Optimizing Indoor Air Quality and Energy Efficiency

A Comprehensive Guide to Assessing Energy Loss and Humidity Control Strategies in Buildings

Source Code: https://github.com/star-3gg/dehumidifier-analysis
Author: [redacted]
Date: October 24, 2023

[redacted]

Contents

1 Introduction

Efficient energy management is crucial for both environmental sustainability and economic viability. Among the various factors affecting energy usage in buildings, humidity control is particularly vital but often overlooked. This paper aims to fill that gap by examining the thermal energy losses associated with ventilation as compared to the energy consumption of dehumidifying systems in colder climates.

1.1 Purpose and Intent

The primary objective of this paper is to serve as a comprehensive guide for optimizing energy efficiency in buildings situated in colder climates. To facilitate this, a Python 3 program was developed that can quantitatively evaluate thermal energy losses and compare various humidity control methods. This paper aims to challenge the prevailing notion that increased building automation necessarily results in higher energy consumption. Our findings suggest that strategic automation, especially in humidity control systems, can improve air quality and overall building efficiency while reducing total energy use. For those interested in a deeper analysis or contributing to the project, the codebase is accessible at the GitHub repository:

https://github.com/star-3gg/dehumidifier-analysis

1.2 Significance of Humidity Control

Humidity control in buildings is not merely a matter of comfort but has significant implications for health and structural integrity. Maintaining optimal indoor humidity levels can drastically reduce the occurrence of mold and unpleasant odors, thereby improving the air quality. This, in turn, positively impacts the health and quality of life of the building's occupants. Our work is supported by data, offering practical insights that can be immediately applied to existing building management practices.

2 Calculating the Thermal Energy Loss due to Ventilation

2.1 Calculation

$$\Delta E = m \cdot c \cdot \Delta T$$

This notation adheres to standard physics conventions for energy calculations involving specific heat capacity.
Where:

- ΔE represents the energy loss in joules (J).
- m represents the mass of the air in kilograms (kg).
- c represents the specific heat capacity of air in J/(kg·°C).
- ΔT represents the temperature difference in degrees Celsius (°C).

2.2 Python Implementation

```
def calc_thermal_energy_loss_in_joules(
    specific_heat_cap_joule_kilogram,
    volume_in_cubic_meters, temperature_difference)
    :
   energy_loss_in_joules =
       specific_heat_cap_joule_kilogram *
       volume_in_cubic_meters *
       temperature_difference
   return energy_loss_in_joules
```

Where:

- $specific\_heat\_cap\_joule\_kilogram$ is the specific heat capacity of air.
- $volume\_in\_cubic\_meters$ is the volume of the building.
- $temperature\_difference$ is the change in temperature.

3 Conversion of Relative Humidity Reduction to Water Volume in Liters

The reduction in relative humidity (RH) after ventilating a building can be converted into liters of water, representing the moisture removed from the air.

3.1 Mathematical Models

3.1.1 Fast Algorithm

The fast algorithm uses the Antoine equation to approximate vapor pressure (P_{mmHg}) as follows:

$$P_{\text{mmHg}} = 10^{(A - \frac{B}{T+C})}$$

3.2 Python Implementation

```python
# Python code for Accurate Algorithm
def
    calc_accurate_absolute_humidity_in_gram_cubic_meter
    (rh,temperature):
    """
    Calculate the accurate absolute humidity using
        Wagner and Pruss equation.

    Parameters:
        rh (float): Relative Humidity in percentage
        temperature (float): Temperature in Celsius

    Returns:
        float: Absolute humidity in g/m^3
    """
    # Specific gas constant for water vapor in J/(kg
        *K)
    Rw = 461.5
    # Critical pressure for water in Pa
    Pc = 22.064e6
    # Critical temperature for water in K
    Tc = 647.096
    # Empirical constants for Wagner and Pruss
        equation
    a1 = -7.85951783
    a2 = 1.84408259
    a3 = -11.7866497
    a4 = 22.6807411
    a5 = -15.9618719
    a6 = 1.80122502
    # Temperature in Kelvin
    T = temperature + 273.15
    # tau for Wagner and Pruss equation
    tau = 1 - T / Tc
    # Saturation vapor pressure using Wagner and
        Pruss equation in Pa
    Ps = Pc * math.exp(Tc / T * (a1 * tau + a2 * tau
        **1.5 + a3 * tau**3 + a4 * tau**3.5 + a5 *
        tau**4 + a6 * tau**7.5))
    # Calculate absolute humidity in g/m^3
    AH = rh * Ps / (Rw * T) * 10
    return AH
```

Where:
- `rh` is the Relative Humidity in percentage.
- `temperature` is the temperature in Celsius.
- `volume` is the volume of the building in cubic meters.

