Fuzzy vs. Nonfuzzy Logic

In this section...

"The Basic Tipping Problem" on page 1-12
"The Nonfuzzy Approach" on page 1-12
"The Fuzzy Logic Approach" on page 1-16
"Problem Solution" on page 1-17

The Basic Tipping Problem

To illustrate the value of fuzzy logic, examine both linear and fuzzy approaches to the following problem:

What is the right amount to tip your waitperson?

First, work through this problem the conventional (nonfuzzy) way, writing MATLAB commands that spell out linear and piecewise-linear relations. Then, look at the same system using fuzzy logic.

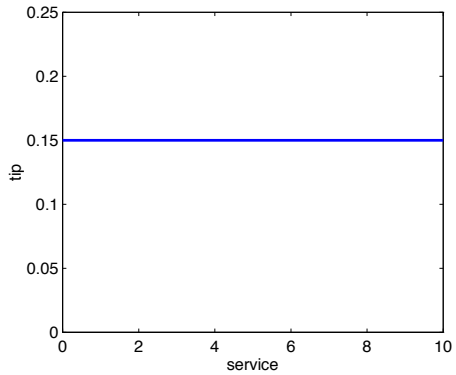
The Basic Tipping Problem. Given a number between 0 and 10 that represents the quality of service at a restaurant (where 10 is excellent), what should the tip be?

Note This problem is based on tipping as it is typically practiced in the United States. An average tip for a meal in the U.S. is 15%, though the actual amount may vary depending on the quality of the service provided.

The Nonfuzzy Approach

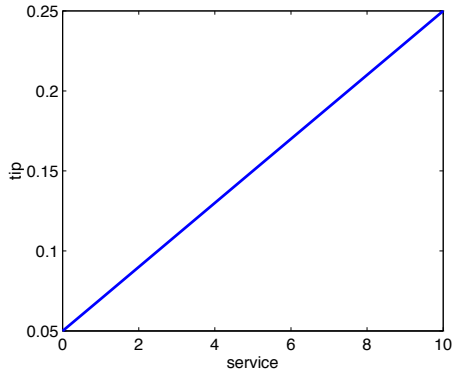
Begin with the simplest possible relationship. Suppose that the tip always equals 15% of the total bill.

```
tip = 0.15
```



This relationship does not take into account the quality of the service, so you need to add a new term to the equation. Because service is rated on a scale of 0 to 10, you might have the tip go linearly from 5% if the service is bad to 25% if the service is excellent. Now the relation looks like the following plot:

```
tip=0.20/10*service+0.05
```

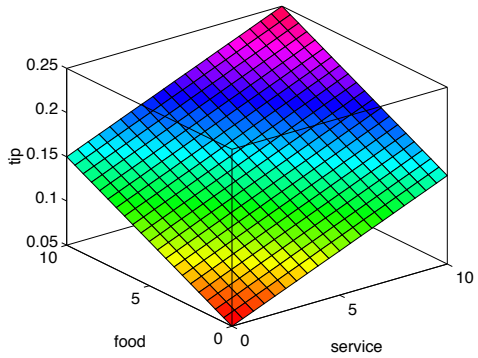


The formula does what you want it to do, and is straightforward. However, you may want the tip to reflect the quality of the food as well. This extension of the problem is defined as follows.

The Extended Tipping Problem. Given two sets of numbers between 0 and 10 (where 10 is excellent) that respectively represent the quality of the service and the quality of the food at a restaurant, what should the tip be?

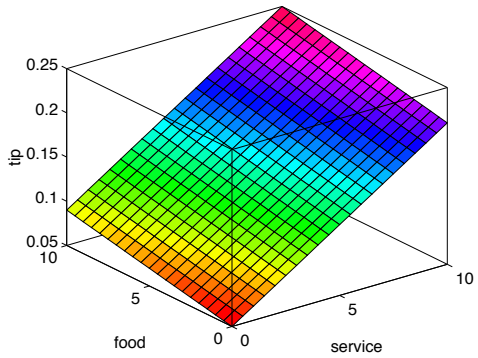
See how the formula is affected now that you have added another variable. Try the following equation:

```
tip = 0.20/20*(service+food)+0.05;
```



In this case, the results look satisfactory, but when you look at them closely, they do not seem quite right. Suppose you want the service to be a more important factor than the food quality. Specify that service accounts for 80% of the overall tipping grade and the food makes up the other 20%. Try this equation:

```
servRatio=0.8;
tip=servRatio*(0.20/10*service+0.05) + ...
    (1-servRatio)*(0.20/10*food+0.05);
```

