Documentation of Simulations in 'Optimal Taxation in Theory and Practice'

N. Gregory Mankiw   Matthew Weinzierl   Danny Yagan

November 2009

(Questions to Danny Yagan at yagan@fas.harvard.edu.)

1 Introduction

We simulate optimal income tax schedules in the setup of Mirrlees (1971). Mirrleesian analytic results are typically complex, and numerical simulations can help to reveal the shape and level of the optimal tax schedule.

One approach to simulating Mirrleesian optimal tax schedules involves direct numerical maximization of the planner's social welfare function under constraints. The algorithm is as follows. A continuous ability distribution is binned into a discrete approximation. MATLAB, or a similar optimization program, is fed that discrete ability distribution and maximizes a social welfare function by allocating a consumption-labor bundle to each ability bin, subject to a feasibility constraint and incentive compatibility constraints. This algorithm has the virtue of being straightforward, as it recreates the analytical social planner's problem. However, it is computationally intensive. Economists are forced to use coarse approximations to the underlying ability distribution, yielding correspondingly imprecise approximations of the true optimal tax schedule. This is particularly problematic for simulating optimal income taxes on high earners.

We take a complementary approach. We extend a method recommended to us by Jeffrey Liebman from experience in the quasi-linear utility case. Our algorithm uses the Saez (2001) first-order condition for optimal marginal income tax rates and an updating rule to find a fixed-point optimal tax schedule.

The algorithm is as follows. We assume a simple initial tax schedule. Given that initial tax schedule, agents choose optimal labor supply. Given this labor supply and the initial tax schedule, the social planner's first-order condition suggests a new tax schedule. This loop is repeated until a fixed-point optimal tax schedule is found. The algorithm executes quickly and generates precise optimal tax schedules. We confirm all of our fixed-point results using the direct optimization approach above.

2   The Fixed-Point Algorithm

The fixed-point algorithm proceeds according to the following steps. Each is discussed in detail below. The MATLAB code we use to implement the algorithm is in Section 3.

1. Load into MATLAB a matrix of wages (abilities) and the CDF evaluated at each wage. Use this matrix to approximate numerically the true wage distribution with a probability mass function.

2. Using a starting marginal tax schedule and lump-sum transfer, calculate optimal labor supply at each wage. Calculate resulting utility at each wage.

3. Given utility at each wage, use the social planner's first-order condition to compute an alternative marginal tax schedule. Use this to update the original marginal tax schedule and lump-sum transfer.

4. Repeat Steps 2 and 3 until the updating is trivially small. The result is the fixed-point tax schedule.

5. Confirm that this is the optimal tax schedule by checking the second-order condition that before-tax income is non-decreasing in wages.

We have no general results that guarantee convergence to a fixed-point. However, if a fixed-point schedule is found and the second-order condition is satisfied, it is the optimal schedule. This result relies on the agent utility function satisfying the single-crossing property (see Salanie 2003).

2.1   Step 1: Discretize the wage distribution

Each simulation requires an exogenous distribution of wages (ability). We use either a lognormal approximation of the U.S. wage distribution or a modification of the lognormal approximation in which we use a Pareto tail above the 95th percentile wage. We use March CPS data and the *lognfit* Stata maximum-likelihood fitting function to parameterize the U.S. wage distribution. The lognormal parameterization for 2007 is $(\mu, \sigma) = (2.757, 0.5611)$, where wages are in dollars per hour. The lognormal parameterization for 1979 is $(\mu, \sigma) = (2.721, 0.4880)$, in 2007 dollars. For the lognormal-Pareto distribution, we append a Pareto tail with parameter $a = 2$, taken from Saez (2001), above wage \$42.50; the Pareto tail is scaled so that the resulting lognormal-Pareto distribution is continuous. We assume only "disabled" agents earn less than \$3.50 per hour. We assume that an atom of disabled agents have wage \$0.01 and are five percent of the population, which according to the Social Security Administration is approximately the percentage of total workers on public disability insurance. Finally, the distribution is scaled by a constant factor to ensure that it integrates to 1.

