DENSO

Crafting the Core

Coupling MATLAB with KULI for ORC System Optimisation

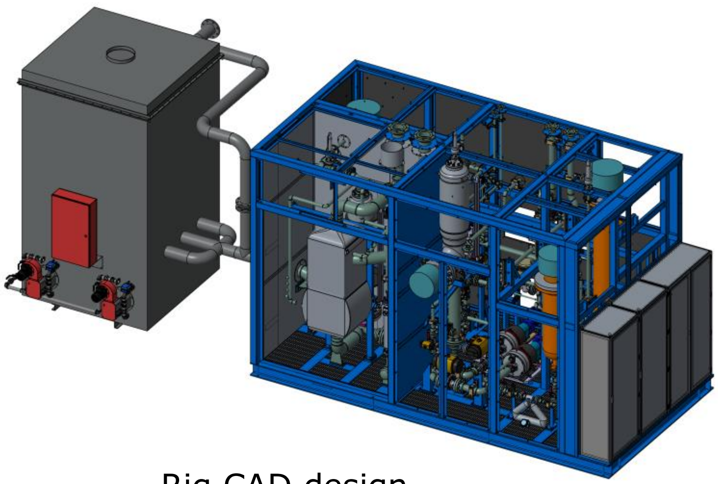
G Yu

May 2021

Backgrounds & Objectives

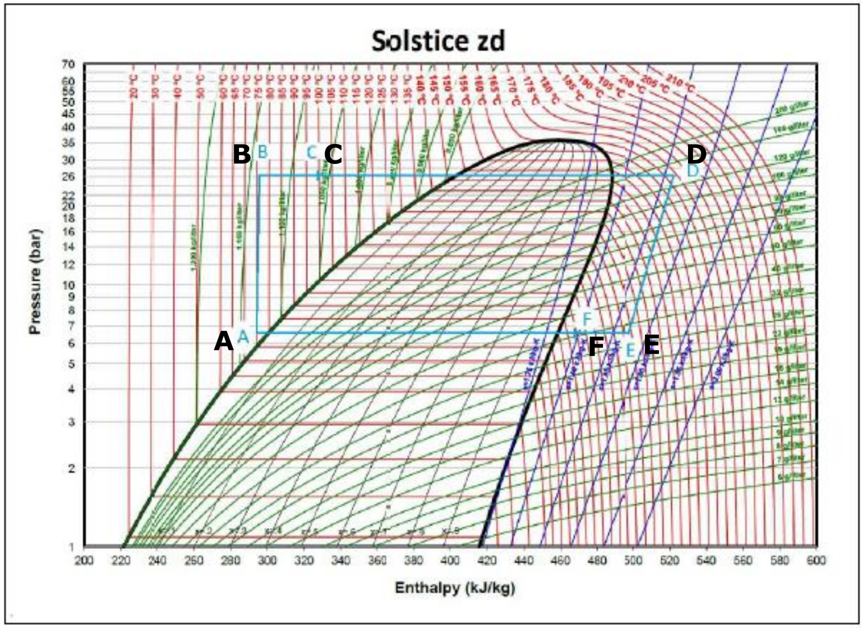
- An organic Rankine cycle (ORC) test rig was designed and being constructed in DENSO Marston Ltd for test of phase change heat exchangers, such as boilers and condensers, used in ORC systems for heavy duty off-highway vehicles.
- To operate this complex system to meet customer specs is not possible through trial and error for example for an operating condition with 6 variables.
- The requested test working conditions can be identified by KULI's optimisation process.
- It was found that KULI optimiser is too sensitive to the bound values of design parameters therefore can not always find the optimal solutions. A professional optimiser was recommended for this purpose.
- A method was developed to couple KULI with MATLAB's optimisation algorithm as an alternative.

