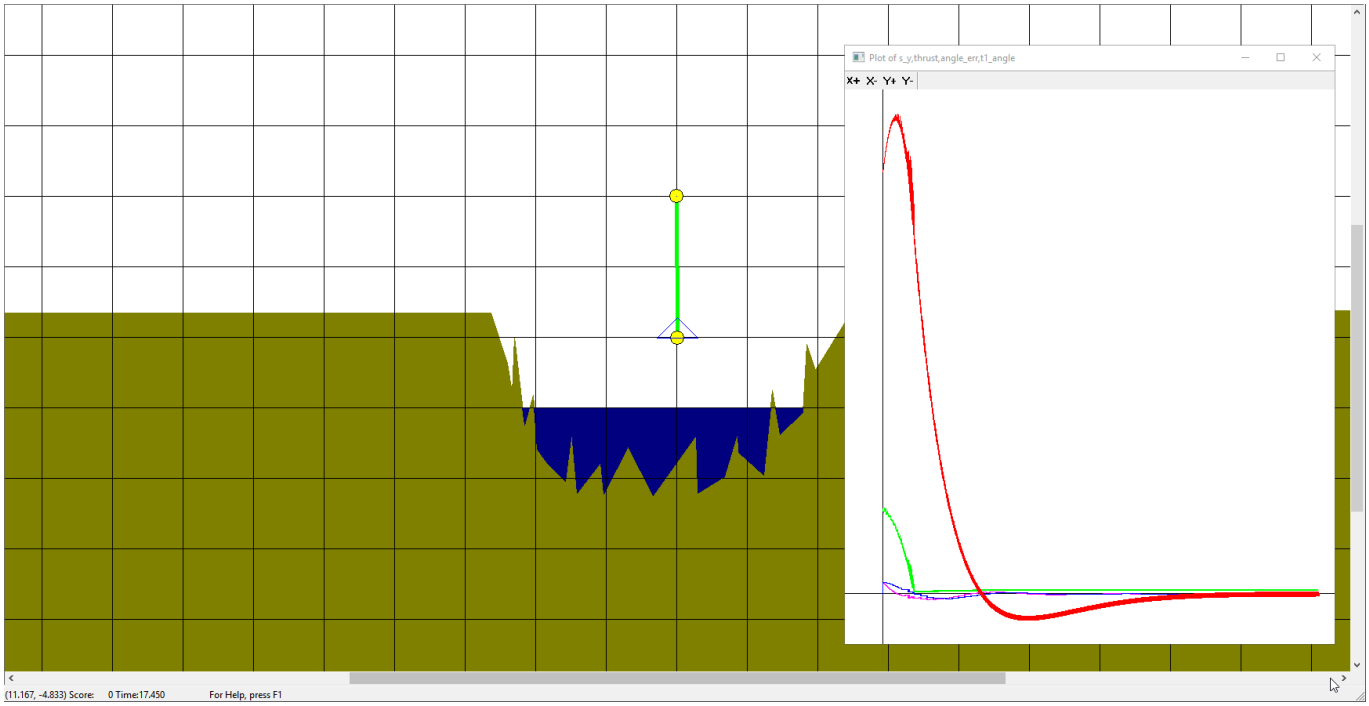
Simulation: PID controller in SimStructure



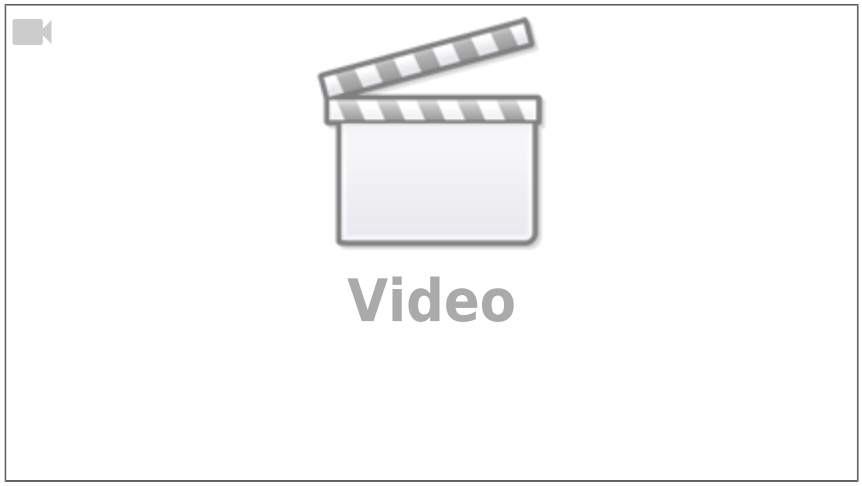
files:

- simstrucuture.zip
- ardugeek_rocket1.zip

A cool simulator for learning the principles of the PID controller: https://tools.softinery.com/PIDSIM/

SimStructure author: Terry A. Davis

Source:



Founding

The simulation is to show the use of a PID controller in a rocket which is to have compensation for changes in acceleration and angular alignment. The simulation is done in two dimensions and calculations are performed in two dimensions.