3.2.1 Accurate Algorithm

The accurate algorithm uses the Wagner and Pruss equation for saturation vapor pressure (P_s) as follows:

$$P_s = P_c \exp\left(\frac{T_c}{T}\left(a_1\tau + a_2\tau^{1.5} + a_3\tau^3 + a_4\tau^{3.5} + a_5\tau^4 + a_6\tau^{7.5}\right)\right)$$

Where:
- A, B, C are Antoine equation constants for water.
- $P_c, T_c, a_1, a_2, a_3, a_4, a_5, a_6$ are Wagner and Pruss equation constants.
- T is the temperature in Kelvin.
- $\tau = 1 - \frac{T}{T_c}$

3.3 Python Implementation

```python
# Python code for Fast Algorithm
def calc_fast_absolute_humidity_in_gram_cubic_meter
    (rh,temperature):
    """
    Calculate the fast approximation of absolute
        humidity using Antoine equation.

    Parameters:
        rh (float): Relative Humidity in percentage
        temperature (float): Temperature in Celsius

    Returns:
        float: Absolute humidity in g/m^3
    """
    # Antoine equation constants for water
    A_water = 8.07131
    B_water = 1730.63
    C_water = 233.426
    # Calculate vapor pressure using Antoine
        equation (in mmHg)
    P_mmHg = 10**(A_water - (B_water / (temperature
        + C_water)))
    # Convert vapor pressure to Pascals
    P_Pa = P_mmHg * 133.322
    # Calculate the partial pressure of water vapor
        (in Pa)
    partial_pressure = rh / 100 * P_Pa
    # Convert temperature to Kelvin
    temperature_K = temperature + 273.15
    # Universal Gas constant (R) in J/(mol*K)
    R = 8.314
    # Molar mass of water in kg/mol
    M_water = 18.01528 / 1000
    # Calculate absolute humidity in g/m^3
    absolute_humidity = (partial_pressure * M_water)
        / (R * temperature_K) * 1e3
    return absolute_humidity
```

Where:
- `rh` is the Relative Humidity in percentage.
- `temperature` is the temperature in Celsius.
- `volume` is the volume of the building in cubic meters.

Both algorithms are encapsulated in a function that computes the total water content in liters, taking the building's volume into account.

3.4 Python Implementation

```
# Python code for function wrapper and liter
    conversion using the accurate solver
def rh_to_ah_in_liters(rh, temperature, volume):
    """
    Convert relative humidity to absolute humidity
        in liters for a given volume.

    Parameters:
        rh (float): Relative Humidity in percentage
        temperature (float): Temperature in Celsius
        volume (float): Volume in cubic meters

    Returns:
        float: Absolute humidity in liters
    """
    # Calculate accurate absolute humidity in g/m^3
    ah_gram_per_cubic_meter =
        calc_accurate_absolute_humidity_in_gram_cubic_meter
        (rh, temperature)
    # Calculate the total water content in the given
         volume in grams
    total_water_content_grams =
        ah_gram_per_cubic_meter * volume
    # Convert the total water content to liters (
        since 1 liter of water is 1000 grams)
    total_water_content_liters =
        total_water_content_grams / 1000
    return total_water_content_liters
```

4 Comparing the Energy Losses

4.1 Calculation

$$\Delta E = E_{\text{loss}} - E_{\text{dehum}}$$

Where:
- ΔE is the difference in energy consumption between the two methods.
- E_{loss} is the energy lost due to ventilation.
- E_{dehum} is the energy consumed by the dehumidifier.

4.2 Python Implementation

```
def simulate_and_compare(duration, volume,
    inside_temp, measured_temp, inside_hum_rh,
    measured_hum_rh):
    # Implementation of the simulation and
        comparison
```

Where:
- 'duration', 'volume', '*inside_temp*', '*measured_temp*', '*inside_hum_rh*', '*measured_hum_rh*' are the input parameters for the simulation taken from real world measurements and estimates.

5 Conclusion

5.1 Experimental Data and Sample Input

The following Python code snippet provides a sample input used for the calculations. These variables are crucial for analyzing the energy efficiency of both airing and using a dehumidifier in a building environment.

