BFW Equilibrium Gender LFP and Wage Code Companion

Sonia R. Bhalotra, Manuel Fernández, and Fan Wang

2022-02-19

Contents

Preface

This is a work-in-progress Matlab package consisting of functions that solve the equilibrium gender labor force participation and wage model in Bhalotra, Fernández and Wang (2022). Tested with Matlab 2021b (The MathWorks Inc, 2021).

All functions are parts of a matlab toolbox that can be installed:

> Download and install the Matlab toolbox: PrjLabEquiBFW.mltbx

The Code Companion can also be accessed via the bookdown site and PDF linked below:

> **bookdown site**, **bookdown pdf**, **MathWorks File Exchange**

This bookdown file is a collection of mlx based vignettes for functions that are available from PrjLabEquiBFW. Each Vignette file contains various examples for invoking each function.

The package relies on MEconTools, which needs to be installed first. The package does not include allocation functions, only simulation code to generate the value of each stimulus check increments for households.

The site is built using Bookdown (Xie, 2020).

Please contact FanWangEcon for issues or problems.

Chapter 1

Introduction

1.1 Bhalotra, Fernández, and Wang (2022)

In Bhalotra, Fernández, and Wang (2022).

Chapter 2

Core Functions

2.1 CES Demand Core Functions

This is the example vignette for function: **bfw_mp_func_demand** from the **PrjLabEquiBFW Package.** This function generates a container map with key CES demand-side equations for a particular sub-nest.

2.1.1 Default Test

Default test

```
bl_verbose = true;
mp_func_demand = bfw_mp_func_demand(bl_verbose);
```

```
----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_func Functions
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                                i       idx
                               ----    ----    ---------------------------------------------------------

    fc_OMEGA                   "1"     "1"     "@(p1,p2,rho,beta_1,beta_2)p1.*fc_d1(p1,p2,1,1,rho,be
    fc_d1                      "2"     "2"     "@(p1,p2,Y,Z,rho,beta_1,beta_2)(Y/Z).*(beta_1+beta_2.
    fc_d2                      "3"     "3"     "@(p1,p2,Y,Z,rho,beta_1,beta_2)(Y/Z).*(beta_1.*((p2./
    fc_lagrange_x1             "4"     "4"     "@(p1,rho,beta_1,beta_2,x_1,x_2)p1/(((beta_1*x_1^(rho
    fc_lagrange_x2             "5"     "5"     "@(p2,rho,beta_1,beta_2,x_1,x_2)p2/(((beta_1*x_1^(rho
    fc_output_nest             "6"     "6"     "@(q1,q2,rho,beta_1,beta_2)((beta_1)*q1^(rho)+beta_2*
    fc_p1_foc                  "7"     "7"     "@(lagrangem,rho,beta_1,beta_2,x_1,x_2)lagrangem*(((b
    fc_p2_foc                  "8"     "8"     "@(lagrangem,rho,beta_1,beta_2,x_1,x_2)lagrangem*(((b
    fc_share_given_elas_foc    "9"     "9"     "@(rho,p1,p2,x1,x2)fc_share_given_elas_foc_Q(rho,p1,p
    fc_w1dw2                   "10"    "10"    "@(x_1,x_2,rho,beta_1,beta_2)(x_2/x_1)^(1-rho)*(beta_
    fc_yz_ratio                "11"    "11"    "@(p1,p2,q1,q2,rho,beta_1,beta_2)fc_revenue(p1,p2,q1,
```

2.2 Multinomial Logit Core Functions

This is the example vignette for function: **bfw_mp_func_supply** from the **PrjLabEquiBFW Package.** This function generates a container map with key multinomial logit supply-side equations.

2.2.1 Test BL_LOG_WAGE is false

Default test

```
bl_log_wage = false;
```

```
bl_verbose = true;
mp_func_supply = bfw_mp_func_supply(bl_log_wage, bl_verbose);
```

```
----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_func Functions
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                         i      idx
                        ---     ---    ------------------------------------------------------------

    fc_ar_prob_wrk      "1"     "1"    "@(arpsi0,psi1,mtwage,probdenom)fc_v_occ(reshape(arpsi0,[1,len
    fc_log_pmdpo_occ    "2"     "2"    "@(psi0,psi1,arwage,pie1,pie2,pie3,pie4,pie5,pie6,t,prbchd,prb
    fc_prob_denom       "3"     "3"    "@(arpsi0,psi1,arpie,arwage1,arwage2,arwage3,t,prbchd,prbmar,p
    fc_prob_lei         "4"     "4"    "@(arpie,t,prbchd,prbmar,prbapp,prbjsy,probdenom)fc_v_lei(arpi
    fc_s1               "5"     "5"    "@(p1,G_1,zeta_1_0,zeta_1_1)G_1./(1+(exp(-zeta_1_0-zeta_1_1.*p
    fc_s2               "6"     "6"    "@(p2,G_2,zeta_2_0,zeta_2_1)G_2./(1+(exp(-zeta_2_0-zeta_2_1.*p
    fc_supply           "7"     "7"    "@(potlabor,prob)potlabor.*prob"
```

2.2.2 Test BL_LOG_WAGE is false

Default test

```
bl_log_wage = true;
mp_func_supply = bfw_mp_func_supply(bl_log_wage, bl_verbose);
```

```
----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_func Functions
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                         i      idx
                        ---     ---    ------------------------------------------------------------

    fc_ar_prob_wrk      "1"     "1"    "@(arpsi0,psi1,mtwage,probdenom)fc_v_occ(reshape(arpsi0,[1,len
    fc_log_pmdpo_occ    "2"     "2"    "@(psi0,psi1,arwage,pie1,pie2,pie3,pie4,pie5,pie6,t,prbchd,prb
    fc_prob_denom       "3"     "3"    "@(arpsi0,psi1,arpie,arwage1,arwage2,arwage3,t,prbchd,prbmar,p
    fc_prob_lei         "4"     "4"    "@(arpie,t,prbchd,prbmar,prbapp,prbjsy,probdenom)fc_v_lei(arpi
    fc_s1               "5"     "5"    "@(p1,G_1,zeta_1_0,zeta_1_1)G_1./(1+(exp(-zeta_1_0-zeta_1_1.*p
    fc_s2               "6"     "6"    "@(p2,G_2,zeta_2_0,zeta_2_1)G_2./(1+(exp(-zeta_2_0-zeta_2_1.*p
    fc_supply           "7"     "7"    "@(potlabor,prob)potlabor.*prob"
```

Chapter 3

Parameters

3.1 bfw_mp_path

This is the example vignette for function: bfw_mp_path from the **PrjLabEquiBFW Package**.

3.1.1 Default Map of Path (Fan path)

```
bl_verbose = true;
mp_path = bfw_mp_path(bl_verbose);
```

```
----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_path_external String
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                             i       idx                                            string
                            ____    ____    ________________________________________________________

    spt_codem               "1"     "1"     "C:\Users\fan\PrjLabEquiBFW\PrjLabEquiBFW\"
    spt_codem_data          "2"     "2"     "C:\Users\fan\PrjLabEquiBFW\PrjLabEquiBFW\_data\"
    spt_codem_doc           "3"     "3"     "C:\Users\fan\PrjLabEquiBFW\PrjLabEquiBFW\doc\"
    spt_output_root         "4"     "4"     "C:\Users\fan\Documents\Dropbox (UH-ECON)\Latex_Bhalotra
    spt_repo_root           "5"     "5"     "C:\Users\fan\PrjLabEquiBFW\"
    spt_simu_outputs_log    "6"     "6"     "C:\Users\fan\Documents\Dropbox (UH-ECON)\Latex_Bhalotra
    spt_simu_outputs_mat    "7"     "7"     "C:\Users\fan\Documents\Dropbox (UH-ECON)\Latex_Bhalotra
    spt_simu_outputs_prf    "8"     "8"     "C:\Users\fan\Documents\Dropbox (UH-ECON)\Latex_Bhalotra
    spt_simu_outputs_vig    "9"     "9"     "C:\Users\fan\Documents\Dropbox (UH-ECON)\Latex_Bhalotra
    spt_simu_results_csv    "10"    "10"    "C:\Users\fan\Documents\Dropbox (UH-ECON)\Latex_Bhalotra
    spt_simu_results_img    "11"    "11"    "C:\Users\fan\Documents\Dropbox (UH-ECON)\Latex_Bhalotra
    st_computer             "12"    "12"    "fan"
```

3.1.2 Map of Path for Alternative Path Installer

Two directories, one for the repo and one for where outputs go, need to be specified.

```
spt_repo_root = "~\PrjLabEquiBFW";
spt_output_root = "~\Dropbox\PrjLabEquiBFW";
bl_verbose = true;
mp_path = bfw_mp_path(spt_repo_root, spt_output_root, bl_verbose);
```

```
----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_path_external String
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
                        i      idx                          string
                       ----    ----     ---------------------------------------------------

    spt_codem          "1"     "1"      "~\PrjLabEquiBFW\PrjLabEquiBFW\"
    spt_codem_data     "2"     "2"      "~\PrjLabEquiBFW\PrjLabEquiBFW\_data\"
    spt_codem_doc      "3"     "3"      "~\PrjLabEquiBFW\PrjLabEquiBFW\doc\"
    spt_output_root    "4"     "4"      "~\Dropbox\PrjLabEquiBFW"
    spt_repo_root      "5"     "5"      "~\PrjLabEquiBFW"
    spt_simu_outputs_log "6"   "6"      "~\Dropbox\PrjLabEquiBFW\PrjLabEquiBFWOutput\log\"
    spt_simu_outputs_mat "7"   "7"      "~\Dropbox\PrjLabEquiBFW\PrjLabEquiBFWOutput\mat\"
    spt_simu_outputs_prf "8"   "8"      "~\Dropbox\PrjLabEquiBFW\PrjLabEquiBFWOutput\prof\"
    spt_simu_outputs_vig "9"   "9"      "~\Dropbox\PrjLabEquiBFW\PrjLabEquiBFWOutput\vig\"
    spt_simu_results_csv "10"  "10"     "~\Dropbox\PrjLabEquiBFW\PrjLabEquiBFWOutput\res\"
    spt_simu_results_img "11"  "11"     "~\Dropbox\PrjLabEquiBFW\PrjLabEquiBFWOutput\img\"
```

3.2 bfw_mp_param_esti

This is the example vignette for function: **bfw_mp_param_esti** from the **PrjLabEquiBFW Package**.

3.2.1 Map of Estimated Parameters

```
bl_log_wage = true;
bl_verbose = true;
mp_func_supply = bfw_mp_param_esti(bl_log_wage, bl_verbose);

pos = 42 ; key = mp_rho_nests
  Map with properties:

        Count: 11
      KeyType: char
    ValueType: any

pos = 43 ; key = mp_rho_nests_init
  Map with properties:

        Count: 8
      KeyType: char
    ValueType: any

----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_params ND Array (Matrix etc)
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                     i     idx    ndim    numel    rowN    colN      sum         mean        std
                     --    ---    ----    -----    ----    ----    --------    --------    -------

    ar_alpha_A001     1     1      2        4        1       4     -0.94699    -0.23675    0.51665
    ar_alpha_A002     2     2      2        4        1       4      -1.4489    -0.36221     0.7982
    ar_alpha_A003     3     3      2        4        1       4     -0.57104    -0.14276    0.31287
    ar_alpha_AA01     4     4      2        4        1       4     -0.67951    -0.16988     0.3633
    ar_alpha_AA02     5     5      2        4        1       4      -0.6718    -0.16795    0.33676
    ar_alpha_B001     6     6      2        4        1       4      -1.2904    -0.32261    0.67446
    ar_alpha_B002     7     7      2        4        1       4      -1.1023    -0.27558    0.57386
    ar_alpha_B003     8     8      2        4        1       4     -0.85037    -0.21259    0.44078
    ar_alpha_B101     9     9      2        4        1       4      -2.7486    -0.68715     1.4441
    ar_alpha_B102    10    10      2        4        1       4      -1.3642    -0.34105    0.66492
    ar_alpha_B103    11    11      2        4        1       4      -1.1457    -0.28641    0.57331
```

arpie_f_s	12	12	2	6	1	6	4.6479	0.77464	6.3115
arpie_f_u	13	13	2	6	1	6	8.0344	1.3391	4.861
arpie_k_s	14	14	2	6	1	6	1.3887	0.23145	1.8386
arpie_k_u	15	15	2	6	1	6	4.7387	0.78979	1.8849
arpsi0_f_s	16	16	2	3	1	3	3.3528	1.1176	1.0974
arpsi0_f_u	17	17	2	3	1	3	20.22	6.74	0.55777
arpsi0_k_s	18	18	2	3	1	3	1.779	0.59299	0.68939
arpsi0_k_u	19	19	2	3	1	3	18.003	6.0009	0.84112

xxx TABLE:ar_alpha_A001 xxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4
r1	0.00013396	-0.0056187	0.068567	-1.0101

xxx TABLE:ar_alpha_A002 xxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4
r1	0.00017171	-0.0079274	0.11544	-1.5565

xxx TABLE:ar_alpha_A003 xxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4
r1	6.9362e-05	-0.0031181	0.04301	-0.611

xxx TABLE:ar_alpha_AA01 xxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4
r1	3.3671e-05	-0.001978	0.03661	-0.71418

xxx TABLE:ar_alpha_AA02 xxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4
r1	9.8127e-06	-0.00029501	0.001573	-0.67309