Declaration of variables

```
 double desired_y = 10, error_y, kd_y = 1, kp_y = 1, ki_y = 0.3; signal s_y;
signal thrust; signal t1_angle; integrator i_y;
```

desired_y - desired height value (initially 10).

error_y - difference between desired and current value.

kp_y, kd_y, ki_y - PID controller coefficients for the y-axis (proportional, differential, integral).

s_y, thrust, t1_angle - signals for further processing and visualisation.

i_y - internal accumulator (integral) of the controller.

PID controller for the _Y axis

```
 desired_y = 15 + 5 * t; error_y = desired_y - p1.y; s_y = error_y; i_y =
error_y; t1.thrust = t1.saturation * ( kp_y * error_y - kd_y * p1.dy + ki_y
* i_y ); thrust = t1.thrust / 100000;
```

The line

```
desired_y = 15 + 5 * t;
```

causes the set height to change linearly in time (t is the simulation time).

Error calculation: ;error_y: Difference between the set height and the current height (p1.y).

Error integration: ;i_y: Simple integration by error assignment (can be extended to include summation in a loop).

Regulator output: ;t1.thrust = factor_saturation × (P - error - D - speed + I - integral).

Scale the thrust signal (thrust) by dividing by 100 000 to get the appropriate units/range.

PID controller for angle

```
 double a, kp_a = 1.0, kd_a = 0.01, ki_a = 0.001, a_err; []a = pi - pi/180 *
```

```
b1.heading + 10 * ( key2("1") - key2("2") ); signal angle_err; angle_err =
50 * a; integrator i_a; i_a = a / 10;

t1.angle = kp_a * a
- kd_a * b1.spin
+ ki_a * i_a;
t1_angle = 50 * t1.angle;
```

a - angle error: difference between the reference angle (pi) and the current angle (b1.heading in radians), plus correction from the keyboard (buttons „1” and „2”).

kp_a, kd_a, ki_a - PID coefficients for the angle.

angle_err - scaled angle error for visualisation.

i_a - integral of angle error (here just a/10, not summing).

Calculation of the angle controller output: ;t1.angle = P - a - D - rotation + I - i_a

t1_angle - additional scaling of the angle signal.

Visualisation and diagrams

```
 @{ line(p1.x-1000, desired_y, p1.x+1000, desired_y, blue, #3); text(#10,
#50, "Thrust: %12.6f N", t1.thrust); text(#10, #10, "Heading:%12.6f degree",
b1.heading); text(#10, #30, "Angle: %12.6f rad", a); text(#10, #70,
"Integrator: %12.6f", i_y); plot s_y, thrust, angle_err, t1_angle; }
```

The function

```
line()
```

draws a horizontal line at the height set with respect to the p1.x position.

Several calls

```
text()
```

displays on the screen:

thrust value (t1.thrust),

the current angle (b1.heading),

angle error (a),

the value of the y-axis error integral (i_y).

```
plot()
```

generates graphs of consecutive signals:

s_y - height error signal,

thrust - thrust force,

angle_err - angle error (scale),

t1_angle - output signal of angle regulator.

Full control code

Code:

```
double desired_y=10,error_y,kd_y=1,kp_y=1,ki_y=0.3;
signal s_y;
signal thrust;
signal t1_angle;
integrator i_y;

desired_y = 15+5*t;


error_y = desired_y-p1.y;
s_y = error_y;
i_y = error_y;
t1.thrust = t1.saturation*(kp_y*error_y-kd_y*p1.dy+ki_y*i_y);
thrust = t1.thrust/100000;
double a,kp_a=1.0,kd_a=0.01,ki_a=0.001,a_err;
[]a = pi-pi/180*b1.heading+10*(key2("1")-key2("2"));
signal angle_err;
angle_err = 50*a;
integrator i_a;
i_a = a/10;

t1.angle = kp_a*a-kd_a*b1.spin+ki_a*i_a;
t1_angle = 50*t1.angle;
@{
    line(p1.x-1000,desired_y,p1.x+1000,desired_y,blue,#3);
    text(#10,#50,"Thrust:  %12.6f N",t1.thrust);
    text(#10,#10,"Heading:%12.6f degree",b1.heading);
    text(#10,#30,"Angle:  %12.6f rad",a);
    text(#10,#70,"Integrator: %12.6f",i_y);
    plot s_y, thrust, angle_err, t1_angle;
}
```