5.1.1 Explanation of Variables

- **Ventilation Variables:**
  - `duration_of_ventilation_in_minutes`: Duration of each ventilation session.
  - `number_of_ventilations_per_day`: The number of times ventilation is performed per day.
- **Building Variables:**
  - `building_area_in_square_meters`: Floor area of the building.
  - `building_height_in_meters`: Ceiling height of the building.
  - `building_volume_in_cubic_meters`: Total volume of the building space.
- **Temperature Variables:**
  - `inside_temperature_at_114_centimeters_room_elevation` Initial inside temperature measured at a height of 114 cm.
  - `measured_temperature_after_duration_at_114_centimete` Measured inside temperature after the duration of ventilation.
  - `specific_heat_capacity_of_air_in_joules_kilogram`: Specific heat capacity of air.
- **Humidity Variables:**
  - `inside_humidity_at_114_centimeters_elevation_in_perc` Initial inside humidity level.
  - `measured_humidity_after_duration_in_percentage_relat` Humidity level after the duration of ventilation.
- **Dehumidifier Variables:**
  - `dehumidifier_wh_per_litre_removed`: Energy consumed by the dehumidifier per liter of water removed.
- **Heating System Efficiency:**
  - `hydronic_radiant_efficiency`: Efficiency of the hydronic radiant heating system.

5.1.2 Sample Input

```
# Ventilation variables
duration_of_ventilation_in_minutes = 10
number_of_ventilations_per_day = 3
# Building variables
building_area_in_square_meters = 84 # float
building_height_in_meters = 2.5 # float
building_volume_in_cubic_meters = building_area_in_square_meters * building_height_in_meters
# Temperature variables
inside_temperature_at_114_centimeters_room_elevation_in_degrees = 22.4
measured_temperature_after_duration_at_114_centimeters_room_elevation_in_degrees = 18.4
specific_heat_capacity_of_air_in_joules_kilogram=1005
# Humidity variables
inside_humidity_at_114_centimeters_elevation_in_percentage_relative_humidity = 68
measured_humidity_after_duration_in_percentage_relative_humidity = 54
# Dehumidifier variables
dehumidifier_wh_per_litre_removed=297.0
# Heater Efficiency
hydronic_radiant_efficiency = 0.95
```

5.2 Experimental Results

Sample output:

```
-- ENERGY LOSS CALCULATION --

      Room volume:   210.0 m^3

      Heat loss(12.0 C):  2532600.00 Joules
                                          703.50 Wh

      Heat loss(12.0 C) after energy conversion compensation (0.95 efficiency):  740.53 Wh

      Humidity reduction:  14% rH

-- COMPARISON --

Comparing airing to dehumidifier for 14 % rH in 210.0 m^3 reduction (litres removed: 0.584 L):

Projected dehumidifier consumption: 173.518 Wh

-- VERDICT --

Dehumidifier consumes 567.008 Wh less
```

5.3 Experimental Findings

The study revealed the following key findings:

- Dehumidifiers can save multiple hundreds of watt-hours every day during fall, winter, and early spring.
- For a room volume of 210.0 m^3, the thermal heat loss at 12.0 °C was found to be 740.53 Wh after accounting for energy conversion with 0.95 efficiency.
- The reduction in humidity was 14% rH, equivalent to 0.584 L of water removed.

5.4 Comparison: Airing vs Dehumidifier

Upon comparing the energy consumption for a 14% rH reduction in 210.0 m^3, the projected dehumidifier consumption was 173.518 Wh.

5.5 Verdict

The use of a dehumidifier resulted in a net energy savings of 567.008 Wh compared to traditional airing methods. This finding further underscores the efficiency and effectiveness of using dehumidifiers, particularly in colder seasons like fall, winter, and early spring.

6 Identified Improvements and Need for Data

More Temperature measurements are needed to verify the long term efficiency of dehumidifiers. One potential improvement would be the prediction of energy savings & losses using only the buildings average air temperature and the regions historical average outside temperature. This prediction engine would be a valuable and could be made available to the public as an online service. People all around the globe would be able to use this tool in order to determine the energy and air quality impact a dehumidifying system could make in their lives.

