Table of Contents


```
[p_sat,T_sat,rho_vap,h_w,~,c_p]=SatWater;
k.n_cell    = 15;
k.switch    = -1;
```

INTEGRATION PARAMETERS

```
%k.T          = 3600*24;
k.T          = 2000*1;
```

AMBIENT CONDITIONS

```
k.p_amb     = 1E05;             % Ambient pressure [Pa]
k.T_amb     = 10 + 273.15;      % Ambient temperature [K]
```

SOLID WASTE PARAMETERS

Molar mass of the waste elemental components [kg/kmol] in the combustable fraction

```
k.Mw_w       = [12.011;          % C
                1.0079;          % H
                32.06;           % S
                35.45;           % Cl
                14.007;          % N
                15.999;          % O
                18.998];         % F
k.nc_w       = length(k.Mw_w);

% Specific heat capacity [J/kgK]
k.cp_w       = 2.26e+03;
```

GAS PARAMETERS

Molar mass [kg/kmol]

```
k.Mw_gas     = [k.Mw_w(2)*2+k.Mw_w(6);          % H2O
                k.Mw_w(5)*2;                    % N2
                k.Mw_w(1)+k.Mw_w(6)*2;          % CO2
                k.Mw_w(6)*2;                    % O2
                k.Mw_w(1)+k.Mw_w(2)*4;          % CH4
                k.Mw_w(2)*2;                    % H2
                k.Mw_w(1)+k.Mw_w(6);            % CO
                k.Mw_w(3)+k.Mw_w(6)*2;          % SO2
                k.Mw_w(2)+k.Mw_w(4);            % HCl
                k.Mw_w(2)+k.Mw_w(7);            % HF
                k.Mw_w(5)+k.Mw_w(6)*2;          % NO2
                k.Mw_w(5)+k.Mw_w(2)*3];         % NH3
k.nc_gas     = length(k.Mw_gas);
% Specific heat capacity [J/kgK]
k.cp_fg      = 1.23e+03;
% Universal gas constant [J/kmol K]
k.R          = 8.314462175*1E3;
```

LIQUID WATER PARAMETERS

```
k.cp_water   = 4.2E+03; % [J/kg K]
k.rho_water = 1E+03;    % [kg/m3]
k.Mw_water   = k.Mw_gas(1);
```

WASTE IN

```
k.w_w_in     = [26;                           % Moisture (H2O)
                20.8;
                3;
                0.3;
```

```
                 0.2;
                 0.4;
                 17.5;
                 0.01;
                 23.8]./100;                  % Ashes
k.w_w_in     = k.w_w_in./sum(k.w_w_in);
```

GRATE

GEOMETRY data from "Ryu, C., Yang, Y.B., Nasserzadeh, V. and Swithenbank, J., 2004 "Thermal reaction modeling of a large municipal solid waste incinerator." Combustion Science and Technology, 176(11), pp.1891-1907.

```
k.W          = 3.76;             % Width of the grate
k.L          = 9.84;             % total grate lenght
k.A          = k.W*k.L;
k.v_grate    = 7.38./3600;       % [m/s]
k.rho_w      = 500;              % density of waste [kg/m3]
h_grate      = 0.3;              % Height of the grate[m] --> assumed
 value
k.A_grate    = k.W.*h_grate;
% Waste and methal conductivity
k.k_w        = 0.1;
k.k_grate    = 50;
% Other data
k.sigma      = 5.67051e-8;       % Stefan-Boltzmann constant [J/(m2 s
 K4)]
k.epsilon_gt= 0.88;              % Emissivity of flue gas  (WHAT VALUES
 SHOULD BE HERE??)
k.alpha_gt   = k.epsilon_gt;     % Absorptivity of flue gas (WHAT
 VALUES SHOULD BE HERE??)
```

Zone dimensions

```
k.M_steel_grate = 1500;              % Mass of steel in the different
 zones of the furnace (this is just a random number)
```

FREE BOARD

```
k.V_FreeBoard = k.W.*k.L*1.5;
k.M_steel_FB  = 2000;                % Mass of steel in the free board
 (this is just a random number)
```

