Originally published in The Technology Teacher, *October 2004, by the International Technology Education Association*

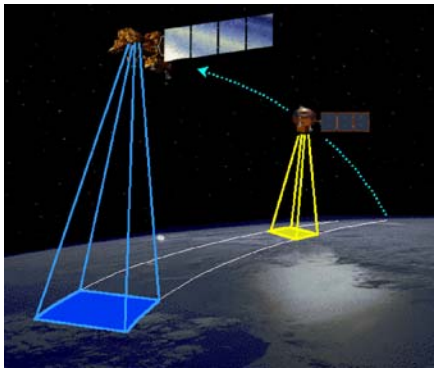
TEACHING MACHINES TO THINK FUZZY

Are you a fuzzy thinker? If you have a human brain, you are. Now scientists are trying to teach computers to think like fuzzy-thinking people, instead of like purely logical machines. To a simple machine, a statement is true or false. A light is on or off. The answer is yes or no. But to a human, a statement may be partly true. A light can be off, dim, bright, or anything in between. And the answer to a question can be "sort of." Computer scientists call this kind of information processing "fuzzy logic."

Fuzzy logic programs for computers make them more human. Computers can then think through messy situations and make smart decisions. It makes computers able to control things the way people do. Fuzzy logic has been used to control subway trains, elevators, washing machines, microwave ovens, and cars. Pretty much all the human has to do is push one button and the computer figures out what to do, how fast, how much, and for how long.

Another really important use for fuzzy logic is in robots. Some robots, especially those that work in space, must navigate and solve problems on their own. For example, two or more spacecraft can work together by flying in a very precise formation. Spacecraft formations can be used as large telescopes to look for planets around other stars. Or they can take pictures of the same scene on Earth's surface looking at it from different angles through the atmosphere. Or they can take 3-D pictures of the clouds. Such spacecraft can use fuzzy logic to keep their positions in the formation.

Earth Observing-1 is a NASA New Millennium Program spacecraft developed at Goddard Space Flight Center in Maryland. It has been testing an advanced new formation flying system that uses fuzzy logic. It has been flying in formation with another satellite, called Landsat-7, for over three years now. It has proven that this new technology works well and can be used on many future space missions.



The Earth Observing-1 satellite flies in formation behind the Landsat-7 satellite, taking pictures of the same ground almost exactly one minute later.

YOUR ASSIGNMENT (SHOULD YOU CHOOSE TO ACCEPT IT) . . .

Here's a story about teaching a robot to think like people think—fuzzily!

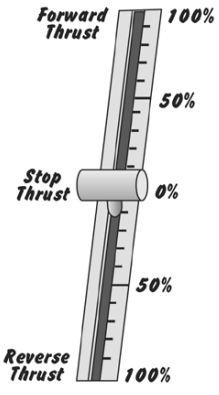
You are part of a crew flying two non-identical orbiting spacecraft to gather important science data about planet Zorgon. Spacecraft "Astroblog" and spacecraft "Beemeup" move in the same orbit, with Beemeup maintaining its station exactly 20 meters directly behind Astroblog throughout the mission.

You are the Orbit Pilot for Beemeup. Your job is to continuously keep your spacecraft exactly 20 meters behind Astroblog throughout all orbit maneuvers.

THE SITUATION:

Your job won't be easy. Irregularities in the upper atmosphere of Zorgon and differences between the two crafts in mass and in surface friction cause the speed of each spacecraft to change in different ways as they orbit the planet. In addition to that complication, Astroblog may occasionally change speed intentionally to accomplish the mission objectives.

Your spacecraft, Beemeup, has a range finder. It can detect the distance to the spacecraft it is supposed to be following. It displays the difference from any pre-set, target distance as an "error." When set to 20 meters, for example, if the target is 20 meters away, the range finder displays "0" error. If the target comes closer than 20 meters, it displays the error (actual distance minus 20 meters) in meters "too close," or m_{tc}. For example, if the target is only 18 meters away, the range finder will show the error as 2_{tc}. If the distance becomes greater than 20 meters, the error is shown in meters "too far," or m_{tf}.