Brief Introduction of the ORC System Test Rig



Rig CAD design

Rig under construction



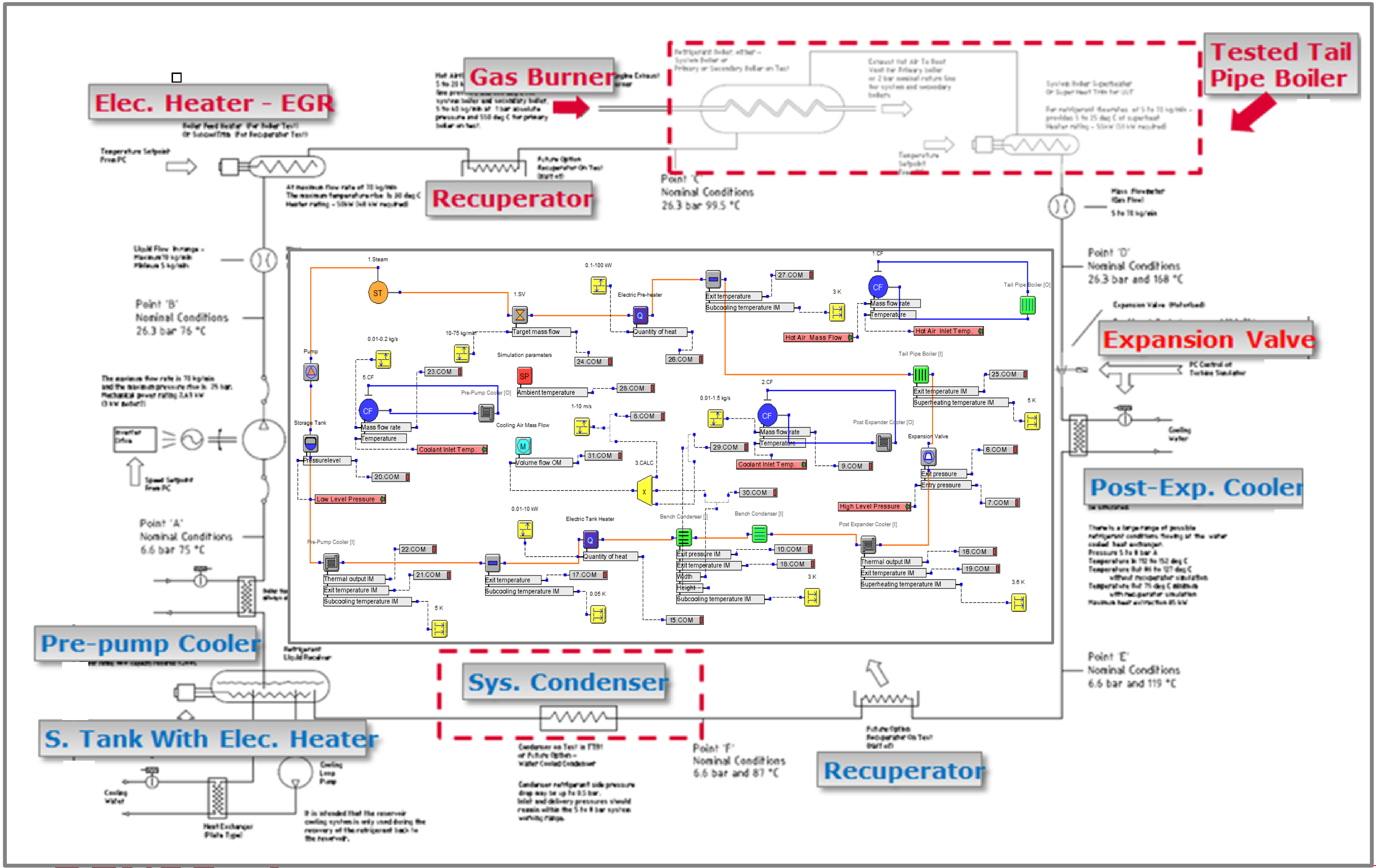
System information:

- Organic fluid: R1233ZD
- Evaporation pressure: 26.3 bar
- Condensation pressure: 6.6 bar
- Hot air as heating source for evaporator
- Ambient air as cooling sink for condenser