7 Python 3 Code

Below is the Python code used to calculate energy loss during building ventilation. The energy loss is compared to a dehumidifier's energy consumption. This code is located in the *main.py* file within the *src/python* directory of the project. All example values used are based on real world measurements and sensible estimates.

```
#!/bin/python
import math
###
# Purpose: This program is designed to calculate the effects of ventilating a building and compares the
    energy loss to that of a dehumidifier
###
###
# Specify Calculation Variables
###
# Ventilation variables
duration_of_ventilation_in_minutes = 10
number_of_ventilations_per_day = 3
# Building variables
building_area_in_square_meters = 84 # float
building_height_in_meters = 2.5 # float
building_volume_in_cubic_meters = building_area_in_square_meters*building_height_in_meters# float
# Temperature variables
#outside_temperature_in_degrees = 11.2 # TODO USE AS PREDICTION INPUT VALUE
inside_temperature_at_114_centimeters_room_elevation_in_degrees = 22.4
# measured_temperature_after_duration_at_114_centimeters_room_elevation_in_degrees = 20.4
measured_temperature_after_duration_at_114_centimeters_room_elevation_in_degrees = 18.4 # last measured
    difference was 4 degrees
specific_heat_capacity_of_air_in_joules_kilogram=1005
# Humidity variables
inside_humidity_at_114_centimeters_elevation_in_percentage_relative_humidity = 68
#outside_humidity_in_percentage_relative_humidity = 92 # WARNING THIS SENSOR IS NOT WORKING CORRECTLY
measured_humidity_after_duration_in_percentage_relative_humidity = 54
# Dehumidifier variables
# Floor area: ~80 m^2
# dehumidifier_wh_per_litre_removed=180.0 # https://www.amazon.de/dp/B096SQ1Q5K 30m^2
dehumidifier_wh_per_litre_removed=297.0 # https://www.amazon.de/dp/B096SQ1Q5K 80m^2
# dehumidifier_wh_per_litre_removed=324.0 # https://www.amazon.de/dp/B082FWPWCR 70m^2
# Heater Efficiency
# Efficiency range is est. 85-95%:
# hydronic_radiant_efficiency = 0.85
hydronic_radiant_efficiency = 0.95

###
# Define functions
###
# Returns the energy lost due to loss of heat energy in joules
def calc_thermal_energy_loss_in_joules(specific_heat_cap_joule_kilogram: float=1005.0,
    volume_in_cubic_meters: float=0, temperature_difference: float=0):
    energy_loss_in_joules = specific_heat_cap_joule_kilogram * volume_in_cubic_meters *
        temperature_difference
    return energy_loss_in_joules

# Converts joules to watts
def calc_joules_to_watt_hours(joules):
    return joules/3600

def calc_fast_absolute_humidity_in_gram_cubic_meter(rh,temperature):
    """
    Calculate the fast approximation of absolute humidity using Antoine equation.

    Parameters:
        rh (float): Relative Humidity in percentage
```

```
        temperature (float): Temperature in Celsius

    Returns:
        float: Absolute humidity in g/m^3
    """
    # Antoine equation constants for water
    A_water = 8.07131
    B_water = 1730.63
    C_water = 233.426
    # Calculate vapor pressure using Antoine equation (in mmHg)
    P_mmHg = 10**(A_water - (B_water / (temperature + C_water)))
    # Convert vapor pressure to Pascals
    P_Pa = P_mmHg * 133.322
    # Calculate the partial pressure of water vapor (in Pa)
    partial_pressure = rh / 100 * P_Pa
    # Convert temperature to Kelvin
    temperature_K = temperature + 273.15
    # Universal Gas constant (R) in J/(mol*K)
    R = 8.314
    # Molar mass of water in kg/mol
    M_water = 18.01528 / 1000
    # Calculate absolute humidity in g/m^3
    absolute_humidity = (partial_pressure * M_water) / (R * temperature_K) * 1e3
    return absolute_humidity

def calc_accurate_absolute_humidity_in_gram_cubic_meter(rh,temperature):
    """
    Calculate the accurate absolute humidity using Wagner and Pruss equation.

    Parameters:
        rh (float): Relative Humidity in percentage
        temperature (float): Temperature in Celsius

    Returns:
        float: Absolute humidity in g/m^3
    """
    # Specific gas constant for water vapor in J/(kg*K)
    Rw = 461.5
    # Critical pressure for water in Pa
    Pc = 22.064e6
    # Critical temperature for water in K
    Tc = 647.096
    # Empirical constants for Wagner and Pruss equation
    a1 = -7.85951783
    a2 = 1.84408259
    a3 = -11.7866497
    a4 = 22.6807411
    a5 = -15.9618719
    a6 = 1.80122502
    # Temperature in Kelvin
    T = temperature + 273.15
    # tau for Wagner and Pruss equation
    tau = 1 - T / Tc
    # Saturation vapor pressure using Wagner and Pruss equation in Pa
