Solving Nested CES Demand Problems (CRS)

Taking advantage of **bfw_crs_nested_ces** from the **PrjLabEquiBFW Package.** This function solves optimal choices in Constant Elasticity of Substitution problems with constant returns and two inputs in each sub-nest. Takes as inputs share and elasticity parameters across layers of sub-nests, as well as input unit costs at the bottom-most layer. Works for CES problems with up to four nest layers, and allows for uneven branches, so that some branches go up to four layers, but others have less layers. Works with BFW (2022) nested labor input problem.

Key Inputs and Outputs for bfw_mp_func_demand

Here are the key inputs for the CES demand solver function:

- **FL_YZ** float output divided by productivity, aggregate single term
- **CL_MN_PRHO** cell array of rho (elasticity) parameter between negative infinity and 1. For example, suppose there are four nest layers, and there are two branches at each layer, then we have 1, 2, 4, and 8 ρ parameter values at the 1st, 2nd, 3rd, and 4th nest layers: size(CL_MN_PRHO{1})= $[1,1]$, size(CL_MN_PRHO{2}) = $[1,2]$, size(CL_MN_PRHO{3}) = $[2,2]$, size(CL_MN_PRHO{4}) = $[2,2,2]$. Note that if the model has 4 nest layers, not all cells need to be specified, some branches could be deeper than others.
- **CL_MN_PSHARE** cell array of share (between 0 and 1) for the first input of the two inputs for each nest. The structure for this is similar to CL_MN_PRHO.
- **CL_MN_PRICE** cell array of wages for both wages for the first and second nest, the last index in each element of the cell array indicates first (1) or second (2) wage. For example, suppose we have four layers, with 2 branches at each layer, as in the example for CL_MN_PRHO, then we have 2, 4, 8, and 16 wage values at the 1st, 2nd, 3rd, and 4th nest layers: size(CL_MN_PRICE{1}) = $[1,2]$, size(CL_MN_PRICE{2}) = $[2,2]$, size(CL_MN_PRICE{3}) = $[2,2,2]$, size(CL_MN_PRICE{4}) = $[2,2,2,2]$. Note that only the last layer of wage needs to be specified, in this case, the 16 wages at the 4th layer. Given optimal solutions, we solve for the 2, 4, and 8 aggregate wages at the higher nest layers. If some branches are deeper than other branches, then can specific NA for non-reached layers along some branches.
- **BL_BFW_MODEL** boolean true by default if true then will output outcomes specific to the BFW 2022 problem.

Here are the key outputs for the CES demand solver function:

- **CL_MN_YZ_CHOICES** has the same dimension as CL_MN_PRICE, suppose there are four layers, the CL_MN_PRICE{4} results at the lowest layer includes quantity choices that might be observed in the data. CL_MN_PRICE cell values at non-bottom layers include aggregate quantity outcomes.
- **CL_MN_PRICE** includes at the lowest layer observed wages, however, also includes higher layer aggregate solved waves. CL_MN_PRHO and CL_MN_PSHARE are identical to inputs.

Single Nest Layer Two Inputs CES Problem

In this first example, we solve a constant returns to scale problem with a single nest, meaning just two inputs and a single output.

```
clc;
close all;
clear all;

% Output requirement
fl_yz = 1;
% rho = 0.5, 1/(1-0.5)=2, elasticity of substitution of 2
cl_mn_prho = {[0.1]};
% equal share, similar "productivity"
cl_mn_pshare = {[0.5]};
% wages for the two inputs, identical wage
cl_mn_price = {[1, 1]};
% print option
bl_verbose = true;
mp_func = bfw_mp_func_demand();
bl_bfw_model = false;
[cl_mn_yz_choices, cl_mn_price, cl_mn_prho, cl_mn_pshare] = ...
    bfw_crs_nested_ces(fl_yz, cl_mn_prho, cl_mn_pshare, cl_mn_price, ...
    mp_func, bl_verbose, bl_bfw_model);
```

```
----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_container_map ND Array (Matrix etc)
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                i    idx    ndim    numel    rowN    colN    sum    mean    std    coefvari    min    max
                _    ___    ____    _____    ____    ____    ___    ____    ___    ________    ___    ___

    price_c1    1     2      2        2       1       2       2      1       0        0         1      1
    yz_c1       2     4      2        2       1       2       2      1       0        0         1      1

xxx TABLE:price_c1 xxxxxxxxxxxxxxxxxx
          c1    c2
          __    __

    r1    1     1

xxx TABLE:yz_c1 xxxxxxxxxxxxxxxxxx
          c1    c2
          __    __

    r1    1     1

----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_container_map Scalars
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 i    idx    value
                 _    ___    _____

    prho_c1      1     1      0.1
    pshare_c1    2     3      0.5
```

