Sumulacja Rakiety w programie SimStructure

pliki:

- simstrucuture.zip
- ardugeek_rocket1.zip

Opis działania kodu sterującego

Ten dokument opisuje działanie fragmentu kodu sterującego dla pojazdu (np. drona lub rakiety) z wykorzystaniem regulatorów PID w dwóch osiach: pionowej (Y) oraz kąta nachylenia (Angle).

Parametry PID dla osi Y

Kod definiuje następujące zmienne sterujące ruchem w osi pionowej:

- `desired_y`

  – zadana pozycja w osi Y. Początkowo ustawiona na 10, następnie dynamicznie zmieniana:

<pre> desired_y = 15 + 5 * t;

```
  </pre>
* <code>error_y</code> – błąd położenia:
  <pre>
```

error_y = desired_y - p1.y;

```
  </pre>
* Współczynniki regulatora PID:
  **Proporcjonalny**: <code>kp_y = 1</code>
  **Różniczkujący**: <code>kd_y = 1</code>
  **Całkujący**: <code>ki_y = 0.3</code>
* <code>i_y</code> – wartość całki błędu.
```

Sterowanie siłą ciągu (thrust)

Siła ciągu liczona jest jako: <pre> t1.thrust = t1.saturation * (kp_y * error_y

1. kd_y * p1.dy

+ ki_y * i_y); thrust = t1.thrust / 100000; </pre>

Regulacja kąta nachylenia

Sterowanie kątem nachylenia realizowane jest osobnym PID-em:

- a

– żądany kąt:

<pre> []a = pi

1. (pi/180) * b1.heading

+ 10 * (key2(„1”) - key2(„2”));

```
  </pre>
* Współczynniki PID:
  **Proporcjonalny**: <code>kp_a = 1.0</code>
  **Różniczkujący**: <code>kd_a = 0.01</code>
  **Całkujący**: <code>ki_a = 0.001</code>
* <code>angle_err</code> – przeskalowany błąd kąta:
  <pre>
```

angle_err = 50 * a;

```
  </pre>
* <code>i_a</code> – integrator kąta:
  <pre>
```

i_a = a / 10;

```
  </pre>
* Obliczenie sygnału sterującego:
  <pre>
```

t1.angle = kp_a * a

1. kd_a * b1.spin

+ ki_a * i_a; t1_angle = 50 * t1.angle;

```
  </pre>
```

Część wizualizacyjna

W bloku graficznym rysowane są:

- Pozioma linia w pozycji

```
desired_y
```

<pre> line(p1.x-1000, desired_y, p1.x+1000, desired_y, blue, #3)

```
  </pre>
* Dynamiczne teksty:
  <pre>
```

text(#10,#50,„Thrust: %12.6f N”, t1.thrust) text(#10,#10,„Heading: %12.6f°”, b1.heading) text(#10,#30,„Angle: %12.6f rad”, a) text(#10,#70,„Integrator: %12.6f”, i_y)

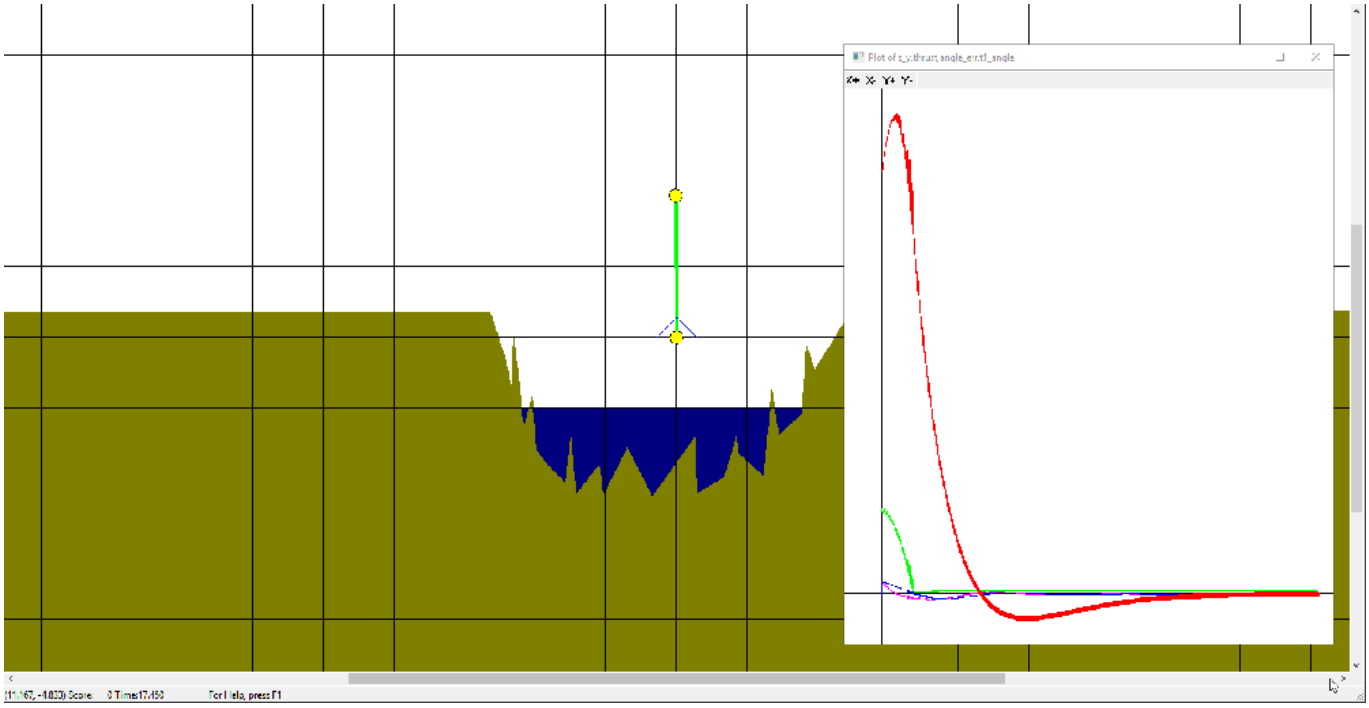
```
  </pre>
* Wykresy zmiennych:
  <pre>
```

plot s_y, thrust, angle_err, t1_angle

```
  </pre>
```

Podsumowanie

- Sterowanie położeniem w osi Y realizowane przez PID (proporcjonalny, różniczkujący, całkujący).
- Sterowanie kątem nachylenia za pomocą oddzielnego PID.
- Wizualizacja parametrów sterowania i stanu pojazdu.



Kod:

```
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;
}
```