The workstation for Beemeup's orbit pilot has a slider bar control for the forward and reverse thrusters on the spacecraft. The control handle slides along a scale with three main positions: "Forward Thrust," "Stop Thrust," and "Reverse Thrust." Also, continuous scales are marked off from Forward to Stop (100%F to 0%) and from Stop to Reverse (0% to 100%R).

Your on-the-job-training manual for station-keeping in orbital space shows two methods:

Method 1: BASIC navigation

The approach is simple and obvious; it's easily done with only a few "rules."

1. *If range error is* ***0****, keep the thruster bar at* ***Stop Thrust****.*
2. *If range error is a number*$_{tc}$ *(you are too close), move the thruster bar to* ***Reverse Thrust****. Then, when range error returns to* ***0****, move the thruster bar to* ***Stop Thrust****.*
3. *If range error is a number*$_{tf}$ *(you are too far behind), move the thruster bar to* ***Forward Thrust****. Then, when range error returns to* ***0****, move the thruster bar to* ***Stop Thrust****.*

While this system is both simple and effective, it probably isn't the one you'll want to use; it has a downside.

As the forward thruster reduces a "too far" range error to zero, for example, and Stop Thrust is selected, the spacecraft's corrective motion continues. This is called "overshoot." A "too close" error now appears and reverse thrust must be selected. This sequence will continue endlessly, requiring the pilot's continuous action and consuming excessive quantities of fuel.

A more comfortable way to navigate requires just a few more rules.

Method 2: REFINED navigation

1. *If range error is* ***0****, set the thruster bar to* ***Stop Thrust****.*
2. *If range error is a* ***larger number***$_{tc}$ *(you are* ***way*** *too close), move the thruster bar* ***a lot toward Reverse Thrust****.*
3. *If range error is a* ***smaller number***$_{tc}$ *(you are a* ***little*** *too close), move the thruster bar a* ***little toward Reverse Thrust****.*
4. *If range error is a* ***larger number***$_{tf}$ *(you are* ***way*** *too far away), move the thruster bar* ***a lot toward Forward Thrust****.*
5. *If range error is a* ***smaller number***$_{tf}$ *(you are a* ***little*** *too far away), move the thruster bar* ***a little toward Forward Thrust****.*

With a little practice you get a feeling for just how much thruster to use to minimize overshoot, and you've become a qualified station keeping pilot! With the new rules, less frequent action is required for station-keeping, but your continuous, alert attention is still needed to note changes in range that can happen at any time.

But you're getting tired. And hungry. And bored! If you hope to sleep, to eat, and to experience the space wonders going on, it's clear that you'll require some assistance.

HELP IS ON THE WAY . . . MAYBE

Every member in the small crew has a full assignment. The Captain, however, is able to spare a small, old fashioned, but user-friendly robot to serve as your Assistant Orbit Pilot. You will have to completely train the robot in piloting, though. The Captain calls it Mr. Bot.

Training should be easy. Mr. Bot has jacks that can connect into both range finder and thrust control circuits. It's just a matter of downloading your list of navigation rules to Mr. Bot's memory and determining whether he understands the job.

After connecting to the circuits, Mr. Bot processes your download, then produces a lengthy printout for you:

It seems that Mr. Bot doesn't appreciate your "fuzzy" way of thinking

```
ATTN  PILOT:
RULE  #1  IS  UNDERSTOOD
RULE  #2  CANNOT  PROCESS—YOUR  PARAMETERS
  ARE  FUZZY!
'LARGER'  IS  UNDEFINED!
'A  LOT'  IS  UNDEFINED!
RULE  #3  CANNOT  PROCESS—YOUR  PARAMETERS
  ARE  FUZZY!
'SMALLER'  IS  UNDEFINED!
'A  LITTLE'  IS  UNDEFINED!
RULE  #4  CANNOT  PROCESS—YOUR  PARAMETERS
  ARE  FUZZY!
'LARGER'  IS  UNDEFINED!
'A  LOT'  IS  UNDEFINED!
RULE  #5  CANNOT  PROCESS—YOUR  PARAMETERS
  ARE  FUZZY!
'SMALLER'  IS  UNDEFINED!
'A  LITTLE'  IS  UNDEFINED!
```

