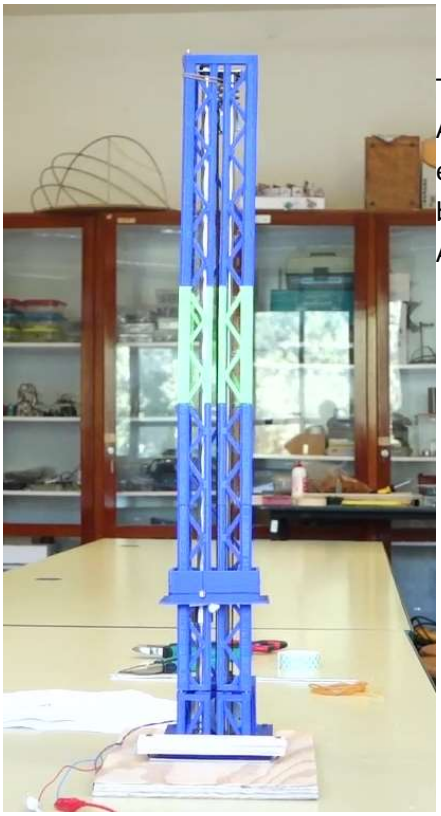
VRIJE ASO.SCHOOL

Let's go for a ride! The physics of roller-coasters

Creating a scale model and performing measurements on the model

Oil Tower 1

The Model:



The model is composed of two aluminum pole as support structures. At the top of it we secured a pulley which is connected with the engine by a belt. The external structure (tower and carriage) has been projected and printed with a 3D printer software. We used Arduino to program the engine.

Analysis:

- First shot

(A) is the starting point

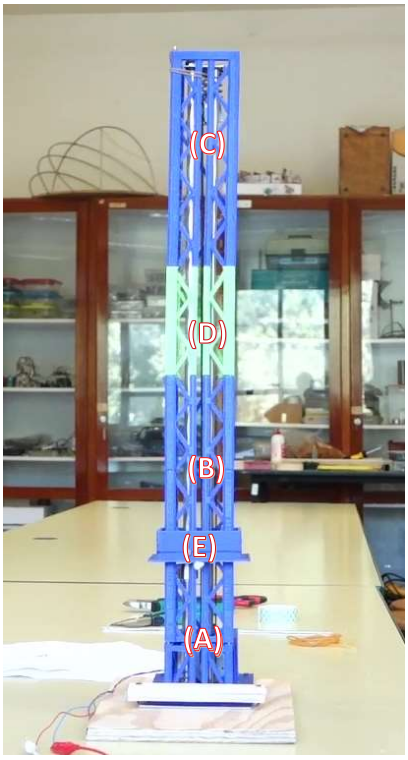
(B) is the point where the lift has the maximum speed

(C) is the point where the lift has reached the top of the tower (maximum height).

