1-30-1994

The existence and uniqueness theorem - Picard iteration.

Name: ______________________________

This project is due by 3 p.m., Friday, February 18.

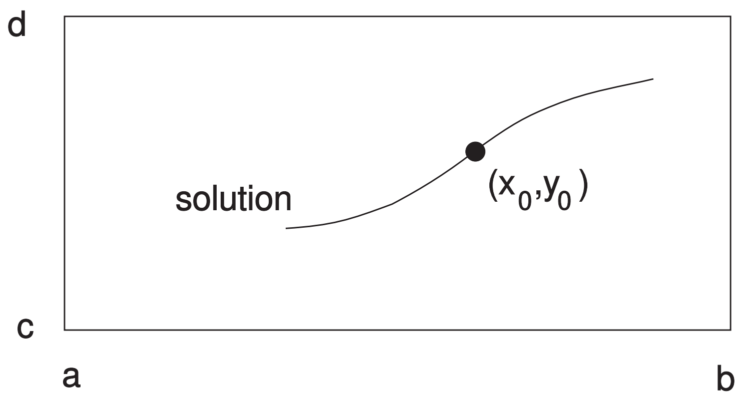
The **existence and uniqueness theorem** for first-order equations gives conditions under which there is a unique solution to

$$y' = f(x, y) \quad \text{satisfying the initial condition} \quad y(x_0) = y_0.$$

It says, in particular, that if f and $\dfrac{\partial f}{\partial y}$ are continuous in an open rectangle

$$\{(x, y) \mid a < x < b, c < y < d\}$$

containing the initial condition point (x_0, y_0), then there is a unique solution passing through the point. The solution will be valid on some interval about x_0 contained in the interval $a < x < b$.



This project will give you an idea of why the theorem is true. The proof of the theorem constructs a solution using a method called **Picard iteration**. It is related to the iterative method for solving equations, which I'll review first.

1. Solving equations by iteration.

It's often possible to solve equations of the form

$$x = f(x)$$

by **iterating** the function f. Here's an example.

Example. $x = \cos x$

There is no way to solve this equation algebraically, but it's easy to approximate a solution to any degree of accuracy. You can do this on a calculator: Set your calculator to radian mode, and enter 1. Then press the "**cos**" button repeatedly. You will get a sequence of numbers like this:

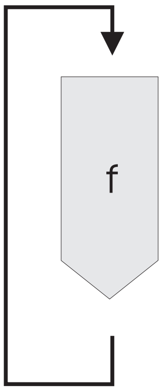
$$1, 0.54030, 0.85755, 0.65429, 0.79348, 0.70137, \ldots$$

You will notice the numbers converging. After pressing the "**cos**" button around 37 times, the numbers will be stable to around 5 places. You should get 0.73909. This is (approximately) a solution to the equation $x = \cos x$, as you can verify by plugging it into both sides.

The numbers in the sequence above are the **iterates** of $f(x) = \cos x$:

$$1, \cos 1, \cos(\cos 1), \cos(\cos(\cos 1)), \ldots$$

(Note that $\cos(\cos 1) \neq (\cos 1)^2$.) You take the **output** of the last step and feed it back in as the **input** of the next step.



The *Mathematica* command to iterate a function is called `NestList`. Here is how to use it to construct a list of the first 40 iterates of $f(x) = \cos x$ starting at $x = 1$:

```
NestList[Cos, 1., 40]
```

You have to write "`1.`" to force *Mathematica* to evaluate the results in decimal form.

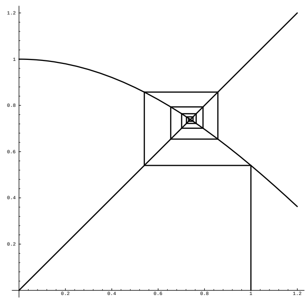
You can represent this process graphically by constructing a **cobweb diagram**. Draw the reference line $y = x$, and draw a graph of the function $y = f(x)$ that you're iterating (In this example, it would be $y = \cos x$.)

Begin at the starting value $x = 1$. Draw a **vertical line** from $x = 1$ to the curve $y = \cos x$. This corresponds to plugging x into $f(x) = \cos x$.

