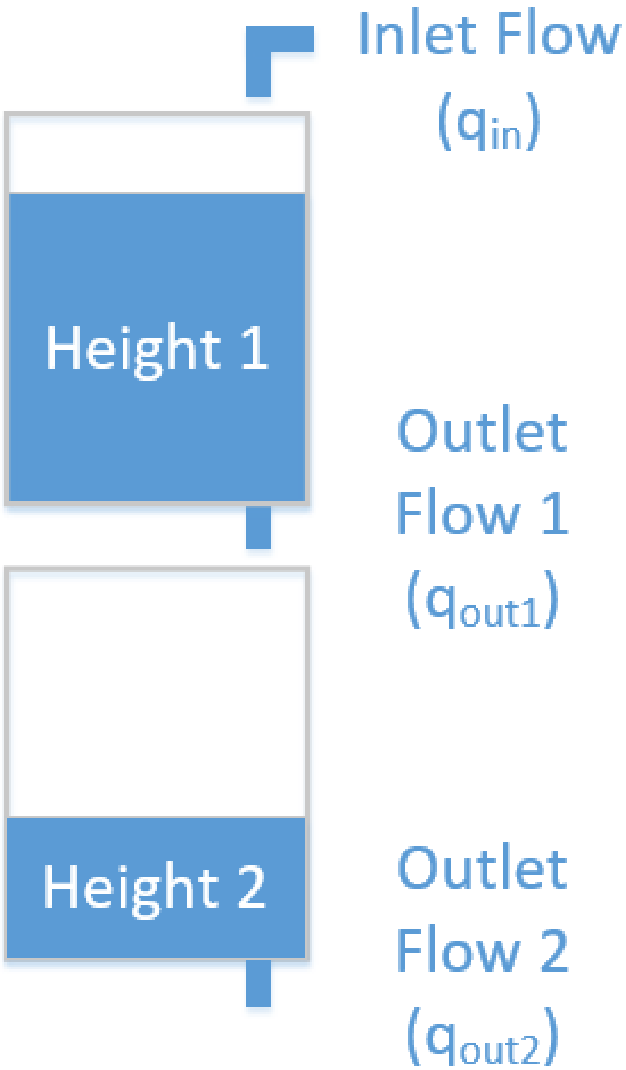
Fluid Process

Simple

Simulation Example

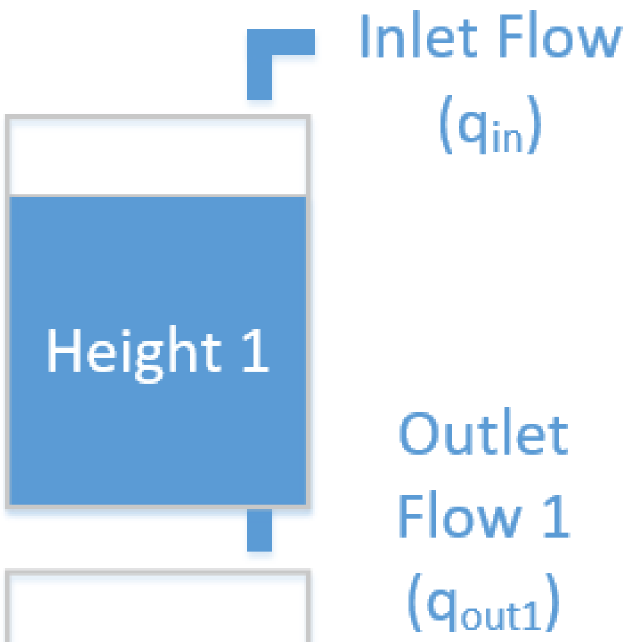
Fluid Process Simulation



Chemical plants often have processes running in reaction tanks. How do you predict if your process will function properly?

Simulate it!

Fluid Height

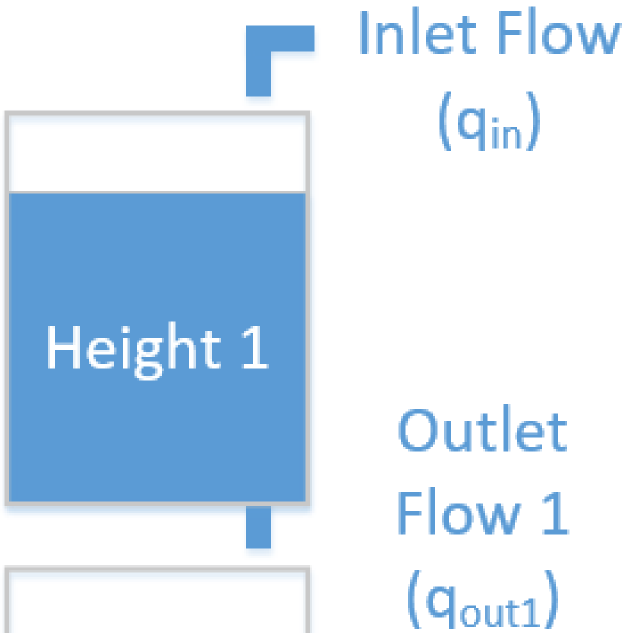


The rate of fluid height growth is proportional to the relative flows divided by the surface area of the tank.