xxx TABLE:ar_alpha_B001 xxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4
r1	7.1149e-05	-0.0031771	0.046411	-1.3337

xxx TABLE:ar_alpha_B002 xxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4
r1	7.7753e-05	-0.0032235	0.036755	-1.1359

xxx TABLE:ar_alpha_B003 xxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4
r1	4.3028e-05	-0.0018888	0.02499	-0.87352

xxx TABLE:ar_alpha_B101 xxxxxxxxxxxxxxxxxx

```
            c1            c2          c3         c4
        -----------   ----------   -------    ------

    r1  -1.7675e-05   -0.0011106   0.10452    -2.852

xxx TABLE:ar_alpha_B102 xxxxxxxxxxxxxxxxxx
            c1           c2          c3           c4
        -----------   ---------   ---------   -------

    r1  -0.00010096   0.0046709   -0.030629   -1.3382

xxx TABLE:ar_alpha_B103 xxxxxxxxxxxxxxxxxx
            c1           c2          c3           c4
        -----------   --------   ----------   -------

    r1  -7.5369e-05   0.002346   -0.0015487   -1.1464

xxx TABLE:arpie_f_s xxxxxxxxxxxxxxxxxx
          c1     c2     c3       c4        c5        c6
        ------   --   ------   -------   -------   -------

    r1  11.145   0    2.7351   0.26746   -8.3485   -1.1508

xxx TABLE:arpie_f_u xxxxxxxxxxxxxxxxxx
          c1     c2      c3         c4         c5        c6
        ------   --   --------   --------   -------   -------

    r1  11.145   0    -0.25662   -0.26519   -2.0749   -0.5135

xxx TABLE:arpie_k_s xxxxxxxxxxxxxxxxxx
          c1     c2      c3         c4         c5        c6
        ------   --   ---------   -------   -------   ------

    r1  2.4457   0    -0.043896   0.91566   -3.0311   1.1023

xxx TABLE:arpie_k_u xxxxxxxxxxxxxxxxxx
          c1     c2     c3        c4       c5        c6
        ------   --   -------   ------   -------   -------

    r1  2.4457   0    -2.2809   3.0169   0.84513   0.71184

xxx TABLE:arpsi0_f_s xxxxxxxxxxxxxxxxxx
        c1     c2       c3
        --   ------   ------

    r1  0    1.1592   2.1936

xxx TABLE:arpsi0_f_u xxxxxxxxxxxxxxxxxx
          c1       c2       c3
        ------   ------   ------

    r1  7.3697   6.5422   6.3081

xxx TABLE:arpsi0_k_s xxxxxxxxxxxxxxxxxx
        c1     c2         c3
        --   -------   ------

    r1  0    0.42958   1.3494
```

```
xxx TABLE:arpsi0_k_u xxxxxxxxxxxxxxxxxx
            c1        c2        c3
          ------    ------    ------

    r1    6.6935    6.2443    5.0649

----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_params Scalars
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                                        i     idx     value
                                        --    ---    --------

    bl_log_wage                          1    20            1
    fl_rho_abstract_vs_manualroutine     2    21     0.031411
    fl_rho_gen_abstract                  3    22      0.65979
    fl_rho_gen_manual                    4    23     0.083519
    fl_rho_gen_routine                   5    24      0.21769
    fl_rho_routine_vs_manual             6    25     -0.15438
    fl_rho_skill_abstract                7    26      0.30231
    fl_rho_skill_manual                  8    27      0.73852
    fl_rho_skill_routine                 9    28      0.30052
    fl_yzagg_y1989                      10    29       1.4905
    fl_yzagg_y1992                      11    30       1.4602
    fl_yzagg_y1994                      12    31       1.6493
    fl_yzagg_y1996                      13    32       1.7686
    fl_yzagg_y1998                      14    33       1.8018
    fl_yzagg_y2000                      15    34       2.0599
    fl_yzagg_y2002                      16    35       2.0597
    fl_yzagg_y2004                      17    36       2.2803
    fl_yzagg_y2005                      18    37       2.3392
    fl_yzagg_y2008                      19    38       2.4908
    fl_yzagg_y2010                      20    39       2.7153
    fl_yzagg_y2012                      21    40        2.822
    fl_yzagg_y2014                      22    41       2.8707
    psi1                                23    44      0.96625
```

Chapter 4

Data

4.1 bfw_mp_data

This is the example vignette for function: bfw_mp_data from the **PrjLabEquiBFW Package**.

4.1.1 Get All Data

```
bl_verbose = false;
mp_data = bfw_mp_data(bl_verbose);
```

4.1.2 Dataset 1

```
disp(mp_data('tb_data_pq'));
```

year	category	numberWorkers	meanWage
1989	{'C001'}	1.4486e+06	1.942
1989	{'C002'}	1.1256e+06	3.2247
1989	{'C003'}	1.5156e+06	3.3738
1989	{'C004'}	8.4266e+06	NaN
1989	{'C011'}	9199	2.1604
1989	{'C012'}	1.1011e+05	5.6589
1989	{'C013'}	4.816e+05	5.8023
1989	{'C014'}	2.533e+05	NaN
1989	{'C101'}	4.4275e+06	2.3157
1989	{'C102'}	3.1277e+06	3.2178
1989	{'C103'}	1.9279e+06	4.329
1989	{'C104'}	4.8562e+05	NaN
1989	{'C111'}	96487	4.5245
1989	{'C112'}	2.7718e+05	5.4146
1989	{'C113'}	1.3868e+06	8.0437
1989	{'C114'}	1.187e+05	NaN
1992	{'C001'}	1.7431e+06	1.8431
1992	{'C002'}	1.3773e+06	3.4764
1992	{'C003'}	1.428e+06	4.079
1992	{'C004'}	8.7758e+06	NaN
1992	{'C011'}	18205	4.5495
1992	{'C012'}	1.6703e+05	5.752
1992	{'C013'}	6.2931e+05	7.0257
1992	{'C014'}	3.657e+05	NaN
1992	{'C101'}	4.7927e+06	2.052
1992	{'C102'}	4.0642e+06	2.9976

1992	{'C103'}	1.6709e+06	4.7971
1992	{'C104'}	5.5e+05	NaN
1992	{'C111'}	74782	4.0857
1992	{'C112'}	3.3189e+05	7.9404
1992	{'C113'}	1.4371e+06	10.001
1992	{'C114'}	1.3064e+05	NaN
1994	{'C001'}	2.5091e+06	1.9678
1994	{'C002'}	1.5404e+06	3.5099
1994	{'C003'}	1.5569e+06	4.3758
1994	{'C004'}	8.8237e+06	NaN
1994	{'C011'}	10653	2.8112
1994	{'C012'}	2.4128e+05	6.9136
1994	{'C013'}	7.5302e+05	8.6943
1994	{'C014'}	4.1888e+05	NaN
1994	{'C101'}	5.3134e+06	2.1107
1994	{'C102'}	4.0308e+06	3.1178
1994	{'C103'}	1.6829e+06	4.8591
1994	{'C104'}	7.1707e+05	NaN
1994	{'C111'}	1.5239e+05	7.0725
1994	{'C112'}	4.3682e+05	11.505
1994	{'C113'}	1.557e+06	12.719
1994	{'C114'}	1.2058e+05	NaN
1996	{'C001'}	2.8324e+06	1.459
1996	{'C002'}	2.1046e+06	2.4083
1996	{'C003'}	1.753e+06	2.7709
1996	{'C004'}	8.7805e+06	NaN
1996	{'C011'}	57074	2.3762
1996	{'C012'}	2.5339e+05	4.8631
1996	{'C013'}	9.465e+05	5.8817
1996	{'C014'}	5.1589e+05	NaN
1996	{'C101'}	5.4919e+06	1.7407
1996	{'C102'}	4.4873e+06	2.385
1996	{'C103'}	1.9182e+06	3.3137
1996	{'C104'}	6.9559e+05	NaN
1996	{'C111'}	2.0215e+05	5.7586
1996	{'C112'}	4.858e+05	6.221
1996	{'C113'}	1.6429e+06	7.9771
1996	{'C114'}	1.7307e+05	NaN
1998	{'C001'}	3.1189e+06	1.3076
1998	{'C002'}	2.0101e+06	2.5758
1998	{'C003'}	2.0265e+06	2.9886
1998	{'C004'}	8.8847e+06	NaN
1998	{'C011'}	36132	3.694
1998	{'C012'}	3.2575e+05	5.0667
1998	{'C013'}	9.3514e+05	5.6322
1998	{'C014'}	5.4905e+05	NaN
1998	{'C101'}	5.5182e+06	1.7357
1998	{'C102'}	4.8667e+06	2.4162
1998	{'C103'}	2.1473e+06	3.3496
1998	{'C104'}	6.7234e+05	NaN
1998	{'C111'}	1.6247e+05	4.0171
1998	{'C112'}	5.3722e+05	7.4345
1998	{'C113'}	1.6662e+06	8.7309
1998	{'C114'}	1.9321e+05	NaN
2000	{'C001'}	2.7625e+06	1.659
2000	{'C002'}	2.7297e+06	2.5901
2000	{'C003'}	2.2657e+06	3.2971
2000	{'C004'}	9.3772e+06	NaN

2000	{'C011'}	77107	2.8732
2000	{'C012'}	4.0734e+05	5.2881
2000	{'C013'}	1.1005e+06	6.5806
2000	{'C014'}	7.5089e+05	NaN
2000	{'C101'}	5.6807e+06	1.8978
2000	{'C102'}	5.3498e+06	2.4629
2000	{'C103'}	2.2554e+06	3.968
2000	{'C104'}	6.7471e+05	NaN
2000	{'C111'}	2.1108e+05	3.8076
2000	{'C112'}	6.6682e+05	7.0165
2000	{'C113'}	2.2414e+06	10.509
2000	{'C114'}	1.9925e+05	NaN
2002	{'C001'}	3.6671e+06	1.6863
2002	{'C002'}	2.5202e+06	2.826
2002	{'C003'}	2.4393e+06	3.292
2002	{'C004'}	9.291e+06	NaN
2002	{'C011'}	1.0685e+05	3.7516
2002	{'C012'}	4.5408e+05	5.83
2002	{'C013'}	1.3436e+06	7.9012
2002	{'C014'}	5.9194e+05	NaN
2002	{'C101'}	5.9945e+06	2.0088
2002	{'C102'}	5.2352e+06	2.7613
2002	{'C103'}	2.2663e+06	4.1455
2002	{'C104'}	6.7629e+05	NaN
2002	{'C111'}	2.4805e+05	4.0453
2002	{'C112'}	5.9178e+05	7.1763
2002	{'C113'}	2.0465e+06	8.9213
2002	{'C114'}	3.2278e+05	NaN
2004	{'C001'}	3.5389e+06	1.755
2004	{'C002'}	2.5059e+06	2.6069
2004	{'C003'}	2.5599e+06	3.1199
2004	{'C004'}	9.5136e+06	NaN
2004	{'C011'}	1.4496e+05	3.4155
2004	{'C012'}	4.5696e+05	5.4516
2004	{'C013'}	1.8123e+06	6.7
2004	{'C014'}	7.7668e+05	NaN
2004	{'C101'}	5.9652e+06	2.215
2004	{'C102'}	5.7124e+06	2.8839
2004	{'C103'}	2.3318e+06	3.8541
2004	{'C104'}	9.677e+05	NaN
2004	{'C111'}	2.8065e+05	5.1077
2004	{'C112'}	5.9455e+05	6.7843
2004	{'C113'}	2.2218e+06	8.6393
2004	{'C114'}	2.6115e+05	NaN
2005	{'C001'}	3.604e+06	1.8015
2005	{'C002'}	2.9152e+06	2.6792
2005	{'C003'}	2.4463e+06	3.3468
2005	{'C004'}	9.2417e+06	NaN
2005	{'C011'}	1.2085e+05	2.4982
2005	{'C012'}	5.9567e+05	4.9431
2005	{'C013'}	1.6771e+06	6.3435
2005	{'C014'}	8.2842e+05	NaN
2005	{'C101'}	5.9621e+06	2.2032
2005	{'C102'}	5.4187e+06	2.7741
2005	{'C103'}	2.5829e+06	3.7258
2005	{'C104'}	9.6341e+05	NaN
2005	{'C111'}	3.5414e+05	3.7752
2005	{'C112'}	6.5345e+05	6.9592

