UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Mechanical Design 444
System Simulation Program Assignment

Quasi-steady simulation of some aspects of a yogurt incubator

Danie Els

Dept of Mechanical & Mechatronic Engineering
Stellenbosch University

2018
Version 1.2

Contents

1. Background

The objective of this project is for students to develop an appreciation of the value of quasi-steady one-dimensional simulations in design. A further objective is to integrate the knowledge gained in Heat Transfer, Fluid Mechanics, Mechatronics, Numerical Methods and Computer Programming. The objective of this type of project in engineering practice would be to optimise the system design, to determine control parameters and to quantify time-dependent behaviour.

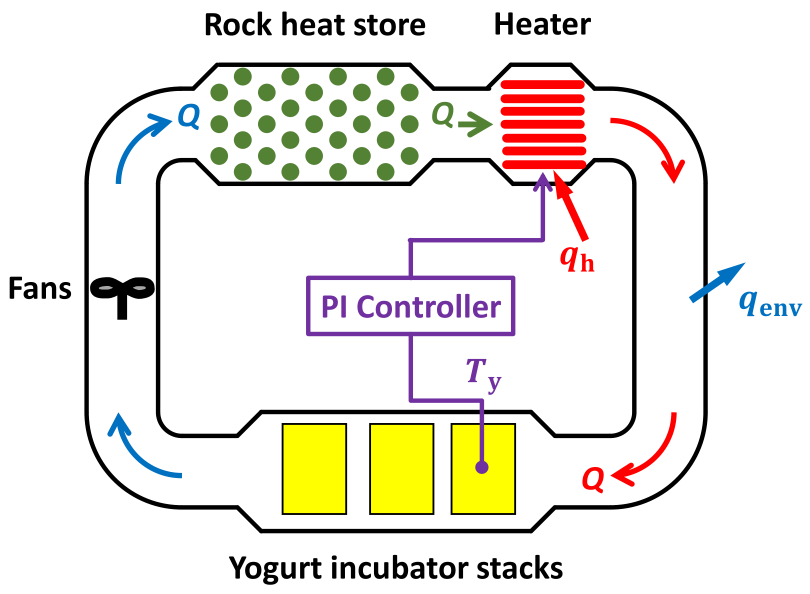

Figure 1: Schematic Layout

2. Assignment

Write a SciLab program that calculates the time-dependent temperature of the air, yogurt and heat store in an incubator during the heating-up phase. Figure 1 shows a schematic layout of the system that must be simulated. Some aspects of the system must be modified, as describe below, to achieve the academic objectives:

- The incubator is divided in 3 control volumes in series that are made up of small yogurt containers that are stacked on trays.
- Assume that all the yogurt containers in a control volume are at the same average temperature, but that the temperatures can vary among the different control volumes. One-dimensional heat transfer between the yogurt containers en the air is computed. Use the average temperature per control volume.
- The heat store is divided into 3 control volumes. One-dimensional heat transfer between the heat store material en the air is computed. Use the average temperature per control volume.
- The parameters of the control volumes of the yogurt stacks and heat store, and the initial temperatures will be given in the simulation program's input file. The

temperature history of the control volumes must be computed. This is one of the major objectives of the program.

- The thermal inertia of the air is neglected. The air temperature can rise or fall through the heat store. It rises through the electrical heater. It falls due to losses to the environment before the stacks, and finally it falls through the yogurt stacks. Other changes in temperature along the flow path are neglected.
- The air temperature after the electrical heater must be limited by a PI controller to a set point given in the input file. The PI controller sets the power of the electrical heater.
- Heat loss to the environment is modelled as a simple convection loss. The loss is proportional to the difference in air temperature before the stacks and the environment. The proportionality constant will be given in the input file.
- Changes in the air density due to pressure changes can be neglected. A reference condition, with air density, will be given in the input file.
- The fixed fan speed will be given in the input file. The simulation programme must use the fan laws and calculate the air flow rate.
- Only the pressure drops over the yogurt stacks and through the heat store should be taken into account. Other pressure drops along the flow path, as a simplification, can be accounted for through a factor applied to the total pressure drop across the heat store and yogurt stacks.
- Assume that the air volume flow rate through the system is constant. Changes in volume flow rate due to temperature and pressure changes can be neglected. A reference condition, with air density, will be given in the input file.
- The effects of humidity can be neglected.

