Manual for the use of the optimization code

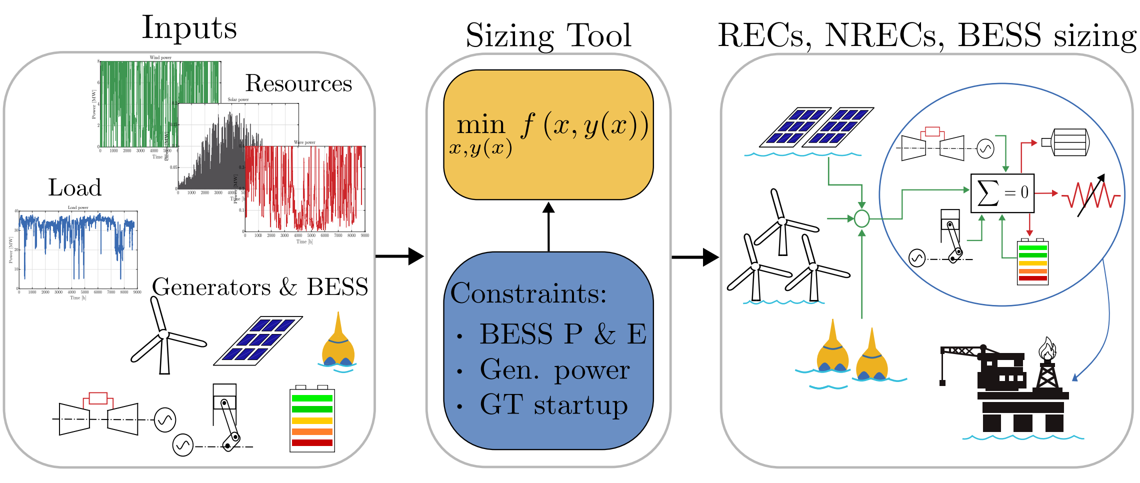
Andreetta Niccolò
niccolo.andreetta@ntnu.no

August 11, 2025

Petronas-NTNU-IEL Collaboration within the SAFER Project

Energy Storage Sizing/Control for 100% Renewable Supplied Isolated Grids

Coordinated Converter Control for Stability and Power Quality in Inertia-less Grids



This manual and the related code is a work in progress, which is continuously improved by the authors. It is not a finished work and may therefore contain defects or "bugs" inherent to this type of development. For this reason the work is provided without warranties of any kind concerning the work, including without limitation merchantability, fitness for a particular purpose, absence of defects or errors, and accuracy.

Contents

1 Code Capabilities

The focus of this manual is on the code capabilities and structure, so the reader can gain an understanding on how to use the code.
The code is developed requiring as input data series of the load and the environmental resources, the different type of assets (generators and storages) that can be installed, and their costs, and provides as output the optimal assets combination to support the load with the given specification. The code is able to perform parametric sweep of some user-defined parameters.
From the technical perspective the code builds and solves an optimization problem expressed as a Mixed Integer Linear Programming (MILP).

2 Prerequisites

The following Matlab products are required:

- Optimization toolbox
- Statistics and Machine Learning Toolbox
- Parallel Computing Toolbox
- Global Optimization Toolbox

2.1 Available assets

Management of the different devices: Battery Energy Storage System (BESS), Solar PV, Wind Turbine (WT), Diesel Generator (DG), Gas Turbine (GT), and potential extension to Hydrogen Fuel Cell.

2.2 Configurations

Different device configurations can be selected by the user before running the optimization (e.g. `WT+DG`, `REC-U`, `REC-U+DG`, `REC-C+DG24`).

2.3 Parameter Sweeps

It is possible to perform parameter sweep. Examples include risk-related variables, carbon tax, PV cost. When operating on parametric sweep it might time advantageous to parallelize hte execution of ht ecode.

3 Code Structure

3.1 General Workflow

1. Data set import
2. Device definition
3. Optimization problem formulation
4. Post-processing
5. Results visualization

3.2 Initialization

Main file: `main_decarbonization.m`
Select parallel or sequential execution: `enable_parallel_computing` (Figure 1)

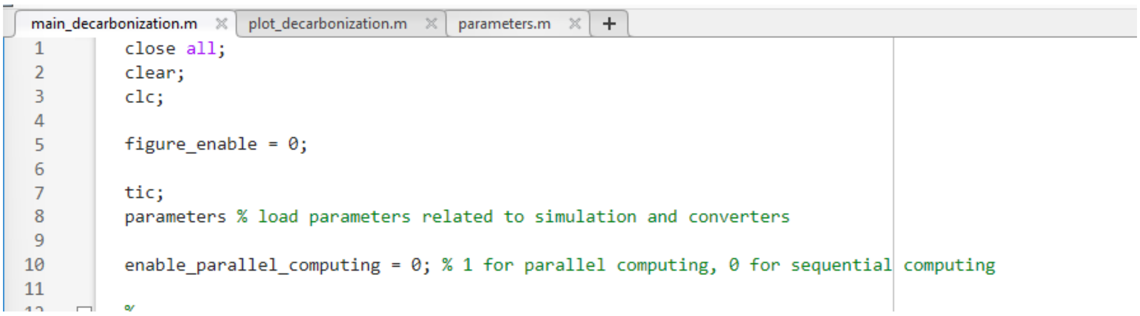


Figure 1: Main project file structure

3.3 Converters

Defined in `define_converters_corepower_MW_formulation` with all parameters Figure 2. Converters and storages are defined in classes Figure 3.