One option could be to download your Method 1 BASIC navigation rule set to Bot's memory. He'd probably love it—no fuzzy words there. But with those rules, Bot would give the crew a jerky ride that would exhaust the spacecraft's thruster fuel before the mission was over.

Another option is to find a way to get Mr. Bot to think "fuzzy." That is, we need to teach him to take these fuzzy input parameters and do some kind of calculations on them that tell him exactly how to control the thrusters.

Maybe he raises a legitimate question; if the range measurement number is supposed to be 20, exactly what *is* a "smaller" range error number? Is it 10?…or is it 5?…or 3?…or does it much matter to us? Whatever we decide, if it turns out too big, we'll try a smaller number. Same for "a little" thrust. We might decide that means 60%, but change to 20% when we get a "feeling" for it.

Mr. Bot of course doesn't have the freedom to make up rules and change them later. But maybe we can do that for him. Let's try by modifying Rules #2 and #3. Let's just pick some numbers and say:

2. *When range error is more than 5_{tc} (too close), move the thruster bar to 100% Reverse Thrust.*
3. *When range error is less than or equal to (*< or *=) 5_{tc}, set the thruster bar percentage in proportion to the range error, based on 100% Reverse Thrust for a range error of 5_{tc}.*

This means that for a range error of 2_{tc}, for example, Bot would set Reverse Thrust to 40%, and at 0 meters to 0% or Stop Thrust. It's logical enough for Mr. Bot to understand, and it still follows the *larger-smaller-a little-a lot* of our "fuzzy" rules (more or less).

So now we rewrite rules #4 and #5 the same way and download to Mr. Bot for his reaction.

```
ATTN  PILOT:
ALL  RULES  UNDERSTOOD
```

Great. He likes it! And it works! But only to a certain point.

Making Little Men Green

Although Mr. Bot keeps Beemeup well locked onto its station 20 meters behind Astroblog, the crew is getting seasick from the strong, lurching speed changes Mr. Bot is making.

Perhaps we can slow Mr. Bot's responses if we give him more maneuvering room. Let's pick new range numbers. Let's change 5 meters to 10 meters (or to whatever). Then Rules #2 and #3 become:

2. *When range error is more than 10_{tc} (too close), move the thruster bar to 100% (Reverse Thrust).*
3. *When range error is less than or equal to (*< or *=) 10_{tc}, set the thruster bar percentage in proportion to the range, based on 100% (reverse) thrust for a range error of 10_{tc}.*

So now we rewrite rules #4 and #5 the same way and download to Mr. Bot for his reaction.

This download works. Everyone is happy. Mr. Bot's adjustments are smooth. Life for an Orbit Pilot is tranquil!

Bot Not for Long!

Things change! Astroblog's mission now requires frequent adjustments in speed. This has caused Beemeup's station adjustments to become erratic—new errors now occur before previous ones are corrected. Bot's responses sometimes make the errors worse.

When moving toward or away from Astroblog, speed of that motion has been controlled primarily by Beemeup's thrusters. Now, changes are also caused by Astroblog's movements. This complication isn't covered in our original navigation rules. Additional rules are needed—and fast!

If we are moving too fast toward Astroblog, it seems obvious that we should use some amount of reverse thrust to slow us down; and, conversely, some amount of forward thrust is needed when speeding too quickly away. They really do seem fuzzy, but these are our new speed adjustment rules. Mr. Bot could deal with them—if we could somehow say it with numbers. Mr. Bot needs numbers—even fuzzy ones! If we could only find a way to measure the speed of our motion toward and away from Astroblog.