2005	{'C113'}	2.3148e+06	8.3387
2005	{'C114'}	2.8472e+05	NaN
2008	{'C001'}	3.9395e+06	1.8657
2008	{'C002'}	2.8968e+06	2.6475
2008	{'C003'}	2.361e+06	3.1947
2008	{'C004'}	9.2787e+06	NaN
2008	{'C011'}	1.5621e+05	3.0013
2008	{'C012'}	6.771e+05	5.3544
2008	{'C013'}	1.9227e+06	6.8198
2008	{'C014'}	9.0351e+05	NaN
2008	{'C101'}	6.0495e+06	2.3736
2008	{'C102'}	5.8662e+06	2.9056
2008	{'C103'}	2.4905e+06	3.7731
2008	{'C104'}	1.2219e+06	NaN
2008	{'C111'}	2.8368e+05	3.9143
2008	{'C112'}	7.9417e+05	6.3566
2008	{'C113'}	2.4155e+06	8.3053
2008	{'C114'}	2.6468e+05	NaN
2010	{'C001'}	3.9036e+06	1.7636
2010	{'C002'}	2.8717e+06	2.4062
2010	{'C003'}	2.7349e+06	2.8429
2010	{'C004'}	9.9169e+06	NaN
2010	{'C011'}	1.2713e+05	3.1825
2010	{'C012'}	6.661e+05	4.7299
2010	{'C013'}	2.2114e+06	6.1872
2010	{'C014'}	1.2068e+06	NaN
2010	{'C101'}	6.6858e+06	2.263
2010	{'C102'}	5.9638e+06	2.5991
2010	{'C103'}	2.4368e+06	3.6533
2010	{'C104'}	1.4088e+06	NaN
2010	{'C111'}	3.6653e+05	3.5758
2010	{'C112'}	7.4601e+05	6.2607
2010	{'C113'}	2.7576e+06	8.101
2010	{'C114'}	3.7913e+05	NaN
2012	{'C001'}	5.1813e+06	1.7308
2012	{'C002'}	3.049e+06	2.4089
2012	{'C003'}	3.0537e+06	2.7185
2012	{'C004'}	8.7224e+06	NaN
2012	{'C011'}	1.9743e+05	3.3489
2012	{'C012'}	7.3753e+05	4.1924
2012	{'C013'}	2.3311e+06	6.4194
2012	{'C014'}	1.0551e+06	NaN
2012	{'C101'}	7.139e+06	2.1453
2012	{'C102'}	6.2508e+06	2.5302
2012	{'C103'}	2.5895e+06	3.1115
2012	{'C104'}	1.512e+06	NaN
2012	{'C111'}	4.3101e+05	3.2287
2012	{'C112'}	9.0347e+05	5.0768
2012	{'C113'}	2.7373e+06	7.5722
2012	{'C114'}	3.9649e+05	NaN
2014	{'C001'}	4.5694e+06	1.7262
2014	{'C002'}	3.2584e+06	2.4145
2014	{'C003'}	2.8512e+06	2.6173
2014	{'C004'}	9.733e+06	NaN
2014	{'C011'}	2.5971e+05	2.9667
2014	{'C012'}	8.2213e+05	5.6007
2014	{'C013'}	2.5873e+06	6.1866
2014	{'C014'}	1.3462e+06	NaN

2014	{'C101'}	7.0339e+06	2.212
2014	{'C102'}	6.3219e+06	2.5069
2014	{'C103'}	2.7689e+06	3.1292
2014	{'C104'}	1.5334e+06	NaN
2014	{'C111'}	4.3522e+05	3.3786
2014	{'C112'}	8.8807e+05	5.4313
2014	{'C113'}	3.0431e+06	8.6421
2014	{'C114'}	4.8022e+05	NaN

4.1.3 Dataset 1 Aux

```
disp(mp_data('tb_category2sexskillocc_key'));
```

category	sex	skill	occupation	categoryhigher	nesttier
{'C001'}	{'Female'}	{'unskilled'}	{'Manual' }	{'B001' }	3
{'C002'}	{'Female'}	{'unskilled'}	{'Routine' }	{'B002' }	3
{'C003'}	{'Female'}	{'unskilled'}	{'Analytical' }	{'B003' }	3
{'C004'}	{'Female'}	{'unskilled'}	{'Home Production'}	{0x0 char}	3
{'C011'}	{'Female'}	{'skilled' }	{'Manual' }	{'B101' }	3
{'C012'}	{'Female'}	{'skilled' }	{'Routine' }	{'B102' }	3
{'C013'}	{'Female'}	{'skilled' }	{'Analytical' }	{'B103' }	3
{'C014'}	{'Female'}	{'skilled' }	{'Home Production'}	{0x0 char}	3
{'C101'}	{'Male' }	{'unskilled'}	{'Manual' }	{'B001' }	3
{'C102'}	{'Male' }	{'unskilled'}	{'Routine' }	{'B002' }	3
{'C103'}	{'Male' }	{'unskilled'}	{'Analytical' }	{'B003' }	3
{'C104'}	{'Male' }	{'unskilled'}	{'Home Production'}	{0x0 char}	3
{'C111'}	{'Male' }	{'skilled' }	{'Manual' }	{'B101' }	3
{'C112'}	{'Male' }	{'skilled' }	{'Routine' }	{'B102' }	3
{'C113'}	{'Male' }	{'skilled' }	{'Analytical' }	{'B103' }	3
{'C114'}	{'Male' }	{'skilled' }	{'Home Production'}	{0x0 char}	3
{'B001'}	{'All' }	{'unskilled'}	{'Manual' }	{'A001' }	2
{'B002'}	{'All' }	{'unskilled'}	{'Routine' }	{'A002' }	2
{'B003'}	{'All' }	{'unskilled'}	{'Analytical' }	{'A003' }	2
{'B101'}	{'All' }	{'skilled' }	{'Manual' }	{'A001' }	2
{'B102'}	{'All' }	{'skilled' }	{'Routine' }	{'A002' }	2
{'B103'}	{'All' }	{'skilled' }	{'Analytical' }	{'A003' }	2
{'A001'}	{'All' }	{'All' }	{'Manual' }	{'AA01' }	1
{'A002'}	{'All' }	{'All' }	{'Routine' }	{'AA01' }	1
{'AA01'}	{'All' }	{'All' }	{'ManualRoutine' }	{'AA02' }	0
{'A003'}	{'All' }	{'All' }	{'Analytical' }	{'AA02' }	0
{'AA02'}	{'All' }	{'All' }	{'All' }	{0x0 char}	NaN

4.1.4 Dataset 2

```
disp(mp_data('tb_supply_potwrklei'));
```

year	group	gender	skill	numberPotentialWorkers	shareMarried	shareChi
1989	{'G00'}	"Female"	"unskilled"	1.2516e+07	0.88971	0.
1992	{'G00'}	"Female"	"unskilled"	1.3324e+07	0.90306	0.
1994	{'G00'}	"Female"	"unskilled"	1.443e+07	0.89015	0.
1996	{'G00'}	"Female"	"unskilled"	1.547e+07	0.88061	0.
1998	{'G00'}	"Female"	"unskilled"	1.604e+07	0.86928	0.
2000	{'G00'}	"Female"	"unskilled"	1.7135e+07	0.85884	0.
2002	{'G00'}	"Female"	"unskilled"	1.7918e+07	0.85017	0.

```
    2004    {'G00'}    "Female"    "unskilled"         1.8118e+07            0.82951             0.
    2005    {'G00'}    "Female"    "unskilled"         1.8207e+07            0.82472             0.
    2008    {'G00'}    "Female"    "unskilled"         1.8476e+07            0.82096             0.
    2010    {'G00'}    "Female"    "unskilled"         1.9427e+07            0.81832             0.
    2012    {'G00'}    "Female"    "unskilled"         2.0006e+07            0.81686             0.
    2014    {'G00'}    "Female"    "unskilled"         2.0412e+07            0.82762             0.
    1989    {'G01'}    "Female"    "skilled"           8.5421e+05            0.87768             0.
    1992    {'G01'}    "Female"    "skilled"           1.1802e+06            0.82152             0.
    1994    {'G01'}    "Female"    "skilled"           1.4238e+06            0.81836             0.
    1996    {'G01'}    "Female"    "skilled"           1.7729e+06             0.8449             0.
    1998    {'G01'}    "Female"    "skilled"           1.8461e+06            0.82028              0
    2000    {'G01'}    "Female"    "skilled"           2.3358e+06            0.83775             0.
    2002    {'G01'}    "Female"    "skilled"           2.4965e+06            0.81416             0.
    2004    {'G01'}    "Female"    "skilled"           3.1909e+06            0.81625             0.
    2005    {'G01'}    "Female"    "skilled"           3.2221e+06            0.79637             0.
    2008    {'G01'}    "Female"    "skilled"           3.6595e+06            0.77916             0.
    2010    {'G01'}    "Female"    "skilled"           4.2115e+06            0.76377              0
    2012    {'G01'}    "Female"    "skilled"           4.3212e+06            0.77164             0.
    2014    {'G01'}    "Female"    "skilled"           5.0153e+06            0.78454             0.
    1989    {'G10'}    "Male"      "unskilled"         9.9687e+06            0.94927             0.
    1992    {'G10'}    "Male"      "unskilled"         1.1078e+07            0.95081             0.
    1994    {'G10'}    "Male"      "unskilled"         1.1744e+07            0.93806             0.
    1996    {'G10'}    "Male"      "unskilled"         1.2593e+07            0.94471             0.
    1998    {'G10'}    "Male"      "unskilled"         1.3205e+07            0.94507             0.
    2000    {'G10'}    "Male"      "unskilled"         1.3961e+07            0.93873             0.
    2002    {'G10'}    "Male"      "unskilled"         1.4172e+07             0.9392             0.
    2004    {'G10'}    "Male"      "unskilled"         1.4977e+07            0.93382             0.
    2005    {'G10'}    "Male"      "unskilled"         1.4927e+07            0.93314             0.
    2008    {'G10'}    "Male"      "unskilled"         1.5628e+07            0.91783             0.
    2010    {'G10'}    "Male"      "unskilled"         1.6495e+07            0.92582             0.
    2012    {'G10'}    "Male"      "unskilled"         1.7491e+07            0.91384             0.
    2014    {'G10'}    "Male"      "unskilled"         1.7658e+07            0.91783             0.
    1989    {'G11'}    "Male"      "skilled"           1.8792e+06             0.9391             0.
    1992    {'G11'}    "Male"      "skilled"           1.9744e+06            0.94708             0.
    1994    {'G11'}    "Male"      "skilled"           2.2668e+06            0.92429             0.
    1996    {'G11'}    "Male"      "skilled"           2.5039e+06            0.92112             0.
    1998    {'G11'}    "Male"      "skilled"           2.5591e+06            0.90398             0.
    2000    {'G11'}    "Male"      "skilled"           3.3185e+06            0.90086             0.
    2002    {'G11'}    "Male"      "skilled"           3.2091e+06            0.90837             0.
    2004    {'G11'}    "Male"      "skilled"           3.3582e+06            0.89767             0.
    2005    {'G11'}    "Male"      "skilled"           3.6072e+06            0.89235             0.
    2008    {'G11'}    "Male"      "skilled"            3.758e+06            0.86831             0.
    2010    {'G11'}    "Male"      "skilled"           4.2492e+06            0.87325             0.
    2012    {'G11'}    "Male"      "skilled"           4.4683e+06            0.83196             0.
    2014    {'G11'}    "Male"      "skilled"           4.8466e+06            0.86615             0.
```

4.1.5 Dataset 2 Aux

```
disp(mp_data('tb_group2category_key'));
```

```
     group           groupName          category        sex            skill
    _______    ____________________    ________    __________    _____________

    {'G00'}    {'female-unskilled'}    {'C001'}    {'female'}    {'unskilled'}
    {'G00'}    {'female-unskilled'}    {'C002'}    {'female'}    {'unskilled'}
    {'G00'}    {'female-unskilled'}    {'C003'}    {'female'}    {'unskilled'}
    {'G00'}    {'female-unskilled'}    {'C004'}    {'female'}    {'unskilled'}
    {'G01'}    {'female-skilled'  }    {'C011'}    {'female'}    {'skilled'  }
```

```
{'G01'}    {'female-skilled'  }    {'C012'}    {'female'}    {'skilled'  }
{'G01'}    {'female-skilled'  }    {'C013'}    {'female'}    {'skilled'  }
{'G01'}    {'female-skilled'  }    {'C014'}    {'female'}    {'skilled'  }
{'G10'}    {'male-unskilled'  }    {'C101'}    {'male'  }    {'unskilled'}
{'G10'}    {'male-unskilled'  }    {'C102'}    {'male'  }    {'unskilled'}
{'G10'}    {'male-unskilled'  }    {'C103'}    {'male'  }    {'unskilled'}
{'G10'}    {'male-unskilled'  }    {'C104'}    {'male'  }    {'unskilled'}
{'G11'}    {'male-skilled'    }    {'C111'}    {'male'  }    {'skilled'  }
{'G11'}    {'male-skilled'    }    {'C112'}    {'male'  }    {'skilled'  }
{'G11'}    {'male-skilled'    }    {'C113'}    {'male'  }    {'skilled'  }
{'G11'}    {'male-skilled'    }    {'C114'}    {'male'  }    {'skilled'  }
```

Chapter 5

Demand

5.1 Solve Nested CES Optimal Demand (CRS)

Testing the **bfw_crs_nested_ces** function from the **PrjLabEquiBFW Package.** This function solves optimal choices given CES production function under cost minimization. Works with Constant Elasticity of Substitution problems with constant returns, up to four nest layers, 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 with Constant Elasticity of Substitution problems with constant returns, up to four nest layers, and two inputs in each sub-nest. 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.

5.1.1 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.