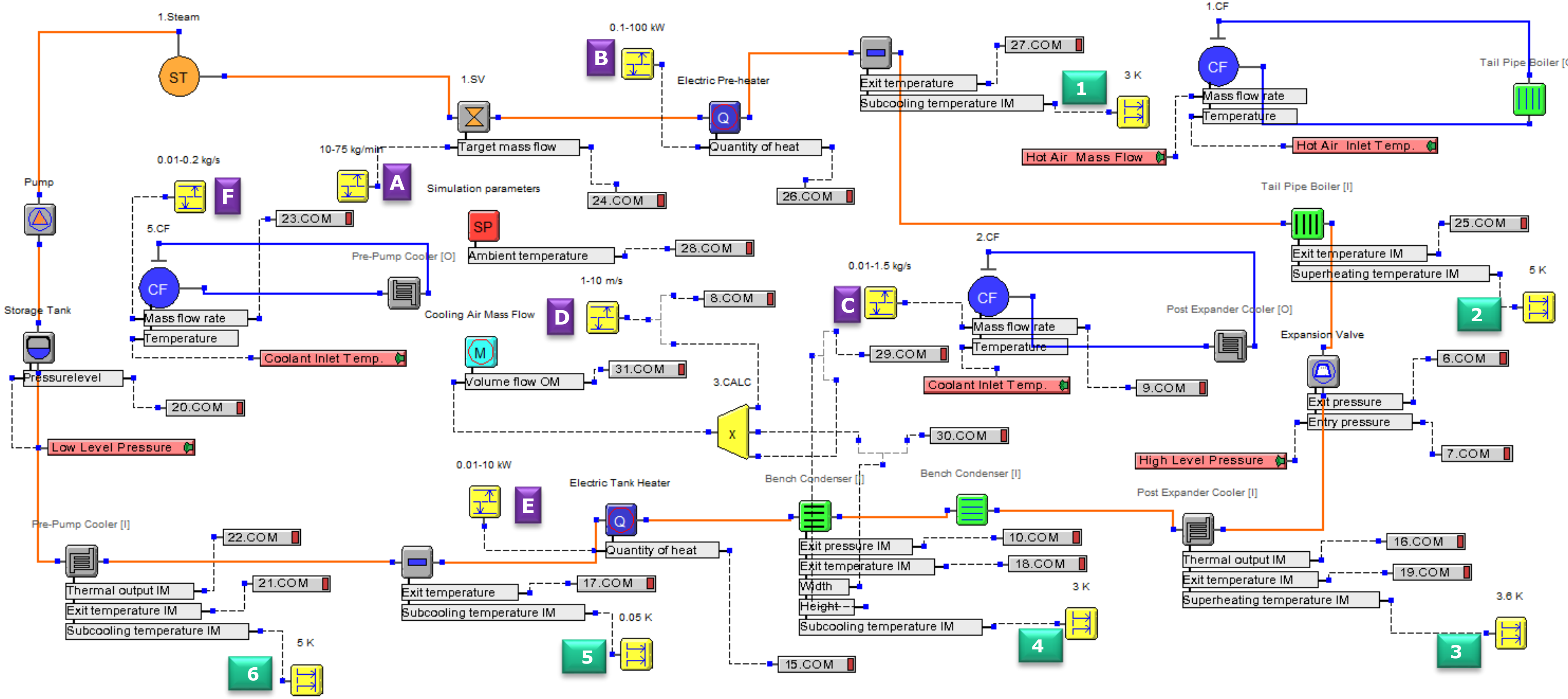
System processes & components

- Pump (A->B)
- Electric heater + Recuperator (B->C)
- Evaporator (C->D)
- Expansion valve + water cooled cooler (D->E)
- Recuperator (E->F)
- Condenser + Storage tank with electric heater + Water cooled cooler (F->A)

System Diagram for Tail Pipe Boiler Test



System Simulation - Tail Pipe Boiler Test



Design Targets

1. Boiler entry sub cool (K)
2. Boiler exit super heat (K)
3. Condenser entry super heat (K)
4. Condenser exit sub cool (K)
5. Storage tank sub cool (K)
6. Pump entry sub cool (K)

Design Parameters

A. Refrigerant mass flow rate (kg/min)

B. EGR emulator (modelled by elec. Heater) power input (kW)

C. Water mass flow rate – post expansion cooler circuit (kg/s)

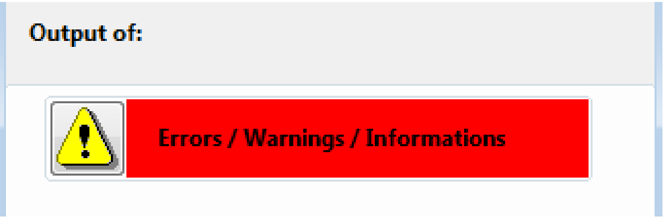
D. Cooling air velocity (m/s)

E. Storage tank electric power input (kW)

F. Water mass flow rate – pre-pump cooler circuit (kg/s)

Why Coupling MATLAB with KULI

When building the KULI evaporator component, if the input data covers a wider range, its discontinuity from non-phase to phase change or vice versa may lead to the KULI model failing to find the optimal solutions as shown below:



