Optimal One-gas Model Furnace

January 11, 2021

A problem that most players face in Stationeers is automating a furnace for smelting alloys, where the issue is reaching a specific pressure target and temperature target. With a naïve setup, we combust fuel to raise the pressure and temperature, cool the system if need be, then remove pressure until both are in range. A better, more precise setup is demanded for more difficult alloys.

1 Motivation

The immediate question is: given the initial conditions of a furnace, can I add a specifically formated mixture of gas in order to reach those targets without guess-work? The answer is yes: though we may need to remove some gas first, there is such a perfect mixture. However, embedded within this problem of temperature and pressure is an issue of specific heat. A volume M comprised of a mix of the seven gasses in the game may have a particular state (parameters pressure, temperature and moles per gas), but there are many different gas compositions such that the specific heat s_M will satisfy these conditions. In-fact, all such possible compositions lie within an $\mathbb{R}^7$ subspace, whose support is on what gasses exist in the volume.

$$s_M = \vec{n_M} \cdot \vec{c} = (n_M(g_0), \cdots, n_M(g_6)) \cdot (c_{g_0}, \cdots, c_{g_6})$$
$$\vec{n_{Mg}} = n_M(g). \qquad \text{(The } g^{\text{th}} \text{ component)}$$

The dimension of this subspace depends on the number of gasses in the system, thus intentionally restricting the gasses (as per filtering) allows us to reduce the dimension as we see fit. This, along with the practical purposes of only using one gas (namely, removing gases which have useful applications outside pressurizing a volume) is why I've decided to implement a one-gas model furnace. In such a system, the dimension of the specific heat subspace is reduced to a single dimension, and calculation of the energy from combining two compositions is made significantly simpler. If we fix g^* the sole gas of the system, then the specific heat s_M of a volume M becomes:

$$n_M = \sum_{g \in G} n_M(g) = n_M(g^*), \quad \vec{n_{Mg}} = \begin{cases} n_M, & \text{if } g = g^* \\ 0, & \text{if } g \neq g^* \end{cases} \Rightarrow s_M = \vec{n_M} \cdot \vec{c} = n_M c_{g^*}.$$

Furthermore, equation defining the result of combining two volumes with different temperatures becomes significantly simpler, where specific heat is replaced simply by moles:

$$t_C s_C = t_A s_A + t_B t_B \;\Rightarrow\; t_C n_C c_{g^*} = t_A n_A c_{g^*} + t_B n_B c_{g^*} \;\Rightarrow\; t_C n_C = t_A n_A + t_B n_B.$$

2 Implementation and optimization

Now we move into the specifics of implementing such a system in the game, involving furnaces and pipe networks. Considering a system of only one gas with a hot source H of temperature t_H and a cold source C of temperature t_C, we can calculate an optimal formulation for bringing a furnace F (with volume v_F, initial pressure p_F, initial temperature t_F, and initial moles $n_F = \frac{p_F v_F}{R t_T}$, where R is the ideal gas constant) to a desired pressure p_T and temperature t_T. This is accomplished by removing an amount n_R from the furnace and/or adding an amount n_I at a specific temperature t_I, where I is composed from amounts n_H and n_C from the H and C sources. Several gas-law derived equations constrain this process:

$$t_T n_T = t_F(n_F - n_R) + t_I n_I \tag{1}$$

$$n_T = n_F - n_R + n_I \tag{2}$$

$$t_I n_I = t_H n_H + t_C n_I \tag{3}$$

Where n_R, t_I and $n_I = n_H + n_C$ are to be determined. Note that each term in this system obeys some constraints:

$$t_C < t_M < t_H, \quad 0 \leq n_M, \quad 0 \leq n_R \leq n_F.$$

Solving Equation 1 for n_I provides a surface bounded in two dimensions by $0 \leq n_R \leq n_F$ and $t_C \leq t_I \leq t_H$, but where $0 \leq n_I$ is potentially unbounded.

$$(n_R, t_I, f): \quad f(n_R, t_I) = n_I = \frac{t_T n_T + t_F(n_R - n_F)}{t_I}.$$

Equation 2 further restricts potential solutions, but given a satisfactory amount to remove n_R provides a particular solution for the amount to add: $n_I = n_T - n_F + n_R$. Then we instead solve Equation 1 for t_I, and as a result these restricted solutions lie within a curve embedded within the surface.

$$(n_R, h, g): \quad g(n_R) = n_I = n_T - n_F + n_R, \quad h(n_R) = t_I = \frac{t_T n_T - t_F(n_F - n_R)}{n_T - n_F + n_R}.$$