Given a continuous wage distribution, we discretize it for use in MATLAB, so we approximate the true probability density function $f(w)$ with a discrete probability mass function $\pi(w)$. We discuss in Step 3 the consequences of this approximation. We divide the wage PDF into bins centered from $w_1 = \$0.01$ to $w_N = \$500.51$ with bin widths of $\Delta = \$3.50$. This means that our simulation has $N = 144$ wage bins. The input MATLAB file ("input_data.txt" in our code) is a vector of the $N+1$ wages from $\left(w_1 - \frac{\Delta}{2}\right)$ to $\left(w_N + \frac{\Delta}{2}\right)$ and an associated vector of the true CDF $F(w)$ evaluated at each of those wages. After this is input to MATLAB, a PMF is generated with N points of support $w_1, w_2, ..., w_N$, where the mass $\pi(w)$ on each wage is equal to $F\left(w + \frac{\Delta}{2}\right) - F\left(w - \frac{\Delta}{2}\right)$. Let $\Pi(w)$ be the CDF associated with PMF $\pi(w)$. All subsequent calculations are performed at the wages $w_1, w_2, ..., w_N$.

2.2 Step 2: Compute optimal labor supply and resulting utility at each wage

This step begins with a marginal tax schedule and a lump-sum transfer. In the first iteration, these are exogenously specified; we begin ours with a flat tax of 35% at each wage[1] and a lump-sum transfer of almost zero. In all subsequent iterations, the marginal tax schedule and lump-sum transfer are inherited from the previous iteration.

We now find optimal consumption and labor supply at each wage, given the marginal tax at that wage and the lump-sum transfer. Each agent has the same additively separable preferences over consumption and labor such that the single-crossing property holds (see Salanie 2003). In our simulations, the utility function is:

$$U(c,l) = u(c) + v(l) = \frac{c^{1-\gamma} - 1}{1 - \gamma} - \frac{\alpha l^\sigma}{\sigma}$$

with $\gamma = 1.5$, $\alpha = 2.55$, and $\sigma = 3$. Thus utility is constant relative risk aversion in consumption with a coefficient of relative risk aversion of 1.5 and isoelastic in labor with a Frisch elasticity of labor supply of 0.5. The parameter α is an innocuous scaling factor that affects the level of l; it has no effect on the shape of optimal tax schedules. In Step 3, we will use $U(w)$ to denote utility of agent w.

Beginning at the lowest wage, MATLAB calculates each agent's optimal labor supply l by maximizing utility under the budget constraint $c = y - T(y) + [\text{lump-sum transfer}]$, where pre-tax income is $y = wl$ and tax liability $T(y)$ is a function of income.[2] The function $T(y)$ is not well-defined because the full economy's incomes are not yet known. As a substitute, we calculate $T(y)$ for the purposes of computing optimal labor

---

[1] Note that in the simulations, marginal tax rates are associated with wages even though the planner cannot observe wages. This is a computational short-cut that is benign because the optimal tax schedule requires that agents reveal their wages. Incentive compatibility of consumption-income bundles is confirmed at the end of the exercise.

[2] Some optimal income tax papers define the tax schedule $T(y)$ as inclusive of the lump-sum transfer. This makes no difference in the analysis.

supply as follows. Suppose that optimal labor supply has been calculated for agents $w_1, w_2, ..., w_{i-1}$ and MATLAB is now finding optimal labor supply for an agent with wage w_i and the given marginal tax rate T_i'. We construct the income tax schedule $T(y)$ using tax brackets such that marginal tax rate T_1' applies on income earned from 0 to y_1, marginal tax rate T_2' applies on income earned from y_1 to y_2, marginal tax rate T_3' applies on income earned from y_2 to y_3, etc., and marginal tax rate T_i' applies on all income earned above y_{i-1}. In this way, MATLAB calculates $T(y_i)$ for any guess of optimal labor supply l_i as it searches for the optimum. This process is repeated until optimal labor supply is found for agents of each wage. This process is not necessarily without loss of generality because the order in which optimal labor supply is chosen can matter. However, at the optimal income tax schedule, before-tax income is non-decreasing in wages, so this algorithm is natural for its purpose. Optimality of the fixed-point tax schedule is confirmed in Step 5.

Using optimal labor supply at each wage, we compute consumption and utility at each wage.