PRIMARY AIR

```
RH_aI        = 90;               % The air is taken from the waste
 bunker --> very humid
k.w_a        = CALC_w_ha(k.T_amb,RH_aI,k.Mw_gas);  % The temperature of
 the air in the bunker is assumed equal to ambient temperature
k.T_aI       = 110 +  273.15;    % Pre-heating of primary air
```

```
y_a         = CALC_y_a(k.w_a,k.Mw_gas);
Mw_a        = sum(k.Mw_gas.*y_a);
```

Primary air fan

```
k.Pmax_aI = 114.1.*1e3; % maximum power the fan can deliver [W]
k.I_aI = 82.8; % moment of inertia [kg*m^2]
rpm_T_p = 400; % lowest value of rpm for which Torque equals Power/
omega
k.omega_T_aI = rpm_T_p*2*pi/60;
% estimating volume flow as function of rpm, assuming a linear
 relationship

% insert data points from fan data
Q_est = [1136 747 447]; % [m^3/min]
rpm_est1 = [1383 931 560]; % [rpm]
k.map_aI = mean(Q_est./rpm_est1); % propornality constant [m^3/
(min*rpm)]

% estimating torque drag function, assumed to be quadratic (T =
% alpha*omega^2)

% insert datapoints from torque diagram
torq_est_fg = [370]; % [N*m]
% insert corresponding rpm
rpm_est2_fg= [931]; % [rpm]
omega_est_fg = rpm_est2_fg.*(2*pi/60); % change to angular velocity
 [rad/s]
k.alpha_est_aII = mean(torq_est_fg./(omega_est_fg.^2)); % estimated
 propotionality constant [J*s]

rpm_in_p = 1; % initial rotation
omega_in_p = rpm_in_p*2*pi/60;

k.mu_eff    = 0.8;
```

Secondary air fan

```
k.Pmax_aII = 34.4; % maximum power the fan can deliver [kW]
k.I_aII = 3.13; % moment of inertia [kg*m^2]
rpm_T_s = 500; % lowest value of rpm for which Torque equals Power/
omega
k.omega_T_aII = rpm_T_s*2*pi/60;

% estimating volume flow as function of rpm, assuming a linear
 relationship

% insert data points from fan data
Q_est = [333]; % [m^3/min]
rpm_est1 = [2960]; % [rpm]
k.map_aII = mean(Q_est./rpm_est1); % propornality constant [m^3/
(min*rpm)]
```

```
% estimating torque drag function, assumed to be quadratic (T =
% alpha*omega^2)

% insert datapoints from torque diagram
torq_est_fg = [111.4]; % [N*m]
% insert corresponding rpm
rpm_est2_fg= [2960]; % [rpm]
omega_est_fg = rpm_est2_fg.*(2*pi/60); % change to angular velocity
 [rad/s]
k.alpha_est_s = mean(torq_est_fg./(omega_est_fg.^2)); % estimated
 propotionality constant [J*s]

rpm_in_s = 1; % initial rotation
omega_in_s = rpm_in_s*2*pi/60;
```

INITIAL CONDITIONS

Zone 1

```
k.T_0        = 850+273.15;
k.M_fg_IC    = k.p_amb.*k.V_FreeBoard./(k.R.*k.T_0).*Mw_a;
```

EVAPORATOR GEOMETRY (need data here)

```
k.p_drum     = 51e5;
M_H2O_l      = 148e3;             % Mass of liquid water in the boiler
 [kg] at T=T_amb
k.V_H2O      = M_H2O_l./k.rho_water; % Volume of the vessel occupied by
 water [m3]
k.V_EVA      = 203;               % Volume of vessel
k.V_gas      = k.V_EVA-k.V_H2O;  % Volume of the vessel eccupied by air
y_sat        = CALC_xsat(k.T_amb);
p_vap        = y_sat*k.p_amb;
rho_steam    = interp1(T_sat,rho_vap,k.T_amb,'linear','extrap');
p_steam      = interp1(T_sat,p_sat,k.T_amb,'linear','extrap');
Z_steam      = p_steam.*k.Mw_water./(rho_steam.*k.R.*k.T_amb);
M_H2O_v      = p_vap.*k.V_gas./(Z_steam.*k.R.*k.T_amb).*k.Mw_water;
k.M_H2O      = M_H2O_l + M_H2O_v;
k.x_vap_0    = M_H2O_v/k.M_H2O;
k.epsilon_EVA=0.7;               % Evaporator effectiveness
k.M_steel    = 200000;            % Total mass of metal in the boiler
 [kg]
k.cp_steel   = 460;               % Heat capacity steel [J/kg K]

k.N_air      = (k.p_amb-p_vap).*k.V_gas./(k.R.*k.T_amb);
k.Mw_air     = 28.014.*0.79 + 15.999*0.21;
k.M_air      = k.N_air.*k.Mw_air;
```

