SpuPilot Reference

The SpuPilot offers a way to communicate with Spumone mathematically. We can use it to provide mathematical rules for the spuCraft to turn on thrusters, activate constraints, and operate actuators. Mathematical precision provided by these rules is required to solve most challenges in Spumone.

Entering Mathematical Equations

In order to express your strategies mathematically, you are going to have to write your equations in a way that Spumone's equation parser can understand them. The rules for writing equations for Spumone are mostly the same as writing mathematical equations in most programming languages. Below, I list important points to keep in mind when writing equations, things that students often have problems with.

- ***Variable Names***. When you write equations, you will often use variables that represent positions, velocities, forces, or other physical quantities. SpuPilot, will list the names of variables that you have available for your equations. You may not use variables that are not available. Furthermore, the variable names are case sensitive. That means the variable `g` is not the same as `G`. There are some instances where Spumone is forgiving, but if you do not type the variable correctly, you should not expect it to work.
- ***Variables are read only***. You can only read the value of a variable. You cannot write to it. You should think of the variables as measurements. For example, you might have a variable that represents the velocity of the spuCraft. In this case, you can think of the variable as the reading on a speedometer. You would not use the speedometer on your car to set the speed of your car. Instead you would use the gas pedal and brake. Likewise you do not set the value of a variable; you read from it.
- ***Multiplication is ****. We humans are accustomed to writing equations like this: F = m a. We interpret the right hand side as multiplication of variables m and a. However, on the computer, you would write the multiplication as: `m*a` . Symbols for addition subtraction, and division are the usual ones: `+`, `-`, `/`, respectively.
- ***Exponents***. If you want to write v^2, then you can simply type `v^2`.
- ***"Unary minus" works***. It's perfectly legal to write `-m*g`, for example.
- ***Precedence***. Mathematical operations occur in a specific order. Here is the order of precedence: (1.) Parentheses; (2.) exponentiation; (3.) unary minus; (4.) multiplication and division; and (5.) addition and subtraction.
  Here are some examples:

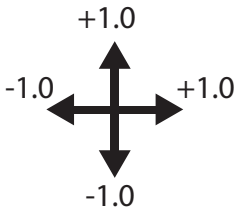
`a*b+c`	is equivalent to	$ab + c$.
`a*(b+c)`	is equivalent to	$a(b + c)$.
`a+b^2`	is equivalent to	$a + b^2$.
`(a+b)^3`	is equivalent to	$(a + b)^3$.
`(a+b)^-3`	does not make sense.	
`(a+b)^(-3)`	is equivalent to	$(a + b)^{-3}$.

`5.5/c+d`	is equivalent to	$\frac{5.5}{c} + d$.
`5.5/(c+d)`	is equivalent to	$\frac{5.5}{(c+d)}$.
`2.7/(a*(b+c))`	is equivalent to	$\frac{2.7}{a(b+c)}$.
`2.7/a/(b+c)`	is equivalent to	$\frac{2.7}{a(b+c)}$ (same as above).

- **Built-In Functions.** The equation parser in Spumone has a bunch of built-in functions. A list:

`abs(A)`	Absolute value of A.
`acos(A)`	Arc-cosine of A.
`acosh(A)`	Arc-hyperbolic cosine of A.
`asin(A)`	Arc-sine of A.
`asinh(A)`	Arc-hyperbolic sine of A.
`atan(A)`	Arc-tangent of A. Returns angle in radians between $-\pi/2$ to $\pi/2$.
`atan2(A,B)`	Principal arc-tangent of A/B using the signs of the two arguments to determine quadrant of the result. Returns angle in radians between $-\pi$ and π.
`atanh(A)`	Arc-hyperbolic tangent of A.
`cbrt(A)`	Cube root of A.
`cos(A)`	Cosine of A, where A is measured in radians.
`cosh(A)`	Hyperbolic cosine of A.
`cot(A)`	Cotangent of A, where A is measured in radians.
`csc(A)`	Cosecant of A, where A is measured in radians.
`exp(A)`	Exponential of A.
`exp2(A)`	Base 2 exponential of A.
`log(A)`	Natural (base e) logarithm of A.
`log2(A)`	Base 2 logarithm of A.
`log10(A)`	Base 10 logarithm of A.
`max(A,B)`	If A>B, the result is A, else B.
`min(A,B)`	if(A<B, the result is A, else B.
`pow(A,B)`	Exponentiation (A raised to the power B).
`sec(A)`	Secant of A, where A is measured in radians.
`sin(A)`	Sine of A, where A is measured in radians.
`sinh(A)`	Hyperbolic sine of A.
`sqrt(A)`	Square root of A.
`tan(A)`	Tangent of A, where A is measured in radians.
`tanh(A)`	Hyperbolic tangent of A.

- **Signals from Gamepad Thumbsticks.** In piloting your spuCraft, you can get signals directly from the thumbsticks on your gamepad. The signals are stored in variables `stickHL, stickVL, stickHR, stickVR.`



`stickHL`	Contains signal from horizontal axis of left thumbstick. The number is between -1.0 (stick all the way to the left) and +1.0 (stick to the right). When one releases the stick, allowing the spring to return it to the center position, this function returns 0.0.
`stickVL`	Contains signal from vertical axis of left thumbstick. The number is between -1.0 (stick all the way down) and +1.0 (stick up). When one releases the stick, allowing the spring to return it to the center position, this function returns 0.0.
`stickHR`	Contains signal from horizontal axis of left thumbstick. The number is between -1.0 (stick all the way to the left) and +1.0 (stick to the right). When one releases the stick, allowing the spring to return it to the center position, this function returns 0.0.
`stickVR`	Contains signal from vertical axis of left thumbstick. The number is between -1.0 (stick all the way down) and +1.0 (stick up). When one releases the stick, allowing the spring to return it to the center position, this function returns 0.0.

- **Signals from keyboard arrows.** Also, you can get input from the arrow keys on your keyboard.

arrowUp	Variable has value 1.0 when the keyboard up arrow is pressed, 0.0 otherwise.
arrowDown	Variable has value 1.0 when the keyboard down arrow is pressed, 0.0 otherwise.
arrowLeft	Variable has value 1.0 when the keyboard left arrow is pressed, 0.0 otherwise.
arrowRight	Variable has value 1.0 when the keyboard right arrow is pressed, 0.0 otherwise.