5.1.2 Single Nest Layer Two Inputs CES Problem (Demand)

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 = {[1.5, 0.75]};
% print option
bl_verbose = true;
mp_func = bfw_mp_func_demand();
bl_bfw_model = false;
[cl_mn_yz_choices, cl_mn_price] = ...
    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
                -    ---    ----    -----    ----    ----    ------    ------    -------    --------

    price_c1    1     2      2        2        1       2       2.25      1.125    0.53033     0.4714
    yz_c1       2     4      2        2        1       2     2.1343     1.0671    0.55403    0.51918

xxx TABLE:price_c1 xxxxxxxxxxxxxxxxxx
          c1      c2
          ---    ----

    r1    1.5    0.75

xxx TABLE:yz_c1 xxxxxxxxxxxxxxxxxx
            c1         c2
          -------    ------

    r1    0.67537    1.4589

----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_container_map Scalars
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 i    idx    value
                 -    ---    -----
```

```
    prho_c1      1     1      0.1
    pshare_c1    2     3      0.5
```

5.1.3 Single Nest Layer Two Inputs CES Problem, Vary Share and Elasticity (Demand)

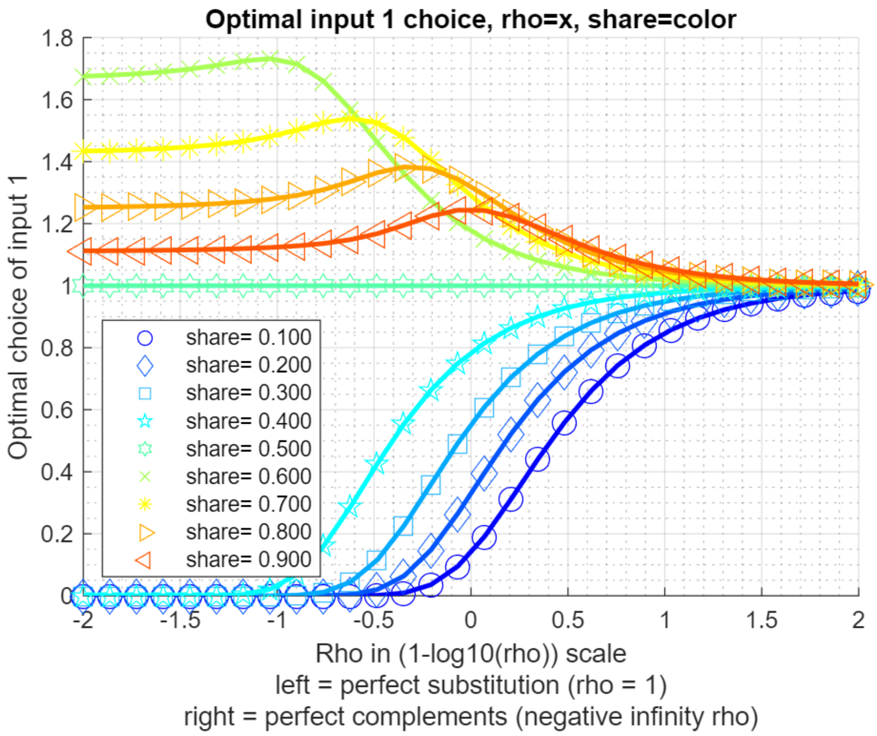
In this second example, we test over different rho values, explore optimal relative choices, as share and elasticity change. In this exercise, we also check, at every combination of rho and share parameter, whether the FOC condition is satisfied by the optimal choices. Also check if at the optimal choices,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

```
        if (~if_is_close(fl_output, fl_yz))
            error('There is an error, output is not equal to required expenditure minimizing output'
        end

    end
end
```

Key results: (1) As share parameter 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);
```



5.1.4 Doubly Nest Layer Two Inputs Each Sub-nest CES Problem (Demand)

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      coefvar
                 -    ---    ----    -----    ----    ----    ------    ------    -------    -------

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

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
```

5.1.5 Doubly Nest Layer Two Inputs Each Sub-nest CES Problem–Solution Check (Demand)

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))^(1/f
fl_output_2 = ((fl_pshare_2)*fl_opti_x21^(fl_prho_2) + (1-fl_pshare_2)*fl_opti_x22^(fl_prho_2))^(1/f
fl_output_0 = ((fl_pshare_0)*fl_output_1^(fl_prho_0) + (1-fl_pshare_0)*fl_output_2^(fl_prho_0))^(1/f
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_1));
fl_rela_opti_foc_2 = (((fl_pshare_2/(1-fl_pshare_2)))*(fl_wage_x22/fl_wage_x21))^(1/(1-fl_prho_2));
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 acro
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 across ne
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 across ne
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_pr
    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 across ne
end
```

5.1.6 BFW (2022) Nested Three Branch (Four Layer) Problem (Demand)

The model BFW 2022 has three branches and four layers. one of the branches go down only three layers, the other two branches go down four layers.

First, we prepare the various inputs:

```
% Controls
bl_verbose = true;
bl_bfw_model = true;

% Given rho and beta, solve for equilibrium quantities
bl_log_wage = false;
mp_func = bfw_mp_func_demand(bl_log_wage);

% Following instructions in: PrjFLFPMexicoBFW\solvedemand\README.md

% Nests/layers
it_nests = 4;

% Input cell of mn matrixes
it_prho_cl = 1;
it_pshare_cl = 2;
it_price_cl = 3;
for it_cl_ctr = [1,2,3]

    cl_mn_cur = cell(it_nests,1);

    % Fill each cell element with NaN mn array
    for it_cl_mn = 1:it_nests

        bl_price = (it_cl_ctr == it_price_cl);

        if (~bl_price && it_cl_mn == 1)
            mn_nan = NaN;
        elseif (~bl_price && it_cl_mn == 2) || (bl_price && it_cl_mn == 1)
            mn_nan = [NaN, NaN];
        elseif (~bl_price && it_cl_mn == 3) || (bl_price && it_cl_mn == 2)
            mn_nan = NaN(2,2);
        elseif (~bl_price && it_cl_mn == 4) || (bl_price && it_cl_mn == 3)
            mn_nan = NaN(2,2,2);
        elseif (~bl_price && it_cl_mn == 5) || (bl_price && it_cl_mn == 4)
            mn_nan = NaN(2,2,2,2);
        elseif (~bl_price && it_cl_mn == 6) || (bl_price && it_cl_mn == 5)
            mn_nan = NaN(2,2,2,2,2);
        end
        cl_mn_cur{it_cl_mn} = mn_nan;
    end

    % Name cell arrays
    if (it_cl_ctr == it_prho_cl)
        cl_mn_prho = cl_mn_cur;
    elseif (it_cl_ctr == it_pshare_cl)
        cl_mn_pshare = cl_mn_cur;
    elseif (it_cl_ctr == it_price_cl)
        cl_mn_price = cl_mn_cur;
    end
end

% Initialize share matrix
```

```
rng(123);
for it_cl_mn = 1:it_nests
    mn_pshare = cl_mn_pshare{it_cl_mn};
    if it_cl_mn == 4
        mn_pshare(2,:,:) = rand(2,2);
    else
        mn_pshare = rand(size(mn_pshare));
    end
    cl_mn_pshare{it_cl_mn} = mn_pshare;
end

% Initialize rho matrix
rng(456);
for it_cl_mn = 1:it_nests
    mn_prho = cl_mn_prho{it_cl_mn};
    if it_cl_mn == 4
        mn_prho(2,:,:) = rand(2,2);
    else
        mn_prho = rand(size(mn_prho));
    end
    % Scalling rho between 0.7500 and -3.0000
    % 1 - 2.^(linspace(-2,2,5))
    mn_prho = 1 - 2.^(mn_prho*(4) - 2);
    cl_mn_prho{it_cl_mn} = mn_prho;
end

% Initialize wage matrix
rng(789);
for it_cl_mn = 1:it_nests
    mn_price = cl_mn_price{it_cl_mn};
    if it_cl_mn == 3
        mn_price(1,:,:) = rand(2,2);
    elseif it_cl_mn == 4
        mn_price(2,:,:,:) = rand(2,2,2);
    end
    % Scalling rho between 3 amd 5
    mn_price = mn_price*(2) + 3;
    cl_mn_price{it_cl_mn} = mn_price;
end

% Initialize yz matrix
rng(101112);
fl_yz = rand();
```

Second, display created inputs:

```
disp(['fl_yz=' num2str(fl_yz)]);
```

```
fl_yz=0.89726
```

```
celldisp(cl_mn_prho);
```

```
cl_mn_prho

cl_mn_prho{1} =

    0.5017
```

```
cl_mn_prho{2} =

    0.6071   -1.1955



cl_mn_prho{3} =

   -1.3523   -0.3346
   -0.4167   -1.9136



cl_mn_prho{4} =


(:,:,1) =

       NaN       NaN
   -1.0512    0.5869


(:,:,2) =

       NaN       NaN
    0.6209    0.1633



celldisp(cl_mn_pshare);

cl_mn_pshare

cl_mn_pshare{1} =

    0.6965



cl_mn_pshare{2} =

    0.2861    0.2269



cl_mn_pshare{3} =

    0.5513    0.4231
    0.7195    0.9808



cl_mn_pshare{4} =


(:,:,1) =
```

```
       NaN       NaN
    0.6848    0.4809


(:,:,2) =

       NaN       NaN
    0.3921    0.3432



celldisp(cl_mn_price);

cl_mn_price

cl_mn_price{1} =

   NaN   NaN



cl_mn_price{2} =

   NaN   NaN
   NaN   NaN



cl_mn_price{3} =


(:,:,1) =

    3.6467    3.4605
       NaN       NaN


(:,:,2) =

    4.5876    4.2488
       NaN       NaN



cl_mn_price{4} =


(:,:,1,1) =

       NaN       NaN
    4.9508    4.5178


(:,:,2,1) =

       NaN       NaN
    3.0212    3.0495
```

```
(:,:,1,2) =

      NaN       NaN
   3.2221    4.0763


(:,:,2,2) =

      NaN       NaN
   3.0909    4.1031
```

Third, call function and solve for optimal demand:

```
% 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);
```

```
----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_container_map ND Array (Matrix etc)
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                            i    idx    ndim    numel    rowN    colN      sum         mean       --
                            --   ---    ----    -----    ----    ----    --------    --------    --

    mt_fl_labor_demanded    1     1      2       12       4       3        5.4455     0.45379    0
    prho_c2                 2     3      2        2       1       2      -0.58844    -0.29422
    prho_c3                 3     4      2        4       2       2       -4.0173     -1.0043    0
    prho_c4                 4     5      3        8       2       4           NaN         NaN
    price_c1                5     6      2        2       1       2        35.345      17.673
    price_c2                6     7      2        4       2       2        40.906      10.226
    price_c3                7     8      3        8       2       4        45.403      5.6754
    price_c4                8     9      4       16       2       8           NaN         NaN
    pshare_c2               9    11      2        2       1       2       0.51299      0.2565    0.
    pshare_c3              10    12      2        4       2       2        2.6747     0.66866    0
    pshare_c4              11    13      3        8       2       4           NaN         NaN
    yz_c1                  12    14      2        2       1       2        1.6003     0.80016
    yz_c2                  13    15      2        4       2       2         2.645     0.66124
    yz_c3                  14    16      3        8       2       4        5.1962     0.64953
    yz_c4                  15    17      4       16       2       8           NaN         NaN

xxx TABLE:mt_fl_labor_demanded xxxxxxxxxxxxxxxxxx
              c1          c2          c3
          --------    --------    -------

    r1    0.020122    0.024929     2.1857
    r2    0.060227    0.037985     2.3642
    r3    0.069088    0.093774    0.21107
    r4    0.058349     0.14469    0.17539

xxx TABLE:prho_c2 xxxxxxxxxxxxxxxxxx
             c1         c2
          -------    -------

    r1    0.60709    -1.1955

xxx TABLE:prho_c3 xxxxxxxxxxxxxxxxxx
              c1          c2
```

r1	-1.3523	-0.33464
r2	-0.41668	-1.9136

xxx TABLE:prho_c4 xxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4
r1	NaN	NaN	NaN	NaN
r2	-1.0512	0.58694	0.62089	0.16334

xxx TABLE:price_c1 xxxxxxxxxxxxxxxxxx

	c1	c2
r1	12.695	22.65

xxx TABLE:price_c2 xxxxxxxxxxxxxxxxxx

	c1	c2
r1	8.1518	7.7015
r2	13.522	11.53

xxx TABLE:price_c3 xxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4
r1	3.6467	3.4605	4.5876	4.2488
r2	8.1184	8.5114	5.7986	7.0309

xxx TABLE:price_c4 xxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4	c5	c6	c7	c8
r1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
r2	4.9508	4.5178	3.0212	3.0495	3.2221	4.0763	3.0909	4.1031

xxx TABLE:pshare_c2 xxxxxxxxxxxxxxxxxx

	c1	c2
r1	0.28614	0.22685

xxx TABLE:pshare_c3 xxxxxxxxxxxxxxxxxx

	c1	c2
r1	0.55131	0.42311
r2	0.71947	0.98076

xxx TABLE:pshare_c4 xxxxxxxxxxxxxxxxxx

	c1	c2	c3	c4
r1	NaN	NaN	NaN	NaN
r2	0.68483	0.48093	0.39212	0.34318

```
xxx TABLE:yz_c1 xxxxxxxxxxxxxxxxxx
           c1         c2
          -----    --------

    r1    1.511    0.089284

xxx TABLE:yz_c2 xxxxxxxxxxxxxxxxxx
             c1         c2
          --------    ------

    r1     0.19312    2.2864
    r2    0.057461     0.108

xxx TABLE:yz_c3 xxxxxxxxxxxxxxxxxx
            c1         c2          c3          c4
          -------    -------    --------    -------

    r1    0.21107     2.1857     0.17539     2.3642
    r2    0.06529    0.11907    0.042587    0.03298

xxx TABLE:yz_c4 xxxxxxxxxxxxxxxxxx
             c1          c2          c3          c4          c5         c6          c7          c8
          --------    --------    --------    --------    --------    -------    --------    -------

    r1         NaN         NaN         NaN         NaN         NaN        NaN         NaN         Na
    r2    0.069088    0.093774    0.020122    0.024929    0.058349    0.14469    0.060227    0.03798

----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_container_map Scalars
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 i    idx     value
                 -    ---    -------

    prho_c1      1     2     0.50172
    pshare_c1    2    10     0.69647
```

5.2 Compute Nested CES MPL Given Demand (CRS)

Testing the **bfw_crs_nested_ces_mpl** function from the **PrjLabEquiBFW Package.** Given labor quantity demanded, using first-order relative optimality conditions, find the marginal product of labor given CES production function. Results match up with correct relative wages, but not wage levels. Takes as inputs share and elasticity parameters across layers of sub-nests, as well as quantity demanded at each bottom-most CES nest layer. Works with Constant Elasticity of Substitution problems with constant returns, up to four nest layers, and two inputs in each sub-nest. 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.