GLOBAL HEAT EXCHANGER EFFICIENCY

```
k.eff_HE     = 0.96;
```

ECONOMIZER

```
k.epsilon_ECO= 0.72;            % Economizer effectiveness
k.eta_ECO    = 0.9;
```

HEAT EXCHANGER WITH UTILITIES

```
k.epsilon_HE= 0.953;            % HE effectiveness
```

SUPERHEATER

```
k.epsilon_SH= 0.95;
```

DRUM

Liquid water enthalpy

```
h_w_sat      = interp1(p_sat,h_w,k.p_amb,'linear','extrap');
h_water      = h_w_sat - k.cp_water.*(373.15-k.T_amb);
% Vapour water enthalphy
h_vap        = interp1(T_sat,h_w,k.T_amb,'linear','extrap');
% Initial energy stored in the boiler
k.A_0        = (k.M_H2O.*(1-k.x_vap_0).*h_water +
 k.M_H2O.*k.x_vap_0.*h_vap - k.p_amb.*k.V_EVA );
k.T_guess    = 300;
k.x_guess    = 0;
```

TURBINE

```
k.p_HP       = (-0.256 + 1).*1e5;
k.p_LP       = (-0.546 + 1).*1e5;
k.eta_TURB   = 0.9;
```

HEAT OF REACTIONS

```
k = CALC_DeltaH(k);
```

GAS CLEANING SYSTEM

```
k.epsilon_HE1 = 0.054;
k.epsilon_HE2 = 0.70;
k.eff_HE2     = 0.83;
k.epsilon_HE3 = 0.47;
k.epsilon_HE4 = 0.69;
```

DYNAMIC PARAMETERS FROM PROCESS DATA

```
data2        = importdata('Data_Boiler.txt');
```

```
data3       = importdata('Data_Fg.txt');
data4       = importdata('Data_Turbine.txt');

% Time (in seconds) between data points in the excel files, and also
 the
% total time span of the process data
delta_t     = 5*60;
max_t       = 60*60*24;
```