NB: *Each student must write his/her own simulation program. This assignment is not done in groups.*

3. Deliverables

08:00 on Tuesday 25 September at M&M Reception:

Hand calculation (or by MathCad or SMathStudio) of one "snapshot" of the system as shown in figure 2:

i. Select initial values for fan speed N and temperatures in the heat store T_{b_i} and yogurt stacks T_{y_i}.
ii. Select a volume flow rate Q and initial system temperatures T_i
iii. Calculate the temperature changes through the system for a selected volume flow rate Q and initial heat store T_{b_i} and yogurt stacks T_{y_i}. Temperatures can be iterated until $T_6 = T_1$.
iv. Calculate fan pressure increase ΔP_{fan} and system losses ΔP_{sys} for a selected speed N, volume flow rate Q and system temperatures T_i obtained in (iii).
v. System can be iterated by selecting a new Q and repeating from (iii) until $\Delta P_{\text{fan}} + \Delta P_{\text{sys}} \approx 0$.

08:00 on Tuesday 2 October on SunLearn

1. An electronic PDF report containing:

(a) Title page

(b) Plagiarism declaration

(c) Graph of the yogurt stacks temperatures against time. Import output data into Excel to draw the graph, do not program graphics into your main program.

(d) Graph of heat store control volume temperatures against time

(e) The final source code

2. The complete source code of the computer programme in a zipped file.

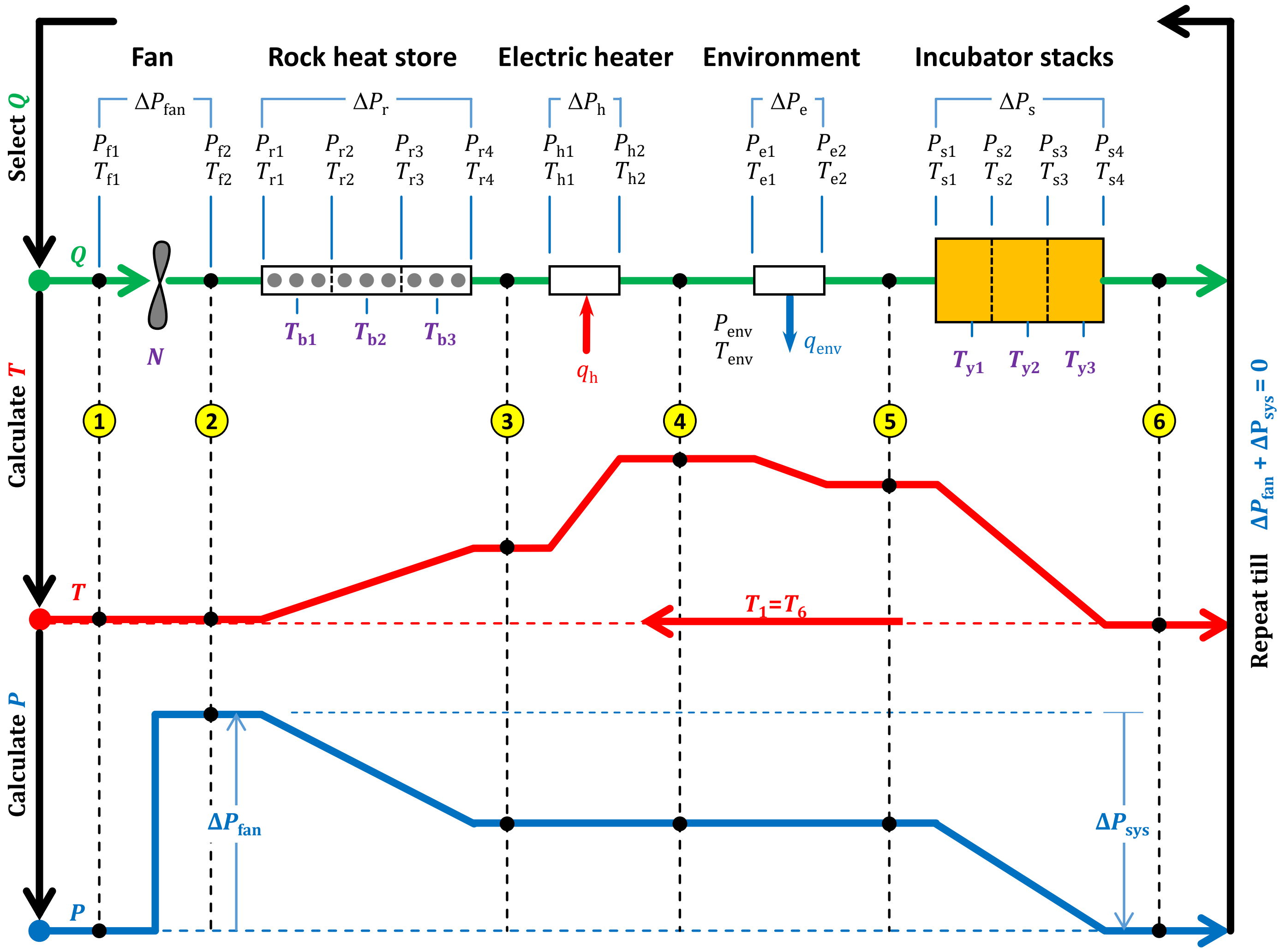

Figure 2: System Snapshot

4. Preliminary marks allocation

The final marks allocation depends on the help given during the contact sessions in class. The break down given below is an indication of the factors that will be taken into account.

- Programming techniques:
  - Sensible variable names
  - Sensible use of subprograms
  - Sensible use of comments
  - Readability of the code (indents, spacing, etc)