$$A_c \frac{dh}{dt} = q_{in} - q_{out}$$

Where: A_c = tank surface area

Flow Rate



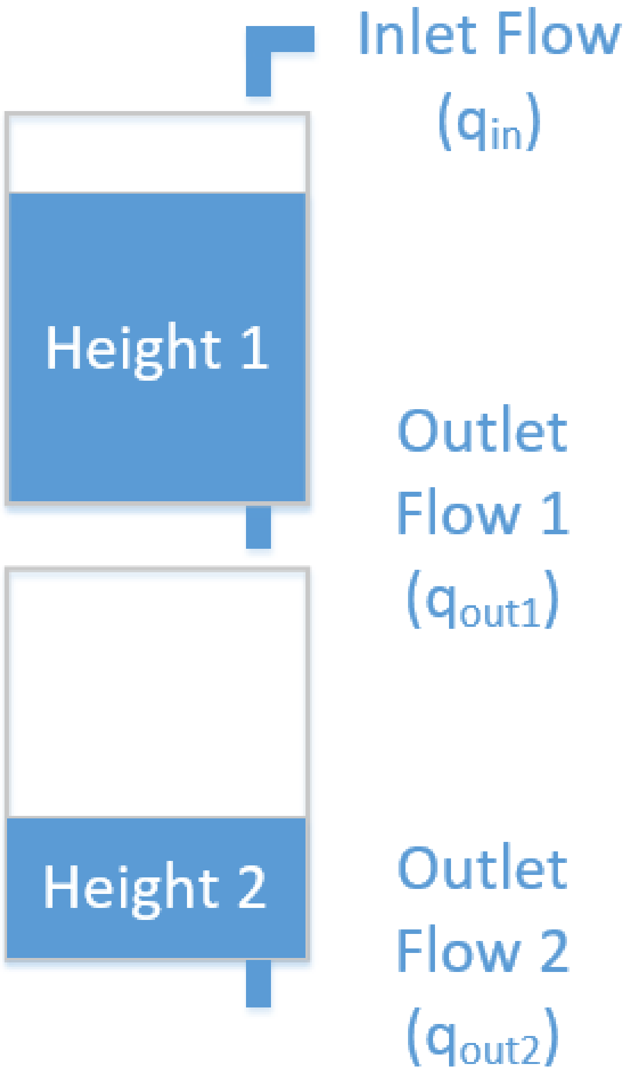
The outlet flow rate depends on the restriction size, fluid viscosity and the height of the fluid column.

Bernoulli's equation for incompressible fluids:

$$q_{out} = c\ \sqrt{h}$$

Where: h – height of the fluid column
c - flow constant based on the fluid type and restriction size.

Fluid Process Problem



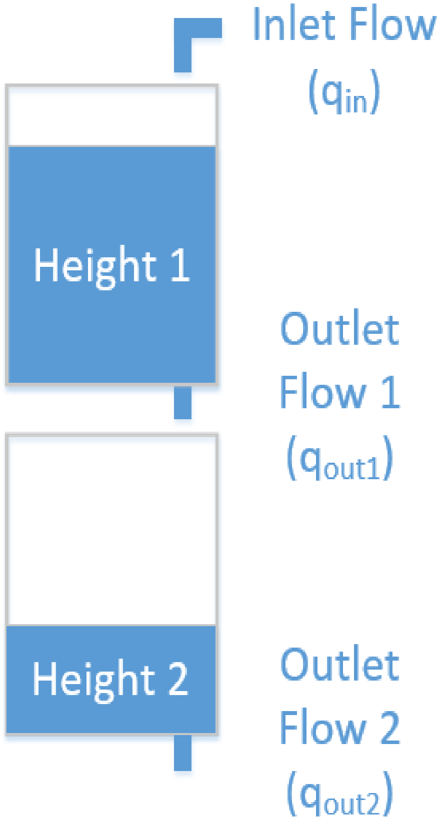
$A_c = 2\ m^2$ - tank surface area (1 & 2)
h = 1 m - tank height (1 & 2)
$q_{in} = 0.5\ m^3/hr$ - inlet flow rate
$c_1 = 0.13$ - flow constant for outlet 1
$c_2 = 0.20$ - flow constant for outlet 2

Plot the height of tank 1 & tank 2

- From 0 to 40 minutes
- Using a 0.5 hour simulation rate

Note: tanks are open top and can overflow

Setup the differential equations



$A_c = 2\ m^2$ - tank surface area (1 & 2)
$q_{in} = 0.5\ m^3/hr$ - inlet flow rate
$c_2 = 0.20$ - flow constant for outlet 2
$h = 1\ m$ - tank height (1 & 2)
$c_1 = 0.13$ - flow constant for outlet 1

Let: x_1, x_2 – the height of the corresponding tank
$dx_1\ dx_2$ – the change in the height of the tank

Change in level of the top tank. (inflow - outflow)/area
In flow of 0.5, out flow based on 0.13*sqrt(height)
Height growth is the overall flow difference / area of the tank

$$dx_1 = (0.5 - 0.13 * \mathrm{sqrt}(x_1)) / 2.0$$

Change in level of the 2nd tank. (inflow - outflow)/area
In flow is outflow from the previous tank
Height growth is the overall flow difference / area of the tank

$$dx_2 = (0.13 * \mathrm{sqrt}(x_1) - .20 * \mathrm{sqrt}(x_2)) / 2.0$$

Note: This does not address the “overfilling” issue

Simulation equation in C

```
/*************************************************************
Differential Equations of a dual tank filler with overflow.
        double *x  - current state
                x[0] - tank 1 level
                x[1] - tank 2 level

        double *dx - derivative of states, results are returned here
                dx[0] – change in tank 1 level
                dx[1] – change in tank 2 level
*************************************************************/
void tanks(double *x, double *dx) {
   // change in level of the top tank. (inflow - outflow)/area
   // In flow of 0.5, out flow based on 0.13*sqrt(height)
   // Height growth is the overall flow difference / area of the tank
   dx[0] = (0.5 - 0.13 * sqrt(x[0])) / 2.0;

   // change in level of the 2nd tank. (inflow - outflow)/area
   // In flow is out flow from the previous tank
   // Height growth is the overall flow difference / area of the tank
   dx[1] = (0.13 * sqrt(x[0]) - .20 * sqrt(x[1])) / 2.0;
}
```

Euler –C Implementation

Euler

$$k_1 = f(t_k, x_k)$$

$$x_{k+1} = x_k + hk_1$$

```
/*-----------------------------------------------------------------------------------------------------------
  Executes ONE cycle of simulation
      double h    - simulation increment
     double *x0 - value of state variables at current time,  AND
                       the approximate solution time t0+h is returned  (x0 is overwritten)
 -------------------------------------------------------------------------*/
void eu(double h, double *x0 ){
    double *xp;                               /* temporary vector */
    xp = malloc(2 * sizeof(double));      /* tanks have 2 state variables */

    /* Compute Euler update - return in x0 */
    /* Evaluate the Differential equations at the current time      k1 = f(tk, xk)  */
    tanks(x0, xp);                           /* pass in the current state, return the change */

    /* Add the values to the current integration solution          xk+1 = xk+hk1    */
    for ( int i = 0; i < 2; i++ ){
        x0[i] += h * xp[i];
   }
} /* End eu */
```