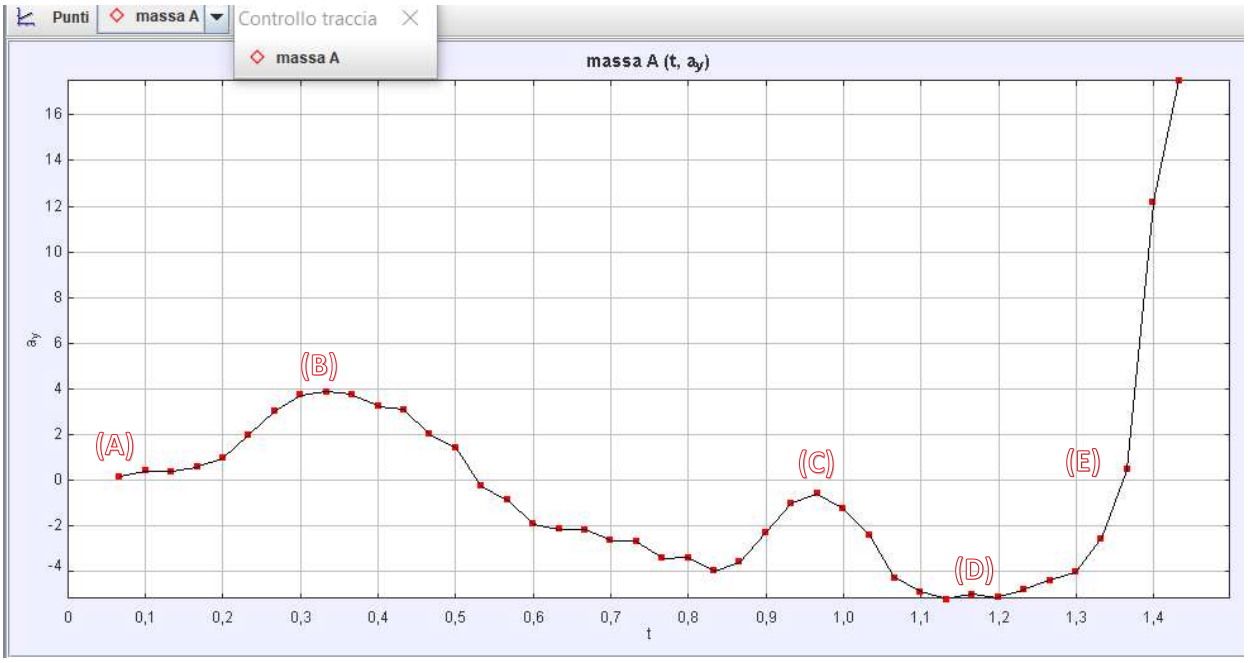
- First fall

(D) is the point where the lift starts to slow down.

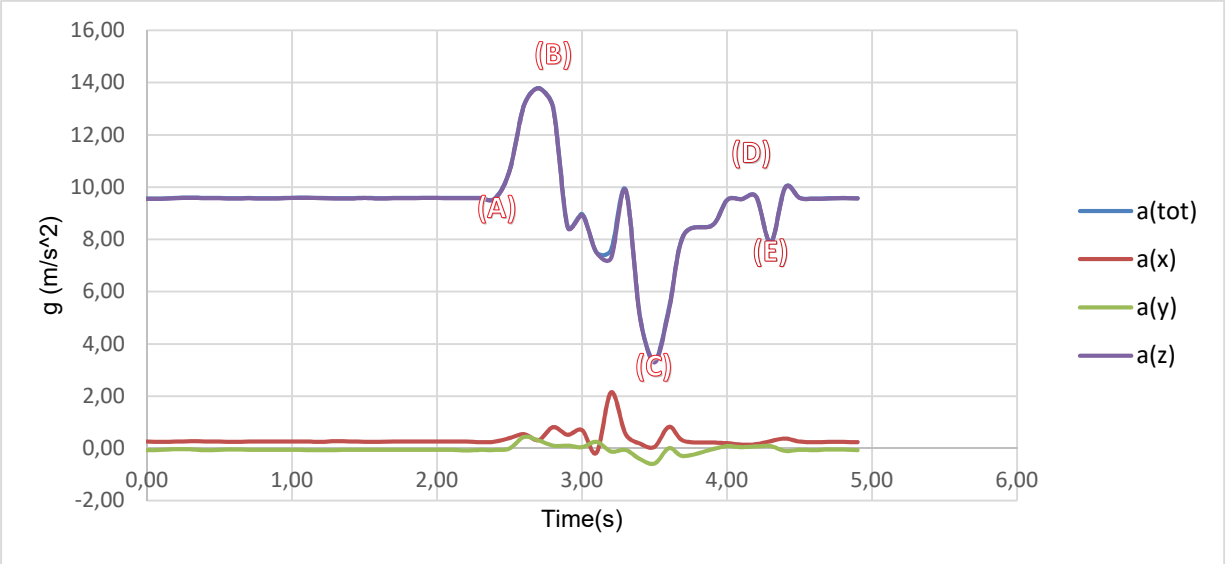
(E) is the lowest point reached by the lift in the first fall.



Graphics:



