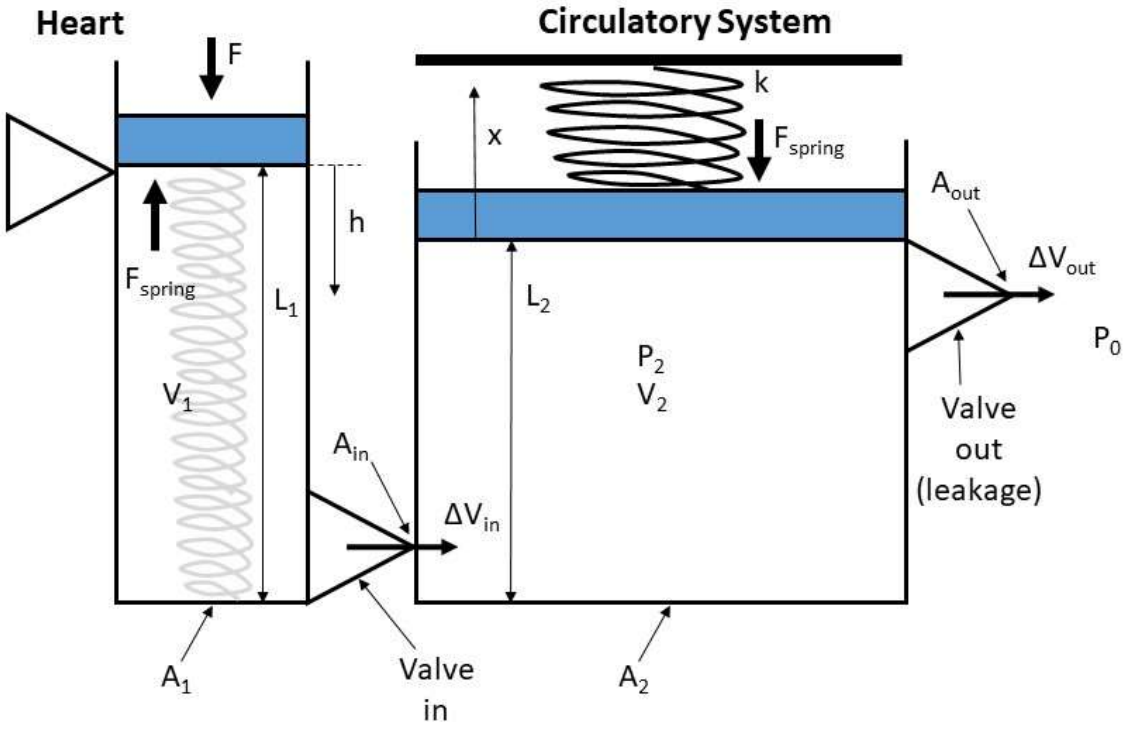
Modeling CPR With a Piston-Cylinder Apparatus



Constants: A_1 , A_2 , L_2 , k , A_{in} , A_{out} , P_0 , ρ

Volume flow into chamber 2 is equal to volume flow out of chamber 1 which is dependent only on change in h. Because of the one-way valve, as long as Δh is positive…

$$\Delta \boldsymbol{V_{in}} = \Delta \boldsymbol{V_1} = \boldsymbol{A_1} * \Delta \boldsymbol{h} \qquad \textbf{Eq1}$$

Now find volume flow out of chamber 2. Using Bernoulli…

$$P_2 = P_0 + \frac{1}{2}\rho v_{out}^2$$

$$P_0 = 0$$

$$\boldsymbol{v_{out}} = \sqrt{\frac{\boldsymbol{2P_2}}{\boldsymbol{\rho}}} \qquad \textbf{Eq2}$$

$$\Delta \boldsymbol{V_{out}} = \boldsymbol{v_{out}} \boldsymbol{A_{out}} \Delta \boldsymbol{t} \qquad \textbf{Eq3}$$

Now update volume of chamber 2…

$$\boldsymbol{V_2} = \boldsymbol{V_2} + \Delta \boldsymbol{V_{in}} - \Delta \boldsymbol{V_{out}} \qquad \textbf{Eq4}$$

Use volume to find displacement of spring in chamber 2 (x) ...

$$V_2 = A_2(L_2 + x)$$

$$\boldsymbol{x} = \frac{\boldsymbol{V_2}}{\boldsymbol{A_2}} - \boldsymbol{L_2} \qquad \textbf{Eq5}$$