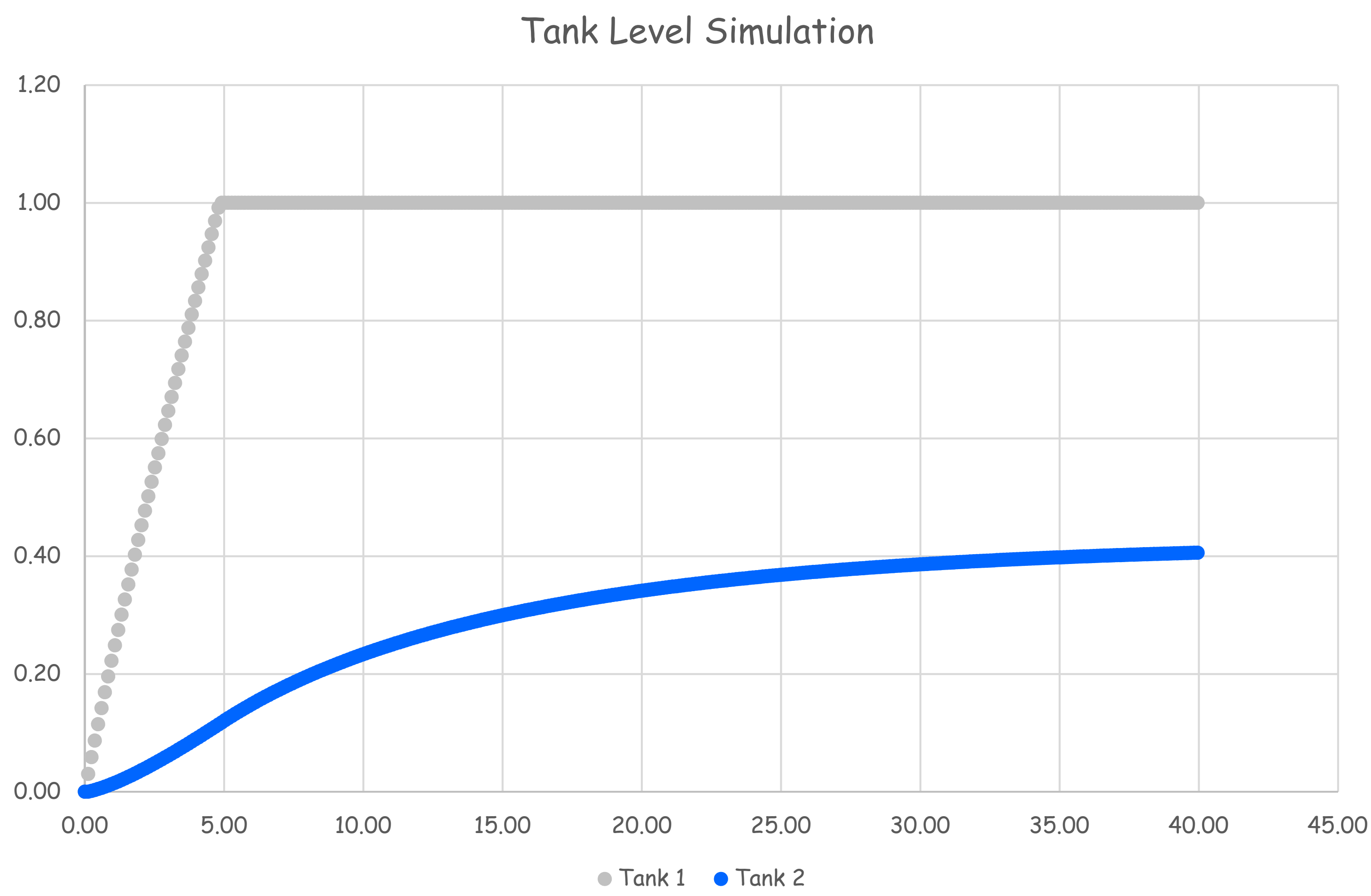
Heun's –C Implementation

Heun's (RK2)

$$k_1 = f(t_k, x_k)$$
$$k_2 = f(t_k + h, x_k + hk_1)$$
$$x_{k+1} = x_k + h(\tfrac{1}{2}k_1 + \tfrac{1}{2}k_2)$$

```
/*------------------------------------------------------------------------------------------------
   Executes ONE cycle of simulation
        double h    - simulation increment
       double *x0 -  state variable
 ------------------------------------------------------------------------------------------------
void rk2(double h, double *x0 ){
double *xtilde, *k1, *k2;                          /* temporary vectors */
xtilde = malloc(2sizeof(double));
k1 = malloc(2*sizeof(double));      k2 = malloc(2*sizeof(double));

/* Evaluate the Differential equations at the current time    k1 = f(tk, xk)  */
tanks(x0, k1);

/* Build the k2 function call parameters                                    xk+hk1 */
for (int i = 0; i < 2; i++ ) {   xtilde[i] = x0[i] + h*k1[i];      }

/* Evaluate the Differential equations at the current time    k2 = f(tk+ h, xk+hk1) */
tanks(xtilde, k2);

/* update dx                                                              xk+1 = xk+h(½k1+½k2)  */
for (int i = 0; i < 2; i++ ) {    x0[i] += h*(k1[i] + k2[i])/2.0;  }
} /* End rk2 */
```