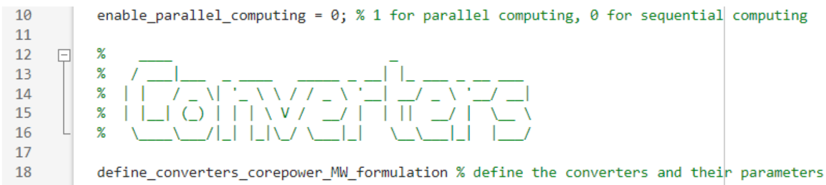


Figure 2: Converter definition

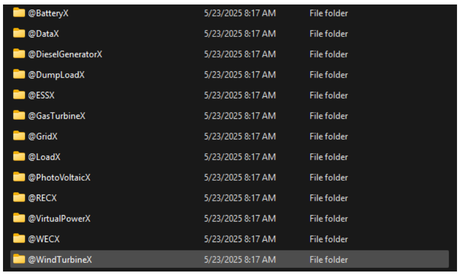


Figure 3: Class definitions for converters and storage

3.4 Dataset Import

`import_datasets_norway_1yr` loads the datasets of load and resources. Structure depends heavily on datasets in use, in the sense that the user has to tune the loading function according to the format of the available dataset Figure 4.

```
define_converters_corepower_MW_formulation % define the converters and their pa

%  ____        _                 _
% |  _ \  __ _| |_ __ _ ___  ___| |_ ___
% | | | |/ _` | __/ _` / __|/ _ \ __/ __|
% | |_| | (_| | || (_| \__ \  __/ |_\__ \
% |____/ \__,_|\__\__,_|___/\___|\__|___/

import_datasets_norway_1yr % load the datasets of the different resources
% import_datasets_dulang_1yr % load the datasets of the different resources
```

Figure 4: Example dataset loading

3.5 Configuration Selection

`case_sim_vec` defines which devices are included in optimization. The user can specify which configuration to use among the listed ones Figure 5. If the user wants to add more configurations he/she has to start doing it from here.

3.6 Sweep Type Selection

Selection of sweep type. The sweep can be performed on the listed quantities:

- `None`: No sweep
- `beta`: Control parameter of the Conditional Value at Risk
- `LPSP`: Lost of Power Supply Probability (percentage)
- `PS`: Peak shaving (percentage)
- `Carbon_tax`: scaling factor for the nominal carbon tax

```matlab
%   ____             __ _                     _
%  / ___|___  _ __  / _(_) __ _   ___  ___| |
% | |   / _ \| '_ \| |_| |/ _` | / __|/ _ \ |
% | |__| (_) | | | |  _| | (_| | \__ \  __/ |_
%  \____\___/|_| |_|_| |_|\__, (_) |___/\___|_(_)
%                         |___/

REC     = RECX();
REC.WT  = WT;
REC.PV  = PV;
REC.WEC = WEC;
REC.DGv.DGv_obj = DGv_obj;
REC.DGv.n_NREC  = 1;
REC.GT.GT_obj   = GT_obj;
REC.GT.n_NREC   = n_GT;
REC.DG.DG_obj   = DG_obj;
REC.DG.n_NREC   = n_DG;
% GT24         & GT only
% GT365        & GT only w/o scenario
% 1GT+WT+DG    & 1 GT + BESS + WT + DG
% 2GT+WT+DG    & 2 GT + BESS + WT + DG
% 1GT+REC+DG   & 1 GT + BESS + WT+PV+WEC + DG
% 2GT+REC+DG   & 2 GT + BESS + WT+PV+WEC + DG
% WT+DG        & WT + BESS + DG
% REC-U        & WT+PV+WEC (uncstr.) + BESS
% REC-U+DG     & WT+PV+WEC (uncstr.) + BESS + DG
% REC-C+DG24   & WT+PV+WEC (cstr.) + BESS + DG w/ scenario
% REC-C+DG365  & WT+PV+WEC (cstr.) + BESS + DG w/o scenario
% DG+REC24     & WT+PV+WEC (cstr.) + BESS + DIESEL w/ scenario
% DG+REC365    & WT+PV+WEC (cstr.) + BESS + DIESEL w/o scenario

case_sim_vec = {'WT+DG', 'REC-U', 'REC-U+DG', 'REC-C+DG24'};
case_sim_vec = {'WT+DG'};
num_sim = length(case_sim_vec);
```