  - Sources of correlations and empirical equations given
  - Reads the prescribed input file
  - Writes the prescribed output file
- System Modelling:
  - Calculation of fan pressure rise
  - Determining the intersection of the fan curve and the system curve
  - Proportional control
  - Calculation of the heat balances in the control volumes
- Implementation of the integration procedure:
  - Temperatures Integral controllers
- Program runs and gives correct answers:

5. Programming guidelines

Build your program up out of subprograms that can easily be tested. Test each subprogram as soon as it is written. Use the following structure:

- The main program script must contain the main program loop and must call the subprograms that handle the input and output. The program IS NOT ALLOWED to require interactive inputs. It must further show clearly when the program has completed the run.
- Subprogram `Input_xx.sci` that handles the input will be provided.
- Write a function `DP_Fan()` that calculates the pressure rise across the fan, using the curve fit (e.g. a cubic polynomial) through the fan characteristic and the fan laws. The arguments for the function should include volume flow rate, rotational speed and fluid density (or temperature). Remember to prevent the programme from extrapolating beyond the start or end points of the curve fit. Test the subprogram by comparing its results with the hand calculation.
- Also write a function `DP_Sys()` that calculates the system resistance for a given combination of density, volume flow rate and temperature. Test the subprogram by comparing its results with the hand calculations.
- Write a function `Cycle_Temp()`, calculates the correct temperatures through the system for a given flow rate, heater setting and rock bed and yogurt temperatures .