2.3 Step 3: Use the planner's FOC to update the tax schedule and transfer

From equation 25 of Saez (2001), the first-order condition of the social planner's problem is:

$$\frac{T'(y(w))}{1-T'(y(w))} = \left(\frac{1+\varepsilon^u(w)}{\varepsilon^c(w)}\right)\left(\frac{U_c(w)}{wf(w)}\right)\int_w^\infty \left(1-\frac{G'(U(\theta))U_c(\theta)}{p}\right)\left(\frac{1}{U_c(\theta)}\right)\left(e^{-\int_w^\theta \frac{l(s)U_{cl}(s)}{sU_c(s)}ds}\right)f(\theta)\,d\theta$$

where $\varepsilon^u(w)$ is the uncompensated labor supply elasticity at w, $\varepsilon^c(w)$ is the compensated labor supply elasticity at w, θ and s index wages in the integrals, $G(U(\theta))$ is the social value of $U(\theta)$, and p is the Lagrange multiplier on the planner's budget constraint. Given a utility function that satisfies the single-crossing property, this first-order condition and the second-order condition of non-decreasing $y(w)$ are sufficient conditions for having found the optimal tax schedule.

We use additively separable utility $U(c,l) = u(c) - v(l)$, so $U_{cl} = 0$ for all w. We also use the linear utilitarian social welfare function $\int_0^\infty U(w)f(w)\,dw$ so that $G(U(w)) = U(w)$ and $G'(U(w)) = 1$ for all w. We can thus rewrite the first-order condition as:

$$
\begin{aligned}
\frac{T'(y(w))}{1-T'(y(w))} &= \left(\frac{1+\varepsilon^u(w)}{\varepsilon^c(w)}\right)\left(\frac{u'(c(w))}{wf(w)}\right)\int_w^\infty \left(1-\frac{u'(c(\theta))}{p}\right)\left(\frac{1}{u'(c(\theta))}\right)f(\theta)\,d\theta \\
&= \left(\frac{1+\varepsilon^u(w)}{\varepsilon^c(w)}\right)\left(\frac{u'(c(w))}{wf(w)}\right)\int_w^\infty \left(\frac{1}{u'(c(\theta))}-\frac{1}{p}\right)f(\theta)\,d\theta \\
&= \left(\frac{1+\varepsilon^u(w)}{\varepsilon^c(w)}\right)\left(\frac{u'(c(w))}{wf(w)}\right)\left[\int_w^\infty \frac{1}{u'(c(\theta))}f(\theta)\,d\theta - (1-F(w))\frac{1}{p}\right]
\end{aligned}
$$

4

The Lagrange multiplier p is often referred to as the marginal value of public funds because it measures the increase in social welfare obtained from an incentive-compatible loosening of the planner's budget constraint. It can be computed as follows. Due to the incentive-compatibility constraints, a marginal increase in public funds is optimally distributed such that each agent's utility rises by the same amount. The cost in consumption terms of raising utility marginally for agent θ is $\frac{1}{u'(c(\theta))}$, so the cost of an incentive-compatible marginal increase in average utility is $\int_0^\infty \frac{1}{u'(\theta)} f(\theta) \, d\theta$. The value to the planner of a marginal unit of public funds is the inverse of this cost, so:

$$p = \frac{1}{\int_0^\infty \frac{1}{u'(\theta)} f(\theta) \, d\theta}$$

Plugging this into the equation above, we obtain:

$$\frac{T'(y(w))}{1 - T'(y(w))} = \left(\frac{1 + \varepsilon^u(w)}{\varepsilon^c(w)} \right) \left(\frac{u'(c(w))}{wf(w)} \right) \left[\int_w^\infty \frac{1}{u'(c(\theta))} f(\theta) \, d\theta - (1 - F(w)) \int_0^\infty \frac{1}{u'(\theta)} f(\theta) \, d\theta \right]$$

We approximate the right-hand side of this equation using our discrete approximation to the underlying wage distribution. Given the consumption-labor allocations derived in Step 2 by agent maximization under a given transfer and marginal tax schedule, we derive an alternative marginal tax schedule at each wage $w_1, w_2, ..., w_N$ by computing the right-hand side of the following equation:

$$\frac{T'(y(w))}{1 - T'(y(w))}$$

$$\approx \left(\frac{1 + \varepsilon^u(w)}{\varepsilon^c(w)} \right) \left(\frac{u'(c(w))}{w \frac{\pi(w)}{\Delta}} \right) \left[\int_{w + \frac{\Delta}{2}}^{w_N + \frac{\Delta}{2}} \frac{1}{u'\left(c\left(\widehat{\theta}\right)\right)} \left(\frac{\pi\left(\widehat{\theta}\right)}{\Delta} \right) d\theta - (1 - \Pi(w)) \int_{w_1 - \frac{\Delta}{2}}^{w_N + \frac{\Delta}{2}} \frac{1}{u'\left(c\left(\widehat{\theta}\right)\right)} \left(\frac{\pi\left(\widehat{\theta}\right)}{\Delta} \right) d\theta \right]$$