Figure 5: Configuration selection

- `PVCost`: scaling factor for the nominal PV cost

```matlab
%  ____                                      _
% / ___|_      _____  ___ _ __    ___  ___| |
% \___ \ \ /\ / / _ \/ _ \ '_ \  / __|/ _ \ |
%  ___) \ V  V /  __/  __/ |_) | \__ \  __/ |_
% |____/ \_/\_/ \___|\___| .__/  |___/\___|_(_)
%                        |_|

% decide which kind of sweep to perform
% None -> No sweep, only 1 simulation
% beta -> Change the CVaR control parameter
% LPSP -> Change the LPSP
% PS   -> Change the power of the PS
% Carbon_tax -> Change the carbon tax
% DGpower -> Change the rated power of the DG
sweep_type  = 'None';
alpha_vec   = 0.8;  % [0.1, 0.5, 0.9];
num_alpha   = size(alpha_vec, 2);
switch sweep_type
  case 'None'
    sweep_vec = 0;
  case 'beta'
    sweep_vec = [0:0.25:1];
  case 'LPSP'
    sweep_vec = [0:0.01:0.05];%[0, 0.1, 0.5, 0.9];
  case 'PS'
    sweep_vec = [0:0.025:0.1];
  case 'Carbon_tax'
    sweep_vec = [1,10,100,1000];
  case 'DGpower'
    sweep_vec = [0.01,0.1,1/3,2/3,1,1.1,1.5,2:1:5,5.5:0.5:8,9,10];
  case 'PVCost'
    sweep_vec = [1,0.7,0.5];
  otherwise
    error('Unknown sweep type')
end
num_sweep = size(sweep_vec, 2);
```

Figure 6: Sweep type selection

3.7 Main Loop

Loop over configurations and sweeps Figure 7. Initially `do_power` computes generated power given the load series and converter models (Figure 8), then `optimization_setup` sets up the optimization parameters for the current case, and finally `CASE_optimization` builds and solves the optimization problem.

```
REC_obj(1:num_alpha,1:num_sweep)      = REC;
scenario_mat(1:num_alpha,1:num_sweep) = scenario;
opt_parameters_vec(1:num_sim)         = opt_parameters;
prob_scens                            = cell(num_alpha, num_sweep);
res_scens                             = cell(num_alpha, num_sweep);
load_scens                            = cell(num_alpha, num_sweep);
REC_tmp                               = cell(num_alpha, num_sweep);
LselScens_N                           = cell(num_alpha, num_sweep);
time_sim                              = cell(num_alpha, num_sweep, num_sim);
seed_increment                        = zeros(num_sweep);
values_solution                       = cell(num_alpha, num_sweep, num_sim);
values_vector                         = cell(num_alpha, num_sweep, num_sim);
values_minvalue                       = cell(num_alpha, num_sweep, num_sim);
CO2_emitted                           = cell(num_alpha, num_sweep, num_sim);
for b = 1:num_sweep
    converged_a = false(num_alpha,1);
  for a = 1:num_alpha

    res_scens{a,b}{1}  = wind.iniVec;
    res_scens{a,b}{2}  = irradiance.iniVec;
    res_scens{a,b}{3}  = reshape(met_data_swh, [], 1);
    res_scens{a,b}{4}  = reshape(met_data_mwp, [], 1);
    load_scenarios.iniVec = LoadA.iniVec;
    load_scens{a,b}    = load_scenarios;
    prob_scens{a,b}    = 1/size(load_scenarios.iniVec, 2)*ones(size(load_scenarios.iniVec, 2), 1);

    [REC_tmp{a,b}] = do_power(REC_obj(a,b), res_scens{a,b});

    for i=1:num_sim
      tic;
      case_sim = case_sim_vec{i};

      % Optimization setup
      [REC_el, load_scenarios_el, opt_parameters_el, scenario_tmp] = optimization_setup('None', alpha_vec, sweep_type, sweep_vec, a, b, REC_tmp{a,b}, load_scens{a,b}, scenario_mat(a, b), opt_parameters_vec(i), 0, case_sim);

      % Run optimization
      [values_solution{a,b,i}, values_vector{a,b,i}, values_minvalue{a,b,i}, CO2_emitted{a,b,i}] = CASE_optimization(prob_scens{a,b}, REC_el, ESS, DL, load_scenarios_el, opt_parameters_el, scenario_tmp(a,b), case_sim);

      % Track time and convergence
      time_sim{a,b,i} = toc;
      fprintf('Simulation, case=%s, a=%4d, b=%3d, time=%5.f [s], stop due to %s\n, convergence=%5.2f\n', case_sim, alpha_vec(a), sweep_vec(b), time_sim{a,b,i}, values_solution{a,b,i}.min_info.message,values_solution{a,b,i}.min_info.relativegap);

    end
  end
end
```

Figure 7: Main loop

```
res_scens{a,b}{1}  = wind.iniVec;
res_scens{a,b}{2}  = irradiance.iniVec;
res_scens{a,b}{3}  = reshape(met_data_swh, [], 1);
res_scens{a,b}{4}  = reshape(met_data_mwp, [], 1);
load_scenarios.iniVec = LoadA.iniVec;
load_scens{a,b}    = load_scenarios;
prob_scens{a,b}    = 1/size(load_scenarios.iniVec, 2)*ones(size(load_scenarios.iniVec, 2), 1);

[REC_tmp{a,b}] = do_power(REC_obj(a,b), res_scens{a,b});
```

Figure 8: Power computation from datasets

4 Optimization: CASE_optimization

4.1 General Notes

Optimization problem formulated as a *problem-based* approach[1].
Different cases handled via `switch` statements.