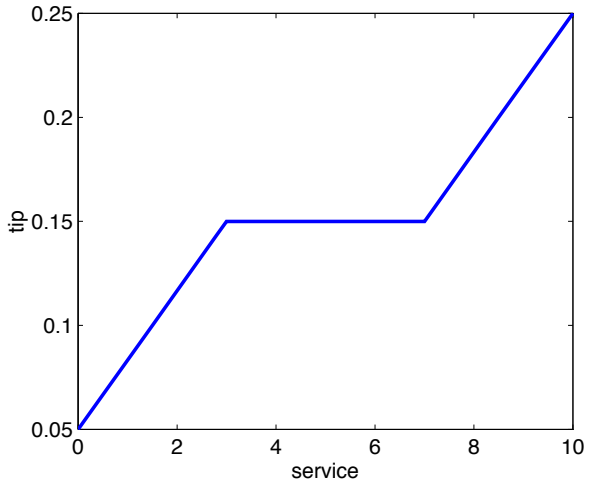


The response is still somehow too uniformly linear. Suppose you want more of a flat response in the middle, i.e., you want to give a 15% tip in general, but

want to also specify a variation if the service is exceptionally good or bad. This factor, in turn, means that the previous linear mappings no longer apply. You can still use the linear calculation with a piecewise linear construction. Now, return to the one-dimensional problem of just considering the service. You can string together a simple conditional statement using breakpoints like this.

```
if service<3,
    tip=(0.10/3)*service+0.05;
elseif service<7,
    tip=0.15;
elseif service<=10,
    tip=(0.10/3)*(service-7)+0.15;
end
```

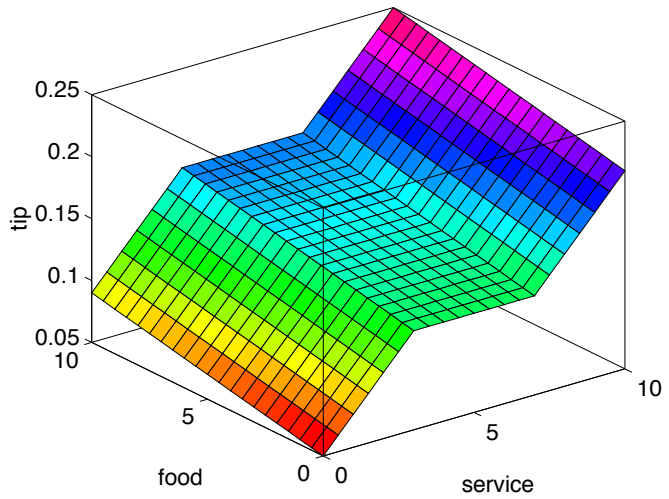
The plot now looks like the following figure:



If you extend this to two dimensions, where you take food into account again, something like the following output results.

```
servRatio=0.8;
if service<3,
    tip=((0.10/3)*service+0.05)*servRatio + ...
        (1-servRatio)*(0.20/10*food+0.05);
elseif service<7,
```

```
    tip=(0.15)*servRatio + ...
        (1-servRatio)*(0.20/10*food+0.05);
else,
    tip=((0.10/3)*(service-7)+0.15)*servRatio + ...
        (1-servRatio)*(0.20/10*food+0.05);
end
```



The plot looks good, but the function is surprisingly complicated. It was a little difficult to code this correctly, and it is definitely not easy to modify this code in the future. Moreover, it is even less apparent how the algorithm works to someone who did not see the original design process.

The Fuzzy Logic Approach

You need to capture the essentials of this problem, leaving aside all the factors that could be arbitrary. If you make a list of what really matters in this problem, you might end up with the following rule descriptions.

Tipping Problem Rules — Service Factor
If service is poor, then tip is cheap
If service is good, then tip is average
If service is excellent, then tip is generous

The order in which the rules are presented here is arbitrary. It does not matter which rules come first. If you want to include the food's effect on the tip, add the following two rules.

Tipping Problem Rules — Food Factor
If food is rancid, then tip is cheap
If food is delicious, then tip is generous

You can combine the two different lists of rules into one tight list of three rules like so.