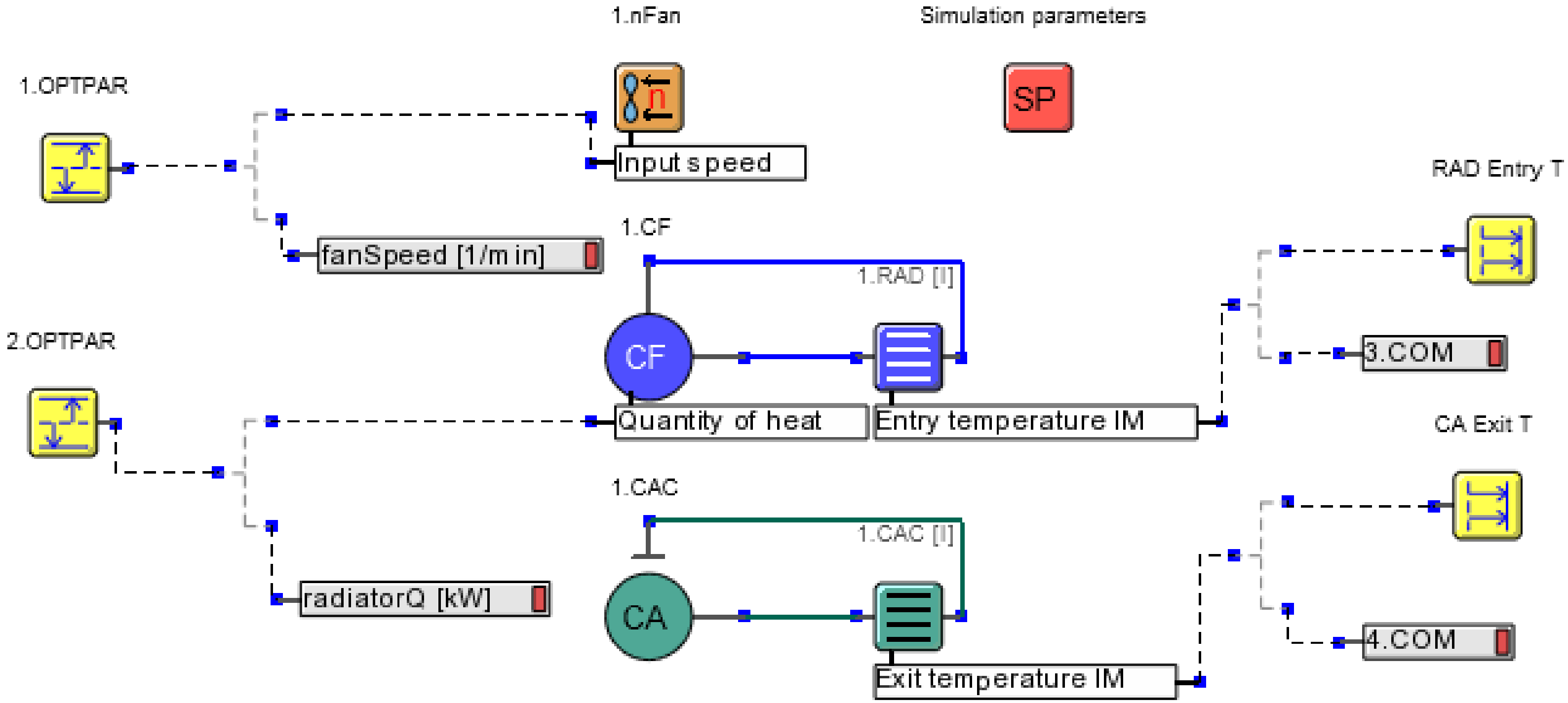
Design Parameters Values

7.OPTPAR: 50072.0395
6.OPTPAR: 0.177032941
5.OPTPAR: 4661.02082
3.OPTPAR: 4.8092379
2.OPTPAR: 0.696466886
1.OPTPAR: 0.885145751

KULI Optimisation Results	Targets
OP 1: 1.Expansion device - is working in mixed phase area!	
Opt. target 1.TUB: Subcooling temperature IM: 20.0741015 [K]	3
Opt. target 4.ACPHE: Subcooling temperature IM: 3.37829305 [K]	5
Opt. target 2.TUB: Subcooling temperature IM: -4.97219476 [K]	0.05
Opt. target 2.CND: Subcooling temperature IM: 1.27063308 [K]	3
Opt. target 2.EVP: Superheating temperature IM: -35.1139483 [K]	5
Opt. target 2.ACPHE: Superheating temperature IM: -14.8099924 [K]	3.6

Solution:
KULI for system simulation only, optimisation process by MATLAB

A Simple KULI System for Feasibility Study



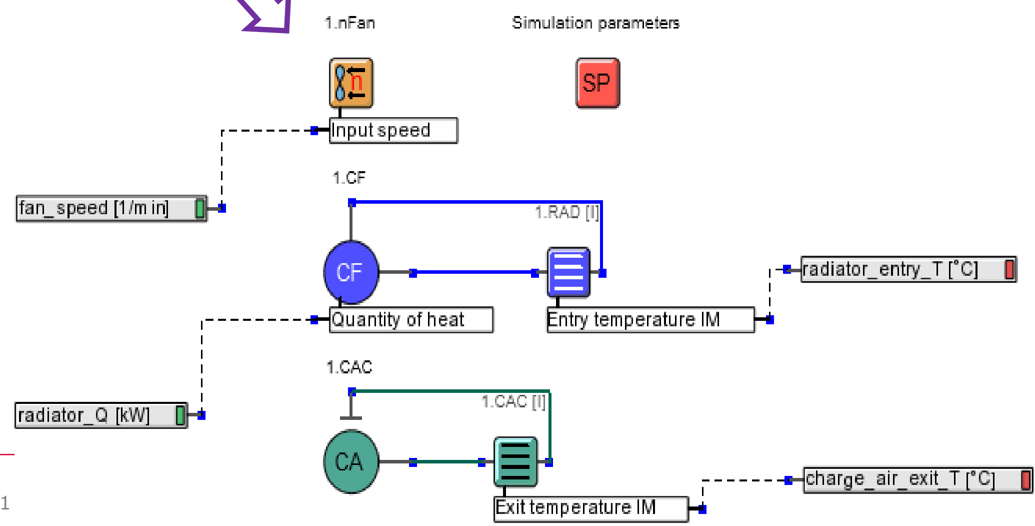
- Two variables to be optimised
  1. coolant circuit heat input – *radiatorQ* (kW)
  2. fan speed – *fanSpeed* (1/min)
- Two targets to be achieved
  1. Radiator coolant entry temperature – 98 Deg C
  2. Charge air exit temperature – 35 Deg C