5.2.1 Key Inputs and Outputs for bfw_crs_nested_ces_mpl

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

- **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_YZ_CHOICES** cell array of quantity demanded for the first and second inputs of the bottom-most layer of sub-nests. The last index in each element of the cell array indicates first (1) or second (2) quantities. 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 quantity values at the 1st, 2nd, 3rd, and 4th nest layers: size(CL_MN_YZ_CHOICES{1})= $[1,2]$, size(CL_MN_YZ_CHOICES{2})= $[2,2]$, size(CL_MN_YZ_CHOICES{3})= $[2,2,2]$, size(CL_MN_YZ_CHOICES{4})= $[2,2,2,2]$. Note that only the last layer of quantities needs to be specified, in this case, the 16 quantities at the 4th layer. Given first order conditions, we solve for the 2, 4, and 8 aggregate quantities 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_MPL_PRICE** has the same dimension as CL_MN_YZ_CHOICES, suppose there are four layers, the CL_MN_MPL_PRICE{4} results at the lowest layer includes wages that might be observed in the data. CL_MN_MPL_PRICE cell values at non-bottom layers include aggregate wages.
- **CL_MN_YZ_CHOICES** includes at the lowest layer observed wages, however, also includes higher layer aggregate solved quantities. CL_MN_PRHO and CL_MN_PSHARE are identical to inputs.

5.2.2 Single Nest Layer Two Inputs CES Problem (MPL)

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;

% 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]};
% levels of the two inputs, Values picked from demand problem parallel
% example.
cl_mn_yz_choices = {[0.67537, 1.4589]};
% print option
bl_verbose = true;
mp_func = bfw_mp_func_demand();
bl_bfw_model = false;
[cl_mn_yz_choices, cl_mn_mpl_price] = ...
    bfw_crs_nested_ces_mpl(cl_mn_prho, cl_mn_pshare, cl_mn_yz_choices, ...
    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      coe
                    -    ---    ----    -----    ----    ----    ------    -------    -------    ---

    mpl_price_c1    1     1      2        2       1       2      1.0678    0.53388    0.25168    0.4
```

```
    yz_c1           2     4      2        2       1       2      2.1343     1.0671    0.55404    0.5

xxx TABLE:mpl_price_c1 xxxxxxxxxxxxxxxxxx
             c1         c2
          -------    -------

    r1    0.71184    0.35592

xxx TABLE:yz_c1 xxxxxxxxxxxxxxxxxx
             c1         c2
          -------    ------

    r1    0.67537    1.4589

----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_container_map Scalars
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 i    idx    value
                 -    ---    -----

    prho_c1      1     2      0.1
    pshare_c1    2     3      0.5
```

5.2.3 Single Nest Layer Two Inputs CES Problem, Vary Share and Elasticity (MPL)

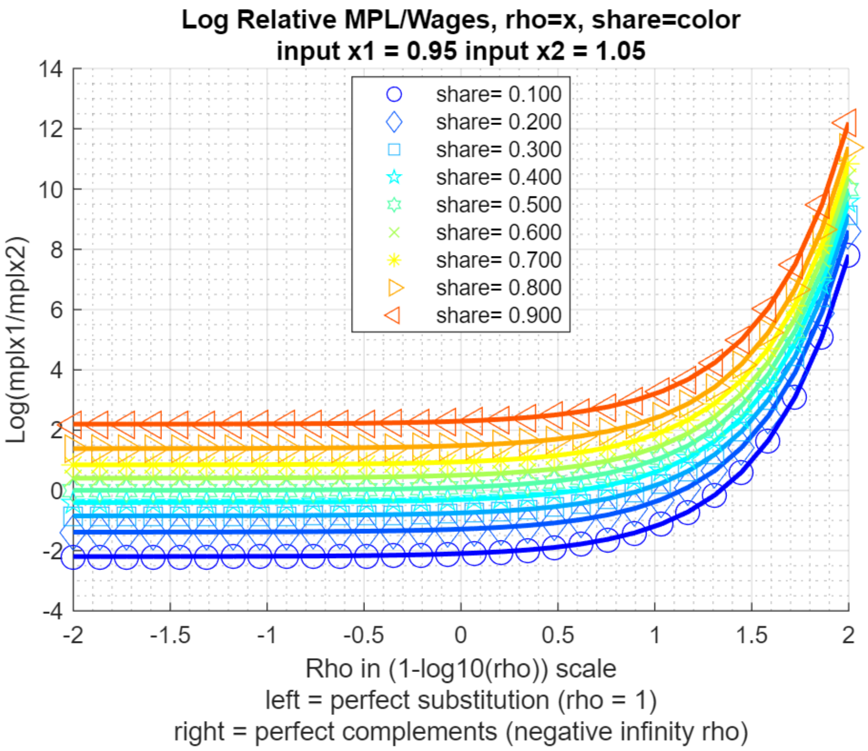
In this second example, we test over different rho values, explore optimal relative choices, as share and elasticity change. In this exercise, we also check, at every combination of rho and share parameter, whether the FOC condition is satisfied by the optimal choices. Also check if at the optimal choices,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

% Input 1 and 2 fixed
fl_x_1 = 0.95;
fl_x_2 = 1.05;

% Define share and rho arrays
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_rela_wage = 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
        % Note that if chosee {[1,1]} below, log(1/1) = log(1) = 0,
```

```
        % elasticity does not matter.
        cl_mn_yz_choices = {[fl_x_1, fl_x_2]};
        % print option
        bl_verbose = false;

        % B. Call function
        [cl_mn_yz_choices, cl_mn_mpl_price] = ...
            bfw_crs_nested_ces_mpl(cl_mn_prho, cl_mn_pshare, cl_mn_yz_choices, ...
            mp_func, bl_verbose, bl_bfw_model);
        % Store results for mpl given input choices
        fl_mpl_x1 = cl_mn_mpl_price{1}(1);
        fl_mpl_x2 = cl_mn_mpl_price{1}(2);
        mt_rela_wage(it_pshare_ctr, it_prho_ctr) = log(fl_mpl_x1/fl_mpl_x2);
    end
end
```

Key results: (1) As share parameter of input 1 goes to zero, input 1 is less productive, and the log(mplx1/mplx2) ratio is lower. (2) Becaus x2 input in this example is larger than x1 input, so as two inputs become more inelastic (more leontief), relative MPL for the lower level input is now larger. At the Leontief extreme, the MPL of the input provided at lower level is infinity.

```
% 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_rela_wage;
% container map settings
mp_support_graph = containers.Map('KeyType', 'char', 'ValueType', 'any');
mp_support_graph('cl_st_graph_title') = {...
    ['Log Relative MPL/Wages, rho=x, share=color'] ...
    ['input x1 = ' num2str(fl_x_1) ' input x2 = ' num2str(fl_x_2)]
    };
mp_support_graph('cl_st_ytitle') = {'Log(mplx1/mplx2)'};
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') = 'best';
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);
```



5.2.4 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. Values picked from demand problem parallel
% example.
cl_mn_yz_choices = {[nan, nan], [0.04789, 6.0934; 2.2044, 0.70496]};
% print option
bl_verbose = true;
[cl_mn_yz_choices, cl_mn_mpl_price] = ...
    bfw_crs_nested_ces_mpl(cl_mn_prho, cl_mn_pshare, cl_mn_yz_choices, ...
    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      coe
                    -    ---    ----    -----    ----    ----    ------    -------    -------    ---

    mpl_price_c1    1     1      2        2       1       2     1.0206    0.51032    0.27499    0.5
    mpl_price_c2    2     2      2        4       2       2     2.3618    0.59045     0.5082    0.8
    prho_c2         3     4      2        2       1       2      -0.65     -0.325    0.95459    -2.
    pshare_c2       4     6      2        2       1       2       1.18       0.59    0.41012    0.6
    yz_c1           5     7      2        2       1       2     4.4862     2.2431    0.68863      0
```

```
    yz_c2           6     8      2        4       2       2      9.0507      2.2627      2.7086      1.

xxx TABLE:mpl_price_c1 xxxxxxxxxxxxxxxxxx
              c1         c2
           -------    -------

    r1     0.31587    0.70476

xxx TABLE:mpl_price_c2 xxxxxxxxxxxxxxxxxx
              c1         c2
           -------    -------

    r1      1.3121    0.13121
    r2     0.39362    0.52484

xxx TABLE:prho_c2 xxxxxxxxxxxxxxxxxx
            c1     c2
           ----    --

    r1     0.35    -1

xxx TABLE:pshare_c2 xxxxxxxxxxxxxxxxxx
           c1      c2
           ---    ----

    r1     0.3    0.88

xxx TABLE:yz_c1 xxxxxxxxxxxxxxxxxx
            c1       c2
           ----    ------

    r1     2.73    1.7562

xxx TABLE:yz_c2 xxxxxxxxxxxxxxxxxx
              c1         c2
           -------    -------

    r1     0.04789     6.0934
    r2      2.2044    0.70496

----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_container_map Scalars
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                 i    idx    value
                 -    ---    -----

    prho_c1      1     3      0.1
    pshare_c1    2     5      0.4

% there are four optimal choices, they are
fl_mpl_x11 = cl_mn_mpl_price{2}(1,1);
fl_mpl_x12 = cl_mn_mpl_price{2}(1,2);
fl_mpl_x21 = cl_mn_mpl_price{2}(2,1);
fl_mpl_x22 = cl_mn_mpl_price{2}(2,2);
% display
st_print = strjoin(...
    ["completed double nest test:", ...
```

```
    ['nest 1 input 1, fl_mpl_x11=' num2str(fl_mpl_x11)], ...
    ['nest 1 input 2, fl_mpl_x12=' num2str(fl_mpl_x12)], ...
    ['nest 2 input 1, fl_mpl_x21=' num2str(fl_mpl_x21)], ...
    ['nest 2 input 2, fl_mpl_x22=' num2str(fl_mpl_x22)], ...
    ['nest 1 input 1, fl_mpl_x11/fl_mpl_x11=' num2str(fl_mpl_x11/fl_mpl_x11)], ...
    ['nest 1 input 2, fl_mpl_x12/fl_mpl_x11=' num2str(fl_mpl_x12/fl_mpl_x11)], ...
    ['nest 2 input 1, fl_mpl_x21/fl_mpl_x11=' num2str(fl_mpl_x21/fl_mpl_x11)], ...
    ['nest 2 input 2, fl_mpl_x22/fl_mpl_x11=' num2str(fl_mpl_x22/fl_mpl_x11)], ...
    ], ";");
st_out = st_print;
ar_ch_out = char(strsplit(st_print,";")');
disp(ar_ch_out);
```

```
completed double nest test:
nest 1 input 1, fl_mpl_x11=1.3121
nest 1 input 2, fl_mpl_x12=0.13121
nest 2 input 1, fl_mpl_x21=0.39362
nest 2 input 2, fl_mpl_x22=0.52484
nest 1 input 1, fl_mpl_x11/fl_mpl_x11=1
nest 1 input 2, fl_mpl_x12/fl_mpl_x11=0.099995
nest 2 input 1, fl_mpl_x21/fl_mpl_x11=0.29998
nest 2 input 2, fl_mpl_x22/fl_mpl_x11=0.39999
```

5.2.5 BFW (2022) Nested Three Branch (Four Layer) Problem (MPL)

The model BFW 2022 has three branches and four layers. one of the branches go down only three layers, the other two branches go down four layers.

First, we prepare the various inputs:

```
% Controls
bl_verbose = true;
bl_bfw_model = true;

% Given rho and beta, solve for equilibrium quantities
mp_func = bfw_mp_func_demand();

% Following instructions in: PrjFLFPMexicoBFW\solvedemand\README.md

% Nests/layers
it_nests = 4;

% Input cell of mn matrixes
it_prho_cl = 1;
it_pshare_cl = 2;
it_yz_share_cl = 3;
for it_cl_ctr = [1,2,3]

    cl_mn_cur = cell(it_nests,1);

    % Fill each cell element with NaN mn array
    for it_cl_mn = 1:it_nests

        bl_yz_share = (it_cl_ctr == it_yz_share_cl);

        if (~bl_yz_share && it_cl_mn == 1)
            mn_nan = NaN;
        elseif (~bl_yz_share && it_cl_mn == 2) || (bl_yz_share && it_cl_mn == 1)
            mn_nan = [NaN, NaN];
```