Single Nest Layer Two Inputs CES Problem, Vary Share and Elasticity

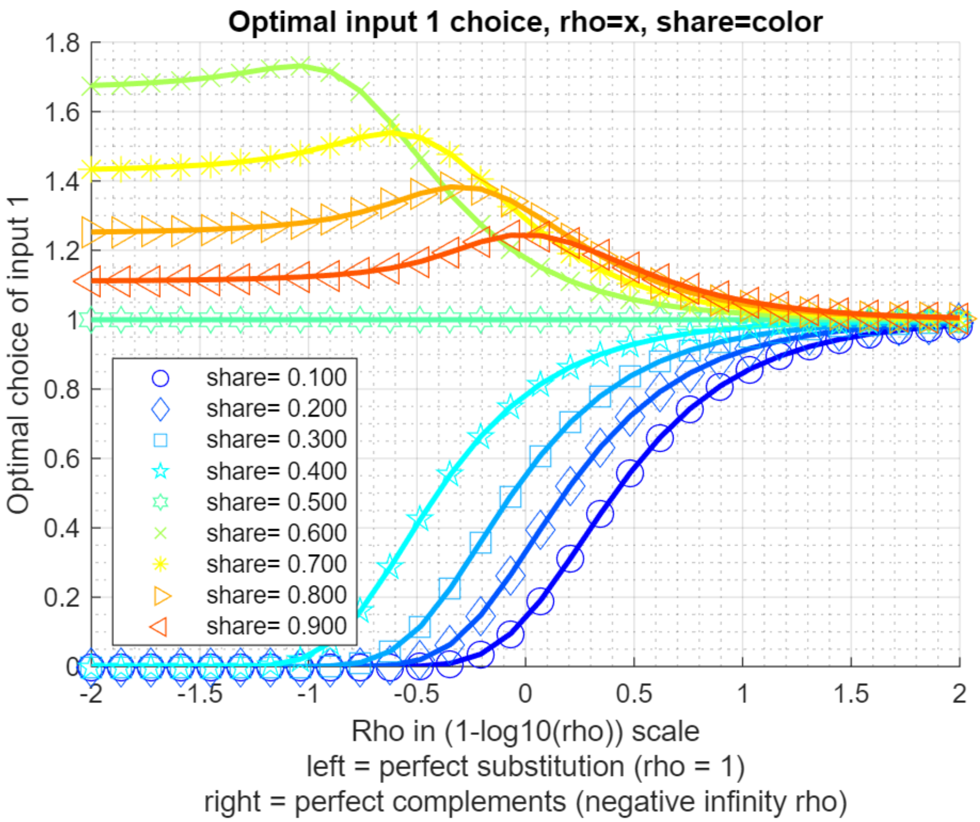
In this second example, we test over different rho values, explore optimal relative choices, as share and elasticity change. In this exercise, we also check, at every combination of rho and share parameter, whether the FOC condition is satisfied by the optimal choices. Also check if at the optimal choices, the minimization output requirement is met.

```
% Approximately close function
rel_tol=1e-09;
abs_tol=0.0;
if_is_close = @(a,b) (abs(a-b) <= max(rel_tol * max(abs(a), abs(b)), abs_tol));

% Define share and rho arrays
fl_yz = 1;
ar_pshare = linspace(0.1, 0.9, 9);
ar_prho = 1 - 10.^(linspace(-2, 2, 30));
% Loop over share and rho values
mt_rela_opti = NaN([length(ar_pshare), length(ar_prho)]);
mt_x1_opti = NaN([length(ar_pshare), length(ar_prho)]);
for it_pshare_ctr = 1:length(ar_pshare)
    for it_prho_ctr = 1:length(ar_prho)

        % A. Parameters
        % rho
        fl_prho = ar_prho(it_prho_ctr);
        cl_mn_prho = {[fl_prho]};
        % share
        fl_pshare = ar_pshare(it_pshare_ctr);
        cl_mn_pshare = {[fl_pshare]};
        % wages for the two inputs, identical wage
        cl_mn_price = {[1, 1]};
        % print option
        bl_verbose = false;

        % B. Call function
        [cl_mn_yz_choices, cl_mn_price, cl_mn_prho, cl_mn_pshare] = ...
            bfw_crs_nested_ces(fl_yz, cl_mn_prho, cl_mn_pshare, cl_mn_price, ...
            mp_func, bl_verbose, bl_bfw_model);
        % Store results for optimal choice
        fl_opti_x1 = cl_mn_yz_choices{1}(1);
        fl_opti_x2 = cl_mn_yz_choices{1}(2);
        mt_x1_opti(it_pshare_ctr, it_prho_ctr) = fl_opti_x1;

        % C. Check if relative optimality FOC condition is met
        fl_rela_opti = fl_opti_x1/fl_opti_x2;
        % From FOC give wages = 1 both
        % Using What is above Equation A.20 in draft.
        fl_rela_opti_foc = (((fl_pshare/(1-fl_pshare)))*(1))^(1/(1-ar_prho(it_prho_ctr)));
        if (~if_is_close(fl_rela_opti_foc, fl_rela_opti))
            error('There is an error, optimal relative not equal to expected foc ratio')
        end

        % D. Check if output quantity requirement is met
        fl_output = ((fl_pshare)*fl_opti_x1^(fl_prho) + (1-fl_pshare)*fl_opti_x2^(fl_prho))^(1/
        if (~if_is_close(fl_output, fl_yz))
```

```
            error('There is an error, output is not equal to required expenditure minimizing ou
        end

    end
end
```