DENSO
Crafting the Core

A Simple KULI System – Setup in MATLAB

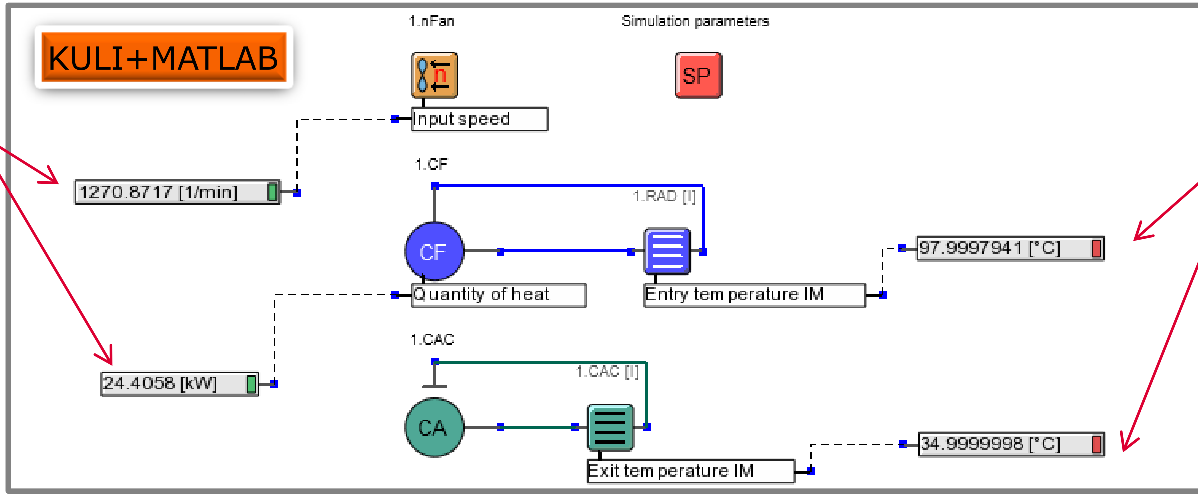
```
% Function to call KULI model
% 2 design parameters: fanspeed & radiatorQ
% 2 targets: watertemp & airtemp
% The function just returns 2 target values given 2 parameter inputs
% The 2 parameter inputs should be provided by MATLAB optimiser
%
function targets = callkuli(parameters)
    fanSpeed = parameters(1);
    radiatorQ = parameters(2);

    global kuli
    kuli = actxserver('KuliAnalysisServer.KuliAnalysisCtr2.13.1');
    kuli.registerevent({'OnMessage', @kuliOnMessage; 'OnError', ...
        @kuliOnError});
    fileName = ...
        'D:\Gary Yu\MATLAB\CoupleKULI\CoolingSystems\ExCAR_opt_2dim_cycle_only.scs';
    kuli.set('KuliFileName', fileName);
    kuli.invoke('Initialize');
%     ok = kuli.invoke('SetValue','1.CircuitWater', 'QuantityHeat', 25000);
%     kuli.invoke('RunAnalysis');
    invoke(kuli,'SetCOMValueByID','fan_speed',fanSpeed);
    invoke(kuli,'SetCOMValueByID', 'radiator_Q', radiatorQ);
    invoke(kuli,'SimulateOperatingPoint',int8(1));
    watertemp = invoke(kuli,'GetCOMValueByID','radiator_entry_T');
    airtemp = invoke(kuli, 'GetCOMValueByID', 'charge_air_exit_T');
    targets(1) = watertemp;
    targets(2) = airtemp;

    kuli.delete;
end
```

- Create a KULI system as shown below
- Accepting two inputs and producing two outputs
- In MATLAB, specify the file name with correct directory
- In MATLAB, use Microsoft Component Object Model (COM) Application Programming Interface(API) to set KULI input values and obtain KULI output values