- Write a function `Workpoint()` that uses the previous three subprograms to find the operating point of the fan for a given fluid density (or temperature) and a given fan speed. Us a systematic iterative method. The maximum and minimum volume flow rate on the fan characteristic, adjusted according to the fan laws, must bound the search region. Test the subprogram by comparing its results with the hand calculations.
- Write a subprogram `Derivatives()` that determines the rate of change of the temperatures in the control volumes and the other variables that have to be integrated, for a given combination of the conditions that occur at an instant (i.e. the state variables). Test this subprogram by comparing its results with the hand calculation. NB: There must only be one subprogram that calculates the time-derivatives of all the variables.
- Now write the main program that handles the numerical integration of the temperatures, as well as the integral controllers. Test the main programme by fixing one or more of the variables that should be integrated and checking that the program reacts sensibly. If unexpected or unreasonable results are given, use the debugger to check the values of the most important variables at each stage of the program execution, or write the values out to the screen ("debug by printing").

NB: *An example input file will be provided.*

The following requirements MUST be met:

- The main programme must be at the top of the printout, followed by `DP_Fan()`, `DP_Sys()`, `Cycle_Temp()`, `Workpoint()` and `Derivatives()` in that order. Subprograms that you wrote yourself must be placed after these. Subprograms given to you by the lecturer (e.g. for air properties, etc.) must be placed last.
- Never use magic numbers in a program. They are numbers from external sources that are not explained in the source code. It is often the easiest to use a descriptive variable name and to assign the value to that name. Also, the source of the number should be given in a comment statement.
- Work consistently only in SI base units, e.g. Watt (and not kW), Kelvin (and not °C). If values in other units are read from or written to files, do the conversion where the input/output is generated.
- Always place a limit on the number of iterations that a loop (e.g. when searching for a root) may execute. NB: Remember to check after the loop whether the iteration limit kicked in and, if it did, output a warning message.
- Convergence limits in iteration loops or search routines should be non-dimensional. It must in any case be chosen to work for any set of input conditions for the programme.
- Remember to close the output file, otherwise all the may not be saved.
- The program must be written in SciLab. It must be able to run on a PC using the compilers available in FIRGA.
- Loops and IF-blocks must be indented.
- The student's initials, surname and student number must be given as close as possible to the top of the first page of source code.
- The source code must be zipped into a single file ("`studenteno.zip`").

- The program must read all the input values from a data file and the results must be written to an output file in the format that will be given during the lectures. The program must read the data from “`Input_xx.sci`” in default folder. Do not use “`A:\Input.sci`” or something similar. An example input file will be given on SUNLearn. The program must write the output to “`Output.txt`” in the default folder. The prescribed format must be adhered to exactly.
- Sufficient comments must be given in the program to make its structure clear. Do not overcomment.
- Remove all commented out source code before handing in.
- Any variable name whose meaning is not obvious, must be explained in comments where it is declared or used for the first time.
- Do not break the program up into an unnecessary number of subprograms. A separate subprogram that is only a few lines long is only sensible if it is called more than once in different parts of other subprograms.