```
        elseif (~bl_yz_share && it_cl_mn == 3) || (bl_yz_share && it_cl_mn == 2)
            mn_nan = NaN(2,2);
        elseif (~bl_yz_share && it_cl_mn == 4) || (bl_yz_share && it_cl_mn == 3)
            mn_nan = NaN(2,2,2);
        elseif (~bl_yz_share && it_cl_mn == 5) || (bl_yz_share && it_cl_mn == 4)
            mn_nan = NaN(2,2,2,2);
        elseif (~bl_yz_share && it_cl_mn == 6) || (bl_yz_share && it_cl_mn == 5)
            mn_nan = NaN(2,2,2,2,2);
        end
        cl_mn_cur{it_cl_mn} = mn_nan;
    end

    % Name cell arrays
    if (it_cl_ctr == it_prho_cl)
        cl_mn_prho = cl_mn_cur;
    elseif (it_cl_ctr == it_pshare_cl)
        cl_mn_pshare = cl_mn_cur;
    elseif (it_cl_ctr == it_yz_share_cl)
        cl_mn_yz_choices = cl_mn_cur;
    end
end

% Initialize share matrix
rng(123);
for it_cl_mn = 1:it_nests
    mn_pshare = cl_mn_pshare{it_cl_mn};
    if it_cl_mn == 4
        mn_pshare(2,:,:) = rand(2,2);
    else
        mn_pshare = rand(size(mn_pshare));
    end
    cl_mn_pshare{it_cl_mn} = mn_pshare;
end

% Initialize rho matrix
rng(456);
for it_cl_mn = 1:it_nests
    mn_prho = cl_mn_prho{it_cl_mn};
    if it_cl_mn == 4
        mn_prho(2,:,:) = rand(2,2);
    else
        mn_prho = rand(size(mn_prho));
    end
    % Scalling rho between 0.7500 and -3.0000
    % 1 - 2.^(linspace(-2,2,5))
    mn_prho = 1 - 2.^(mn_prho*(4) - 2);
    cl_mn_prho{it_cl_mn} = mn_prho;
end

% Initialize quantities matrix
rng(789);
for it_cl_mn = 1:it_nests
    mn_yz_choices = cl_mn_yz_choices{it_cl_mn};
    if it_cl_mn == 3
        mn_yz_choices(1,:,:) = rand(2,2);
    elseif it_cl_mn == 4
        mn_yz_choices(2,:,:,:) = rand(2,2,2);
    end
```

```
    % Scalling quantities between 3 amd 5
    mn_yz_choices = mn_yz_choices*(2) + 3;
    cl_mn_yz_choices{it_cl_mn} = mn_yz_choices;
end

% Initialize yz matrix
rng(101112);
```

Second, display created inputs:

```
celldisp(cl_mn_prho);


cl_mn_prho{1} =

    0.5017



cl_mn_prho{2} =

    0.6071   -1.1955



cl_mn_prho{3} =

   -1.3523   -0.3346
   -0.4167   -1.9136



cl_mn_prho{4} =


(:,:,1) =

       NaN       NaN
   -1.0512    0.5869


(:,:,2) =

       NaN       NaN
    0.6209    0.1633



celldisp(cl_mn_pshare);


cl_mn_pshare{1} =

    0.6965



cl_mn_pshare{2} =
```

```
    0.2861    0.2269



cl_mn_pshare{3} =

    0.5513    0.4231
    0.7195    0.9808



cl_mn_pshare{4} =


(:,:,1) =

       NaN       NaN
    0.6848    0.4809


(:,:,2) =

       NaN       NaN
    0.3921    0.3432



celldisp(cl_mn_yz_choices);


cl_mn_yz_choices{1} =

   NaN   NaN



cl_mn_yz_choices{2} =

   NaN   NaN
   NaN   NaN



cl_mn_yz_choices{3} =


(:,:,1) =

    3.6467    3.4605
       NaN       NaN


(:,:,2) =

    4.5876    4.2488
       NaN       NaN
```

```
cl_mn_yz_choices{4} =


(:,:,1,1) =

      NaN       NaN
   4.9508    4.5178


(:,:,2,1) =

      NaN       NaN
   3.0212    3.0495


(:,:,1,2) =

      NaN       NaN
   3.2221    4.0763


(:,:,2,2) =

      NaN       NaN
   3.0909    4.1031
```

Third, call function and solve for optimal demand:

```
% Call function
[cl_mn_yz_choices, cl_mn_mpl_price] = ...
    bfw_crs_nested_ces_mpl(cl_mn_prho, cl_mn_pshare, cl_mn_yz_choices, ...
    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
                    --    ---    ----    -----    ----    ----    --------    --------    --------

    mpl_price_c1     1      1      2        2       1       2        1.0002      0.5001     0.28686
    mpl_price_c2     2      2      2        4       2       2        1.0009     0.25022     0.17949
    mpl_price_c3     3      3      3        8       2       4        1.0088      0.1261     0.10191
    mpl_price_c4     4      4      4       16       2       8           NaN         NaN         NaN
    prho_c2          5      6      2        2       1       2      -0.58844    -0.29422      1.2746
    prho_c3          6      7      2        4       2       2       -4.0173     -1.0043     0.76195
    prho_c4          7      8      3        8       2       4           NaN         NaN         NaN
    pshare_c2        8     10      2        2       1       2       0.51299      0.2565    0.041923
    pshare_c3        9     11      2        4       2       2        2.6747     0.66866     0.24087
    pshare_c4       10     12      3        8       2       4           NaN         NaN         NaN
    yz_c1           11     13      2        2       1       2        8.0897      4.0448       0.173
    yz_c2           12     14      2        4       2       2        16.015      4.0039     0.19166
    yz_c3           13     15      3        8       2       4        31.235      3.9044     0.51337
    yz_c4           14     16      4       16       2       8           NaN         NaN         NaN

xxx TABLE:mpl_price_c1 xxxxxxxxxxxxxxxxxx
           c1         c2
          -------    -------
```

```
    r1    0.70294    0.29725

xxx TABLE:mpl_price_c2 xxxxxxxxxxxxxxxxxx
             c1         c2
          --------    -------

    r1     0.19946    0.50351
    r2    0.080381    0.21754

xxx TABLE:mpl_price_c3 xxxxxxxxxxxxxxxxxx
             c1         c2          c3          c4
          --------    -------    --------    ---------

    r1     0.13727    0.24893    0.065108      0.25809
    r2    0.050551    0.21139    0.031132    0.0063057

xxx TABLE:mpl_price_c4 xxxxxxxxxxxxxxxxxx
            c1          c2          c3          c4            c5          c6          c7          c8
          -------    --------    --------    ---------    --------    -------    --------    --------

    r1        NaN         NaN         NaN          NaN         NaN        NaN         NaN          N
    r2    0.02507    0.099481    0.012272    0.0025507    0.027845    0.11203    0.018861    0.00380

xxx TABLE:prho_c2 xxxxxxxxxxxxxxxxxx
            c1         c2
          -------    -------

    r1    0.60709    -1.1955

xxx TABLE:prho_c3 xxxxxxxxxxxxxxxxxx
             c1          c2
          --------    --------

    r1     -1.3523    -0.33464
    r2    -0.41668     -1.9136

xxx TABLE:prho_c4 xxxxxxxxxxxxxxxxxx
            c1         c2         c3         c4
          -------    -------    -------    -------

    r1        NaN        NaN        NaN        NaN
    r2    -1.0512    0.58694    0.62089    0.16334

xxx TABLE:pshare_c2 xxxxxxxxxxxxxxxxxx
            c1         c2
          -------    -------

    r1    0.28614    0.22685

xxx TABLE:pshare_c3 xxxxxxxxxxxxxxxxxx
            c1         c2
          -------    -------

    r1    0.55131    0.42311
    r2    0.71947    0.98076

xxx TABLE:pshare_c4 xxxxxxxxxxxxxxxxxx
```

```
           c1         c2         c3         c4
        -------    -------    -------    -------

    r1      NaN        NaN        NaN        NaN
    r2  0.68483    0.48093    0.39212    0.34318

xxx TABLE:yz_c1 xxxxxxxxxxxxxxxxxx
           c1        c2
        ------    ------

    r1  3.9225    4.1672

xxx TABLE:yz_c2 xxxxxxxxxxxxxxxxxx
           c1        c2
        ------    ------

    r1  4.0073    3.8887
    r2  3.8468    4.2727

xxx TABLE:yz_c3 xxxxxxxxxxxxxxxxxx
           c1        c2        c3        c4
        ------    ------    ------    ------

    r1  3.6467    3.4605    4.5876    4.2488
    r2    4.23    4.2863    3.0635    3.7118

xxx TABLE:yz_c4 xxxxxxxxxxxxxxxxxx
           c1        c2        c3        c4        c5        c6        c7        c8
        ------    ------    ------    ------    ------    ------    ------    ------

    r1     NaN       NaN       NaN       NaN       NaN       NaN       NaN       NaN
    r2  4.9508    4.5178    3.0212    3.0495    3.2221    4.0763    3.0909    4.1031

----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_container_map Scalars
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                i    idx     value
                -    ---    -------

    prho_c1     1     5     0.50172
    pshare_c1   2     9     0.69647
```

Chapter 6

Supply

6.1 bfw_mlogit

This is the example vignette for function: bfw_mlogit from the **PrjLabEquiBFW Package.**

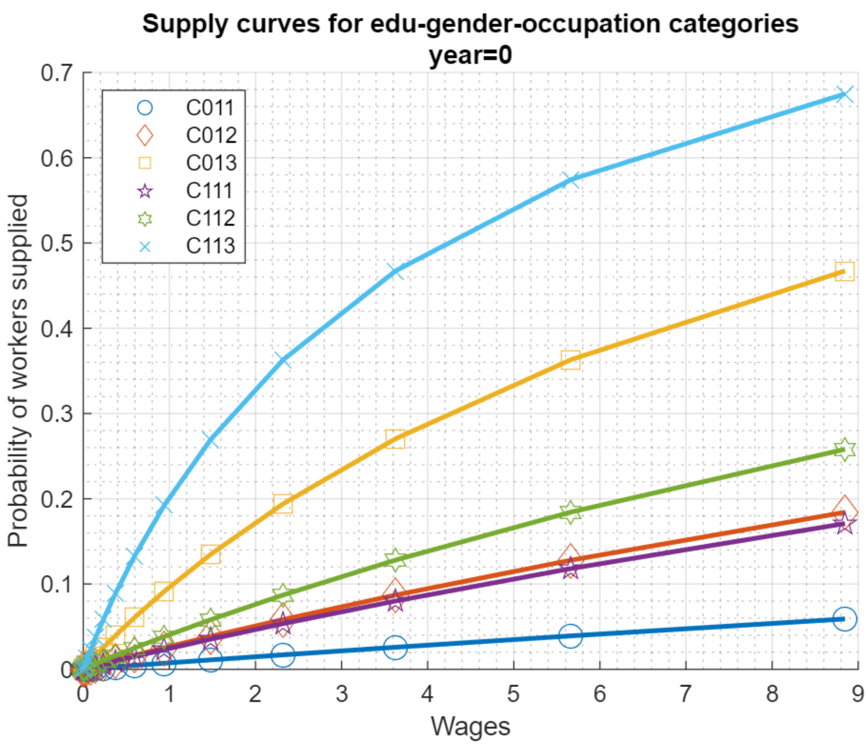
6.1.1 Default

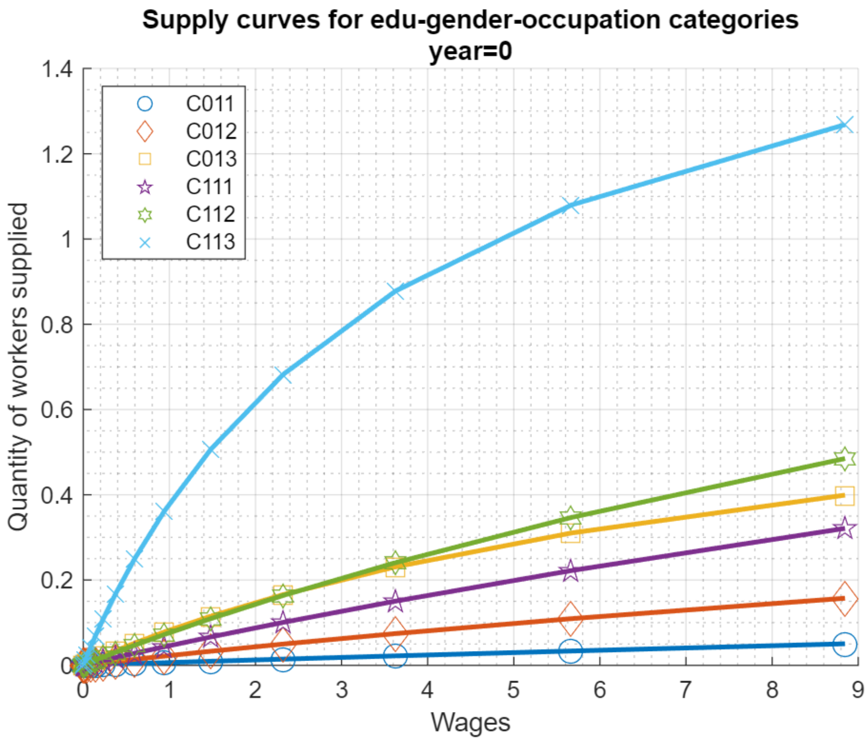
```
[mp_fl_labor_occprbty,mp_fl_labor_supplied] = bfw_mlogit();
```