Result



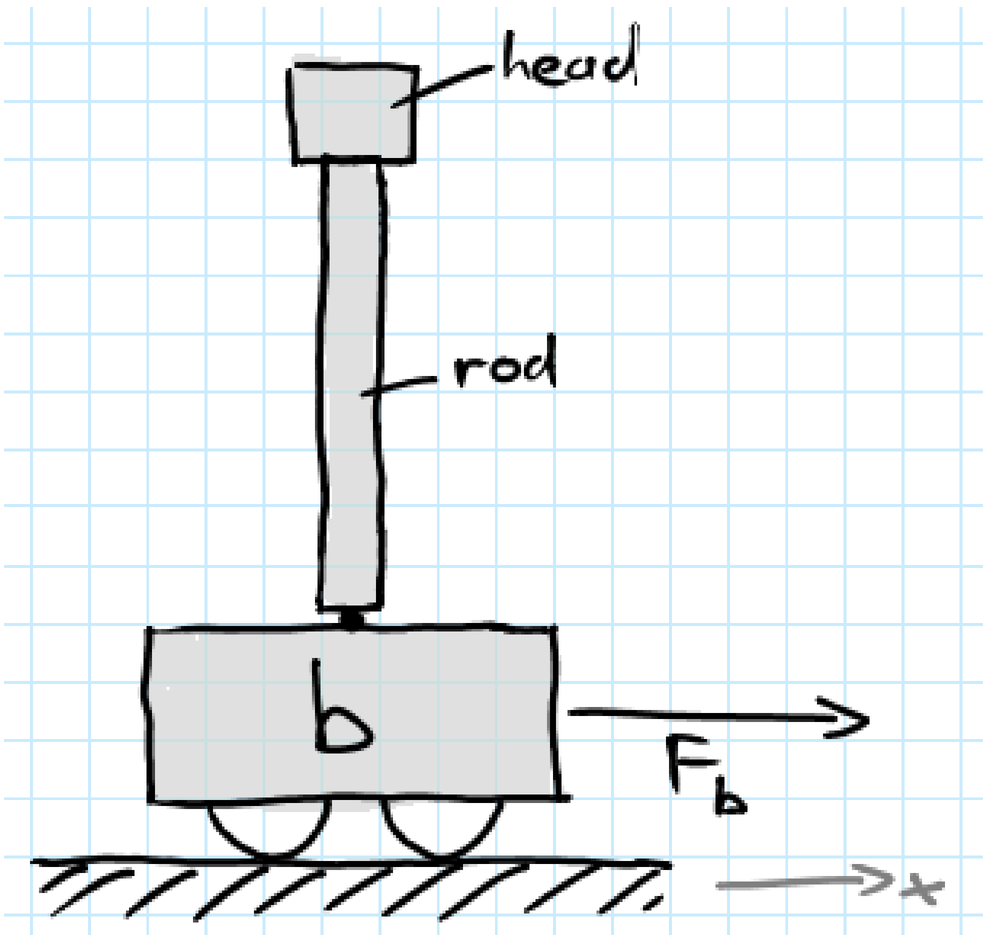
Tank Level Simulation
1.20
1.00
0.80
0.60
0.40
0.20
0.00
0.00
5.00
10.00
15.00
20.00
25.00
30.00
35.00
40.00
45.00
Tank 1
Tank 2

Inverted Pendulum

Complex

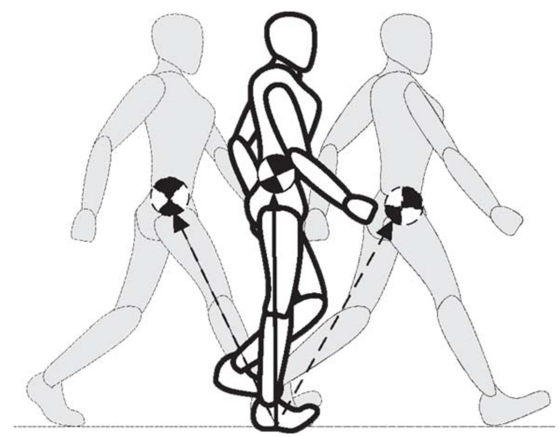
Simulation Example

Inverted Pendulums



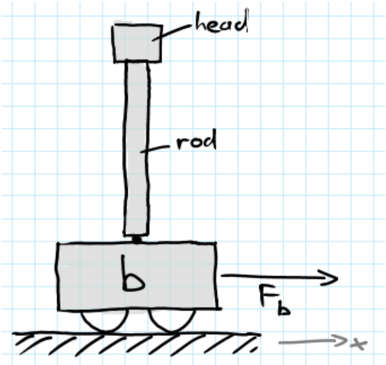
https://i.stack.imgur.com/P7BFu.png

hackedgadgets.com/wp-content/vox1.jpg



www.frontiersin.org/files/Articles/153280/frobt-02-00021-HTML/image_m/frobt-02-00021-g001.jpg

Inverted Pendulum



https://i.stack.imgur.com/P7BFu.png

- Inherently Unstable
- Requires an active control system to maintain stability

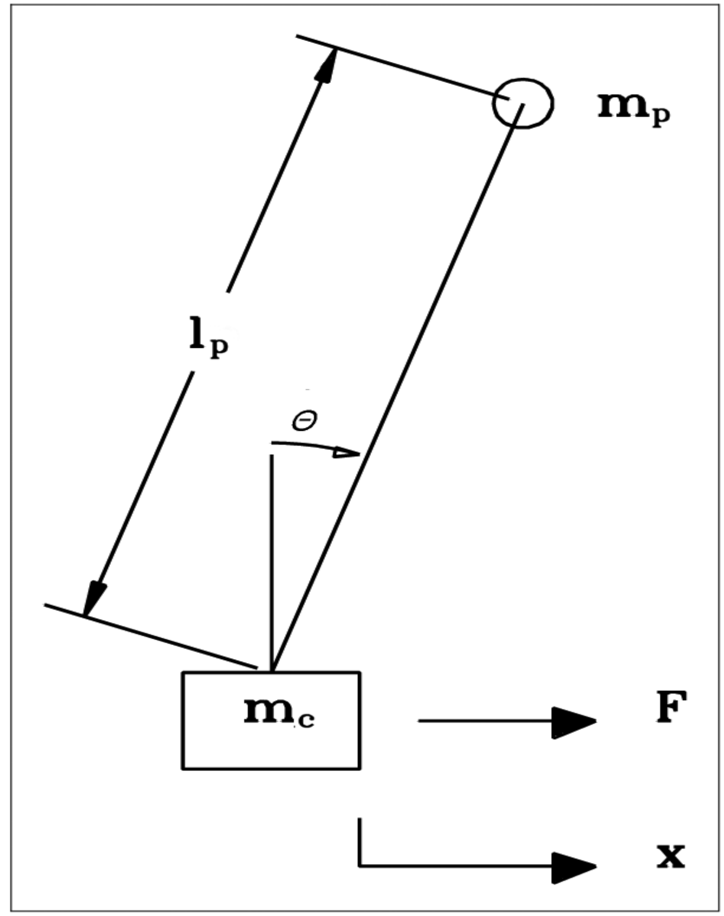
- System performance depends on:
  - Leaver arm
  - Mass
  - Acceleration of gravity
  - Control System Performance