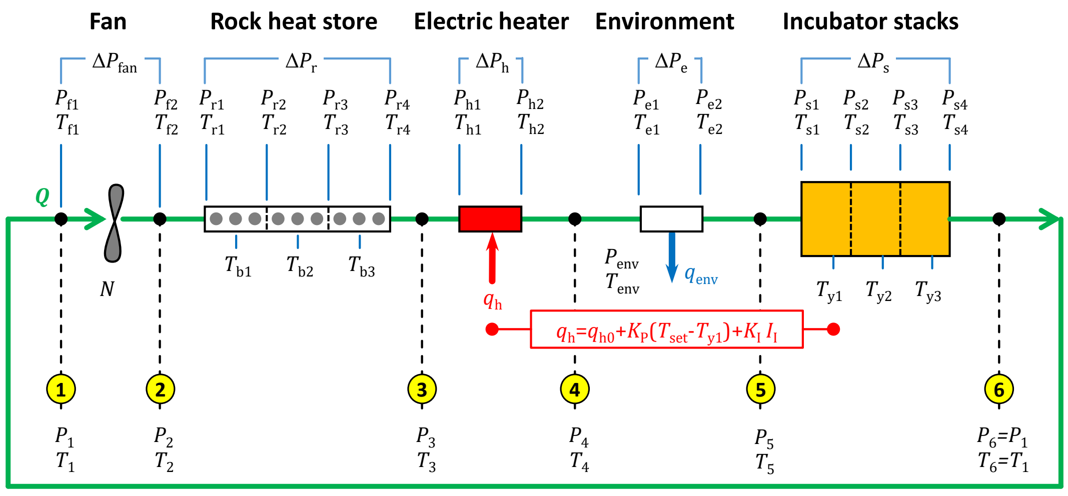

Figure 3: System Diagram

6. Program frameworks

6.1. Main program: `Main()`

1. Read input data, `Input_xx.sci`, ONLY in the main program.
2. Calculate the constants parameters that do not change in the remainder of the programme.
3. Assign initial values to all parameters determined through integration.

$$X_0 = [T_{b1},\ T_{b2},\ T_{b3},\ T_{y1},\ T_{y2},\ T_{y3},\ I_I]$$

with T_{bi} the temperatures in the heat store and T_{yi} the temperatures in the yogurt stacks.

4. "`Open`" the output file.
5. Loop: repeat for each step until $t > t_{\max}$:
   i. Use `Workpoint()` to balance the system at the initial or updated conditions.
   ii. Write the required output, i.e. the prescribed parameters that must be written to a file for each time step. During the previous time step, a complete "snapshot" of all the variables would have been calculated based on a real state. This is therefore the best place to write the output.
   iii. Update the state by means of 4th order Runge-Kutta

$$\begin{aligned}
[dX_1] &= \texttt{Derivatives}(t, && X_t, && \cdots), && K_1 = \Delta t \cdot dX_1\\
[dX_2] &= \texttt{Derivatives}(t{+}\Delta t/2, && X_t + K_1/2, && \cdots), && K_2 = \Delta t \cdot dX_2\\
[dX_3] &= \texttt{Derivatives}(t{+}\Delta t/2, && X_t + K_2/2, && \cdots), && K_3 = \Delta t \cdot dX_3\\
[dX_4] &= \texttt{Derivatives}(t{+}\Delta t, && X_t + K_3, && \cdots), && K_4 = \Delta t \cdot dX_4
\end{aligned}$$

$$X_{t+\Delta t} = X_t + (K_1 + 2K_2 + 2K_3 + K_4)/6$$
$$t = t + \Delta t$$

6. “`Close`” the output file.

6.2. Function: `[DX] = Derivatives(t, X, ...)`

The time is t for time dependent functions and X the state variables

$$X = [T_{b1},\ T_{b2},\ T_{b3},\ T_{y1},\ T_{y2},\ T_{y3},\ I_I]$$

as called by Runge-Kutta, e.g. at $t{+}\Delta t/2 \rightarrow$ and $X_t{+}K_1/2 \rightarrow X$, etc.

1. Call `[Q,T]=Workpoint_Q(t, X, T, N, ...)  to find` Q `and` T.

2. Calculate rates of change.

$$DX = \left[\frac{\mathrm{d}T_{b1}}{\mathrm{d}t},\ \frac{\mathrm{d}T_{b2}}{\mathrm{d}t},\ \frac{\mathrm{d}T_{b3}}{\mathrm{d}t},\ \frac{\mathrm{d}T_{y1}}{\mathrm{d}t},\ \frac{\mathrm{d}T_{y2}}{\mathrm{d}t},\ \frac{\mathrm{d}T_{y3}}{\mathrm{d}t},\ \frac{\mathrm{d}I_I}{\mathrm{d}t}\right]$$

3. Return `DX` to the main program

6.3. Function: `[Q,T, ...] = Workpoint(t, X, T, N, ...)`

For the current fan speed N, calculate the minimum and maximum volume flow rates ($Q_{\min}$ and $Q_{\max}$) that the curve fit is valid for, using the fan laws.