Beemeup's technician has uncovered a simple toy radar speedometer in a recreation gear storage locker. It was probably used by earlier crews to measure speeds of things like thrown balls, other crew members, or whatever. The technician has cleverly targeted the radar's beam on Astroblog, with an output connected directly to Mr. Bot. The radar's digital display will now show us how fast we are approaching or moving away from Astroblog. The radar has a low speed calibration that covers from 30 meters per minute closing in (mpm_c) to zero to 30 meters per minute falling back (mpm_f). The speeds we are observing all seem to fall within that range.

Adding Speed to the Fuzzy Equation

We now have rules for dealing with speed change and a way to measure the changing speeds. While both the speed and the position rules are followed at the same time,

it's obvious to us that the speed adjustment rules will be more important, or less important, depending on what kind of range error we see:

- If we're closing in at high speed and we are already 10 meters too close, a lot of reverse thrust makes sense.
- On the other hand, if we're 10 meters too far away and we're closing in at high speed, a lot of reverse thrust might slow correction or make the error worse.

We make these decisions easily. It's just plain common sense. But how do we teach this kind of fuzz to Mr. Bot?

Let's consider a typical situation that Mr. Bot will deal with, such as when we're too close to Astroblog and we're moving in too fast.

In this case, our range rules say:

1. When range error is 10_{tc} or less, set the thruster bar percentage in proportion to the range error. Or, to rephrase that so Mr. Bot will understand it:
   - *For range error < or = 10_{tc}, thrust = (100%R / 10) x range error.*

What thrust correction should we make if we're moving toward our station a lot faster or a little faster than planned? Since our speedometer measures up to 30 mpm, let's choose to use maximum thrust for speeds over 20 mpm_c (closing in) and make a rule like this:

2. When speed is less than or equal to 20 mpm_c, set the thruster bar to reverse thrust in proportion to the speed. Or, in Mr. Bot's terms:
   - *For speed < or = 20 mpm_c, thrust = (100%R / 20) x speed.*

How should that rule be written in case we are moving away from our station too fast?

Mr. Bot needs to combine the range and the speed rules and perform both at once. A very simple way is just to add both results, and, ignoring the portion greater than 100%, their sum equals the total thrust to be applied.*

Here's an example of what happens with range error of 3_{tc} and speed of 8 mpm_c:

For RANGE ERRORS < or = 10 m_{tc} Thrust = (100%R / 10 m) x Range Error therefore Thrust = (100%R / 10 m) x 3 m_{tc} = **30%R**	**For SPEEDS** < or = 20 mpm_c Thrust = (100%R / 20 mpm_c) x Speed therefore Thrust = (100%R /20 mpm_c) x 8 mpm_c = **40%R**
So adding . . . Range Correction Thrust = **30%R** Speed Correction Thrust = **40%R** Total Correction Thrust = **70%R**	

That seems reasonable! We would need a total of 70% reverse thrust: 30% reverse thrust just to move away from Astroblog and an additional 40% reverse thrust to slow our rapid motion toward Astroblog. If it doesn't work smoothly, the numbers we've selected can be changed until it does. That's the beauty of fuzz!

What would the total correction thrust be for this situation?

Range error 5_{tc}, speed 15 mpm_c? _____ %

(Answer is given at end of article.)

It Gets Fuzzier

Mr. Bot can understand this stuff so far and execute it quite well. Some of the other situations he faces may be a lot more challenging however. This table shows all nine potential situations where our rules either (1) say to do the same action (agree), (2) one rule requires nothing and the other rule says do something (no conflict), or (3) one rule says to do one thing and the other rule says to do just the opposite (conflict).