```
----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_wages Scalars
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            i    idx     value
            -    ---    ------

    C011    1     1     2.1604
    C012    2     2     5.6589
    C013    3     3     5.8023
    C111    4     4     4.5245
    C112    5     5     5.4146
    C113    6     6     8.0437

BFW_SUPPLY_LEVELS_BF18;it_supplier_group=1;SNW_MP_CONTROL=;C011;time=;G01;fl_wage=2.1604
Supply data;potwrker=0.85421;shrmarid=0.87768;shrufive=0.54077;applianc=0.95588;jobscrys=0.613
BFW_SUPPLY_LEVELS_BF18;it_supplier_group=1;SNW_MP_CONTROL=;C012;time=;G01;fl_wage=5.6589
Supply data;potwrker=0.85421;shrmarid=0.87768;shrufive=0.54077;applianc=0.95588;jobscrys=0.613
BFW_SUPPLY_LEVELS_BF18;it_supplier_group=1;SNW_MP_CONTROL=;C013;time=;G01;fl_wage=5.8023
Supply data;potwrker=0.85421;shrmarid=0.87768;shrufive=0.54077;applianc=0.95588;jobscrys=0.613
BFW_SUPPLY_LEVELS_BF18;it_supplier_group=2;SNW_MP_CONTROL=;C111;time=;G11;fl_wage=4.5245
Supply data;potwrker=1.8792;shrmarid=0.9391;shrufive=0.54027;applianc=0.93209;jobscrys=0.613
BFW_SUPPLY_LEVELS_BF18;it_supplier_group=2;SNW_MP_CONTROL=;C112;time=;G11;fl_wage=5.4146
Supply data;potwrker=1.8792;shrmarid=0.9391;shrufive=0.54027;applianc=0.93209;jobscrys=0.613
BFW_SUPPLY_LEVELS_BF18;it_supplier_group=2;SNW_MP_CONTROL=;C113;time=;G11;fl_wage=8.0437
Supply data;potwrker=1.8792;shrmarid=0.9391;shrufive=0.54027;applianc=0.93209;jobscrys=0.613
----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_fl_labor_occprbty Scalars
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            i    idx      value
            -    ---    --------

    C011    1     1     0.015821
```

```
    C012    2     2      0.12787
    C013    3     3      0.36854
    C111    4     4     0.097357
    C112    5     5      0.17795
    C113    6     6      0.65443

----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_fl_labor_supplied Scalars
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            i    idx     value
            -    ---    --------

    C011    1     1     0.013514
    C012    2     2      0.10923
    C013    3     3      0.31481
    C111    4     4      0.18296
    C112    5     5      0.33441
    C113    6     6       1.2298

----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_fl_labor_supplied_3v0f Scalars
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            i    idx     value
            -    ---    --------

    C011    1     1     0.013514
    C012    2     2      0.10923
    C013    3     3      0.31481
    C111    4     4      0.18296
    C112    5     5      0.33441
    C113    6     6       1.2298

----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_fc_labor_occprbty_3v0f Functions
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
             i      idx                                     functionString
            ---     ---     -----------------------------------------------------------------------------

    C011    "1"     "1"     "@(w1,w2,w3)fc_ar_prob_wrk(fl_psi0_manual,psi1,w1,fc_prob_denom_wage(w1,w2
    C012    "2"     "2"     "@(w1,w2,w3)fc_ar_prob_wrk(fl_psi0_routine,psi1,w2,fc_prob_denom_wage(w1,w
    C013    "3"     "3"     "@(w1,w2,w3)fc_ar_prob_wrk(fl_psi0_analytical,psi1,w3,fc_prob_denom_wage(w
    C111    "4"     "4"     "@(w1,w2,w3)fc_ar_prob_wrk(fl_psi0_manual,psi1,w1,fc_prob_denom_wage(w1,w2
    C112    "5"     "5"     "@(w1,w2,w3)fc_ar_prob_wrk(fl_psi0_routine,psi1,w2,fc_prob_denom_wage(w1,w
    C113    "6"     "6"     "@(w1,w2,w3)fc_ar_prob_wrk(fl_psi0_analytical,psi1,w3,fc_prob_denom_wage(w

----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_fc_labor_supplied_3v0f Functions
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
             i      idx                                  functionString
            ---     ---     -----------------------------------------------------------------------------

    C011    "1"     "1"     "@(w1,w2,w3)fc_supply(fl_potwrklei_potwrker,fc_labor_occprbty_3v0f(w1,w2,w
    C012    "2"     "2"     "@(w1,w2,w3)fc_supply(fl_potwrklei_potwrker,fc_labor_occprbty_3v0f(w1,w2,w
    C013    "3"     "3"     "@(w1,w2,w3)fc_supply(fl_potwrklei_potwrker,fc_labor_occprbty_3v0f(w1,w2,w
```

```
    C111    "4"    "4"    "@(w1,w2,w3)fc_supply(fl_potwrklei_potwrker,fc_labor_occprbty_3v0f(w1,w2,w
    C112    "5"    "5"    "@(w1,w2,w3)fc_supply(fl_potwrklei_potwrker,fc_labor_occprbty_3v0f(w1,w2,w
    C113    "6"    "6"    "@(w1,w2,w3)fc_supply(fl_potwrklei_potwrker,fc_labor_occprbty_3v0f(w1,w2,w

----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_fc_labor_occprbty_1v2f Functions
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
             i      idx                                       functionString

            ---     ---    ---------------------------------------------------------------------------

    C011    "1"    "1"    "@(wage)fc_ar_prob_wrk(fl_psi0_manual,psi1,wage,fc_prob_denom_wage(wage,fl
    C012    "2"    "2"    "@(wage)fc_ar_prob_wrk(fl_psi0_routine,psi1,wage,fc_prob_denom_wage(fl_w1,
    C013    "3"    "3"    "@(wage)fc_ar_prob_wrk(fl_psi0_analytical,psi1,wage,fc_prob_denom_wage(fl_
    C111    "4"    "4"    "@(wage)fc_ar_prob_wrk(fl_psi0_manual,psi1,wage,fc_prob_denom_wage(wage,fl
    C112    "5"    "5"    "@(wage)fc_ar_prob_wrk(fl_psi0_routine,psi1,wage,fc_prob_denom_wage(fl_w1,
    C113    "6"    "6"    "@(wage)fc_ar_prob_wrk(fl_psi0_analytical,psi1,wage,fc_prob_denom_wage(fl_

----------------------------------------
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
CONTAINER NAME: mp_fc_labor_supplied_1v2f Functions
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
             i      idx                              functionString

            ---     ---    ----------------------------------------------------------------------

    C011    "1"    "1"    "@(wage)fc_supply(fl_potwrklei_potwrker,fc_labor_occprbty_1v2f(wage))"
    C012    "2"    "2"    "@(wage)fc_supply(fl_potwrklei_potwrker,fc_labor_occprbty_1v2f(wage))"
    C013    "3"    "3"    "@(wage)fc_supply(fl_potwrklei_potwrker,fc_labor_occprbty_1v2f(wage))"
    C111    "4"    "4"    "@(wage)fc_supply(fl_potwrklei_potwrker,fc_labor_occprbty_1v2f(wage))"
    C112    "5"    "5"    "@(wage)fc_supply(fl_potwrklei_potwrker,fc_labor_occprbty_1v2f(wage))"
    C113    "6"    "6"    "@(wage)fc_supply(fl_potwrklei_potwrker,fc_labor_occprbty_1v2f(wage))"
```

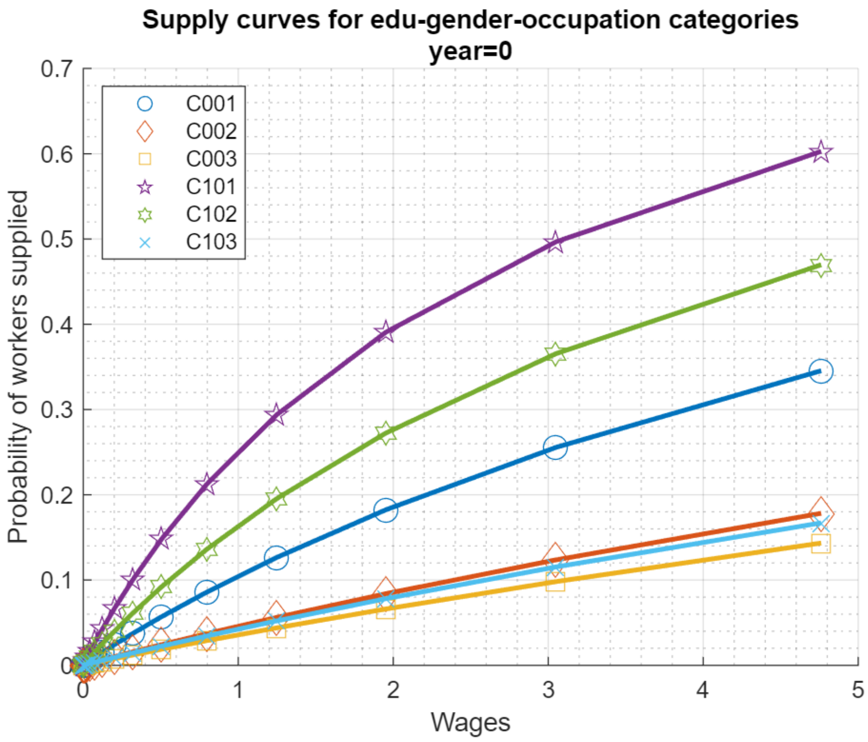


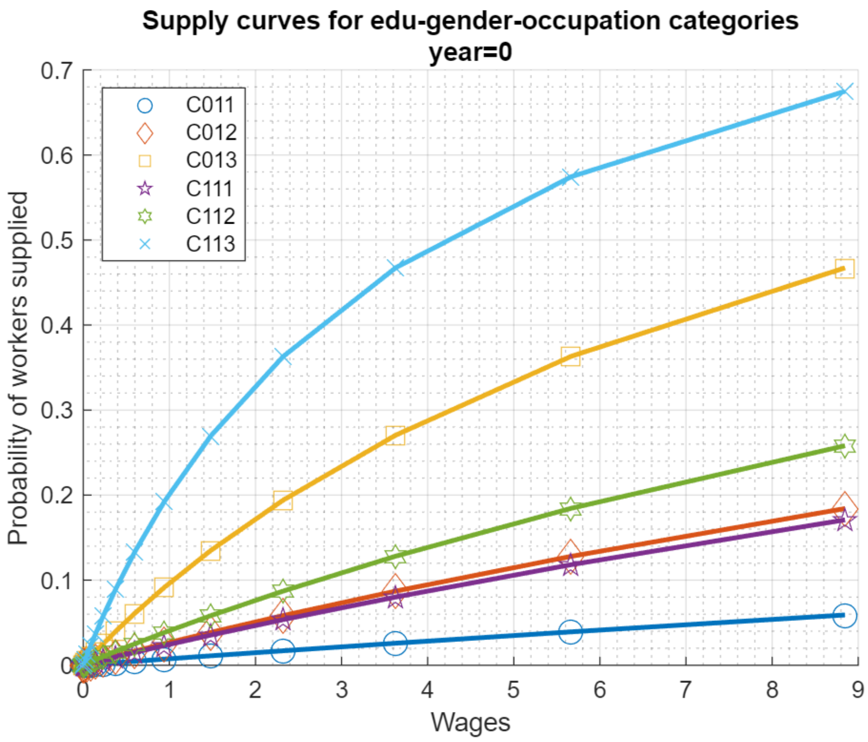


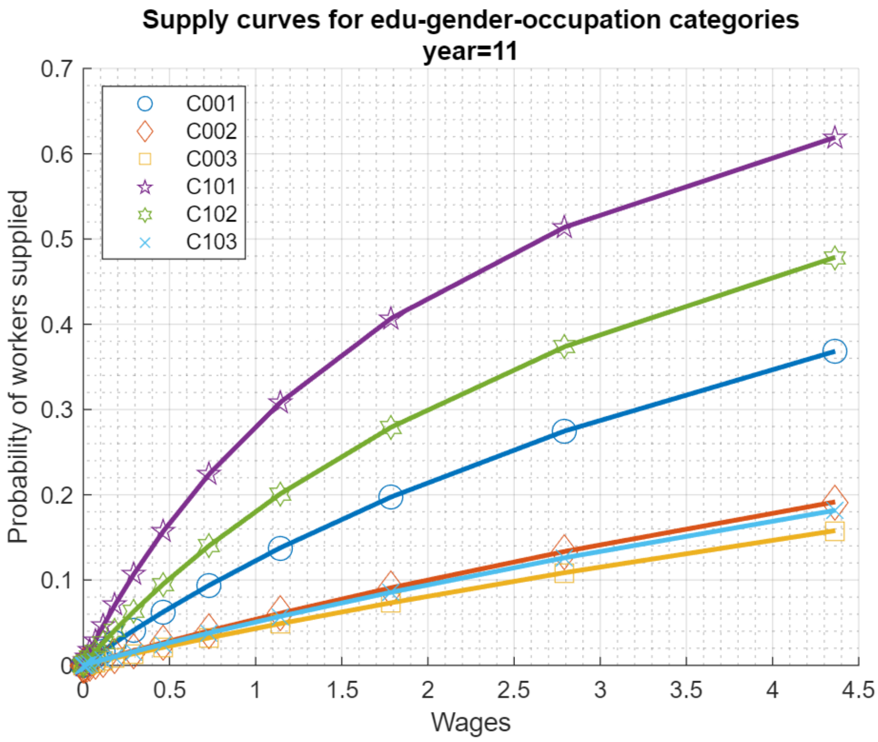
6.1.2 Visualize Supply Curves Different Years

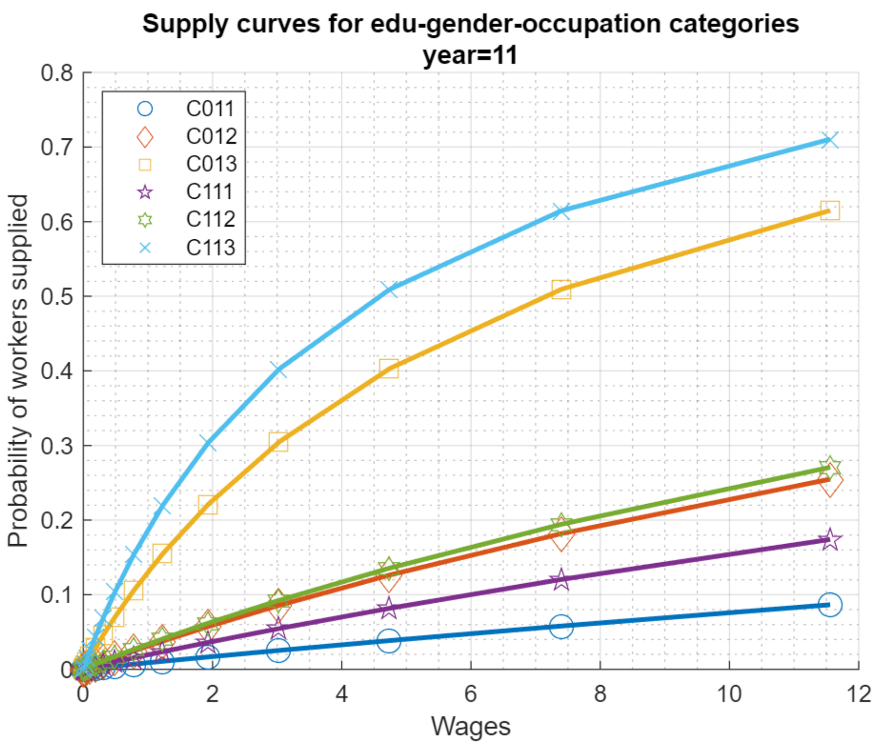
```
% 1. Print and Graph options
bl_verbose = false;
bl_graph = true;
ar_it_prob_or_quant = [1];