1. Use $Q_{\min}$ and $Q_{\max}$ as lower and upper bounds for a zero finding search routine:
   i. Increment $i = i + 1$
   ii. Estimate Q_i where $Q_{\min} \leq Q_i \leq Q_{\max}$, according to the selected algorithm.
   iii. Call `[T]=Cycle_Temp(t, X, T, Q`$_i$`, ...)` to balance the system temperatures at Q_i.
   iv. Call `[`Δ`P`$_{sys}$`]=DP_Sys(T, Q`$_i$`, ...)` and `[`Δ`P`$_{fan}$`]=DP_Fan(T, Q`$_i$`, N, ...)`.
   v. Test whether the difference in pressure drop is within the convergence limit ε,

      $$|\Delta P_{\text{fan}} + \Delta P_{\text{sys}}| < \varepsilon$$

      and whether the maximum number of iterations has been reached, $i \geq i_{\max}$. If not, return to (i).
2. If the loop ended because the iteration limit was reached, output an error message.
3. Return the calculated volume flow rate $Q = Q_i$ and system temperature T at Q_i.

6.4. Function: `[T, ...]=Cycle_Temp(t, X, T, Q, ...)`

The aim of the function is to obtain the instantaneous system state (temperatures) at a give time step:

The time is t for time dependent functions and X the state variables

$$X = [T_{b1},\ T_{b2},\ T_{b3},\ T_{y1},\ T_{y2},\ T_{y3},\ I_I]$$

1. Select an initial value for the temperature stations through the system (the previous iteration's values are good enough)

   $$T = [T_1, T_2, \dots, T_n]$$

   Compute the electrical heater output at this condition

   $$q_h = q_{h0} + K_P \cdot (T_{\text{set}} - T_{y1}) + K_I \cdot I_{\text{I}}$$

   with $0 \le q_h \le q_{h\text{max}}$
2. Loop:
   i. Calculate $T = [T_1, T_2, \dots, T_n]$ at T_{w1} and Q
   ii. When $|(T_1 - T_n)/T_n| < \varepsilon$ the loop has converged, otherwise set $T_1 = T_n$ and repeat

6.5. Function: `[DP, ...] = DP_Fan(T, Q, N, ...)`

Using the fan laws: scale the given volume flow rate to the rotational speed N and density of the fitted fan characteristic.

1. Calculate the pressure rise using the polynomial fit.
2. Using the fan laws: scale the pressure rise obtained from the curve fit to the actual rotational speed and density at T and Q.
3. Return the calculated pressure rise ΔP_{fan}.

6.6. Function: `[DP, ...] = DP_Sys(T, Q, ...)`

1. Calculate the pressure drop across the rock heat store, ΔP_{r} at T and Q.
2. Calculate the pressure drop across the yogurt stacks, ΔP_{s} at T and Q.
3. Return the total pressure loss, $\Delta P_{\text{sys}} = \Delta P_{\text{r}} + \Delta P_{\text{s}}$.

7. Program outputs

Write a file "`Output.txt`" that contains the following data. The first 3 comment lines need not be included. You can use the data to draw graphs, if necessary.

```
%  t     T1      T2   ..  Tb1     Tb2     Tb3     Ty1     Ty2     Ty3       II    qh
% [s]   [K]     [K]  ..  [K]     [K]     [K]     [K]     [K]     [K]     [K.s]  [W]
%----  ------  ------   ------  ------  ------  ------  ------  ------  ----
---  -----
  0  348.04  349.04 .. 348.15  348.15  348.15  348.15  348.15  348.15    0.000  500.0
 200  348.06  349.06 .. 348.17  348.15  348.15  348.17  348.15  348.15   19.875  519.8
 400  348.11  349.11 .. 348.22  348.16  348.15  348.22  348.16  348.15   67.736  565.6
 600  348.16  349.15 .. 348.27  348.17  348.15  348.27  348.17  348.15  117.178  608.1
  :      :       :   ..   :       :       :       :       :       :       :        :
  :      :       :   ..   :       :       :       :       :       :       :        :
  :      :       :   ..   :       :       :       :       :       :       :        :
```

This table is NOT a sample output! It is just random numbers.