Primary air

temperature

```
data_ex = Find_val('1HLA30DT001_PV',data2); % Read process data from
 t=0 to
% t = max_t
T_aI.signals.values = data_ex + 273.15; % Celsius to Kelvin
T_aI.time = delta_t*[0:(length(T_aI.signals.values)-1)]';
% Flow rate of primary air to the different zones. each zone is given
 a
% struct
data_ex = Find_val('1HLA41FF001_',data2); %left 1
data_ex_r = Find_val('1HLA51FF001_',data2);
% merge and scale to [kg/s]
F_aI_z1.signals.values = (data_ex + data_ex_r)./(3600*22.414).*28.966;
F_aI_z1.time = delta_t*[0:(length(F_aI_z1.signals.values)-1)]';
% repeat for the other 4
data_ex = Find_val('1HLA42FF001_',data2); %left 2
data_ex_r = Find_val('1HLA52FF001_',data2); % right 2
F_aI_z2.signals.values = (data_ex + data_ex_r)./(3600*22.414).*28.966;
F_aI_z2.time = delta_t*[0:(length(F_aI_z2.signals.values)-1)]';
data_ex = Find_val('1HLA43FF001_',data2); %left 1
data_ex_r = Find_val('1HLA53FF001_',data2); % right 1
F_aI_z3.signals.values = (data_ex + data_ex_r)./(3600*22.414).*28.966;
F_aI_z3.time = delta_t*[0:(length(F_aI_z3.signals.values)-1)]';
data_ex = Find_val('1HLA44FF001_',data2); %left 1
data_ex_r = Find_val('1HLA54FF001_',data2); % right 1
F_aI_z4.signals.values = (data_ex + data_ex_r)./(3600*22.414).*28.966;
F_aI_z4.time = delta_t*[0:(length(F_aI_z4.signals.values)-1)]';
data_ex = Find_val('1HLA45FF001_',data2); %left 1
data_ex_r = Find_val('1HLA55FF001_',data2); % right 1
F_aI_z5.signals.values = (data_ex + data_ex_r)./(3600*22.414).*28.966;
F_aI_z5.time = delta_t*[0:(length(F_aI_z5.signals.values)-1)]';
% merge all to one struct as well
F_aI.signals.values = [F_aI_z1.signals.values, F_aI_z2.signals.values,
 F_aI_z3.signals.values, F_aI_z4.signals.values,
 F_aI_z5.signals.values];
F_aI.signals.dimensions = 5;
F_aI.time = delta_t*[0:(length(F_aI_z5.signals.values)-1)]';
% total air flow from primary air source
F_aI.totave = sum(mean(F_aI.signals.values));
% total primary air struct
F_aI_tot.signals.values = sum(F_aI.signals.values,2);
```

```
F_aI_tot.time = F_aI.time;
```

Secondary air

```
data_ex = Find_val('1HLA81CF001_F_',data2);
  % Flow secondary air front wall
data_ex_r = Find_val('1HLA82CF001_F_',data2);
  % Flow secondary air rear wall
F_aII.signals.values = (data_ex + data_ex_r)./(3600*22.414).*28.966;
  % Secondary air flows from both walls are summed
F_aII.time = delta_t*[0:(length(F_aII.signals.values)-1)]';
F_aII.totave = mean(F_aII.signals.values);
F_a_tot = (F_aII.signals.values
+F_aI_tot.signals.values).*(3600*22.414)./28.966;
% Temperature
data_ex = Find_val('1HLA80CT001_F_',data2); %(correct??)
T_aII.signals.values = data_ex + 273.15;
  % Celsius to Kelvin
T_aII.time = delta_t*[0:(length(T_aII.signals.values)-1)]';
```

Waste

```
data_ex = Find_val('1HFB10FF002',data2); % rear wall
F_w_in.signals.values = (data_ex) ./3.6;
F_w_in.time = delta_t*[0:(length(F_aII.signals.values)-1)]';
```

Grate cooling

Temperature upstream grate cooling

```
T_c     = Find_val('1HLC03CT001_',data2);
% Cooling water flow in the different zones
F_z1    = Find_val('1HLC51CF001_',data2);
F_z2    = Find_val('1HLC52CF001_',data2);
F_z3    = Find_val('1HLC53CF001_',data2);
% Cooling water temperature in the different zones
T_z1    = Find_val('1HLC51CT001_',data2);
T_z2    = Find_val('1HLC52CT001_',data2);
T_z3    = Find_val('1HLC53CT001_',data2);
% C_min in the different zones
C_z1    = F_z1.*4.2e3;
C_z2    = F_z2.*4.2e3;
C_z3    = F_z3.*4.2e3;
% Heat exchanged in the different zones
Q_z1    = C_z1.*(T_z1-T_c);
Q_z2    = C_z2.*(T_z2-T_c);
Q_z3    = C_z3.*(T_z3-T_c);
% Make bus signals
Q_grate.signals.values = [Q_z1,Q_z2,Q_z3];
Q_grate.time           = delta_t*[0:
(length(Q_grate.signals.values)-1)]';
```

District heating water

```
F_from_DH.signals.values= Find_val('1NDA10CF010_A',data4);          %
 Water flow in district heating
F_from_DH.time          = delta_t*[0:
(length(F_from_DH.signals.values)-1)]';
T_from_DH.signals.values= Find_val('1NDB10CT003_',data4) + 273.15;  %
 Water flow in district heating
T_from_DH.time          = delta_t*[0:
(length(T_from_DH.signals.values)-1)]';
```