4.2 Structure

1. Extract information from input
2. Define variables
3. Define costs
4. Compose objective function
5. Define constraints
6. Solve optimization

[1] `https://uk.mathworks.com/help/optim/problem-based-approach.html`

7. Post-process results

4.3 Information Extraction

Defines installed devices, load, and cost annualization factors Figure 9.

```
% Annualization-related variables
r = opt_parameters.r; % daily interest rate
L = opt_parameters.L; % investment lifetime
p = (365*24)*L/scenario.h_star; % recoupling period
T = scenario.T;  % number of data in each scenario
W = size(Pload, 2); % number of scenarios (e.g. days)
AF.CRF     = (r*(1 + r)^p)/((1 + r)^p - 1);  % capital recovery factor (daily value
AF.gamma   = scenario.h_star/scenario.h;      % rescale the scenario to cost to dail
AF.YtD     = scenario.d_o;                    % rescale from yearly to daily cost
```

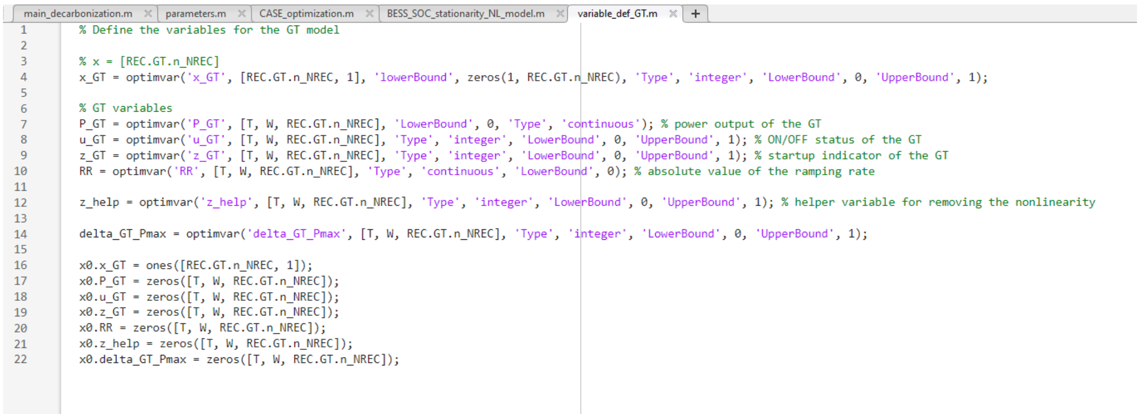
Figure 9: Cost annualization factors

4.4 Variable Definition

Variables for each device defined in its own function Figure 10, as for example in the GT case in Figure 11:.

```
% __     __         _       _     _
% \ \   / /_ _ _ __(_) __ _| |__ | | ___  ___
%  \ \ / / _` | '__| |/ _` | '_ \| |/ _ \/ __|
%   \ V / (_| | |  | | (_| | |_) | |  __/\__ \
%    \_/ \__,_|_|  |_|\__,_|_.__/|_|\___||___/

Pch   = 0; % charging power of the battery
Pdc   = 0; % discharging power of the battery
prob  = optimproblem;
cvar_zeta = 0; % CVaR constraint
cvar_s = zeros([W, 1]); % CVaR constraint
P_REC_max = 0;
P_GT  = 0;
Pbt   = 0;

switch case_constraint
  case {'GT24','GT365'} % Only GT
    variable_def_GT;
    variable_def_Pns_Pct;

  case {'1GT+WT+DG', '2GT+WT+DG', '1GT+REC+DG', '2GT+REC+DG'} % 1/2 GT + REC + BESS
    variable_def_GT;
    x_GT = ones(REC.GT.n_NREC, 1); % impose the number of GTs
    variable_def_REC;
    variable_def_Pns_Pct;
    if BESS_NL_model == 1
      variable_def_BESS_NL_model;
    else
      variable_def_BESS;
    end

  case {'WT+DG', 'REC-U', 'REC-U+DG', 'REC-C+DG24', 'REC-C+DG365'} % REC + BESS
    variable_def_REC;
    variable_def_Pns_Pct;
    if BESS_NL_model == 1
      variable_def_BESS_NL_model;
    else
      variable_def_BESS;
    end

  case {'DG+REC24', 'DG+REC365'} % REC + BESS + DIESEL
    variable_def_REC;
    variable_def_DGv;
    variable_def_Pns_Pct;
    if BESS_NL_model == 1
      variable_def_BESS_NL_model;
    else
      variable_def_BESS;
    end

  otherwise
    error('Case name not implemented yet\n');

end
```