$$= \left(\frac{1 + \varepsilon^u(w)}{\varepsilon^c(w)} \right) \left(\frac{u'(c(w))}{w \frac{\pi(w)}{\Delta}} \right) \left[\sum_{w_i = w+1}^{w_N} \frac{\pi(w_i)}{u'(c(w_i))} - (1 - \Pi(w_i)) \sum_{w_i = w_1}^{w_N} \frac{\pi(w_i)}{u'(c(w_i))} \right]$$

and solving for $T'(y(w))$, where $\widehat{\theta}$ is the midpoint of the wage bin into which wage θ falls. This equation illuminates the sense in which our simulations are approximations of the true optimal tax schedule. First, we consider only a bounded subset of the true wage distribution. Second, we assume that all agents within the same wage bin obtain identical utility, which means that the resulting marginal tax schedule is constant within wage bins. With a wide enough wage range $(w_N - w_1)$ and a small enough bin width Δ, the deviations are minor. The approximation is exact as $w_1 \to -\infty$, $w_N \to \infty$, and $\Delta \to 0$.

We create a new marginal tax schedule at each wage by averaging the tax schedule from Step 2 with the alternative tax schedule derived from the social planner's first-order condition. This will be our new marginal tax schedule when repeating Step 2. We derive the new lump-sum transfer by finding optimal labor supply under the new marginal tax schedule and old lump-sum transfer and using that to compute

pre-transfer government revenue. We set the new lump-sum transfer equal to this pre-transfer government revenue.[3]

2.4 Step 4: Arrive at the fixed-point tax schedule

We repeat Steps 2 and 3 until we arrive at a tax schedule that is updated trivially in Step 3 and a government budget constraint that is sufficiently close to holding with equality. In our simulations, we halt the loop when no wage's marginal tax rate is updated by more than 10^{-6} percent and when government revenue is within 10^{-6} units of the transfer.

2.5 Step 5: Confirm optimality of the fixed-point tax schedule

Given a fixed-point tax schedule, all that is left is to confirm the second order condition that $y(w)$ is non-decreasing. We also double-check that no incentive constraint is violated.

In interpreting the optimal tax schedule, it is important to remember that the optimal marginal tax on the highest wage w_N is always zero in such simulations. This is the "no distortion at the top" result that holds in every Mirrleesian economy with bounded wages. If the true wage distribution is unbounded and takes certain shapes (see the discussion in Lesson 2 of the main paper), this may be misleading.

3 MATLAB Implementation

The following four MATLAB files implement the fixed-point algorithm.[4] These files, along with a sample input file of wages, can be found at http://www.people.fas.harvard.edu/~yagan in their original formats. For help in navigating them:

- "FP_MTR_sim_exec.m" is the execution file.

- Function "FP_find_opt_l_.m" finds optimal labor supply for all agents given a tax schedule and transfer. It is called by the execution file.

---

[3]In our simulations, the social planner has no independent purchases to make, so tax revenue goes entirely to finance the lump-sum transfer.

[4]We thank Jan Duras for correcting sign errors in an earlier version of this code that cancelled out and, therefore, did not affect the results.

- Function "FP_opt_l_obj_.m" is the objective function called by "FP_find_opt_l_.m".

- Function "FP_consump_.m" calculates a single agent's consumption given his labor supply, the tax schedule, the transfer, and a vector of the incomes of agents with lower wages. It is called by "FP_opt_l_obj_.m" and the execution file.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FIXED-POINT MIRRLEES SIMULATIONS
%
% EXECUTION FILE
% FP_MTR_sim_exec.m
%
% Mankiw-Weinzierl-Yagan "Optimal Taxation in Theory and Practice"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


delete diary_FP_MTR_sim
diary diary_FP_MTR_sim;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% SETUP

% SET FORMATS.
format long;
format compact;
warning off;

% LOAD DATA.
clear;
clc;
data = load('input_data.txt');
options=optimset('TolCon',1e-13,'TolFun',1e-13,'TolX',1e-13,'Display','off',
'MaxFunEvals',10000000,'MaxIter',200);