Key results: (1) As share of input 1 goes to zero, optimal choice goes to zero when inputs are elastic; (2) When inputs are inelasticty, even very low share input 1 asymptote to equal input 2; (3) When input 1 is more productive (higher share), actually hire less as productivity (share) increases, becasue less of it is needed to achieve production for high rho, elastictic production function; (4) For inelastic production, monotonic relationship between input and shares.

```
% Visualize
% Generate some Data
rng(456);
ar_row_grid = ar_pshare;
ar_col_grid = log(1-ar_prho)/log(10);
rng(123);
mt_value = mt_x1_opti;
% container map settings
mp_support_graph = containers.Map('KeyType', 'char', 'ValueType', 'any');
mp_support_graph('cl_st_graph_title') = {'Optimal input 1 choice, rho=x, share=color'};
mp_support_graph('cl_st_ytitle') = {'Optimal choice of input 1'};
mp_support_graph('cl_st_xtitle') = {'Rho in (1-log10(rho)) scale', ...
    'left = perfect substitution (rho = 1)', ...
    'right = perfect complements (negative infinity rho)'};
mp_support_graph('st_legend_loc') = 'southwest';
mp_support_graph('bl_graph_logy') = false; % do not log
mp_support_graph('st_rowvar_name') = 'share=';
mp_support_graph('it_legend_select') = 5; % how many shock legends to show
mp_support_graph('st_rounding') = '6.3f'; % format shock legend
mp_support_graph('cl_colors') = 'jet'; % any predefined matlab colormap
% Call function
ff_graph_grid(mt_value, ar_row_grid, ar_col_grid, mp_support_graph);
```