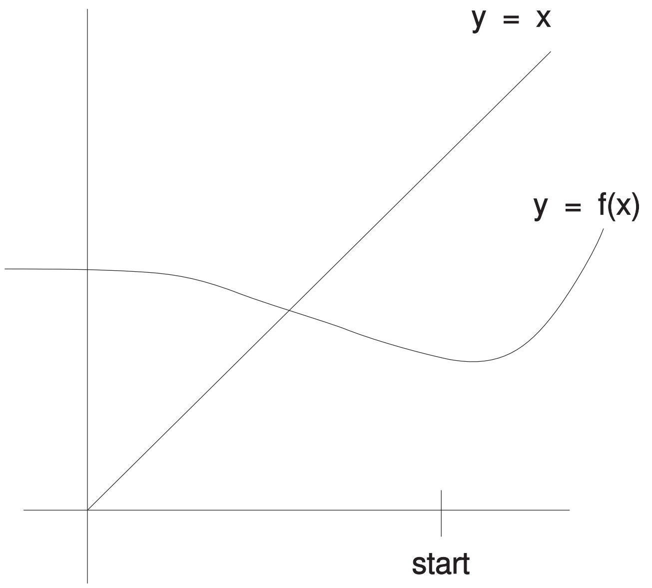
Starting at your current position on the curve, draw a horizontal line to the reference line $y = x$. This corresponds to making the output (y) the new input (x).

Keep going. Draw a vertical line from your current position to the curve $y = \cos x$. Then draw a horizontal line from your current position to the reference line $y = x$. And so on.

The first few steps of this procedure are illustrated in the picture below.



1. (a) Draw the cobweb diagram for the function $y = f(x)$ which is pictured below. Begin at the point on the x-axis labelled "**start**".



(b) Use the **NestList** function and iteration to approximate the solution to $x = e^{-x^2}$. Find the solution to 5 decimal places. (Hint: Begin by defining `f[x_] := E^(-x^2)`. You will need around 100 iterates to get 5 place accuracy.)

2. Picard iteration.

In this section, you'll see how **Picard iteration** is used to generate the solution to a differential equation

$$y' = f(x, y) \quad \text{with initial condition} y(0) = 0.$$

(The initial condition $y(0) = 0$ may seem "special", but it really isn't. If, for instance, the initial condition was $y(2) = -7$, you could substitute $u = x - 2$ and $v = y + 7$. This would yield a differential equation in u and v with the initial condition $v(0) = 0$.)

Suppose $y = \phi(x)$ is the solution. Since it's the solution, it satisfies

$$\frac{d}{dx}\phi(x) = f(x, \phi(x)).$$

Integrate both sides from $x = 0$ to $x = t$:

$$\phi(t) = \int_0^t f(x, \phi(x))\, dx.$$

The last equation is interesting because it has the same form as the iteration equation $x = f(x)$:

$$\square \;=\; f(\square)$$

$$\square \;=\; \int_0^t f(x, \square)\ dx$$

The similarity in form suggests constructing a solution to the differential equation by iteration. In this case, the input/output would be a *function*, rather than a *number*.

Begin by defining a function `Picard` which will perform the iteration for us*. It takes as inputs the function $f(x, y)$ from the differential equation $y' = f(x, y)$, the initial guess $\phi(x)$, and the number of iterations n.

```
Picard[f_, phi_, n_] := Integrate[f[x, Picard[f, phi, n-1][x]],
    {x, 0, #}]&/; n > 0
```

This definition will do the iteration, but I have to make sure it *begins* with $\phi(x)$:

```
Picard[f_, phi_, 0] := phi
```

Notice that the output of `Picard` is a function. It is the function whose input will be `#`, in fact; the "`#`" and "`&`" which appear in the definition are *Mathematica*'s "pure function" notation.