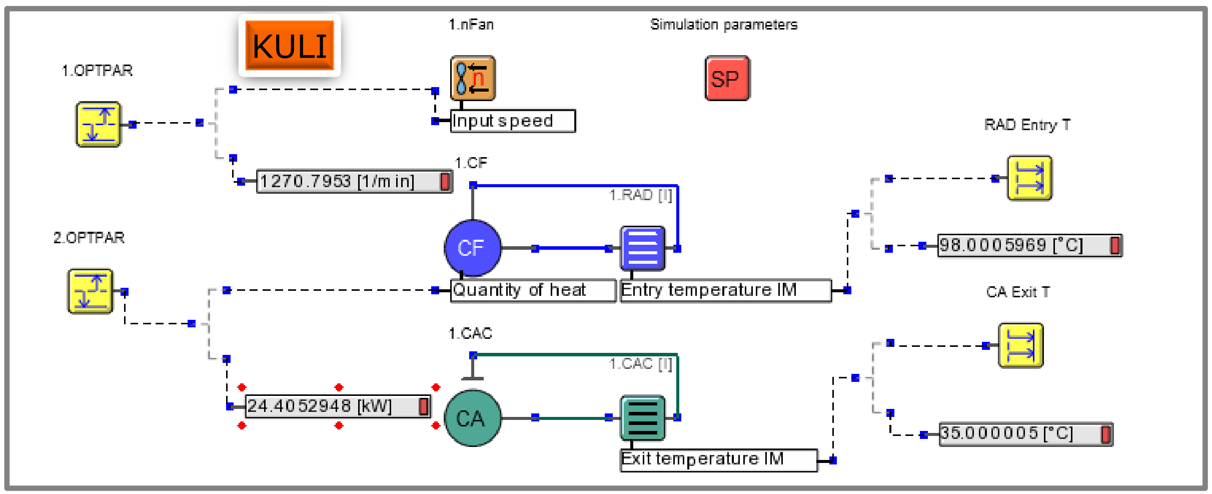


A Simple KULI System – Comparison Between MATLAB & KULI Optimisation

Optimal values from MATLAB optimiser



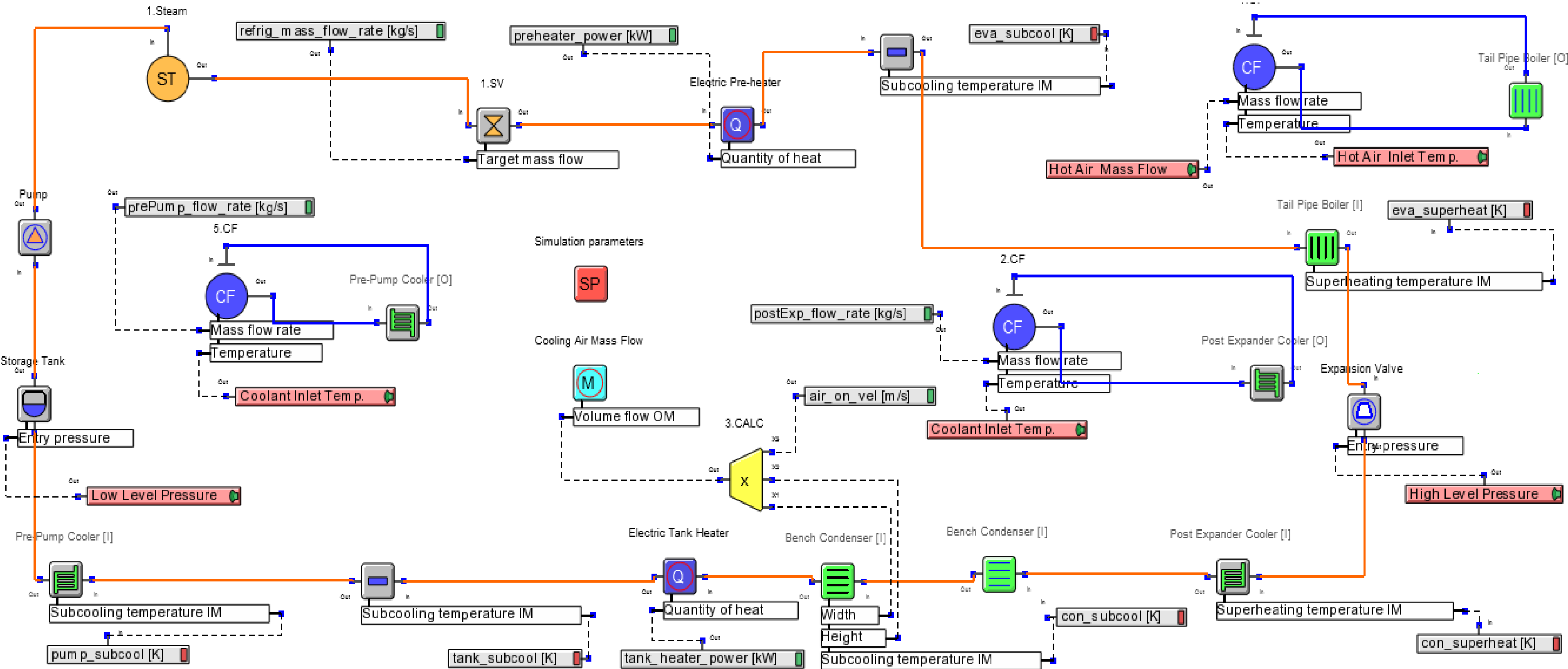
Target (output) values from KULI system simulation



