrix}$$

then Figure 1 shows that the equations determined by $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ have four real solutions.

```
> implicitplot(convert(F,list), x=-3..3, y=-2.5..2.5, color=[red,
  black], scaling=constrained, title="Figure 1");
```

Figure 1



Maple determines these four solutions to be

```
> sol := solve(convert(F,set),{x,y},explicit);
```

$$sol := \left\{x=\frac{3}{5}\sqrt{5}, y=\frac{4}{5}\sqrt{5}\right\}, \left\{x=\frac{3}{5}\sqrt{5}, y=-\frac{4}{5}\sqrt{5}\right\}, \left\{x=-\frac{3}{5}\sqrt{5}, y=\frac{4}{5}\sqrt{5}\right\}, \left\{x=-\frac{3}{5}\sqrt{5}, y=-\frac{4}{5}\sqrt{5}\right\}$$

Using exact arithmetic, these solutions can be written as the vectors

```
> seq(eval(X,sol[k]),k=1..4);
```

$$\begin{bmatrix} \frac{3}{5}\sqrt{5} \\ \frac{4}{5}\sqrt{5} \end{bmatrix}, \begin{bmatrix} \frac{3}{5}\sqrt{5} \\ -\frac{4}{5}\sqrt{5} \end{bmatrix}, \begin{bmatrix} -\frac{3}{5}\sqrt{5} \\ \frac{4}{5}\sqrt{5} \end{bmatrix}, \begin{bmatrix} -\frac{3}{5}\sqrt{5} \\ -\frac{4}{5}\sqrt{5} \end{bmatrix}$$

or as the vectors

```
> seq(eval(X,evalf(sol[k])),k=1..4);
```

$$\begin{bmatrix}1.341640786\\1.788854382\end{bmatrix}, \begin{bmatrix}1.341640786\\-1.788854382\end{bmatrix}, \begin{bmatrix}-1.341640786\\1.788854382\end{bmatrix}, \begin{bmatrix}-1.341640786\\-1.788854382\end{bmatrix}$$

in floating-point arithmetic.

The first-quadrant solution can be found by fixed-point iteration if the equation $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ is cast into the form $\mathbf{x} = \mathbf{g}(\mathbf{x})$ by defining the vector $\mathbf{g}(\mathbf{x})$ as follows.

```
> g := <solve(F[1],x)[1], solve(F[2],y)[1]>;
```

$$g := \begin{bmatrix}\sqrt{-y^2+5}\\ \frac{2}{3}\sqrt{-x^2+9}\end{bmatrix}$$

That the first-quadrant solution is a fixed point of this $\mathbf{g}(\mathbf{x})$ is demonstrated by the calculation

```
> simplify(eval(X = g, sol[1]));
```

$$\begin{bmatrix}\frac{3}{5}\sqrt{5}\\ \frac{4}{5}\sqrt{5}\end{bmatrix} = \begin{bmatrix}\frac{3}{5}\sqrt{5}\\ \frac{4}{5}\sqrt{5}\end{bmatrix}$$

We next provide several Maple implementations of fixed-point iteration for the vector-valued function $\mathbf{g}(\mathbf{x})$. First, we use the vector $\mathbf{g}(\mathbf{x})$ as an *expression* into which we must make substitutions of the form $x = x_k, y = y_k$. To obtain the substitution rules at each step of the iteration, we use the top-level **Equate** command. This command takes two lists or vectors, and forms the equations determined by equating like components. Thus, we equate the components of the vectors

$$\mathbf{X} = \begin{bmatrix}x\\y\end{bmatrix} \text{ and } \mathbf{x}_k = \begin{bmatrix}x_k\\y_k\end{bmatrix}$$

and start the fixed-point iteration at $\mathbf{x}_0 = \begin{bmatrix}0.1\\0.1\end{bmatrix}$, so that we have

```
> XX := <.1,.1>;
  for k from 1 to 5 do
  XXX := Equate(X,XX);
  XX := eval(g,XXX);
  print(XX);
  od:
```

$$XX := \begin{bmatrix}0.1\\0.1\end{bmatrix}$$

$$\begin{bmatrix} 2.233830790 \\ 1.998888580 \end{bmatrix}$$

$$\begin{bmatrix} 1.002219759 \\ 1.334998960 \end{bmatrix}$$

$$\begin{bmatrix} 1.793816539 \\ 1.885094227 \end{bmatrix}$$

$$\begin{bmatrix} 1.202671923 \\ 1.603083449 \end{bmatrix}$$

$$\begin{bmatrix} 1.558885325 \\ 1.832251832 \end{bmatrix}$$

an iteration that slowly converges to the first-quadrant fixed point.

Alternatively, in an attempt to make a function out of the vector $\mathbf{g}(\mathbf{x})$, we write

```
> G := unapply(g,x,y):
  'G'(x,y) = G(x,y);
```

$$G(x, y) = \begin{bmatrix} \sqrt{-y^2+5} \\ \frac{2}{3}\sqrt{-x^2+9} \end{bmatrix}$$

The argument of this function is not a vector, and we did not create a vector-valued function of a *vector* argument. The argument of this function is the sequence of coordinates $x, y$. To implement fixed-point iteration with this function, we need to convert the output vectors to a sequence of two coordinates. We do this by converting the output vector to a list, then removing the brackets from the list by appending the brackets [ ]. Thus, the iteration can be written as

```
> X := <.1,.1>;
  for k from 1 to 5 do
  X := G(convert(X,list)[]);
  od;
```

$$X := \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$$

$$X := \begin{bmatrix} 2.233830790 \\ 1.998888580 \end{bmatrix}$$

$$X := \begin{bmatrix} 1.002219759 \\ 1.334998960 \end{bmatrix}$$

$$X := \begin{bmatrix} 1.793816539 \\ 1.885094227 \end{bmatrix}$$

$$X := \begin{bmatrix} 1.202671923 \\ 1.603083449 \end{bmatrix}$$

$$X := \begin{bmatrix} 1.558885325 \\ 1.832251832 \end{bmatrix}$$

A true vector-valued function of a vector argument, namely, $\mathbf{h}(\mathbf{v})$, is most easily created with the following syntax.

```
> h := v-> Vector([sqrt(5-v[2]^2), 2*sqrt(9-v[1]^2)/3]);
```

$$h := v \to Vector\left(\left[\sqrt{-v_2^2+5}, \frac{2}{3}\sqrt{-v_1^2+9}\right]\right)$$

Then, the requisite iteration can be implemented with

```
> X := <.1,.1>;
  for k from 1 to 5 do
  X := h(X);
  end do;
```

$$X := \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$$

$$X := \begin{bmatrix} 2.233830790 \\ 1.998888580 \end{bmatrix}$$

$$X := \begin{bmatrix} 1.002219759 \\ 1.334998960 \end{bmatrix}$$

$$X := \begin{bmatrix} 1.793816539 \\ 1.885094227 \end{bmatrix}$$

$$X := \begin{bmatrix} 1.202671923 \\ 1.603083449 \end{bmatrix}$$

$$X := \begin{bmatrix} 1.558885325 \\ 1.832251832 \end{bmatrix}$$