Doubly Nest Layer Two Inputs Each Sub-nest CES Problem

In this third example, solve for optimal choices for a doubly nested problem. Below, we first solve for the optimal choices, then we do a number of checks, to make sure that the solutions are correct, as expected.

```
% output requirement
fl_yz = 2.1;
% upper nest 0.1, lower nests 0.35 and -1 separately for rho values
cl_mn_prho = {[0.1], [0.35, -1]};
% unequal shares of share values
cl_mn_pshare = {[0.4], [0.3, 0.88]};
% differential wages
% in lower-left nest, not productive and very expensive, not very elastic
% last index for left or right,
cl_mn_price = {[nan, nan], [10, 1;3, 4]};
% print option
bl_verbose = true;
[cl_mn_yz_choices, cl_mn_price, cl_mn_prho, cl_mn_pshare] = ...
    bfw_crs_nested_ces(fl_yz, cl_mn_prho, cl_mn_pshare, cl_mn_price, ...
    mp_func, bl_verbose, bl_bfw_model);
```

```
----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_container_map ND Array (Matrix etc)
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                i    idx    ndim    numel    rowN    colN     sum      mean      std      coefvari      min
                _    ___    ____    _____    ____    ____    ______    ______    _______    ________    ________
```

prho_c2	1	2	2	2	1	2	-0.65	-0.325	0.95459	-2.9372	-1
price_c1	2	3	2	2	1	2	7.7788	3.8894	2.0959	0.53886	2.4074
price_c2	3	4	2	4	2	2	18	4.5	3.873	0.86066	1
pshare_c2	4	6	2	2	1	2	1.18	0.59	0.41012	0.69512	0.3
yz_c1	5	7	2	2	1	2	4.4862	2.2431	0.68863	0.307	1.7561
yz_c2	6	8	2	4	2	2	9.0506	2.2626	2.7086	1.1971	0.047893

xxx TABLE:prho_c2 xxxxxxxxxxxxxxxxxx

	c1	c2
r1	0.35	-1

xxx TABLE:price_c1 xxxxxxxxxxxxxxxxxx

	c1	c2
r1	2.4074	5.3714

xxx TABLE:price_c2 xxxxxxxxxxxxxxxxxx

	c1	c2
r1	10	1
r2	3	4

xxx TABLE:pshare_c2 xxxxxxxxxxxxxxxxxx

	c1	c2
r1	0.3	0.88

xxx TABLE:yz_c1 xxxxxxxxxxxxxxxxxx

	c1	c2
r1	2.73	1.7561

xxx TABLE:yz_c2 xxxxxxxxxxxxxxxxxx

	c1	c2
r1	0.047893	6.0934
r2	2.2044	0.70496

----
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_container_map Scalars
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

	i	idx	value
prho_c1	1	1	0.1
pshare_c1	2	5	0.4

```
% there are four optimal choices, they are
fl_opti_x11 = cl_mn_yz_choices{2}(1,1);
fl_opti_x12 = cl_mn_yz_choices{2}(1,2);
fl_opti_x21 = cl_mn_yz_choices{2}(2,1);
fl_opti_x22 = cl_mn_yz_choices{2}(2,2);
% display
st_print = strjoin(...
    ["completed double nest test:", ...
```

```
    ['nest 1 input 1, fl_opti_x11=' num2str(fl_opti_x11)], ...
    ['nest 1 input 2, fl_opti_x12=' num2str(fl_opti_x12)], ...
    ['nest 2 input 1, fl_opti_x21=' num2str(fl_opti_x21)], ...
    ['nest 2 input 2, fl_opti_x22=' num2str(fl_opti_x22)], ...
    ], ";");
st_out = st_print;
ar_ch_out = char(strsplit(st_print,";")');
disp(ar_ch_out);
```

```
completed double nest test:
nest 1 input 1, fl_opti_x11=0.047893
nest 1 input 2, fl_opti_x12=6.0934
nest 2 input 1, fl_opti_x21=2.2044
nest 2 input 2, fl_opti_x22=0.70496
```