roboticdreams.files.wordpress.com/2015/05/broom-balancing.jpg

Schematic – 1-D

System parameters:

Symbol	Description
m_p	The mass [Kg] of the top platform
l_p	The distance [m] from the pivot to the center of gravity of the platform
m_c	The mass [Kg] of the rover base
θ	Angle [rad] of the pivot from vertical
F	Force Applied [N]
x	Platform position [M]
g	Gravitational force [m/s^2]
T_d	An external torque disturbance [Nm]

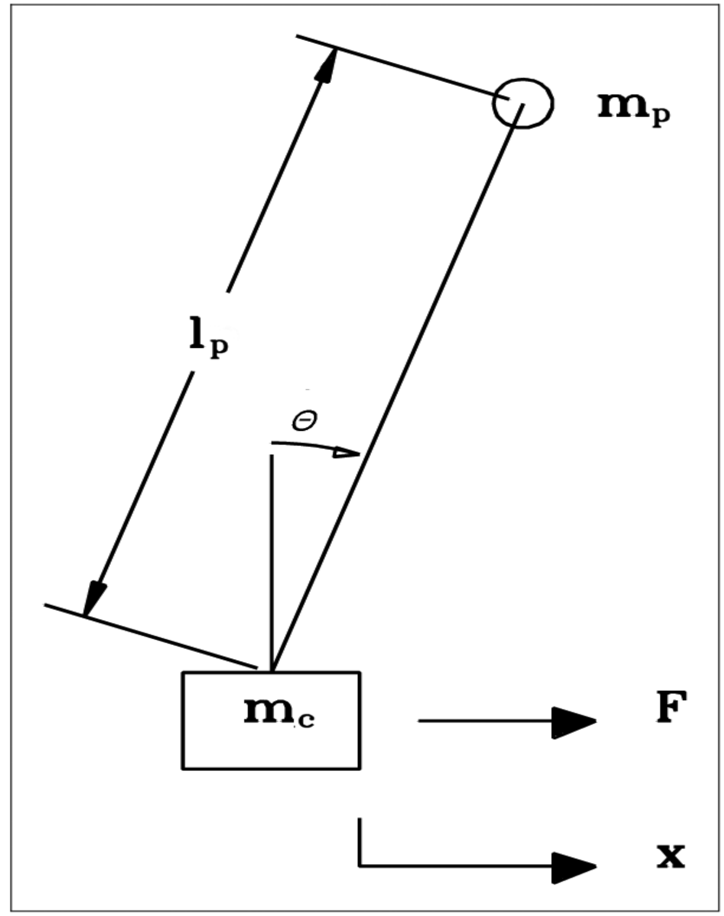


Values & Forces

- On Mars

twinqu.com/wp-content/uploads/2011/06/58.jpg

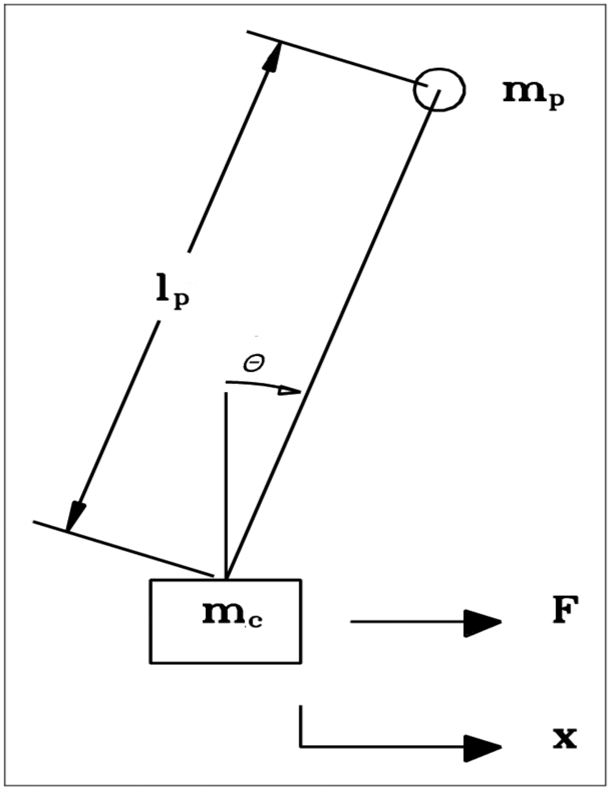
Symbol	Description
m_p	1.2 Kg
l_p	0.84 m
M_c	0.5 Kg
g	3.8 m/s^2 -Gravitational constant of Mars
u(t)	= (F(t), T_d(t)) - An external forcing function which applies a force and a torque at some time (t)
T_d	An external torque disturbance [Nm]



State Variables

- Use to fully record the state of the simulation at each step.
  - Linear and angular positions
  - Linear and angular velocities