Tipping Problem — Both Service and Food Factors
If service is poor or the food is rancid, then tip is cheap
If service is good, then tip is average
If service is excellent or food is delicious, then tip is generous

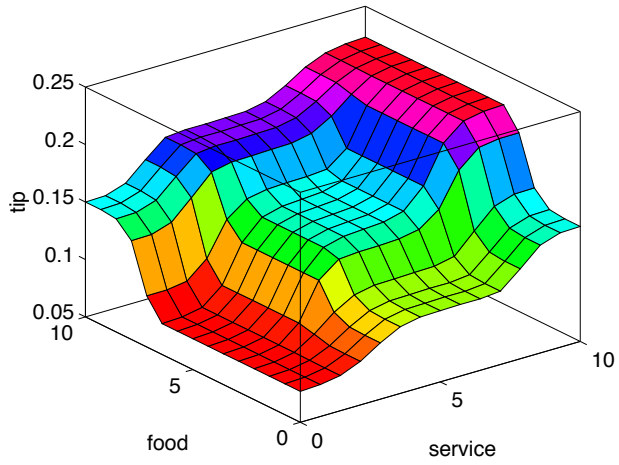
These three rules are the core of your solution. Coincidentally, you have just defined the rules for a fuzzy logic system. When you give mathematical meaning to the linguistic variables (what is an average tip, for example?) you have a complete fuzzy inference system. The methodology of fuzzy logic must also consider:

- How are the rules all combined?
- How do I define mathematically what an average tip is?

The next few chapters provide detailed answers to these questions. The details of the method don't really change much from problem to problem—the mechanics of fuzzy logic aren't terribly complex. What matters is that you understand that fuzzy logic is adaptable, simple, and easily applied.

Problem Solution

The following plot represents the fuzzy logic system that solves the tipping problem.



This plot was generated by the three rules that accounted for both service and food factors. The mechanics of how fuzzy inference works is explained in "Overview" on page 2-2 of "Foundations of Fuzzy Logic" on page 2-2. In "Build Mamdani Systems (GUI)" on page 2-34, the entire tipping problem is worked through using the Fuzzy Logic Toolbox graphical tools.

Observations

Consider some observations about the example so far. You found a piecewise linear relation that solved the problem. It worked, but it was problematic to derive, and when you wrote it down as code, it was not very easy to interpret. Conversely, the fuzzy logic system is based on some common sense statements. Also, you were able to add two more rules to the bottom of the list that influenced the shape of the overall output without needing to undo what had already been done, making the subsequent modification was relatively easy.

Moreover, by using fuzzy logic rules, the maintenance of the structure of the algorithm decouples along fairly clean lines. The notion of an average tip might change from day to day, city to city, country to country, but the underlying logic is the same: if the service is good, the tip should be average.

Recalibrating the Method

You can recalibrate the method quickly by simply shifting the fuzzy set that defines average without rewriting the fuzzy logic rules.

You can shift lists of piecewise linear functions, but there is a greater likelihood that recalibration will not be so quick and simple.

In the following example, the piecewise linear tipping problem slightly rewritten to make it more generic. It performs the same function as before, only now the constants can be easily changed.

```
% Establish constants
lowTip=0.05; averTip=0.15; highTip=0.25;
tipRange=highTip-lowTip;
badService=0; okayService=3;
goodService=7; greatService=10;
serviceRange=greatService-badService;
badFood=0; greatFood=10;
foodRange=greatFood-badFood;

% If service is poor or food is rancid, tip is cheap
if service<okayService,
    tip=(((averTip-lowTip)/(okayService-badService)) ...
        *service+lowTip)*servRatio + ...
        (1-servRatio)*(tipRange/foodRange*food+lowTip);
% If service is good, tip is average
elseif service<goodService,
    tip=averTip*servRatio + (1-servRatio)* ...
        (tipRange/foodRange*food+lowTip);
% If service is excellent or food is delicious, tip is generous
else,
    tip=(((highTip-averTip)/ ...
        (greatService-goodService))* ...
        (service-goodService)+averTip)*servRatio + ...
        (1-servRatio)*(tipRange/foodRange*food+lowTip);
end
```

As with all code, the more generality that is introduced, the less precise the algorithm becomes. While you can improve clarity by adding more comments, or perhaps rewriting the algorithm in slightly more self-evident ways, but the piecewise linear methodology is not the optimal way to resolve this issue.

If you remove everything from the algorithm except for three comments, what remain are exactly the fuzzy logic rules you previously wrote down.

```
% If service is poor or food is rancid, tip is cheap
% If service is good, tip is average
% If service is excellent or food is delicious, tip is generous
```

If, as with a fuzzy system, the comment is identical with the code, think how much more likely your code is to have comments. Fuzzy logic lets the language that is clearest to you, high level comments, also have meaning to the machine, which is why it is a very successful technique for bridging the gap between people and machines.

By making the equations as simple as possible (linear) you make things simpler for the machine but more complicated for you. However, the limitation is really no longer the computer—it is your mental model of what the computer is doing. Computers have the ability to make things hopelessly complex; fuzzy logic reclaims the middle ground and lets the machine work with your preferences rather than the other way around.