Doubly Nest Layer Two Inputs Each Sub-nest CES Problem--Solution Check

Checking output equality, if there are problems, would output an error.

```
% A. Check output Equality
fl_pshare_0 = cl_mn_pshare{1}(1);
fl_pshare_1 = cl_mn_pshare{2}(1);
fl_pshare_2 = cl_mn_pshare{2}(2);
fl_prho_0 = cl_mn_prho{1}(1);
fl_prho_1 = cl_mn_prho{2}(1);
fl_prho_2 = cl_mn_prho{2}(2);
fl_output_1 = ((fl_pshare_1)*fl_opti_x11^(fl_prho_1) + (1-fl_pshare_1)*fl_opti_x12^(fl_prho_1)
fl_output_2 = ((fl_pshare_2)*fl_opti_x21^(fl_prho_2) + (1-fl_pshare_2)*fl_opti_x22^(fl_prho_2)
fl_output_0 = ((fl_pshare_0)*fl_output_1^(fl_prho_0) + (1-fl_pshare_0)*fl_output_2^(fl_prho_0)
if (~if_is_close(fl_output_0, fl_yz))
    error('There is an error, output is not equal to required expenditure minimizing output')
end
```

Checking FOC within-nest optimality, if there are problems, would output an error.

```
% B. Check FOC Optimality inner nest
fl_wage_x11 = cl_mn_price{2}(1,1);
fl_wage_x12 = cl_mn_price{2}(1,2);
fl_wage_x21 = cl_mn_price{2}(2,1);
fl_wage_x22 = cl_mn_price{2}(2,2);

% B1. Checking via Method 1
fl_rela_opti_foc_1 = (((fl_pshare_1/(1-fl_pshare_1)))*(fl_wage_x12/fl_wage_x11))^(1/(1-fl_prho_
fl_rela_opti_foc_2 = (((fl_pshare_2/(1-fl_pshare_2)))*(fl_wage_x22/fl_wage_x21))^(1/(1-fl_prho_
if (~if_is_close(fl_rela_opti_foc_1, fl_opti_x11/fl_opti_x12))
    error('B1. There is an error, optimal relative not equal to expected foc ratio, nest 1')
end
if (~if_is_close(fl_rela_opti_foc_2, fl_opti_x21/fl_opti_x22))
    error('B1. There is an error, optimal relative not equal to expected foc ratio, nest 2')
end

% B2. Equation left to right, right to left, checking via method 2
% Check FOC Optimality cross nests (actually within) T1
fl_dy_dx11 = fl_pshare_1*(fl_opti_x11^(fl_prho_1-1));
```

```
fl_dy_dx12 = (1-fl_pshare_1)*(fl_opti_x12^(fl_prho_1-1));
fl_rwage_x11dx12 = fl_dy_dx11/fl_dy_dx12;
if (~if_is_close(fl_rwage_x11dx12, fl_wage_x11/fl_wage_x12))
    error('B2. There is an error, relative price x11 and x12 does not satisfy within optimality
end
```

Generate aggregate prices, if there are problems, would output an error.

```
% C. Aggregate prices and optimality within higher tier
% Is optimality satisfied given aggregate prices?
fl_rela_wage_share_11 = ...
    ((fl_wage_x11/fl_wage_x12)*((1-fl_pshare_1)/(fl_pshare_1)))^(fl_prho_1/(1-fl_prho_1));
fl_rela_wage_share_12 = ...
    ((fl_wage_x12/fl_wage_x11)*((fl_pshare_1)/(1-fl_pshare_1)))^(fl_prho_1/(1-fl_prho_1));
fl_agg_prc_1 = ...
    fl_wage_x11*(fl_pshare_1 + (1-fl_pshare_1)*(fl_rela_wage_share_11))^(-1/fl_prho_1) + ...
    fl_wage_x12*(fl_pshare_1*(fl_rela_wage_share_12) + (1-fl_pshare_1))^(-1/fl_prho_1);

fl_rela_wage_share_21 = ...
    ((fl_wage_x21/fl_wage_x22)*((1-fl_pshare_2)/(fl_pshare_2)))^(fl_prho_2/(1-fl_prho_2));
fl_rela_wage_share_22 = ...
    ((fl_wage_x22/fl_wage_x21)*((fl_pshare_2)/(1-fl_pshare_2)))^(fl_prho_2/(1-fl_prho_2));
fl_agg_prc_2 = ...
    fl_wage_x21*(fl_pshare_2 + (1-fl_pshare_2)*(fl_rela_wage_share_21))^(-1/fl_prho_2) + ...
    fl_wage_x22*(fl_pshare_2*(fl_rela_wage_share_22) + (1-fl_pshare_2))^(-1/fl_prho_2);

% What is returned by the omega function that is suppose to have aggregate prices?
mp_func = bfw_mp_func_demand();
params_group = values(mp_func, {'fc_OMEGA', 'fc_d1', 'fc_d2'});
[fc_OMEGA, fc_d1, fc_d2] = params_group{:};

% Aggregate price
fl_aggregate_price_1 = fc_OMEGA(...
    fl_wage_x11, fl_wage_x12, ...
    fl_prho_1, ...
    fl_pshare_1, 1 - fl_pshare_1);

fl_aggregate_price_2 = fc_OMEGA(...
    fl_wage_x21, fl_wage_x22, ...
    fl_prho_2, ...
    fl_pshare_2, 1 - fl_pshare_2);
```