Analysis made with Tracker



Analysis made with an acceleration sensor

Programme of the scale model:

```
/*
  Oil Tower 1 programming code
*/

// voltage: 12,5 V

/*
  constant declaration, they never change until someone changes the programme
*/

const int MAX_PWM = 255; // max value permitted to handle motor power
const int MIN_PWM = 0; // min value permitted to handle motor power
const int CONV_S = 1000; // milliseconds-to-seconds conversion value
const int T_MAX = 1800; // max time for a ride

/*
  define which pin has to do
*/

int pin_PWM = 11; // pin used for actual power's value (0=0%, 255 = 100%)
int pin_orario = 12; // pin used for clockwise rotation
int pin_antiorario = 13; // pin used for counter-clockwise rotation
int pin_tasto = 7; // pin used for start button

/*
  variable declaration, they could change during programme execution
*/



int pwm_value; // power-counter variable
double t; // time passed(in milliseconds)

/*
  real program, it's the part of the programming-code that does what it has been programmed to do
  divided in 2 parts:
  -the setup: initialize all Arduino's settings and instructions
  -the loop: it's the main part of the program, it runs in an uninterrupted loop that ends only when you switch off the power or you change programme
*/

void setup() {

  Serial.begin(9600); // wait 9,6 seconds to run the program, default
  Serial.println("start"); // printing to serial monitor when Arduino is ready to work
  pinMode(pin_PWM, OUTPUT); // output-only pin 11 setting
  pinMode(pin_orario, OUTPUT); // output-only pin 12 setting
  pinMode(pin_antiorario, OUTPUT); // output-only pin 13 setting
  pinMode(7, INPUT_PULLUP); // input-only pin 2 setting


}


/*
   function that control all the ride
*/
```

```
void ride(){
  t=0; // setting the variable of the time to value 0
  do{

    pwm_value = pwm_value_sin(t); // calling the function that returns the power level (between 0 and 255)
    t += 5; // increasing the time, 5  milliseconds per cycle

    if (pwm_value >= 0) { // controlling if the power level is over or below 0

      /*
        if it's above, the motor has a clockwise rotation
      */

      Serial.println(pwm_value);

      digitalWrite(pin_orario, HIGH); // declaring that pin 12 is the phase
      digitalWrite(pin_antiorario, LOW); // declaring that pin 13 is the neutral

      analogWrite(pin_PWM, pwm_value); // giving the power value to pin 11

    } else {

      /*
        if it's below, the motor has a counter-clockwise rotation
      */

      if(t<=1330){ // when the time is below this value, the motor pushes down the carriage with an acceleration higher than the gravity one


        digitalWrite(pin_orario, LOW); // declaring that pin 12 is the neutral
        digitalWrite(pin_antiorario, HIGH); // declaring that pin 13 is the phase

        analogWrite(pin_PWM, -pwm_value); // giving the power value to pin 11

      }else{ // when the time is over the value, the motor controls the carriage fall and stops it before it slams down
        if(t<=1450){
          digitalWrite(pin_orario, HIGH); // declaring that pin 12 is the phase
          digitalWrite(pin_antiorario, LOW); // declaring that pin 13 is the neutral

          analogWrite(pin_PWM, 145); // declaring that PWM value is 145 out of 255 to contrast the gravity acceleration,controls and stops the fall

        }else{
          digitalWrite(pin_orario, HIGH); // declaring that pin 12 is the phase
          digitalWrite(pin_antiorario, LOW); // declaring that pin 13 is the neutral

          analogWrite(pin_PWM, 50); // declaring that PWM value is 50 out of 255 to contrast the gravity acceleration,controls and stops the fall

        }
      }
    }

    delay(5); // everytime the programme has completed its run, Arduino waits 5 milliseconds before starting another cycle

  }while(t!=T_MAX); // exit from the loop and to the function only if the time has reached the max value

}


/*
  function that transforms the time(in seconds) to the equals value in radians, one shot and one fall is like 2π radians
*/

double a_rad(double t) {

  return t * 2 * PI / T_MAX; // returning the value of the time transformed to radians

}

/*
  function that returns the power value between -255 and 255
*/

int pwm_value_sin(double t) {

  return MIN_PWM + MAX_PWM * sin(a_rad(t)); // returning the power value as a multiplication between the max power and the sin value in a particular time

}

/*
  here starts the uninterrupted loop
*/

void loop() {

  if(digitalRead(7) == LOW){ // controlling if the button has been pulled. HIGH = pulled, LOW = not pulled
    Serial.println("start"); // printing to the serial monitor when the ride is starting




    delay(1000); // waiting for 1 second to start the ride
    ride(); // calling the function that starts the ride
    digitalWrite(pin_orario, HIGH);
    digitalWrite(pin_antiorario, LOW);
    analogWrite(pin_PWM, 11);
    /*
      setting clockwise rotation to keep rest the motor
    */

  }

}


/*
  programmers: Elisabetta Donati, Lanconelli Luca.

  thanks to Enzo Cortesi, Peter Ulf Johan Helgesson, Elisa Capucci, Marco Vassura and Elisabetta Medici for their patience and their help during all the project.

*/
```

Screaming Eagle

In this file we are going to discuss the results of our scale model. We got the data from the analyses of the program *tracker*. We calculated three kinds of energy. Kinetic, potential and mechanical energy. We made our conclusions based on the difference of these energy. We could not calculate V_x because there's no horizontal movement.