With respect to t_I, h is monotone decreasing, but monotone increasing with respect to n_R. Thus minimizing with respect to n_I in the domain is a matter of minimizing n_R and maximizing t_I. However, $0 \leq n_R$ is naïve, and Equation 3 imparts a narrower lower-bound.

$$t_C n_I \leq t_I n_I$$
$$n_F(n_F - t_C) - n_T(t_T - t_C) \leq n_R(t_F - t_C)$$
$$n_{R_C} = n_F - \frac{n_T(t_T - t_C)}{t_F - t_C} \leq n_R \tag{4}$$

$$t_I n_I \leq t_H n_I$$
$$t_T n_T - t_F(n_F - n_R) \leq t_H(n_T - n_F + n_R)$$
$$n_{R_H} = n_F - \frac{n_T(t_H - t_T)}{t_H - t_F} \leq n_R \tag{5}$$

Thus $\max(n_{R_C}, n_{R_H}) \leq n_R$. In-fact, the curve crosses a hyperplane boundary at a point (n_R, t_I, n_I) which minimizes n_I and for which $n_R = \max(0, n_{R_C}, n_{R_H})$. We then easily solve for the remaining coordinates.

In the end, finding the n_I minimizing point (n_R, t_I, n_I):

$$n_R = \max(0, n_{R_C}, n_{R_H}) \tag{6}$$

$$t_I = \frac{t_T n_T - t_F(n_F - n_R)}{n_T - n_F + n_R} \tag{7}$$

$$n_I = n_T - n_F + n_R. \tag{8}$$

Which hyperplane the curve crosses corresponds to a certain procedure of the gas mixer that in the end we are making: the curve either:

- Crosses $n_R = 0$: a mixture of H and C is added, or
- Crosses $t_I = t_H$: some volume of F is removed and only H is added, or
- Crosses $t_I = t_C$: some volume of F volume is removed and only C is added.

Lastly, since n_I is composed of moles from H and C, we also need to calculate n_H and n_C, which is trivial given that $n_I = n_H + n_C$ and $t_I n_I = t_H n_H + t_C n_C$.

$$\begin{aligned} t_I &= n_H + n_C \\ \Rightarrow n_C &= n_I - n_H. \end{aligned} \tag{9}$$

$$\begin{aligned} t_I n_I &= t_H n_H + t_C n_C \\ &= t_H n_H + t_C(n_I - n_H) \\ &= t_C n_I + n_H(t_H - t_C) \\ \Rightarrow n_H &= \frac{t_I n_I - t_C n_I}{t_H - t_C}. \end{aligned} \tag{10}$$

And what we are left with is precisely the minimum number of moles n_R to remove from the furnace initially, and (with respect to n_R) n_H and n_C the precise numbers of moles to add from the H and C sources respectively, such that F will achieve pressure p_T and temperature t_T exactly. All that is left is game implementation.

3 Game implementation with MIPS

I've decided to implement this entire mixing program along with controls onto a single IC housing, and implementation details will depend on what devices we specifically need. I've chosen to use the following device inputs:

- the furnace itself, for reading its state,
- two pipe analyzers; one for H and one for C,
- one volume pump for removing the n_R initial furnace moles, and
- two volume pumps; one for H and one for C, both into I.

With the volume pumps instead of a gas mixer we will be able to scale how quickly we add amounts n_H and n_C and with more precision than with a gas mixer which moves moles at a specific amount per tick, and we don't need to calculate the correct ratio that a gas mixer needs. And instead of having a pipe analyzer on I to count up to the correct number of moles, we have the H volume pump on until it's number of moles is less-than or equal to n_{H_0} (the initial number of moles in H) minus n_H, and likewise for C.

For clarity in describing the steps of the MIPS program, every uniquely labeled term of an equation will have a unique register in the code described below, but optimizations will need to be made in reusing registers in order to create a final MIPS program.

3.1 Utility functions

For emptying the input pipe I into the furnace

```
fillFurnace:
yield
s Furnace SettingInput 100
l x IAnalyzer TotalMoles
brgtz -3
s Furnace SettingInput 0
j ra
```

And for emptying the furnace into the filtration system, which filters all CO_2 back into the input pipe.

```
emptyFurnace:
yield
s Furnace SettingOutput 100
l x Furnace TotalMoles
brgtz -3
s Furnace SettingOutput 0
j ra
```

3.2 Acquiring p_T and t_T

Since all six of our device pins are being used already, we have no way to directly set the target pressure and temperature for our program, or even any way to directly start it. Instead we can use a messaging system, where another program (a control program) sends messages which set these

values or start the program. The `db` register (that is, the value corresponding to the IC housing) acts as a channel through which a "sender" can communicate to this one, the "receiver", and vice versa.