Controller

```
% Controller type
%
% Open, AB, Cascade or MPC. Is used by the ChooseController block.
%
reg.controller_type = ControllerType.AB;

% Operating values
%
% The controller action happens around these operating values.
%
reg.op_F_O2 = 4.6345;                   % Oxygen flow (kg/s)
reg.op_F_st = 10.761;                   % Steam flow (kg/s)
reg.op_HHV = 4.133444255e+07;           % Heat value (J/s?)
reg.op_F_aII = 14;                      % Secondary air (kg/s)
reg.op_F_aI = 18;                       % Total primary air (kg/s)
reg.op_v_grate = 0.0025;                % Grate speed (m/s)
reg.op_F_w_in = 4;                      % Waste feed (kg/s)
reg.op_MVs = [...
    reg.op_F_aII;...
    reg.op_F_aI;...                     % Needed for backwards
 compatibility.
    reg.op_v_grate;...                  % Should be fixed.
    reg.op_F_w_in...
 ];
reg.MV_names = ["F_aII" "F_aI"...       % Needed for backwards
 compatibility.
    "v_grate" "F_w"];                   % Should be fixed.

% Constant input values
%
% These are used when the corresponding variable has
% input_type set to InputType.Constant.
%
reg.const_F_aI = reg.op_F_aI;                   % Total primary air
 (kg/s)
reg.const_v_grate = reg.op_v_grate;             % Grate speed (m/s)
reg.const_F_w_in = reg.op_F_w_in;               % Waste feed (kg/s)
reg.const_F_aII = reg.op_F_aII;                 % Secondary air (kg/s)
```

```
reg.const_T_aI = 380;                           % Primary air
 temperature (K)
reg.const_T_aII = 300;                          % Secondary air
 temperature (K)
reg.const_Q_grate = [1.0e+05 2.2e+05 2.5e+05];  % Grate heat exchange
 (J?)
reg.const_F_from_DH = 130;                      % Flow from district
 heating (kg/s?)
reg.const_T_from_DH = 320;                      % Temperature of flow
 from district heating (K)
%
% Primary air flow distribution
%
% The relative distribution between primary air zones.
%
reg.const_F_aI_dist = [0.05 0.3 0.38 0.24 0.03];

% Input signal type
%
% Set the input type for different process variables. Is used by the
% ChooseSignal block.
%   Constant: The signal is determined by reg.const_<variable>.
%   Experimental: The signal is loaded from process data.
%   Controlled: The signal is determined by the controller.
% Note that not all variables can be Controlled.
%
% Controllable:
%
reg.input_type_F_aII = InputType.Controlled;    % Secondary air
reg.input_type_F_aI = InputType.Controlled;     % Primary air
reg.input_type_v_grate = InputType.Controlled;  % Grate speed
reg.input_type_F_w_in = InputType.Controlled;   % Waste feed
reg.input_type_HHV = InputType.Controlled;      % Heating value
reg.input_type_AB = InputType.Controlled;       % Primary-secondary
 air ratio
%
% Not controllable:
%
reg.input_type_T_aI = InputType.Constant;       % Primary air
 temperature
reg.input_type_T_aII = InputType.Constant;      % Secondary air
 temperature
reg.input_type_Q_grate = InputType.Constant;    % Grate heat exchange
reg.input_type_F_from_DH = InputType.Constant;  % Flow from district
 heating
reg.input_type_T_from_DH = InputType.Constant;  % Temperature from
 district heating
%
% Heat value setpoint override
%
% Set to true if you are using the cascade controller want to control
 the
% HHV setpoint manually using reg.op_HHV, instead of getting it from
 the
```

```
% primary air PID. This is useful (and necessary) when tuning the
 slave
% loop.
%
reg.override_sp_HHV = false;

% PID parameters
%
% The critical value is an estimate of the P-constant
% required to get 30% overshoot when performing the SIMC tuning. It