$E_{kin} = m \cdot v^2$

The kinetic energy at the beginning:

Data: m = 60 g = 60 . 10^3 kg

V_b = - 88, 097 m/s

Wanted: $E_{kin, b}$

Solution: $E_{kin, b} = 60 \cdot 10^3 \text{ kg} \cdot (-88{,}97) \text{ m/s} = -5{,}3 \cdot 10^6 \text{ J}$

The kinetic energy in the middle

Data: m = 60 g = 60 . 10^3 kg

v_m = -115 m/s

Wanted: $E_{kin, m}$

Solution: $E_{kin, m} = 60 \cdot 10^3 \text{ kg} \cdot (-115) \text{ m/s} = -6{,}9 \cdot 10^6 \text{ J}$

The kinetic energy at the end

Data: m = 60 g = 60 . 10^3 kg

v_e = - 250 m/s

Wanted: $E_{kin, e}$

Solution: $E_{kin, e} = 60 \cdot 10^3 \text{ kg} \cdot (-250) \text{ m/s} = -1{,}5 \cdot 10^7 \text{J}$

The difference in kinetic energy

$\Delta E_{kin} = E_{kin,\,e} - E_{kin,b}$

$= -1{,}5 \cdot 10^7 - (-5{,}3 \cdot 10^6) = -1{,}4 \cdot 10^7 J$

$E_{pot} = m \cdot g \cdot h$

The potential energy at the beginning

Data: $m = 60\ g = 60 \cdot 10^3\ kg$

$g = 9{,}81\ m/s^2$

$h = 38\ m$

Wanted: $E_{pot,\,b}$

Solution: $E_{pot,\,b} = 60 \cdot 10^3\ kg \cdot 9{,}81\ m/s^2 \cdot 38\ m = 2{,}2 \cdot 10^7\ J$

The potential energy in the middle

Data: $m = 60\ g = 60 \cdot 10^3\ kg$

$g = 9{,}81\ m/s^2$

$h = 18\ m$

Wanted: $E_{pot,\,b}$

Solution: $E_{pot,\,b} = 60 \cdot 10^3\ kg \cdot 9{,}81\ m/s^2 \cdot 18\ m = 1{,}1 \cdot 10^7\ J$

The potential energy at the end

Data: $m = 60\ g = 60 \cdot 10^3\ kg$

$g = 9{,}81\ m/s^2$

$h = 2\ m$

Wanted: $E_{pot,\,b}$

Solution: $E_{pot,\,b} = 60 \cdot 10^3\ kg \cdot 9{,}81\ m/s^2 \cdot 2\ m = 1{,}2 \cdot 10^6\ J$

The difference in potential energy

$\Delta E_{pot} = E_{pot,\,e} - E_{pot,b}$

$= 1{,}2 \cdot 10^6\,J - 2{,}2 \cdot 10^7\,J = -2{,}0 \cdot 10^7\,J$

$E_{mech} = E_{kin} + E_{pot}$

The mechanical energy at the beginning

Data: $E_{kin,\,b} = -5{,}3 \cdot 10^6\,J$

$E_{pot,\,b} = 2{,}2 \cdot 10^7\,J$

Wanted: $E_{mech,\,b}$

Solution: $E_{mech,\,b} = E_{kin,b} + E_{pot,b} = -5{,}3 \cdot 10^6 + 2{,}2 \cdot 10^7\,J = 2{,}7 \cdot 10^7\,J$

The mechanical energy in the middle

Data: $E_{kin,\,m} = -6{,}9 \cdot 10^6 J$

$E_{pot,\,m} = 1{,}1 \cdot 10^7 J$

Wanted: $E_{mech,\,m}$

Solution: $E_{mech,\,m} = E_{kin,m} + E_{pot,m} = -6{,}9 \cdot 10^6 + 1{,}1 \cdot 10^7 J = 5{,}1 \cdot 10^6\,J$

The mechanical energy at the end

Data: $E_{kin,\,e} = -1{,}5 \cdot 10^7 J$

$E_{pot,\,e} = 1{,}2 \cdot 10^6\,J$

Wanted: $E_{mech,\,e}$

Solution: $E_{mech,\,e} = E_{kin,e} + E_{pot,e} = -6{,}9 \cdot 10^6\,J + 1{,}1 \cdot 10^7 J = 4{,}1 \cdot 10^6 J$