    Ps = Pc * math.exp(Tc / T * (a1 * tau + a2 * tau**1.5 + a3 * tau**3 + a4 * tau**3.5 + a5 * tau**4 + a6 *
        tau**7.5))
    # Calculate absolute humidity in g/m^3
    AH = rh * Ps / (Rw * T) * 10
    return AH

def rh_to_ah_in_liters(rh, temperature, volume):
    """
    Convert relative humidity to absolute humidity in liters for a given volume.
```

```
    Parameters:
        rh (float): Relative Humidity in percentage
        temperature (float): Temperature in Celsius
        volume (float): Volume in cubic meters

    Returns:
        float: Absolute humidity in liters
    """
    # Calculate accurate absolute humidity in g/m^3
    ah_gram_per_cubic_meter = calc_accurate_absolute_humidity_in_gram_cubic_meter(rh, temperature)
    # Calculate the total water content in the given volume in grams
    total_water_content_grams = ah_gram_per_cubic_meter * volume
    # Convert the total water content to liters (since 1 liter of water is 1000 grams)
    total_water_content_liters = total_water_content_grams / 1000
    return total_water_content_liters

# Calculates various effects of velntilating a building
def simulate_and_compare(duration: float=0.0,
                volume: float=0.0,
                inside_temp: float=0.0,
                measured_temp: float=0.0,
                inside_hum_rh: float=0,
                measured_hum_rh: float=0,
                ):
    heat_loss_wh=0
    hum_reduction=0
    # Volume
    print(f"--␣ENERGY␣LOSS␣CALCULATION␣--\n\n\tRoom␣volume:\t\t\t{volume}␣m^3")
    # Calculate Energy loss due to temperature difference for building volume
    if measured_temp<inside_temp: # If temperature fell during ventilation
        temperature_difference=(inside_temp-measured_temp)*number_of_ventilations_per_day
        heat_loss_in_jules=calc_thermal_energy_loss_in_joules(
            specific_heat_capacity_of_air_in_joules_kilogram,
                                                    building_volume_in_cubic_meters,
                                                    temperature_difference)
        heat_loss_wh=calc_joules_to_watt_hours(heat_loss_in_jules)
        print(f"\n\tHeat␣loss({temperature_difference:.1f}␣C):\t\t{heat_loss_in_jules:.2f}␣Joules\n\t\t\t\t\t\t
            \t\t{heat_loss_wh:.2f}␣Wh")
        # Compensate for hydronic radiant floor heating energy conversion losses
        heat_loss_wh = heat_loss_wh / hydronic_radiant_efficiency
        print(f"\n\tHeat␣loss({temperature_difference:.1f}␣C)␣after␣energy␣conversion␣compensation␣({
            hydronic_radiant_efficiency}␣efficiency):\t\t{heat_loss_wh:.2f}␣Wh")
    else:
        print(f"\n\tHeat␣loss:\t\t\t\tNONE")
    # Calculate humidity difference
    if measured_hum_rh<inside_hum_rh: # If humidity fell
        humidity_difference=inside_hum_rh-measured_hum_rh
        print(f"\n\tHumidity␣reduction:\t\t{humidity_difference}%\trH")
        hum_reduction=humidity_difference
    else:
        print(f"\n\tHumidity␣reduction:\t\tNONE")
    # Compare energy loss to energy consumption of dehumidifier
    if hum_reduction>0 and heat_loss_wh>0:
        # Project energy loss
        litres_of_water_removed=rh_to_ah_in_liters(hum_reduction,
            inside_temperature_at_114_centimeters_room_elevation_in_degrees,volume)
        print(f"\n--␣COMPARISON␣--\n\nComparing␣airing␣to␣dehumidifier␣for␣{hum_reduction}␣%␣rH␣in␣{volume}␣m
            ^3␣reduction␣(litres␣removed:␣{litres_of_water_removed:.3f}␣L):")
        projected_dehumidifier_energy_loss_wh=dehumidifier_wh_per_litre_removed*litres_of_water_removed
        print(f"\nProjected␣dehumidifier␣consumption:\t{projected_dehumidifier_energy_loss_wh:.3f}␣Wh")
        # Compare efficiency
        energy_loss_delta_ventilation_to_dehum=heat_loss_wh-projected_dehumidifier_energy_loss_wh
```

```
       print("\n--␣VERDICT␣--\n")
       if energy_loss_delta_ventilation_to_dehum>0:
          print(f"Dehumidifier␣consumes␣{abs(energy_loss_delta_ventilation_to_dehum):.3f}␣Wh␣less")
       else:
          print(f"Dehumidifier␣consumes␣{abs(energy_loss_delta_ventilation_to_dehum):.3f}␣Wh␣more")
###
# Run calculation
###
simulate_and_compare(duration_of_ventilation_in_minutes,
          building_volume_in_cubic_meters,
          inside_temperature_at_114_centimeters_room_elevation_in_degrees,
          measured_temperature_after_duration_at_114_centimeters_room_elevation_in_degrees,
          inside_humidity_at_114_centimeters_elevation_in_percentage_relative_humidity,
          measured_humidity_after_duration_in_percentage_relative_humidity)
```