% 2. Get Parameters and data
bl_log_wage = true;
bl_verbose_nest = false;
% Get Parameters
mp_params = bfw_mp_param_esti(bl_log_wage);
mp_param_aux = bfw_mp_param_aux(bl_verbose_nest);
mp_params = [mp_params ; mp_param_aux];
% Get Data
mp_data = bfw_mp_data(bl_verbose_nest);
% Get Functions
mp_func = bfw_mp_func_supply(bl_log_wage, bl_verbose_nest);
% Get Controls
mp_controls = bfw_mp_control();

% 3. Data from which year, only integer year value allowed
% ar_it_data_year = [1989 1994 2000 2008 2014];
ar_it_data_year = [1989 2000 2014];
for it_data_year=ar_it_data_year

    % 4. Which categories to obtain data from, there are 12 possible
    % For non-college equilibrium, six wages, three female, three males
    % gen_occ = gender occupation
    for bl_skilled = [false true]
        if (bl_skilled)
            mt_st_gen_occ_categories = [...
                "C011", "C012", "C013"; ...
                "C111", "C112", "C113"];
```

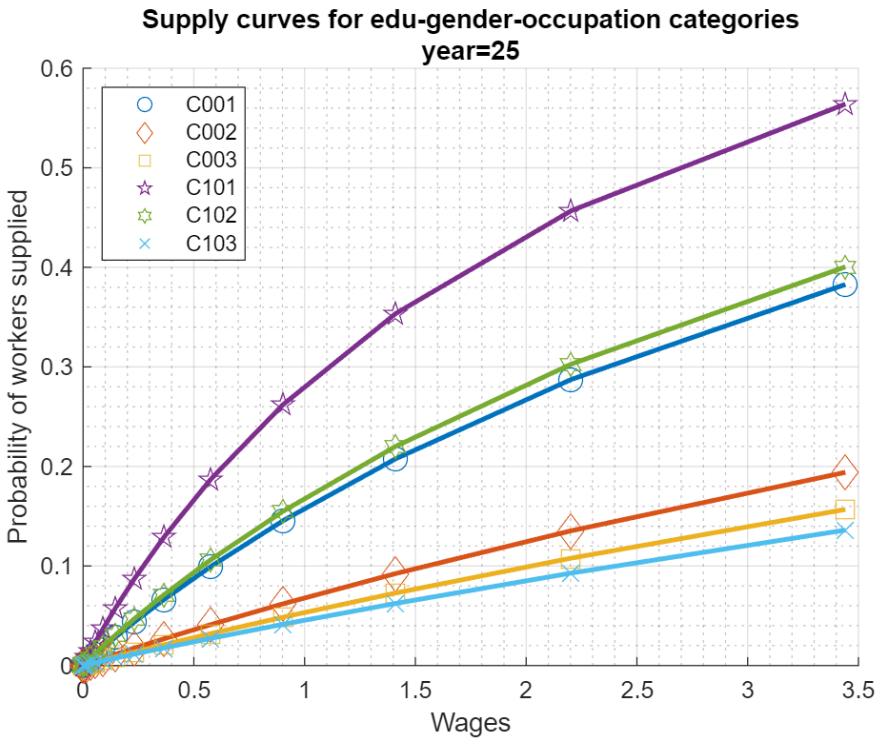
```
        else
            mt_st_gen_occ_categories = [...
                "C001", "C002", "C003"; ...
                "C101", "C102", "C103"];
        end

        % 5. Array of wages, at most, since there are six nests, there are 12
        % prices possible. And there are 12 quantity supplies possible, coming
        % from four tyeps of workers, each supply 3 + home categories.
        mp_wages = containers.Map('KeyType', 'char', 'ValueType', 'any');
        % Obtain some equilibrium wage data as testing inputs
        mp_path = bfw_mp_path();
        spt_codem_data = mp_path('spt_codem_data');
        tb_data_pq = mp_data('tb_data_pq');
        tb_data_pq = tb_data_pq(:, ["year", "category", "numberWorkers", "meanWage"]);
        ar_st_gen_occ_categories = mt_st_gen_occ_categories(:)';
        for st_gen_occ=ar_st_gen_occ_categories
            tb_gen_occ_over_years = tb_data_pq(strcmp(tb_data_pq.category, st_gen_occ),:);
            fl_wage_one_year = tb_gen_occ_over_years(tb_gen_occ_over_years.year == (it_data_year), :
            mp_wages(st_gen_occ) = fl_wage_one_year{1, "meanWage"};
        end

        % Print Wages
        % ff_container_map_display(mp_wages);

        % Get date offset
        params_group = values(mp_data, {'date_esti_offset'});
        [date_esti_offset] = params_group{:};

        % Run function
        [mp_fl_labor_occprbty,mp_fl_labor_supplied] = bfw_mlogit(...
            mp_params, mp_data, mp_func, mp_controls, ...
            mt_st_gen_occ_categories, it_data_year - date_esti_offset, mp_wages, ...
            bl_verbose, bl_graph, ...
            ar_it_prob_or_quant);
    end
end
```

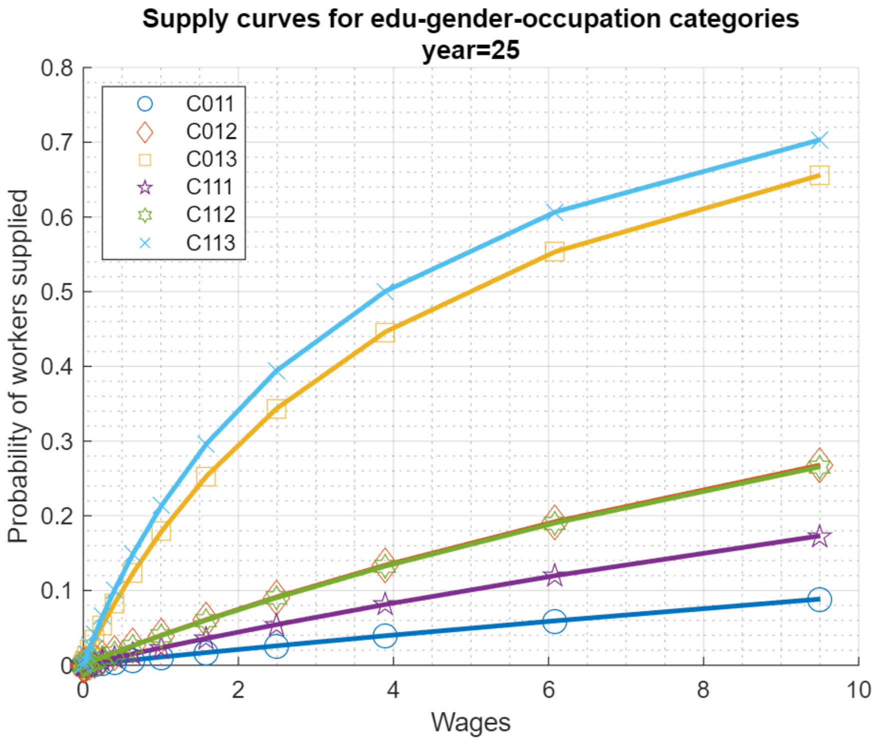

Supply curves for edu-gender-occupation categories
year=0
C001
C002
C003
C101
C102
C103
Probability of workers supplied
Wages
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
1
2
3
4
5


Supply curves for edu-gender-occupation categories
year=0
C011
C012
C013
C111
C112
C113
Probability of workers supplied
Wages
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
1
2
3
4
5
6
7
8
9


Supply curves for edu-gender-occupation categories
year=11
C001
C002
C003
C101
C102
C103
Probability of workers supplied
Wages
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.5
1
1.5
2
2.5
3
3.5
4
4.5


Supply curves for edu-gender-occupation categories
year=11
C011
C012
C013
C111
C112
C113
Probability of workers supplied
Wages
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12


Supply curves for edu-gender-occupation categories
year=25
Probability of workers supplied
Wages
C001
C002
C003
C101
C102
C103
0
0.1
0.2
0.3
0.4
0.5
0.6
0.5
1
1.5
2
2.5
3
3.5


Supply curves for edu-gender-occupation categories
year=25
Probability of workers supplied
Wages
C011
C012
C013
C111
C112
C113
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10

Appendix A

Index and Code Links

A.1 Introduction links

1. The Labor Demand and Supply Problem: **mlx** | **m** | **pdf** | **html**
   - The Labor Demand and Supply Problem

A.2 Core Functions links

1. CES Demand Core Functions: **mlx** | **m** | **pdf** | **html**
   - This function generates a container map with key CES demand-side equation for a particular sub-nest.
   - **PrjLabEquiBFW**: *bfw_mp_func_demand()*
2. Multinomial Logit Core Functions: **mlx** | **m** | **pdf** | **html**
   - This function generates a container map with key multinomial logit supply-side equations.
   - **PrjLabEquiBFW**: *bfw_mp_func_supply()*

A.3 Parameters links

1. bfwx_mp_path: **mlx** | **m** | **pdf** | **html**
   - bfw_mp_path
   - **PrjLabEquiBFW**: *bfw_mp_path()*
2. bfw_mp_param_esti: **mlx** | **m** | **pdf** | **html**
   - bfw_mp_param_esti
   - **PrjLabEquiBFW**: *bfw_mp_param_esti()*

A.4 Data links

1. bfwx_mp_data: **mlx** | **m** | **pdf** | **html**
   - bfw_mp_data
   - **PrjLabEquiBFW**: *bfw_mp_data()*

A.5 Demand links

1. Solve Nested CES Optimal Demand (CRS): **mlx** | **m** | **pdf** | **html**
   - This function solves optimal choices given CES production function under cost minimization.
   - Works with Constant Elasticity of Substitution problems with constant returns, up to four nest layers, 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 with Constant Elasticity of Substitution problems with constant returns, up to four nest layers, and two inputs in each sub-nest.
- **PrjLabEquiBFW**: *bfw_crs_nested_ces()*

2. Compute Nested CES MPL Given Demand (CRS): **mlx** | **m** | **pdf** | **html**
    - Given labor quantity demanded, using first-order relative optimality conditions, find the marginal product of labor given CES production function.
    - Takes as inputs share and elasticity parameters across layers of sub-nests, as well as quantity demanded at each bottom-most CES nest layer.
    - Works with Constant Elasticity of Substitution problems with constant returns, up to four nest layers, and two inputs in each sub-nest.
    - 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.
    - **PrjLabEquiBFW**: *bfw_crs_nested_ces_mpl()*

A.6 Supply links

1. bfwx_mlogit: **mlx** | **m** | **pdf** | **html**
    - bfwx_mlogit
    - **PrjLabEquiBFW**: *bfwx_mlogit()*

Bibliography

The MathWorks Inc (2021). *MATLAB*. Matlab package version 2021b.

Xie, Y. (2020). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.18.