Figure 10: Variable definitions

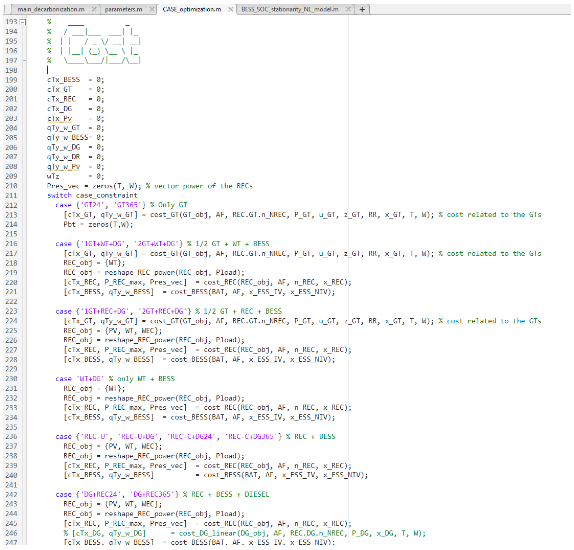


```
% Define the variables for the GT model

% x = [REC.GT.n_NREC]
x_GT = optimvar('x_GT', [REC.GT.n_NREC, 1], 'lowerBound', zeros(1, REC.GT.n_NREC), 'Type', 'integer', 'LowerBound', 0, 'UpperBound', 1);

% GT variables
P_GT = optimvar('P_GT', [T, W, REC.GT.n_NREC], 'LowerBound', 0, 'Type', 'continuous'); % power output of the GT
u_GT = optimvar('u_GT', [T, W, REC.GT.n_NREC], 'Type', 'integer', 'LowerBound', 0, 'UpperBound', 1); % ON/OFF status of the GT
z_GT = optimvar('z_GT', [T, W, REC.GT.n_NREC], 'Type', 'integer', 'LowerBound', 0, 'UpperBound', 1); % startup indicator of the GT
RR = optimvar('RR', [T, W, REC.GT.n_NREC], 'Type', 'continuous', 'LowerBound', 0); % absolute value of the ramping rate

z_help = optimvar('z_help', [T, W, REC.GT.n_NREC], 'Type', 'integer', 'LowerBound', 0, 'UpperBound', 1); % helper variable for removing the nonlinearity

delta_GT_Pmax = optimvar('delta_GT_Pmax', [T, W, REC.GT.n_NREC], 'Type', 'integer', 'LowerBound', 0, 'UpperBound', 1);

x0.x_GT = ones([REC.GT.n_NREC, 1]);
x0.P_GT = zeros([T, W, REC.GT.n_NREC]);
x0.u_GT = zeros([T, W, REC.GT.n_NREC]);
x0.z_GT = zeros([T, W, REC.GT.n_NREC]);
x0.RR = zeros([T, W, REC.GT.n_NREC]);
x0.z_help = zeros([T, W, REC.GT.n_NREC]);
x0.delta_GT_Pmax = zeros([T, W, REC.GT.n_NREC]);
```

Figure 11: Gas Turbine variables

4.5 Cost Definition

Costs defined per asset in stage 1 and stage 2, in separate functions as shown in Figure 12, and visible in the GT example Figure 13. Afterwards, they are composed as in Figure 14.



```
%    ____          _
%   / ___|___  ___| |_
%  | |   / _ \/ __| __|
%  | |__| (_) \__ \ |_
%   \____\___/|___/\__|

cTx_BESS  = 0;
cTx_GT    = 0;
cTx_REC   = 0;
cTx_DG    = 0;
cTx_Pv    = 0;
qTy_w_GT  = 0;
qTy_w_BESS= 0;
qTy_w_DG  = 0;
qTy_w_DR  = 0;
qTy_w_Pv  = 0;
wTz       = 0;
Pres_vec = zeros(T, W); % vector power of the RECs
switch case_constraint
  case {'GT24', 'GT365'} % Only GT
    [cTx_GT, qTy_w_GT] = cost_GT(GT_obj, AF, REC.GT.n_NREC, P_GT, u_GT, z_GT, RR, x_GT, T, W); % cost related to the GTs
    Pbt = zeros(T,W);

  case {'1GT+WT+DG', '2GT+WT+DG'} % 1/2 GT + WT + BESS
    [cTx_GT, qTy_w_GT] = cost_GT(GT_obj, AF, REC.GT.n_NREC, P_GT, u_GT, z_GT, RR, x_GT, T, W); % cost related to the GTs
    REC_obj = {WT};
    REC_obj = reshape_REC_power(REC_obj, Pload);
    [cTx_REC, P_REC_max, Pres_vec]  = cost_REC(REC_obj, AF, n_REC, x_REC);
    [cTx_BESS, qTy_w_BESS]  = cost_BESS(BAT, AF, x_ESS_IV, x_ESS_NIV);

  case {'1GT+REC+DG', '2GT+REC+DG'} % 1/2 GT + REC + BESS
    [cTx_GT, qTy_w_GT] = cost_GT(GT_obj, AF, REC.GT.n_NREC, P_GT, u_GT, z_GT, RR, x_GT, T, W); % cost related to the GTs
    REC_obj = {PV, WT, WEC};
    REC_obj = reshape_REC_power(REC_obj, Pload);
    [cTx_REC, P_REC_max, Pres_vec]  = cost_REC(REC_obj, AF, n_REC, x_REC);
    [cTx_BESS, qTy_w_BESS]  = cost_BESS(BAT, AF, x_ESS_IV, x_ESS_NIV);

  case 'WT+DG' % only WT + BESS
    REC_obj = {WT};
    REC_obj = reshape_REC_power(REC_obj, Pload);
    [cTx_REC, P_REC_max, Pres_vec]  = cost_REC(REC_obj, AF, n_REC, x_REC);
    [cTx_BESS, qTy_w_BESS]  = cost_BESS(BAT, AF, x_ESS_IV, x_ESS_NIV);

  case {'REC-U', 'REC-U+DG', 'REC-C+DG24', 'REC-C+DG365'} % REC + BESS
    REC_obj = {PV, WT, WEC};
    REC_obj = reshape_REC_power(REC_obj, Pload);
    [cTx_REC, P_REC_max, Pres_vec]  = cost_REC(REC_obj, AF, n_REC, x_REC);
    [cTx_BESS, qTy_w_BESS]          = cost_BESS(BAT, AF, x_ESS_IV, x_ESS_NIV);

  case {'DG+REC24', 'DG+REC365'} % REC + BESS + DIESEL
    REC_obj = {PV, WT, WEC};
    REC_obj = reshape_REC_power(REC_obj, Pload);
    [cTx_REC, P_REC_max, Pres_vec]  = cost_REC(REC_obj, AF, n_REC, x_REC);
    % [cTx_DG, qTy_w_DG]      = cost_DG_linear(DG_obj, AF, REC.DG.n_NREC, P_DG, x_DG, T, W);
    [cTx_BESS, qTy_w_BESS]  = cost_BESS(BAT, AF, x_ESS_IV, x_ESS_NIV);
```