- If `db` is `0`, this program is ready to receive a message.
- If `db` is `-1`, this program is awaiting a follow-up value from a sender. New messages should not be sent.
- If `db` is `1/2`, a sender has sent the message that it wants to send a new target pressure/temperature.
- If `db` is `3`, a sender has sent the message for this program to start.

First we define the messages.

```
define MsgReadyMsg 0
define MsgReadyValue -1
define MsgPTarget 1
define MsgTTarget 2
define MsgStart 3
```

Then in the receiver (this program)

```
checkInput:
l x db Setting
brne x MsgPTarget 4 # receive pT
jal waitReceive
move pT x
jal calculateMolesTarget
brne x MsgTTarget 4 # receive tT
jal waitReceive
move tT x
jal calculateMolesTarget
bne x MsgStart main # received start command
# ...

waitReceive:
s db Setting MsgReadyValue
yield
l x db Setting
breq x MsgReadyValue -2
s db Setting MsgReadyMsg
j ra
```

And in the sender (the control program)

```
checkInput:
l x MixerIC Setting
bne x MsgReadyMsg ra # skip if mixer busy
l x PTargetDial Setting # pTarget
breq x pTarget 6
move pTarget x
mul x x 100
```

```
s PTargetDisplay Setting x
move y MsgPTarget
j waitSend
l x TTargetDial Setting # tTarget
breq x tTarget 6
move tTarget x
mul x x 20
s TTargetDisplay Setting x
move y MsgTTarget
j waitSend
l x StartButton Activate # start
breq x start 5
move start x
breqz start 3 # skip if changed to 0
move y MsgStart
s MixerIC Setting y
j ra

waitSend:
s MixerIC Setting y
s db Setting 1
yield
l y MixerIC Setting
brne y MsgReadyValue -2
s MixerIC Setting x
s db Setting 0
j ra
```

3.3 Calculating n_R, t_I, n_I, and n_H

Once the start message has been sent, we carry out the entire gas mixing procedure for the current state of F, H and C.

Equation 4

$$n_{R_C} = n_F - \frac{n_T(t_T - t_C)}{t_F - t_C} \leq n_R$$

Equation 5

$$n_{R_H} = n_F - \frac{n_T(t_H - t_T)}{t_H - t_F} \leq n_R$$

```
calculate:
sub nRC tT tC
mul nRC nT nRC
sub x tF tC
div nRC nRC x
sub nRC nF nRC # nRC=nF-nT(tT-tC)/(tF-tC)

sub nRH tT tH
mul nRH nT nRH
sub x tH tF
div nRH nRH x
sub nRH nF nRH # nRC=nF-nT(tH-tT)/(tH-tF)
```

Equation 6

$$n_R = \max(n_{R_C}, n_{R_H}, 0)$$

```
max nR nRC nRH
max nR nR 0 # nR=max(nRC,nRH,0)
```

Equation 8

$$n_I = n_T - n_F + n_R$$

```
sub nI nT nF
add nI nI nR # nI=nT-nF+nR
```

Equation 7

$$t_I = \frac{t_T n_T - t_F(n_F - n_R)}{n_I}$$

```
mul tI tT nT
sub x nF nR
mul x tF x
div tI tI nI # tI=(tT*nT-tF(nF-nR))/nI
```

Equation 10

$$n_H = \frac{t_I n_I - t_C n_I}{t_H - t_C}$$

```
mul nH tI nI
mul x tC nI
sub nH nH x
sub x tH tC
div nH nH x # nH=(tI*nI-tC*nI)/(tH-tC)
```

Equation 9

$$n_C = n_I - n_H$$

```
sub nC tI nH
```

3.4 Mixing I

With n_H and n_C saved we can now compose I by mixing H and C.

The full mixing loop

TODO: Scale the volume pump settings based on moles per liter and how many moles more need to be moved.

```
mix:
l x HAnalyzer TotalMoles
l y CAnalyzer TotalMoles
sub nH x nH # reduced moles in H target
sub nC y nC # reduced moles in C target
s HPump Setting 100 # TODO: Scale these in loop
s CPump Setting 100
mixLoop:
yield
s HPump On 1
s CPump On 1
l x HAnalyzer TotalMoles
l y CAnalyzer TotalMoles
sgt x x nH # current H > target H
sgt y y nC # current C > target C
brnez x 2 # skip if H not at target
s HPump On 0 # else turn H pump off
brnez y 2 # skip if C not at target
s CPump On 0 # else turn C pump off
and x x y
bnez x mixLoop # loop if either above target
```