The difference in mechanical energy

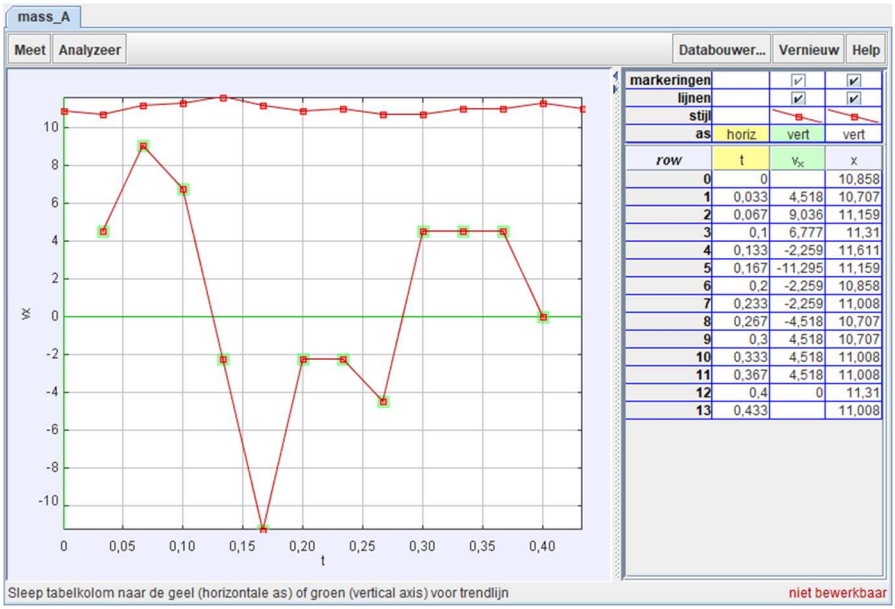
$\Delta E_{mech} = E_{mech,\,e} - E_{mech,b}$

$= 4{,}1 \cdot 10^6 J - 2{,}7 \cdot 10^7 J = -2{,}2 \cdot 10^7 J$

Conclusion

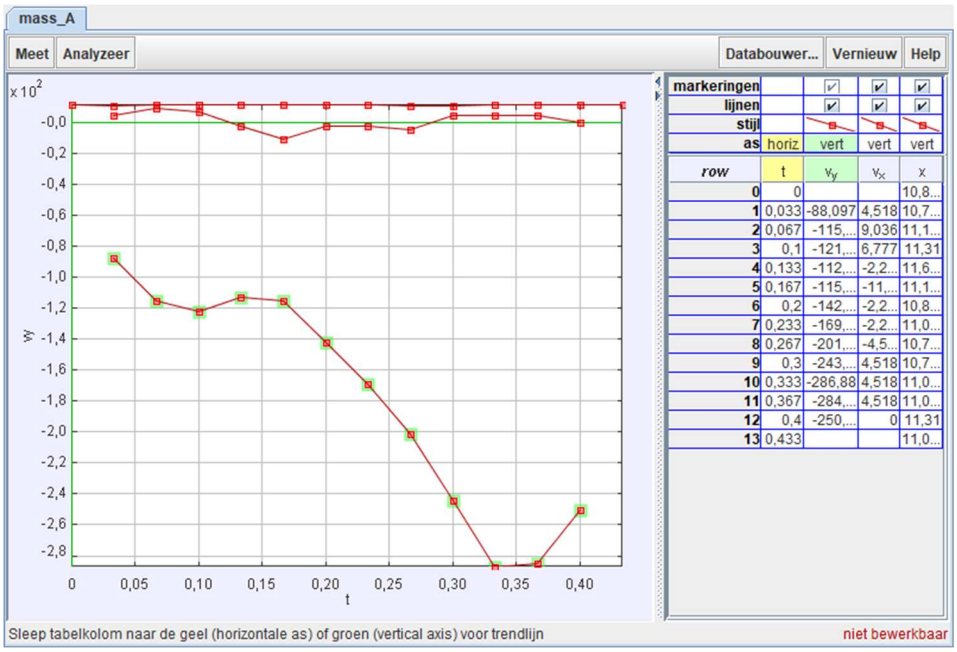
We can conclude that our results are not completely correct, because we used a scale model. We realized that our v_x-graphic is not important, because we have a drop tower and the graphic is based on the left and right movements in its drop. The speed increases when it drops, that's also what we thought, because it only falls because of the gravity. It's not a smooth line because it slipped at the end.

This graph is not useable because there's no horizontal movement.



markeringen		✔	✔
lijnen		✔	✔
stijl			
as	horiz	vert	vert
row	t	v_x	x
0	0		10,858
1	0,033	4,518	10,707
2	0,067	9,036	11,159
3	0,1	6,777	11,31
4	0,133	-2,259	11,611
5	0,167	-11,295	11,159
6	0,2	-2,259	10,858
7	0,233	-2,259	11,008
8	0,267	-4,518	10,707
9	0,3	4,518	10,707
10	0,333	4,518	11,008
11	0,367	4,518	11,008
12	0,4	0	11,31
13	0,433		11,008

This is the V_x(T) graph and shows the speed of the falling cage. It's not a smooth graph because it slipped at the end.



This is the