Figure 12: Cost definitions

4.6 Constraints

Constraints are imposed according to the utilized devices as in Figure 15. For example, for the GT they are as in Figure 16.

4.7 Solving

After having defined costs and constraints, the problem is solved as in Figure 17.

```matlab
function [cTx, qTy_w] = cost_GT(GT_obj, AF, n_GT, P_GT, u_GT, z_GT, RR, x_GT, T, W)
  % cost related to the GTs

  % 1st stage variable cost
  c = zeros(n_GT, 1);
  for g = 1:n_GT
    c(g) = GT_obj{g}.C_I;
  end

  % 2nd stage variable cost
  % cost associated with the GTs
  for g = 1:n_GT
    qTg = [GT_obj{g}.C_per_watt*ones(T, 1); GT_obj{g}.C_on*ones(T, 1); GT_obj{g}.C_start*ones(T, 1); GT_obj{g}.C_RR*ones(T, 1)]';
    y_G(1,1:W,g)  = qTg*[P_GT(:,:,g);u_GT(:,:,g);z_GT(:,:,g);RR(:,:,g)];
  end

  cTx = AF.CRF*c'*x_GT;
  qTy_w = sum(y_G, 3);
end
```

Figure 13: Gas Turbine cost function

```matlab
% fast_constraint;

[qTy_w_Pns_Pct]     = cost_Pns_Pct(Pns, Pct, T, Pload_obj);
[cTx_Pv, qTy_w_Pv]  = cost_DGv(AF, P_DGv, T, P_DGv_max_inst,  REC.DGv.DGv_obj{1}); % virtual power (i.e. diesel cost without intercept)

cTx = cTx_GT + cTx_BESS + cTx_REC + cTx_DG + cTx_Pv;                 % Total 1st stage cost
qTy_w = qTy_w_GT + qTy_w_BESS + qTy_w_Pns_Pct + qTy_w_DG + qTy_w_DR + qTy_w_Pv; % Total 2nd stage cost

f     = cTx + AF.gamma*qTy_w*LscensProbVec_N;
if num_ESS > 0
  [wTz] = cost_soft_constraint(BAT, AF, epsilon_bat);
  f = f + wTz; % add soft constraint in case of presence of the BESS
end

cvar  = cvar_zeta + (1/(1 - cvar_alpha))*sum(LscensProbVec_N.*cvar_s);
F     = (1 - cvar_beta)*f + cvar_beta*cvar;
prob.Objective  = F;
```