A good agreement between KULI and KULI+MATLAB

ORC System Optimisation - MATLAB

KULI file name = ORC tail pipe boiler test for MATLAB.scs

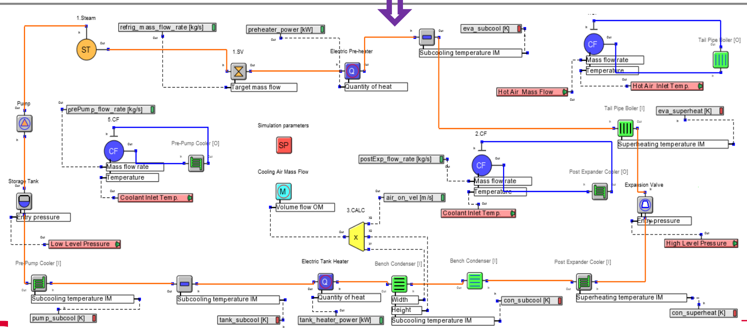


Design Parameters	Variable Name
1. Refrigerant mass flow rate (kg/s)	refri_mass_flow_rate
2. EGR emulator (modelled by elec. Heater) power input (kW)	preheater_power
3. Storage tank electric power input (kW)	tank_heater_power
4. Water mass flow rate – post expansion cooler circuit (kg/s)	postExp_flow_rate
5. Water mass flow rate – pre-pump cooler circuit (kg/s)	prePump_flow_rate
6. Cooling air velocity (m/s)	air_on_vel

Design Targets	Variable Name	Targets
1. Evaporator entry sub cool (K)	eva_subcool	3
2. Evaporator exit super heat (K)	eva_superheat	5
3. Condenser entry super heat (K)	con_superheat	3.6
4. Condenser exit sub cool (K)	con_subcool	3
5. Storage tank sub cool (K)	tank_subcool	0.05
6. Pump entry sub cool (K)	pump_subcool	5

ORC System Optimisation - MATLAB

```
function targets = callkuli(parameters)
%     global kuli
%     For KULI v13 and later
%     kuli = actxserver('KuliAnalysisServer.KuliAnalysisCtr2.13.1');
%     For other earlier KULI versions
    kuli = actxserver('KuliAnalysis2.KuliAnalysisCtr2.12.x64');
%     kuli.registerevent({'OnMessage', @kuliOnMessage; 'OnError', ...
%         @kuliOnError});
    fileName = ...
        'D:\ORC tail pipe boiler test for MATLAB.scs';
    kuli.set('KuliFileName', fileName);
    kuli.invoke('Initialize');

    kuli.invoke('SetCOMValueByID','refrig_mass_flow_rate',parameters(1));
    kuli.invoke('SetCOMValueByID', 'preheater_power', parameters(2));
    kuli.invoke('SetCOMValueByID', 'tank_heater_power', parameters(3));
    kuli.invoke('SetCOMValueByID', 'postExp_flow_rate', parameters(4));
    kuli.invoke('SetCOMValueByID', 'prePump_flow_rate', parameters(5));
    kuli.invoke('SetCOMValueByID', 'air_on_vel', parameters(6));

%     kuli.invoke('RunAnalysis');
    kuli.invoke('SimulateOperatingPoint',int8(1));

    targets(1) = kuli.invoke('GetCOMValueByID','eva_subcool');
    targets(2) = kuli.invoke('GetCOMValueByID', 'eva_superheat');
    targets(3) = kuli.invoke('GetCOMValueByID','con_superheat');
    targets(4) = kuli.invoke('GetCOMValueByID', 'con_subcool');
    targets(5) = kuli.invoke('GetCOMValueByID','tank_subcool');
    targets(6) = kuli.invoke('GetCOMValueByID', 'pump_subcool');

    kuli.invoke('CleanUp');
    kuli.delete;
end
```



KULI HVAC	
KULI AC: Maximum number of circuit equalizations in overall system	400
KULI AC: Truncation condition for enthalpy equalization [%/100]	0.001
KULI AC: Truncation condition for subcooling/superheating equalization [K]	0.001