Check relative price within nest and across nests, if there are problems, would output an error.

```
% D. Check FOC Optimality cross nests

% D1a. Two within-nest relative wages and four cross-nest relative wages
% within
fl_rwage_x11dx12 = fl_wage_x11/fl_wage_x12;
fl_rwage_x21dx22 = fl_wage_x21/fl_wage_x22;
% across
fl_rwage_x11dx21 = fl_wage_x11/fl_wage_x21;
fl_rwage_x11dx22 = fl_wage_x11/fl_wage_x22;
fl_rwage_x12dx21 = fl_wage_x12/fl_wage_x21;
```

```
fl_rwage_x12dx22 = fl_wage_x12/fl_wage_x22;

% D1b. Generate relative wages within nest and across nests own equations
fl_dy_dx1_shared = (fl_pshare_0*(fl_output_1)^(fl_prho_0-1))*((fl_output_1)^(1-fl_prho_1));
fl_dy_dx11 = fl_dy_dx1_shared*(fl_pshare_1*fl_opti_x11^(fl_prho_1-1));
fl_dy_dx12 = fl_dy_dx1_shared*((1-fl_pshare_1)*fl_opti_x12^(fl_prho_1-1));

fl_dy_dx2_shared = ((1-fl_pshare_0)*(fl_output_2)^(fl_prho_0-1))*((fl_output_2)^(1-fl_prho_2))
fl_dy_dx21 = fl_dy_dx2_shared*(fl_pshare_2*fl_opti_x21^(fl_prho_2-1));
fl_dy_dx22 = fl_dy_dx2_shared*((1-fl_pshare_2)*fl_opti_x22^(fl_prho_2-1));

% within
fl_rwage_x11dx12_foc = fl_dy_dx11/fl_dy_dx12;
fl_rwage_x21dx22_foc = fl_dy_dx21/fl_dy_dx22;
% across
fl_rwage_x11dx21_foc = fl_dy_dx11/fl_dy_dx21;
fl_rwage_x11dx22_foc = fl_dy_dx11/fl_dy_dx22;
fl_rwage_x12dx21_foc = fl_dy_dx12/fl_dy_dx21;
fl_rwage_x12dx22_foc = fl_dy_dx12/fl_dy_dx22;

if (~if_is_close(fl_rwage_x11dx21_foc, fl_wage_x11/fl_wage_x21))
    error('There is an error, relative price x11 and x21 does not satisfy cross optimality acro
end
if (~if_is_close(fl_rwage_x12dx22_foc, fl_wage_x12/fl_wage_x22))
    error('There is an error, relative price x12 and x22 does not satisfy cross optimality acro
end

% D2. Check FOC Optimality cross nests, simplified equation
fl_rela_wage_x11_x21 = log((fl_pshare_0/(1-fl_pshare_0))* ...
    ((fl_pshare_1*fl_opti_x11^(fl_prho_1-1)*fl_output_2^(fl_prho_2))/(fl_pshare_2*fl_opti_x21^(
    fl_prho_0*log(fl_output_1/fl_output_2);
if (~if_is_close(fl_rela_wage_x11_x21, log(fl_wage_x11/fl_wage_x21)))
    error('There is an error, relative price x11 and x21 does not satisfy cross optimality acro
end
```