Figure 14: Cost composition

```matlab
%    ____                _             _       _
%   / ___|___  _ __  ___| |_ _ __ __ _(_)_ __ | |_ ___
%  | |   / _ \| '_ \/ __| __| '__/ _` | | '_ \| __/ __|
%  | |__| (_) | | | \__ \ |_| | | (_| | | | | | |_\__ \
%   \____\___/|_| |_|___/\__|_|  \__,_|_|_| |_|\__|___/

Pload_PS = load_PS(Pload, Pps_rem, Pps_add);

switch case_constraint
  case {'GT24','GT365'} % Only GT
    P_gen_tot = sum(P_GT, 3);
    prob = GT_constraints(prob, REC.GT.n_NREC, T, W, GT_obj, u_GT, z_GT, x_GT, P_GT, RR, z_help, case_constraint);

  case {'1GT+WT+DG', '2GT+WT+DG', '1GT+REC+DG', '2GT+REC+DG'} % GT + REC + BESS
    % GT
    P_gen_tot = P_res + sum(P_GT, 3);
    prob = GT_constraints(prob, REC.GT.n_NREC, T, W, GT_obj, u_GT, z_GT, x_GT, P_GT, RR, z_help, case_constraint);

    % REC
    prob = REC_constraints(prob, P_res, P_REC_max, n_REC, REC_obj, x_REC);

    % BESS
    prob = BESS_constraints(prob, BESS_NL_model, E0, Ebt, Pch, Pdc, Pbt, Uch, Udc, PchUch, PdcUdc, BAT, x_ESS_IV, x_ESS_NIV, epsilon_bat);

  case {'WT+DG', 'REC-C+DG24', 'REC-C+DG365'}  % G -> WT + BESS
                          % I -> REC + BESS
    % REC
    P_gen_tot = P_res;
    prob = REC_constraints(prob, P_res, P_REC_max, n_REC, REC_obj, x_REC);

    % BESS
    prob = BESS_constraints(prob, BESS_NL_model, E0, Ebt, Pch, Pdc, Pbt, Uch, Udc, PchUch, PdcUdc, BAT, x_ESS_IV, x_ESS_NIV, epsilon_bat);

  case {'REC-U+DG'} % REC + BESS w/o plant size constraints and possibility to use Pv
    % REC
    P_gen_tot = P_res;
    prob = max_power(prob, 'REC_max_power', P_res, P_REC_max); % Maximum power produced by the REC

    % BESS
    prob = BESS_constraints(prob, BESS_NL_model, E0, Ebt, Pch, Pdc, Pbt, Uch, Udc, PchUch, PdcUdc, BAT, x_ESS_IV, x_ESS_NIV, epsilon_bat);

    case {'REC-U'} % REC + BESS w/o plant size constraints
    % REC
    P_gen_tot = P_res;
    prob = max_power(prob, 'REC_max_power', P_res, P_REC_max); % Maximum power produced by the REC

    % BESS
    prob = BESS_constraints(prob, BESS_NL_model, E0, Ebt, Pch, Pdc, Pbt, Uch, Udc, PchUch, PdcUdc, BAT, x_ESS_IV, x_ESS_NIV, epsilon_bat);

    % Virtual power
    prob.Constraints.Pv_zero = P_DGv == 0;

  case {'DG+REC24', 'DG+REC365'} % REC + BESS + DIESEL
    % Total generated power
    P_gen_tot = P_res + sum(P_DG, 3);
```

Figure 15: General constraints

.

```
main_decarbonization.m | parameters.m | CASE_optimization.m | BESS_SOC_stationarity_NL_model.m | cost_GT.m | GT_constraints.m
function prob = GT_constraints(prob, n_GT, T, W, GT_obj, u_GT, z_GT, x_GT, P_GT, RR, z_help, case_constraint)
% Constraints related to the GT

  switch case_constraint
    case  {'GT24','GT365'}
      prob = GT_offtime(prob, n_GT, T, W, GT_obj, u_GT);
      prob = GT_startUP(prob, n_GT, T, W, z_GT, u_GT);
      prob = GT_P_minmax(prob, n_GT, T, W, P_GT, x_GT, GT_obj, u_GT, z_help);
      prob = GT_rump_UP(prob, n_GT, T, W, P_GT, GT_obj);
      prob = GT_rump_UP_cost_constraint(prob, n_GT, T, W, P_GT, RR);

    case {'1GT+WT+DG', '2GT+WT+DG', '1GT+REC+DG', '2GT+REC+DG'}
      prob = GT_offtime(prob, n_GT, T, W, GT_obj, u_GT);
      prob = GT_startUP(prob, n_GT, T, W, z_GT, u_GT);
      prob = GT_P_minmax_fix_GT_number(prob, n_GT, T, W, P_GT, GT_obj, u_GT);
      prob = GT_rump_UP(prob, n_GT, T, W, P_GT, GT_obj);
      prob = GT_rump_UP_cost_constraint(prob, n_GT, T, W, P_GT, RR);

  end

end
```

Figure 16: Gas Turbine constraints

```
%   ____        _       _   _
%  / ___|  ___ | |_   _| |_(_) ___  _ __
%  \___ \ / _ \| | | | | __| |/ _ \| '_ \
%   ___) | (_) | | |_| | |_| | (_) | | | |
%  |____/ \___/|_|\__,_|\__|_|\___/|_| |_|

fprintf('Start solution\n')

% prob.Constraints.tmp1 = x_REC ==1;
% prob.Constraints.tmp2 = x_ESS_IV == 4e6;
% prob.Constraints.tmp2 = x_ESS_NIV == 4;

opt = optimoptions('intlinprog', 'Display', 'iter', 'AbsoluteGapTolerance', abs_gap_tol, 'RelativeGapTolerance', rel_gap_tol, 'IntegerTolerance', 1e-3, 'MaxTime', 2*3600, 'Heur:
prob.ObjectiveSense = 'minimize';
[values_solution, values_minvalue, tmp, min_info] = solve(prob, 'Options', opt);

values_solution.min_info = min_info;
values_solution.values_minvalue = values_minvalue;

fprintf('Solution ended\n')

if ~isempty(queue)
  % write the messages to be sent to the queue
  tmp_message.case_constraint = case_constraint;
  tmp_message.values_minvalue = values_minvalue;

  send(queue, tmp_message)
end
```