% INPUT WAGES AND CDF EVALUATED AT EACH WAGE.
cdf_orig = data(:,2);
wage_orig = data(:,1);
clear data
wage_orig_min = min(wage_orig);
wage_orig_max = max(wage_orig);

% SET DISTRIBUTION PARAMETERS. BINS MUST BE EQUAL-SIZED.
num_wages = length(wage_orig)-1;
w = (wage_orig(1:num_wages)+wage_orig(2:num_wages+1)) / 2;
pmf = cdf_orig(2:num_wages+1) - cdf_orig(1:num_wages);
bin_width = w(2) - w(1);
wage_min = min(w);
wage_max = max(w);

% NORMALIZE THE PMF SO THAT ITS MASS EQUALS 1.
pmf = pmf./sum(pmf);

% CREATE CDF.
cdf = zeros(num_wages,1);
for k=1:num_wages
    cdf(k) = sum(pmf(1:k));
end

% CREATE pmf_over_bin_width FOR PLANNER'S FOC.
pmf_over_bin_width = pmf/bin_width;

% SET SIZE OF RELEVANT VECTORS.
y = zeros(num_wages,1);
c = zeros(num_wages,1);
l = zeros(num_wages,1);
```

```
tax_paid = zeros(num_wages,1);
IC_check = zeros(num_wages^2,1);
marg_soc_value_above_w = zeros(num_wages,1);
l_iters = zeros(num_wages,1000);
y_iters = zeros(num_wages,1000);
tax_marg_iters = zeros(num_wages,1000);

% SET UTILITY PARAMETERS.
gamma = 1.5;
alpha = 2.55;
sigma = 3;

% SET FMINCON GUESS, BOUNDS, AND LOOP TOLERANCES. tolerance_gov_bc IS IN
% UNITS OF PERCENT OF GNI. tolerance_tax_dev IS IN UNITS OF PERCENT
% DEVIATION OF THE NEW OPTIMAL TAX SCHEDULE AT EACH WAGE FROM THE PREVIOUS
% ITERATION'S OPTIMAL TAX SCHEDULE.
guess_for_lowest_wage = .1;
lb = 0;
ub = inf;
tolerance_gov_bc = .00001;
tolerance_tax_dev = .00001;




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% FIND THE FIXED-POINT TAX SCHEDULE

% STARTING PARAMETER VALUES.
tax_marg = .35*ones(num_wages,1);
tax_marg_iters(:,1) = tax_marg;
transfer = .000001;
loop = 0;
gov_imbalance_percent_gni = -9999999;
max_percent_opt_tax_deviation = -9999999;


% [LOOP]: WHILE THE GOVERNMENT BUDGET CONSTRAINT DEVIATES TOO MUCH FROM
% EQUALITY OR THE TAX SCHEDULE HAS NOT SUFFICIENTLY CONVERGED TO A
% FIXED-POINT, RUN THROUGH THE FOLLOWING LOOP: (1) GIVEN A TAX SCHEDULE AND
% A TRANSFER, FIND EACH WAGE'S OPTIMAL LABOR SUPPLY, (2) USE THE PLANNER'S
% FOC TO FIND AN ALTERNATIVE TAX SCHEDULE GIVEN THE DERIVED LABOR SUPPLY
% AND THE TRANSFER, (3) AVERAGE THE TWO SCHEDULES POINT-WISE TO YIELD A NEW
% TAX SCHEDULE, (4) FIND EACH WAGE'S OPTIMAL LABOR SUPPLY UNDER THE NEW TAX
% SCHEDULE AND SET THE TRANSFER EQUAL TO RESULTING GOVERNMENT REVENUE.

while ( (abs(gov_imbalance_percent_gni)>tolerance_gov_bc) ||
(abs(max_percent_opt_tax_deviation)>tolerance_tax_dev) )
    loop = loop+1

    % FIND EACH PERSON'S OPTIMAL LABOR SUPPLY CONDITIONAL ON THE
    % TRANSFER AND THE MTR SCHEDULE.
    [l, y] = FP_find_opt_l_(num_wages, transfer, tax_marg, w,
guess_for_lowest_wage, lb, ub, options, gamma, alpha, sigma);

    % CALCULATE UTILITY AT EACH WAGE.
    for i=1:num_wages
        c(i) = FP_consump_(l(i), i, w, y, tax_marg, transfer);
    end
    u = (c.^(1-gamma)-1)/(1-gamma) - alpha*l.^sigma/sigma;
```