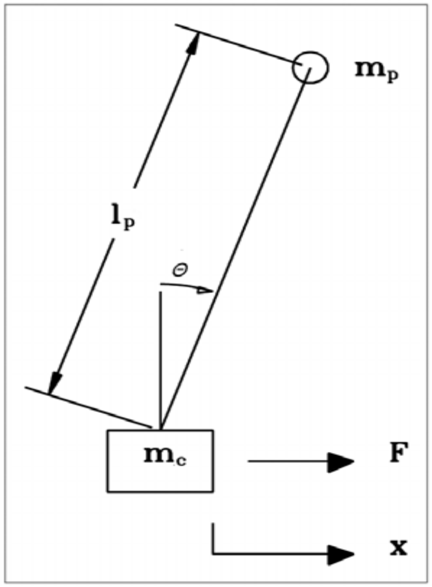
Symbol	Description
$x_1(t)$	Represent the rover position [m]
$x_2(t)$	The platform angle (measured from the upright position) [rad]
$x_3(t)$	The rover velocity [m/s]
$x_4(t)$	The angular velocity of the platform [rad/s]



- Note: “F” is some possible external force.

Controller

- Inverted pendulums are unstable and require an external “active controller” to react to disturbances.
  - Given in the problem
  - Force STRONGLY depends on the angle x_2(t)
  - Force also depends on the torque x_4(t)

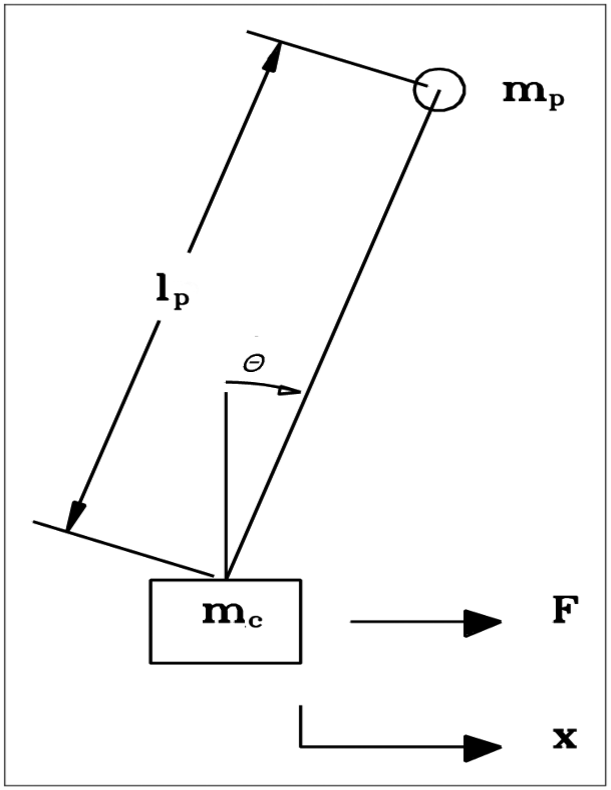


Symbol	Description
$x_1(t)$	Represent the rover position [m]
$x_2(t)$	The platform angle (measured from the upright position) [rad]
$x_3(t)$	The rover velocity [m/s]
$x_4(t)$	The angular velocity of the platform [rad/s]

$$F(t) = 3.16x_1(t) + 51.90x_2(t) + 5.64x_3(t) + 10.88x_4(t)$$

Initial Conditions & Forcing function

Symbol	Description
$x_1(t)$	position = 0.5 [m]
$x_2(t)$	Angle = -π/10 [rad]
$x_3(t)$	Linear velocity = 2 [m/s]
$x_4(t)$	Angular velocity = -1 [rad/s]

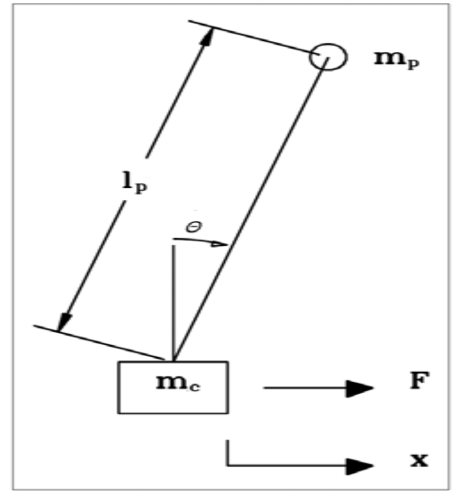


- Forcing Function
  - Given by the problem
  - At $t = 12$ [sec] the rover will ***hit a bump*** and will experiences a ***torque disturbance*** (T_d) of 1.1 Nm
  - Lasting for 0.5 seconds

Differential Equations

- Helper Function: $M(x_2(t)) = m_p(\cos x_2(t))^2 - (m_p + m_c)$
- Note: torque disturbance T_d