As an example, I'll use `Picard` to generate a solution to

$$y' = x^2 + y^2, \quad y(0) = 0.$$

First, enter the function f:

```
f1[x_, y_] := x^2 + y^2
```

Take $\phi(x) = x$ as our initial guess. (Notice that $\phi(0) = 0$, so it satisfies the initial condition.)

```
p1[x_] := x
```

Next, use `Picard` to generate a table of the first 3 iterates:

```
example = Table[Picard[f1, p1, i][t], {i, 0, 3}]
```

Here are the first few iterates worked out by hand. Check them against the entries in the table that generated by the previous command.

$$\phi_0(t) = t$$

$$\phi_1(t) = \int_0^t (x^2 + x^2)\,dx = \int_0^t 2x^2\,dx = \frac{2}{3}t^3$$

$$\phi_2(t) = \int_0^t \left(x^2 + \left(\frac{2}{3}x^3\right)^2\right)\,dx = \frac{1}{3}t^3 + \frac{4}{63}t^7$$

$$\phi_3(t) = \int_0^t \left(x^2 + \left(\frac{1}{3}x^3 + \frac{4}{63}x^7\right)^2\right)\,dx = \frac{1}{3}t^3 + \frac{1}{63}t^7 + \frac{8}{2079}t^{11} + \frac{16}{59535}t^{15}$$

---

* I would like to thank Steve Izen for writing this function.

Notice that at each stage the previous ϕ is substituted for y in $f(x, y) = x^2 + y^2$. Then you integrate to get the next ϕ.

You can check graphically that the iterates are converging to a solution by plotting the successive polynomials:

```
Plot[Evaluate[example], {t, 0, 1}, AspectRatio->Automatic]
```

The proof of the existence and uniqueness theorem uses Picard iteration to construct a solution. The important ideas were illustrated above, but there are many technical points which need to be addressed when you're actually carrying out the proof. While these points are beyond the scope of this discussion, I'll mention one of them by way of example: How do you know that the iteration procedure actually converges?

Consider the simpler question of iterating a function (as with the cobweb diagrams in the first section). It is not true that the iteration will always converge. If you iterate $f(x) = x^2$ starting at $x = 2$, you get:

$$2, 4, 8, 16, 32, 64, \ldots$$

Part of the full-blown proof of the existence and uniqueness theorem addresses the issue of convergence. In fact, under the continuity assumptions of the theorem, the Picard iterates *will* converge to a solution.

2. (a) Use `Picard` to find the first 5 Picard approximations to the solution to

$$y' = y + 2 \quad \text{with the initial condition} \quad y(0) = 0.$$

Here are some hints to get you started. Begin by defining the function f and the initial guess:

```
f2[x_, y_] := y + 2

p2[x_] := x
```

Next, define a table named `problem2` in a manner analogous to `example` above. You should use `{i, 0, 5}` to get the first 5 iterates.

Write down the *last* polynomial iterate below.

(b) Solve the differential equation $y' = y + 2$, $y(0) = 0$ directly, using one of the solution methods you've learned. Show your work below.

(c) Use the `Plot` command to draw a picture of your solution for $0 \leq x \leq 1$. Then use `Plot` to draw a picture of the last polynomial iterate for $0 \leq t \leq 1$ (the polynomial variable will probably be t if you followed the steps in the example). To save typing, you can refer to the polynomial as "`problem2[[5]]`" when you enter it in the `Plot` command.

The pictures should be the same. Draw either of the pictures below.

(d) Fill in the blank in the following sentence: "The polynomial iterates produced by Picard iteration are just the ______________________________ polynomials for the solution in part (b)." (Hint: It's *someone's name.*)

3. (a) Use `Picard` to find the first 2 Picard approximations to the solution to

$$y' = x^2y^2 - xy + 1 \quad \text{satisfying} \quad y(0) = 0.$$

("The first 2" means that in your `Table` command, you should use `{i, 0, 2}`.) Use $\phi(x) = x$ as your initial guess, and write down the *last* iterate below. Use `problem3` as the name of your table of iterates so you can do the next part of the problem.

(If you want to see something mildly amusing, try using `{i, 0, 4}` instead. That 4-th iterate has a lot of terms, doesn't it?)

(b) Use `Plot` to draw a picture of the last polynomial iterate for $0 \le t \le 1$ (the polynomial variable will probably be t if you followed the steps in the example). To save typing, you can refer to the polynomial as "`problem3[[2]]`" when you enter it in the `Plot` command.

Draw the graph below.

(c) If $y(x)$ is the solution to $y' = x^2y^2 - xy + 1$ satisfying $y(0) = 0$, what is the approximate value of $y(1)$? (You can either look at the graph in (b), or you can use

```
problem3[[2]] /.  t->1.0
```