Figure 17: Solving the optimization problem

4.8 Post-Processing

Computes costs including already installed devices, CO_2 emissions, installed power as shown in Figure 18.

```
%  ____           _                                      _
% |  _ \ ___  ___| |_ _ __  _ __ ___   ___ ___  ___ ___(_)_ __   __ _
% | |_) / _ \/ __| __| '_ \| '__/ _ \ / __/ _ \/ __/ __| | '_ \ / _` |
% |  __/ (_) \__ \ |_| |_) | | | (_) | (_|  __/\__ \__ \ | | | | (_| |
% |_|   \___/|___/\__| .__/|_|  \___/ \___\___||___/___/_|_| |_|\__, |
%                    |_|                                        |___/

% Check that the battery does not charge and discharge at the same time
if isfield(values_solution, 'Pch')
  check_charging(values_solution);
end

% Reshape values in a vector
values.n_REC = n_REC;
[values_vector, values_solution] = reshape_data(values_solution, Pload, Pres_vec, BAT);


% % Compute the capacity factors
% values_vector = capacity_factor(values_vector, values_vector, PV, WT, WEC, BAT, W, T);

% Compute the emitted CO2
compute_CO2

% compute the installed power
compute_installed_power;

% Evaluate the costs
compute_final_costs;

% Compute the CVaR
values_solution.CVaR = values_solution.cvar_zeta + (1/(1 - cvar_alpha))*LscensProbVec_N'*values_solution.cvar_s;

% Effective LPSP
values_solution.LPSP_eff = sum(values_vector.Pns)/sum(Pload, 'all');

end
```

Figure 18: Post-processing results

5 Results and Plots

The results can be displayed using `plot_decarbonization` (Figure 19), for producing plots like the one in Figure 20, Figure 21, and Figure 22.

```
    end
  end

  %%
  %  ____  _       _
  % |  _ \| | ___ | |_
  % | |_) | |/ _ \| __|
  % |  __/| | (_) | |_
  % |_|   |_|\___/ \__|


  close all;
  plot_decarbonization
```

Figure 19: Example plot layout

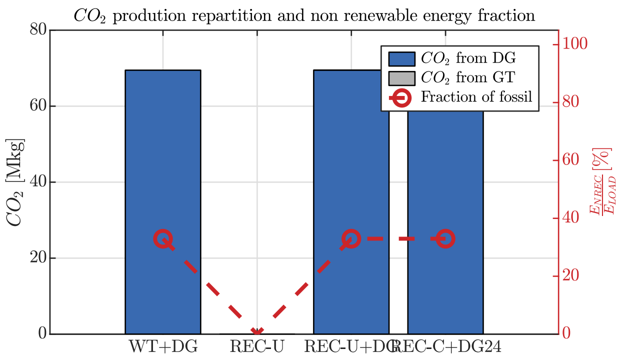


Figure 20: CO_2 fraction

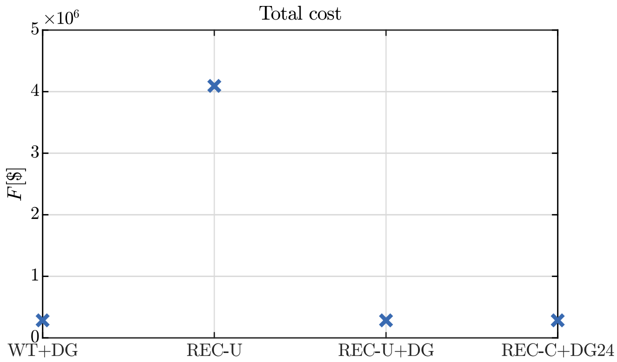


Figure 21: Total cost

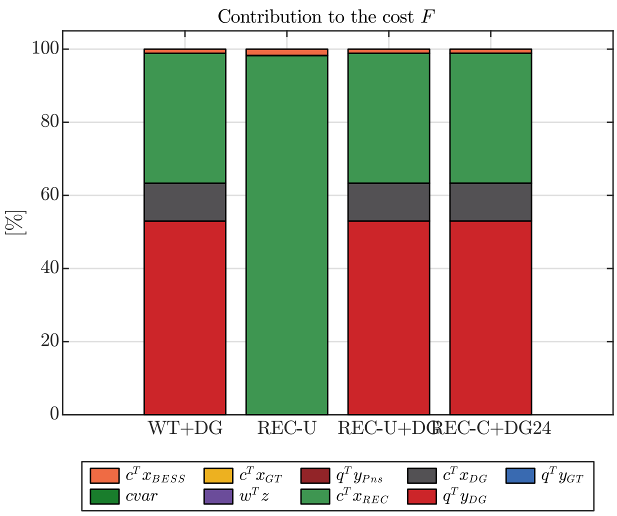


Figure 22: Cost fraction breakdown