  Controller forcing function $F(t)$



- From the Physics of the problem

$$\dot{x}_1(t) = x_3(t)$$

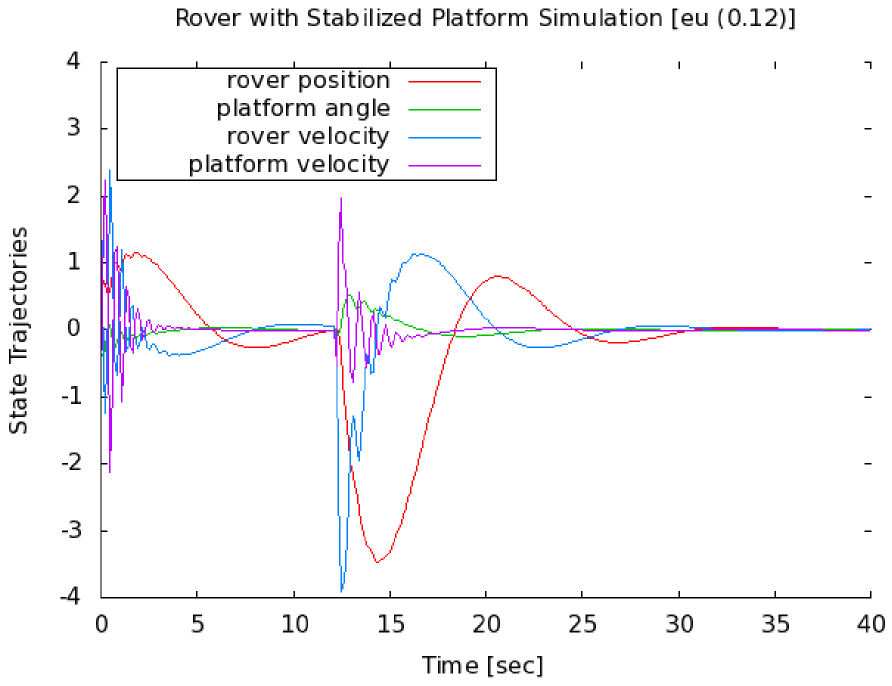
$$\dot{x}_2(t) = x_4(t)$$

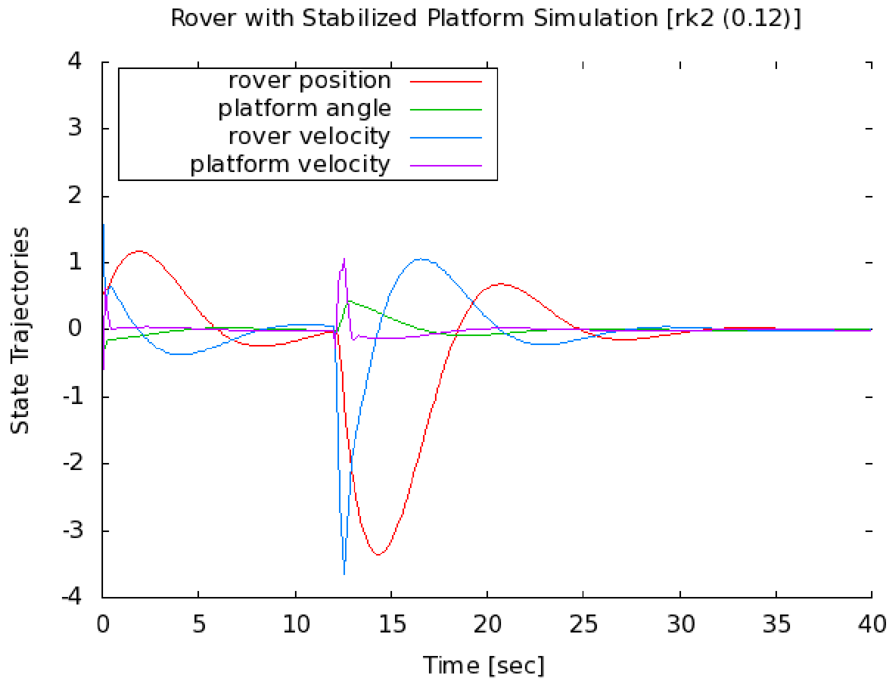
$$[M(x_2(t))]\dot{x}_3(t) = g\, m_p \sin x_2(t) \cos x_2(t) - \ell_p m_p x_4^2(t) \sin x_2(t) - F(t) + \frac{1}{\ell_p}\cos(x_2(t))\, T_d(t)$$

$$[\ell_p M(x_2(t))]\dot{x}_4(t) = -g\,(m_p + m_c)\sin x_2(t) + \ell_p m_p x_4^2(t) \sin x_2(t) \cos x_2(t)$$
$$+ \cos(x_2(t))\, F(t) + \frac{m_p + m_c}{\ell_p m_p} T_d(t)$$

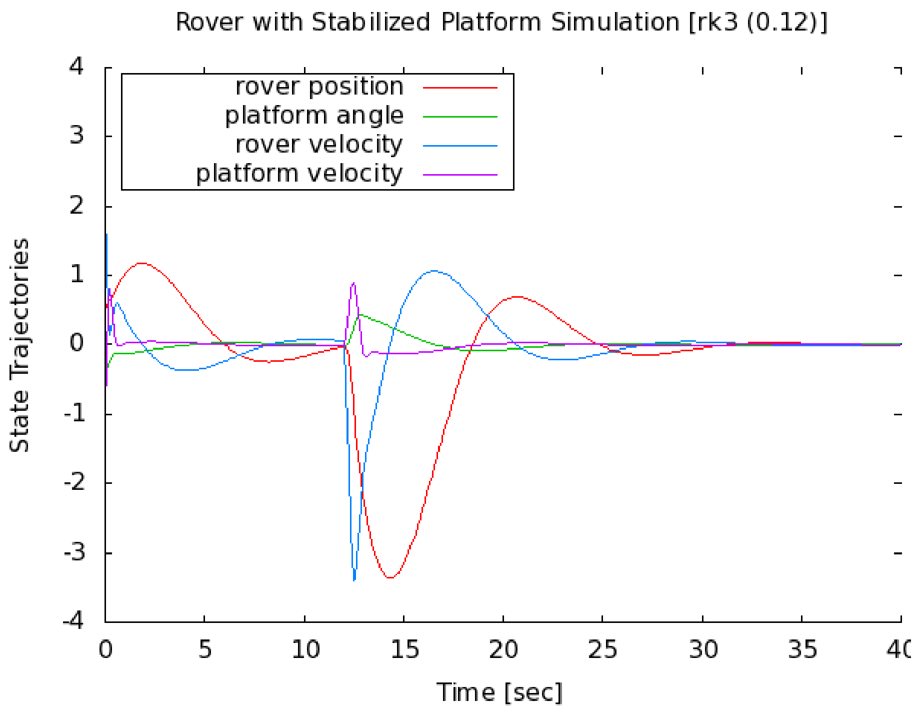
Symbol	Description
$x_1(t)$	Represent the rover position [m]
$x_2(t)$	The platform angle (measured from the upright position) [rad]
$x_3(t)$	The rover velocity [m/s]
$x_4(t)$	The angular velocity of the platform [rad/s]