DIRECTION OF MOTION/ Rule for Action:	**RANGE ERROR / Rule for action:**		
	TOO CLOSE/ Reverse Thrust	JUST RIGHT/ No Action	TOO FAR/ Forward Thrust
CLOSING IN/ Reverse Thrust	AGREE	NO CONFLICT	**CONFLICT!**
NONE/ No Action	NO CONFLICT	NO ACTION	NO CONFLICT
FALLING BACK/ Forward Thrust	**CONFLICT!**	NO CONFLICT	AGREE

Let's see what happens when instructions for RANGE TOO CLOSE and MOTION FALLING BACK are in conflict. Let's use Range Error of 3_{tc} and Speed of 8 mpm_f:

Note: In the case where you are combining forward and reverse thrust, they will tend to cancel each other out. What you will have left is the difference between the two numbers, with the thrust in the direction of the larger number.

For RANGE ERRORS < or = 10 m_{tc} Thrust = (100%R / 10 m) x Range Error therefore Thrust = (100%R / 10 m) x 3 m_{tc} = **30%R**	**For SPEEDS** < or = 20 mpm_f Thrust = (100%F / 20 mpm) x Speed therefore Thrust = (100%F / 20 mpm) x 8 mpm = **40%F**
So adding . . . Range Correction Thrust = **30%R** Speed Correction Thrust = **40%F** Total Correction Thrust = **10%F**	

So Mr. Bot would now select 10 % forward thrust: 30% reverse thrust to move away and 40% forward thrust to slow the excess speed of moving away. That feels just about right.

Think Like Mr. Bot

Now that you've taught Mr. Bot to think fuzzy like you, it's only fair that you try out your own rules as Mr. Bot will have to do.

Again, here are the rules for handling the fuzzy input parameters:

For SPEEDS . . .	For RANGE ERRORS . . .
> 20 mpm_c (closing in) Thrust = 100%R	> 10 m_{tc} (too close) Thrust = 100%R
< or = 20 mpm_c Thrust = (100%R / 20 mpm) x Speed	< or = 10 m_{tc} Thrust = (100%R / 10 m) x Range Error
> 20 mpm_f (falling back) Thrust = 100%F	> 10 m_{tf} (too far) Thrust = 100%F
< or = 20 mpm_f Thrust = (100%F / 20 mpm) x Speed	< or = 10 m_{tf} Thrust = (100%F / 10 m) x Range Error

Fill in the **thrust values** for the following situations. Note that the total correction thrust percentage may add up to more than 100%R or 100%F. If so, just give it all you've got, which can be no more than 100%!

	RANGE ERROR		
SPEED	8 m_{tc} (too close)	0 (just right)	10 m_{tf} (too far)
10 mpm_c (closing in)	Thrust =	Thrust =	Thrust =
0 (none)	Thrust =	Thrust =	Thrust =
8 mpm_f (falling back)	Thrust =	Thrust =	Thrust =

See bottom of article for answers.

After you review how the rules work in all possible situations, you download the rules and Mr. Bot responds:

```
ATTN  PILOT:
THESE  RULES  ARE  REALLY  CRISP  AND  CLEAR!
THANK  YOU!  ☺
```

…and it really works!

Go to The Space Place web site at http://spaceplace.nasa.gov to learn about some of the other advanced technologies on Earth Observing-1.

This article was written by Diane Fisher, writer and designer of The Space Place website at ***spaceplace.nasa.gov****. Alex Novati drew the illustrations. Thanks to Gene Schugart, Space Place advisor, for activity concept and helpful advice. The article was provided through the courtesy of the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, under a contract with the National Aeronautics and Space Administration.*

Answers:

Range error 5_{tc}, speed 15 mpm_c? *125%R or* ***100%R***

	RANGE ERROR		
SPEED	8 m_{tc} (too close)	0 (just right)	10 m_{tf} (too far)
10 mpm_c (closing in)	Thrust = 100%R	Thrust = 50%R	Thrust = 50%F
0 (none)	Thrust = 80%R	Thrust = 0	Thrust = 100%F
8 mpm_f (falling back)	Thrust = 40%R	Thrust = 40%F	Thrust = 100%F

* *While this simple method can do our job, more sophisticated statistical tools are used to produce very smooth, precise maneuvering. These include math techniques such as root-mean-square averaging and others.*