```
    % CALCULATE COMPONENTS OF PLANNER'S FOC.
    elas_comp = 1 ./ (sigma-1+alpha*gamma*l.^sigma.*c.^(gamma-1));
    elas_uncomp = (1-alpha*gamma*l.^sigma.*c.^(gamma-1)) ./ (sigma-
1+alpha*gamma*l.^sigma.*c.^(gamma-1));
    u_prime_c = c.^(-gamma);
    total_marg_soc_value = sum(pmf./u_prime_c);
    for k=1:num_wages
        marg_soc_value_above_w(k) = total_marg_soc_value -
sum(pmf(1:k)./u_prime_c(1:k));
    end

    % CALCULATE ALTERNATIVE MTR SCHEDULE FROM PLANNER'S FOC.
    tax_marg_foc_rhs = (1+elas_uncomp)./elas_comp .* 1./(w.*pmf_over_bin_width) .*
u_prime_c .* (marg_soc_value_above_w-total_marg_soc_value*(1-cdf));
    tax_marg_old = tax_marg;
    tax_marg_alternative = tax_marg_foc_rhs ./ (1+tax_marg_foc_rhs);

    % ADJUST MTR SCHEDULE HALF-WAY TO THE ALTERNATIVE MTR SCHEDULE, STORE
    % VALUES FOR THIS ITERATION, AND UPDATE VALUE FOR THE WHILE LOOP
    % ARGUMENT.
    tax_marg = (tax_marg + tax_marg_alternative) ./ 2;
    max_percent_opt_tax_deviation = max(100*(tax_marg_alternative-
tax_marg)./tax_marg);
    tax_marg_iters(:,loop+1) = tax_marg;
    l_iters(:,2*loop-1) = l;
    y_iters(:,2*loop-1) = y;


    % CALCULATE EACH WAGE'S TAX LIABILITY AT THE NEW MTR SCHEDULE.
    [l, y] = FP_find_opt_l_(num_wages, transfer, tax_marg, w,
guess_for_lowest_wage, lb, ub, options, gamma, alpha, sigma);
    for i=1:num_wages
        c(i) = FP_consump_(l(i), i, w, y, tax_marg, transfer);
    end
    tax_paid = y + transfer - c;
    l_iters(:,2*loop) = l;
    y_iters(:,2*loop) = y;

    % SAVE VALUES.
    gov_rev_iters(loop,1) = sum(tax_paid.*pmf);
    gov_surplus_iters(loop,1) = sum((tax_paid-transfer).*pmf);
    transfer_iters(loop,1) = transfer;

    % SET THE NEW TRANSFER TO EQUAL GOVERNMENT REVENUE. NO SCALING IS
    % NECESSARY BECAUSE THE TRANSFER IS PER-CAPITA. ALSO UPDATE VALUE FOR
    % THE WHILE LOOP ARGUMENT.
    transfer = gov_rev_iters(loop,1);
    gov_imbalance_percent_gni = 100*gov_surplus_iters(loop,1)/sum(y.*pmf);

end
% [LOOP END].


% CHECK THAT EACH i PREFERS HIS (c,y) BUNDLE TO ALL OTHER m's BUNDLES.
% IC_check(i) = {u_i[choosing i's bundle]-u_i[choosing m's bundle]}
% EVERY ELEMENT IN IC_check SHOULD BE NONNEGATIVE. THIS IS USED BELOW.
for i=1:num_wages
    for m=1:num_wages
        IC_check((i-1)*num_wages+m) = ((c(i)^(1-gamma)-1)/(1-gamma)-
alpha*l(i)^sigma/sigma)...
```

```
                                            - ((c(m)^(1-gamma)-1)/(1-gamma)-
alpha*(y(m)/w(i))^sigma/sigma);
    end
end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DISPLAY RESULTS

% DISPLAY INDICATIONS OF SIMULATION SUCCESS.
disp ' '
disp ' '
disp ' '
disp '<<<<<<<<<<<<<<<<<<<<<<<<  INDICATIONS OF SIMULATION SUCCESS
>>>>>>>>>>>>>>>>>>>>>>>>>'
disp '(1) Is y(w) non-deacreasing?  "1" means yes.'
disp '(2) Max IC violation: [utility from misrepresenting] - [utility from
revealing type]'
disp '(3) Min optimal marginal tax rate'
disp '(4) Max optimal marginal tax rate'
disp '(5) Slackness of the government BC: Gov budget imbalance as a percent of GNI'
disp '(6) Max percent deviation of final tax schedule from "optimal" tax schedule'
disp '(7) Transfer'
disp '(8) Total number loops run'