 will
% probably need some adjustment.
%
% Oxygen controller
%
reg.Kp_O2 = 1.9769;             % Critical: 34
reg.Ki_O2 = 0.5054;             % Critical: 0
%
% Steam controller (no cascade)
%
reg.Kp_st = 45.8596;            % Critical: 125
reg.Ki_st = 0.8579;             % Critical: 0
reg.Kd_st = 0;                  % Critical: 0
%
% AB ratio controller (no cascade)
%
reg.Kp_AB = -0.0010;            % Critical: -0.0002
reg.Ki_AB = -2.9786e-05;        % Critical: 0
%
% Heat value controller
%
reg.Kp_HHV = 2.1723e-07;        % Critical: ?
reg.Ki_HHV = 1.4988e-07;        % Critical: 0
%
% Steam controller (cascade)
%
reg.Kp_st_HHV = 1.2459e+07;     % Critical: ?
reg.Ki_st_HHV = 2.4691e+05;     % Critical: 0
reg.Kd_st_HHV = 0;              % Critical: 0
%
% AB ratio controller (cascade)
%
reg.Kp_AB_HHV = -0.0010;        % Critical: -0.001
reg.Ki_AB_HHV = -2.9786e-05; % Critical: 0
%
% Fan controllers
%
reg.Kp_fan = 15;                % Critical: ?
reg.Ki_fan = 15;                % Critical: 0
%
% AB ratio controller output distribution
%
% The PID output is multiplied with these constants to determine their
% control signals.
```

```
%
reg.F_w_ratio = 400;
reg.v_grate_ratio = 1;

% Set points
%
% The controller setpoints. Add to these values to witness the
 controller
% response to setpoint changes.
%
reg.sp_F_O2 = reg.op_F_O2;       % Oxygen setpoint (kg/s)
reg.sp_F_st = reg.op_F_st;       % Steam setpoint (kg/s)
reg.sp_AB = 1.286;               % AB setpoint
reg.sp_HHV = reg.op_HHV;         % HHV setpoint, not used by the
 controller,
                                 % only the tuning script

% Waste feed perturbation
%
% A square wave is applied to the waste feed input
% to emulate composition variation. Set the amplitude to 0 to disable,
 and
% set the frequency to a large value to create a step disturbance.
 Note
% that the square wave is offset 50% down on the y axis.
%
reg.F_w_pert_amp = 0*4*0.1;      % Peak-to-peak amplitude (kg/s)
reg.F_w_pert_freq = 1/400000;    % Frequency (s)


% PID output low pass filters
%
% The discrete PIDs and the MPC create stepping outputs, which can
 upset
% the differentiator. A first order low pass filter is used, but the
% effectiveness of this approach is not determined. Set higher time
% constants to increase smoothing.
%
reg.T_O2_LPF = 1e-01;   % Oxygen PID smoothing
reg.T_st_LPF = 1e-01;   % Steam PID smoothing
reg.T_AB_LPF = 1e-01;   % AB PID smoothing
reg.T_MPC_LPF = 1e-01;  % MPC smoothing

% Initial values
%
% Needed by fan model to avoid spikes at startup.
%
reg.F_aII_fan_PID_int = 1.01e+03;       % Secondary air PID integrator
reg.F_aII_fan_model_int = 16.8;         % Secondary air model
 integrator
reg.F_aI_fan_PID_int = 1.66e+03;        % Primary air PID integrator
reg.F_aI_fan_model_int = 16.8;          % Primary air model integrator

% MPC variables
```

```
%
% The linear system and MPC object is stored between runs. To create
 new
% versions, just overwrite the corresponding file.
%
load('linsys.mat');     % Linear system, see linearize_model.m
load('lmpc.mat');       % MPC object, see config_mpc.m
```

Additional part added by Jorn

```
set_post_step_values; % Set the values after a step
reg.stepTime = 17e3;
reg.include_noise = false;
load("last_mpc_params.mat"); % Parameter for the MPC are needed to
 keep the system form crashing (and the last MPC controller)
mpc_function_type  = mpc_params.mpc_function_parameter_type;
unpack_mpc_params("mpc_params.mpc_function_params")
reg.enforce_parameter_limits = false;
```

Published with MATLAB® R2020a