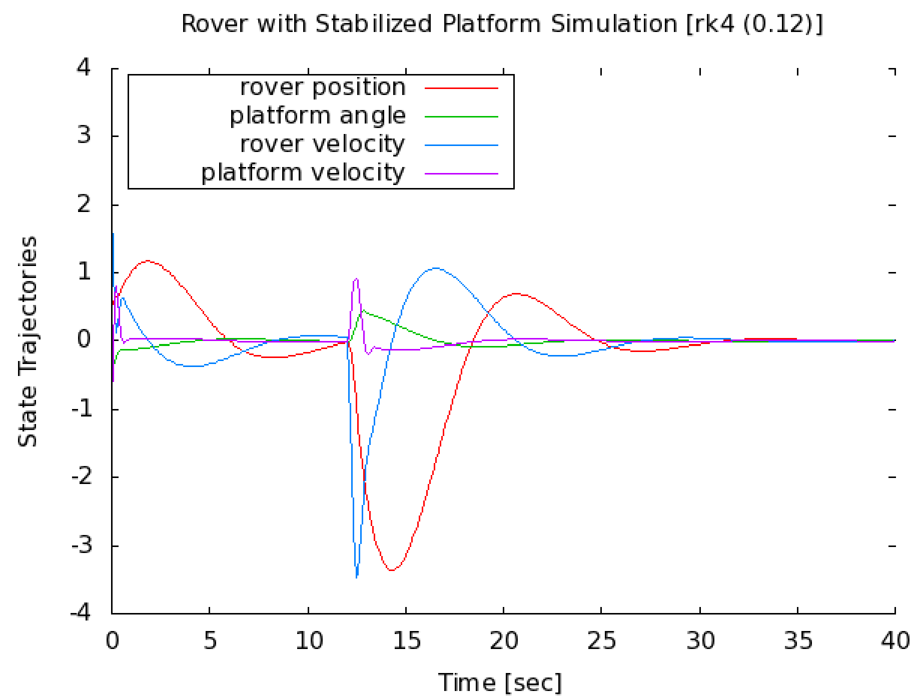
EU – RK2 Comparisons



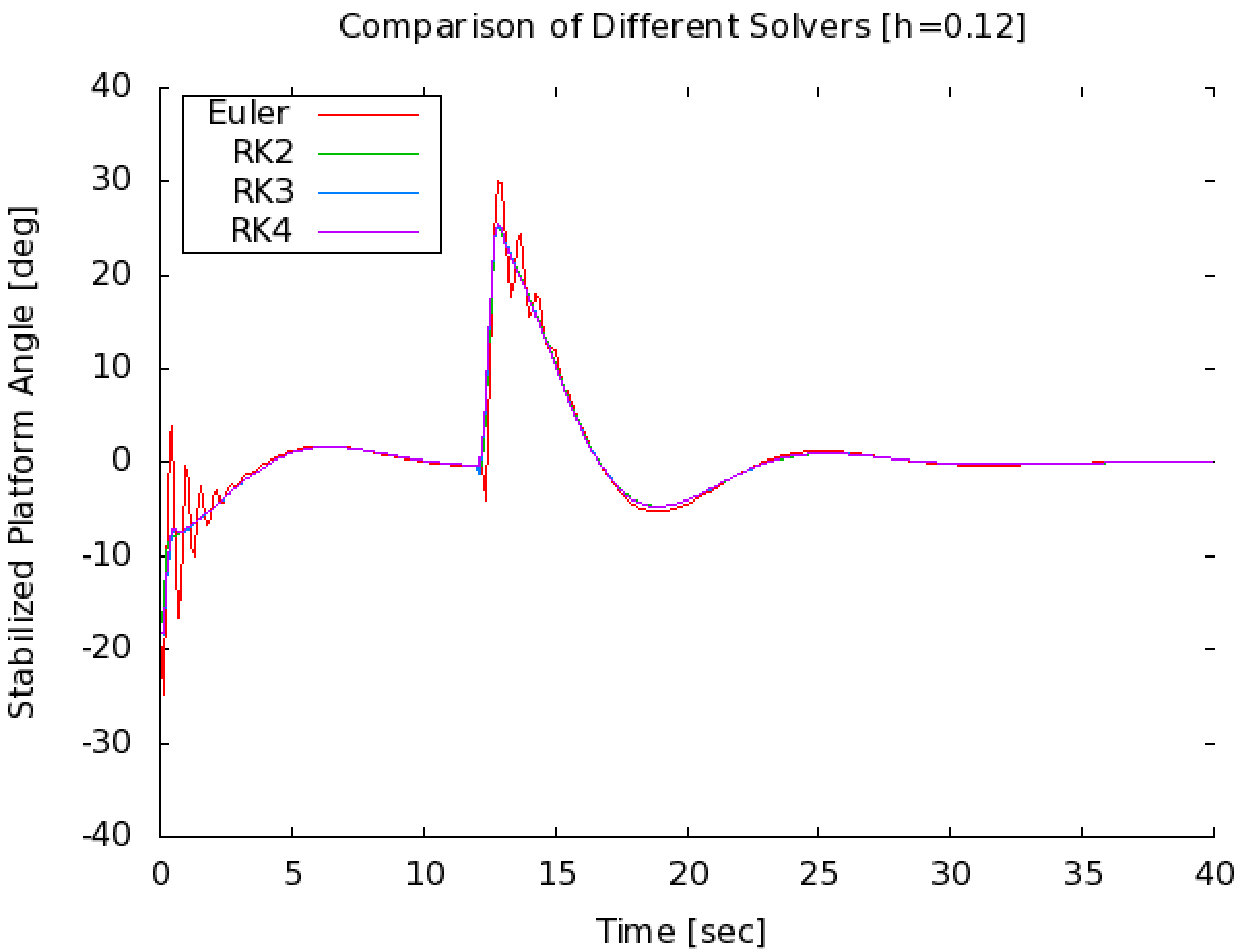


RK3 – RK4 Comparison





All Solvers



Comparison of Different Solvers [h=0.12]
Euler
RK2
RK3
RK4
Stabilized Platform Angle [deg]
40
30
20
10
0
-10
-20
-30
-40
0
5
10
15
20
25
30
35
40
Time [sec]