[1, (min(diff(y))>=0);
2, -min(IC_check);
3, min(tax_marg)
4, max(tax_marg)
5, gov_imbalance_percent_gni;
6, max_percent_opt_tax_deviation;
7, transfer;
8, loop]


% PLOT MTR VS. WAGE.
figure
plot(w,tax_marg)
title('Optimal MTR vs. Wage')
ylabel('MTR')
xlabel('Wage')
saveas(gcf,'MTR_v_Wage.fig')




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% SAVE LATEST OUTPUT AND CLOSE DIARY
save FP_MTR_sim.mat;
diary off;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FIXED-POINT MIRRLEES SIMULATIONS
%
% FUNCTION FOR FINDING OPTIMAL LABOR SUPPLY FOR ALL INDIVIDUALS
% FP_find_opt_l_.m
%
% Mankiw-Weinzierl-Yagan "Optimal Taxation in Theory and Practice"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [l, y] = FP_find_opt_l_(num_wages, transfer, tax_marg, w,
guess_for_lowest_wage, lb, ub, options, gamma, alpha, sigma);


l = zeros(num_wages,1);
y = zeros(num_wages,1);
guess = guess_for_lowest_wage;

for i=1:num_wages
    l(i) = fmincon( 'FP_opt_l_obj_', guess, [],[],[],[], lb, ub, [], options, i, w,
 y, tax_marg, transfer, gamma, alpha, sigma);
    y(i) = w(i)*l(i);
    guess = l(i);
end


end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FIXED-POINT MIRRLEES SIMULATIONS
%
% OBJECTIVE FUNCTION FOR FINDING OPTIMAL LABOR SUPPLY
% FP_opt_l_obj_.m
%
% Mankiw-Weinzierl-Yagan "Optimal Taxation in Theory and Practice"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function neg_utility = FP_opt_l_obj_(l_opt_i, i, w, y, tax_marg, transfer, gamma,
alpha, sigma);

neg_utility = -( ( (FP_consump_(l_opt_i, i, w, y, tax_marg, transfer))^(1-gamma)-1
)/(1-gamma) - alpha/sigma*l_opt_i^(sigma) );

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FIXED-POINT MIRRLEES SIMULATIONS
%
% FUNCTION FOR CALCULATING CONSUMPTION FOR A GIVEN INDIVIDUAL
% FP_consump_.m
%
% Mankiw-Weinzierl-Yagan "Optimal Taxation in Theory and Practice"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [c_i] = FP_consump_(l_i, i, w, y, tax_marg, transfer);


y(i) = w(i)*l_i;

% FIND THE HIGHEST j SUCH THAT y(j) IS LESS THAN y(i). THIS IS AKIN TO
% FINDING THE iTH'S PERSON'S TAX BRACKET j, GIVEN i'S INCOME. IN THIS
% EXERCISE, THERE ARE EXACTLY i TAX BRACKETS SINCE WE CALCULATE EACH
% PERSON'S LABOR SUPPLY AS IF HIS ASSIGNED MARGINAL TAX RATE IS THE
% TAX RATE THAT APPLIES TO ALL INCOMES EARNED ABOVE y(i-1), OR ABOVE 0 IF
% i EQUALS 1. NOTE THAT IF y(1:i-1) IS NOT MONOTONIC, THIS ALGORITHM PUTS i
% INTO THE LOWEST POSSIBLE TAX BRACKET.

j = 1;
while ( (j ~= i)  &&  (y(j) < y(i)) )
    j = j+1;
end

% CALCULATE TAX PAID BY THIS iTH PERSON GIVEN HIS TAX BRACKET.
if (j==1)
    tax_paid_i = tax_marg(1)*y(i);
elseif (j==2)
    tax_paid_i = tax_marg(1)*y(1) + tax_marg(2)*(y(i)-y(1));
else
    tax_paid_i = tax_marg(1)*y(1) + sum(tax_marg(2:j-1).*diff(y(1:j-1))) +
tax_marg(j)*(y(i)-y(j-1));
end

% CALCULATE RESULTING CONSUMPTION.
c_i = transfer + y(i) - tax_paid_i;


end
```