Now use x to find spring force using Hooke's Law…

$$\boldsymbol{F_{spring}} = \boldsymbol{kx} \qquad \textbf{Eq6}$$

Finally, use spring force to find pressure in chamber 2…

$$F_{spring} = F_{pressure}$$

$$P = \frac{F}{A}$$

$$\boldsymbol{P_2 = \frac{F_{spring}}{A_2}} \qquad \textbf{Eq7}$$

Execute **Eq1-7** for each iteration of the loop to update volume and pressure of chamber 2. See below for Arduino script.

```
const int pin_in1 = 5;
const int pin_in2 = 7;

const float area_out = .00002; //m^2
const float area_chamber_1 = 0.01; //m^3 - cross sectional area of the piston-cylinder (chamber 1)
const float initial_height_chamber_2 = .04; //m
const float spring_constant = 5000; // N/m
const float area_chamber_2 = .01; //m^2
const float density = 1000; // kg/m^3

const String comma = ",";

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  // make pins an input:
  pinMode(pin_in1, INPUT);
  pinMode(pin_in2, INPUT);
}

int prevStateIn1 = 0;
int prevStateIn2 = 0;
int depth = 0;

int lastPositive = 0; //the last sensor to become positive. Either 1 or 2. 0 if both at same time

float volume2 = area_chamber_2 * initial_height_chamber_2;
float pressure2 = 0;
long previousTime = 0;

// the loop routine runs over and over again forever:
void loop() {

  int prevDepth = depth; //save previous to find delta

  // read the input pins: Use PIND low level because it's faster than digitalRead
  int stateIn1 = digitalRead(pin_in1);//PIND & in1 ? 1 : 0;
  int stateIn2 = digitalRead(pin_in2);//PIND & in2 ? 1 : 0;

  //we need to interpret sensor changes as the linear encoder moving up/down.
  //when it moves down stateIn1 should become positive, followed by stateIn2
  //becoming positive, then stateIn2 = negative, and finally stateIn2 = negative.
  //Moving up is the opposite, 2 = pos, 1 = pos, 2 = neg, 1 = neg. Problem is that
  //the sensors don't always recognize the full cycle. We account for that
  //below

  int in1Pos = stateIn1 > prevStateIn1 ? 1 : 0; //true (1) if input1 went from neg to pos in the last iteration
  int in1Neg = stateIn1 < prevStateIn1 ? 1 : 0; //true if input1 went from pos to neg in the last iteration
  int in2Pos = stateIn2 > prevStateIn2 ? 1 : 0; //true if input2 went from neg to pos in the last iteration
  int in2Neg = stateIn2 < prevStateIn2 ? 1 : 0; //true if input2 went from pos to neg in the last iteration

  //if both sensors are negative and at least one of them just turned negative
  if(!stateIn1 && !stateIn2 && (in1Neg|| in2Neg)) {
    //if input 1 just turned negative and the last positive was not 2 (could be 1 or both)
    if(in1Neg && !in2Neg && lastPositive != 2) {
      depth--;
    //if input 2 just turned negative and the last positive was not 1 (could be 2 or both)
    } else if(in2Neg && !in1Neg && lastPositive != 1) {
      depth++;
    //if both just turned negative at the same time and the last positive was 1
    } else if(in1Neg && in2Neg && lastPositive == 1) {
      depth--;
    //if both just turned negative and the last positive was 2
    } else if(in1Neg && in2Neg && lastPositive == 2) {
      depth++;
    } else { //otherwise decrease depth to help prevent gradual rise
      depth--;
      lastPositive = 0;
    }
    if(depth < 0) { //prevent gradual decrease in depth
      depth = 0;
    }

    //log depth and pressure to console
    float display_pressure = (pressure2 / 1000.0); //convert to kpa (guage)
    Serial.println(depth + comma + display_pressure);
```

```
  //if both are positive and at least one just turned positive
  } else if(stateIn1 && stateIn2 && (in1Pos || in2Pos)) {
    //if 2 was already positive
    if(in1Pos && !in2Pos) {
      lastPositive = 1;
    //if 1 was already positive
    } else if(in2Pos && !in1Pos) {
      lastPositive = 2;
    //if they both became positive at the same time
    } else {
      lastPositive = 0;
    }
  }

  //update previous state variables
  prevStateIn1 = stateIn1;
  prevStateIn2 = stateIn2;

  int deltaDepth = depth - prevDepth; //change in depth

  float deltaDepthMeters = deltaDepth / 1000.0; //change in depth in meters


  //find time change
  long currentTime = millis();
  float deltaT = (currentTime - previousTime) / 1000.0;
  previousTime = currentTime;

  //find volume and pressure changes

  //volume flow into chamber 2 is equal to the volume flow out of chamber 1
  float volumeFlowIn = deltaDepthMeters > 0 ? area_chamber_1 * deltaDepthMeters : 0;
  float velocityOut = sqrt(2 * pressure2 / density); //bernoulli
  float volumeFlowOut = velocityOut * area_out * deltaT; //dV = v*A*dt
  volume2 = volume2 + volumeFlowIn - volumeFlowOut;
  float springDisplacement = volume2 / area_chamber_2 - initial_height_chamber_2; //V2 = A2(H2 + x)
  float springForce = spring_constant * springDisplacement; //F = kx
  pressure2 = springForce / area_chamber_2; //P = F/A

}
```