0.1 before

```
% Define design parameter lower/upper bounds
% 1......refrig_mass_flow_rate, kg/s [10, 75]/60
% 2......preheater_power, kW [0.1, 100]
% 3......tank_heater_power, kW [0.01, 10]
% 4......postExp_flow_rate. kg/s [0.01, 1.5]
% 5......prePump_flow_rate, kg/s [0.01, 0.2]
% 6......air_on_vel, m/s [1, 10]
% order important

number_of_variables = 6;

% ObjectiveFunc = @objectiveFunction; %dummy objective function
Constraints = @constraints;%define constraints for optimiser
ConstraintTolerance_Data = 1e-4; %1e-4;%set constraint tolerance
StepTolerance_Data = 1e-6;% X step size tolerance
FiniteDifferenceStepSize_Data = [0.01, 0.01, 0.01, 0.01, 0.01, 0.01];

LB = [10/60, 0.1, 0.01, 0.01, 0.01, 1];
UB = [75/60, 100, 10,   1.5,  0.2,  10];

% Define design parameter initial values
x0 = ones(1, number_of_variables);
for i=1:number_of_variables
    x0(i) = 0.5* (LB(i) + UB(i));
end

% Call MATLAB 'fmincon' optimisation algorithm to
% Find a minimum of a constrained nonlinear multivariable function
[x,fval,exitflag,output] = fmincon_opt(@(x)0, x0, LB, UB, ...
    Constraints,ConstraintTolerance_Data, StepTolerance_Data, ...
    FiniteDifferenceStepSize_Data);
% Final run to get the targets
y = callkuli(x);
disp(output);
```

Bound Sensitivity Analysis - MATLAB & KULI

Design Parameters	Bound-1	Bound-2	Bound-3
1. Refrigerant mass flow rate (kg/min)	10 ~ 75	20 ~ 37.5	10 ~ 75
2. EGR emulator (modelled by elec. Heater) power input (kW)	0.1 ~ 100	0.2 ~ 66.67	1 ~ 400
3. Storage tank electric power input (kW)	0.01 ~ 10	0.02 ~ 5	1 ~ 200
4. Water mass flow rate – post expansion cooler circuit (kg/s)	0.01 ~ 1.5	0.02 ~ 0.75	0.01 ~ 10
5. Water mass flow rate – pre-pump cooler circuit (kg/s)	0.01 ~ 0.2	0.05 ~ 0.2	0.01 ~ 5
6. Cooling air velocity (m/s)	1 ~ 10	2~ 10	1 ~ 10

		KULI+MATLAB					
Design Targets	**Targets**	**Values – Bound 1**	**%**	**Values – Bound 2**	**%**	**Values – Bound 3**	**%**
1. Evaporator entry sub cool (K)	3	2.97	99%	3.026	101%	3.0034	100%
2. Evaporator exit super heat (K)	5	5.03	101%	4.98	100%	5.0033	100%
3. Condenser entry super heat (K)	3.6	3.63	101%	3.58	99%	3.6007	100%
4. Condenser exit sub cool (K)	3	3.01	100%	3	100%	3.0189	101%
5. Storage tank sub cool (K)	0.05	0.0626	125%	0.0526	105%	0.0686	137%
6. Pump entry sub cool (K)	5	5.01	100%	5.0029	100%	4.9766	100%

		KULI					
Design Targets	**Targets**	**Values – Bound 1**	**%**	**Values – Bound 2**	**%**	**Values – Bound 3**	**%**
1. Evaporator entry sub cool (K)	3	2.93	98%	2.894	96%	2.711	90%
2. Evaporator exit super heat (K)	5	4.95	99%	5.231	105%	7.099	142%
3. Condenser entry super heat (K)	3.6	3.678	102%	3.769	105%	-0.172	-5%
4. Condenser exit sub cool (K)	3	2.953	98%	3.003	100%	4.251	142%
5. Storage tank sub cool (K)	0.05	0.005	10%	0.0505	101%	-0.399	-798%
6. Pump entry sub cool (K)	5	4.959	99%	4.996	100%	5.669	113%

- Bound-1 initial guess
- Bound-2 improved one based on optimal results from Bound-1, much closer to the optimal results
- Bound-3 mixed between Bound-1 & Bound-2 for algorithm robustness check

- MATLAB has a better consistence
- MATLAB is more flexible, which allows users to set up problem dependent simulation controls to achieve a better solution
- Both can not deal with Design target 5 except for Bound-2, i.e. the bound range is narrower around the true solution
- Compared with KULI, MATLAB takes a longer but affordable simulation time, ~45 minutes on a desktop with 3.06GHz CPU, 48GB RAM, 64 bit system

Conclusions

- KULI optimiser worked well when design parameter bound values were narrow enough around true optimal solutions.
- A method was developed to use KULI for system simulation, MATLAB optimisation algorithm for optimisation process.
- An ORC system with 6 design parameters and 6 design targets was successfully optimised using KULI coupled with MATLAB.
- Bound sensitivity analysis showed that KULI optimiser was more sensitive to the bound values, KULI/MATLAB was more flexible and robust.
- It was also shown that compared with KULI, KULI/MATLAB took a longer but affordable simulation time.
- Possible improvement of KULI optimiser to deal with phase change heat exchangers for ORC system optimisation?

DENSO
Crafting the Core