Replication De Nardi, French and Jones (2010)

Pierre-Yves Yanni

February 2019

1 Model

1.1 Description

Period utility is a function of consumption c and health status h:

$$u(c,h) = \delta(h)\frac{c^{1-\nu}}{1-\nu} \tag{1}$$

with $\nu > 0$ and $\delta(h) = 1 + \delta h$.

When the person dies, her utility is

$$\phi(e) = \theta\frac{(e+\kappa)^{1-\nu}}{1-\nu}. \tag{2}$$

where e is the estate net of taxes, κ is the curvature of the bequest function and θ is the intensity of the bequest motive.

Non-asset income is a function of sex g, permanent income I, and age t:

$$y_t = y(g, I, t) \tag{3}$$

Transition probabilities for health status obey

$$\pi_{j,k,g,I,t} = \Pr(h_{t+1} = k | h_t = j, g, I, t),\ j,k \in \{1, 0\}. \tag{4}$$

$s_{g,h,I,t}$ is the probability of survival.

Medical expenses are given by

$$\ln m_t = m(g, h, I, t) + \sigma(g, h, I, t) \cdot \psi_t \tag{5}$$

where

$$\psi_t = \zeta_t + \xi_t,\ \xi_t \sim N(0, \sigma_\xi^2) \tag{6}$$

$$\zeta_t = \rho_m \zeta_{t-1} + \epsilon_t,\ \epsilon_t \sim N(0, \sigma_\epsilon^2) \tag{7}$$

where ξ_t and ϵ_t are serially and mutually independent.

The timing is as follows:

1. health status and medical expenses are realized
2. the individual consumes and saves
3. survival shock hits; individuals who die leave any remaining assets to their heirs

Next period's assets are given by

$$a_{t+1} = a_t + y_n(ra_t + y_t, \tau) + b_t - m_t - c_t \tag{8}$$

where $y_n(ra_t + y_t, \tau)$ is posttax income, r denotes the risk-free, pretax rate of return, the vector τ describes the tax structure and b_t denotes government transfers. Government transfers provide a consumption floor:

$$b_t = \max\{0, \underline{c} + m_t - [a_t + y_n(ra_t + y_t, \tau)]\}. \tag{9}$$

If transfers are positive, $c_t = \underline{c}$ and $a_{t+1} = 0$. The model is defined in terms of cash on hand x_t:

$$x_t = a_t + y_n(ra_t + y_t, \tau) + b_t - m_t \tag{10}$$

Assets and cash on hand follow $a_{t+1} = x_t - c_t$ and

$$x_{t+1} = x_t - c_t + y_n(r(x_t - c_t) + y_{t+1}, \tau) + b_{t+1} - m_{t+1} \tag{11}$$

To enforce the consumption floor and that assets are non negative for all t, we require $x_t \geq \bar{c}$ and $c_t \leq x_t$. The value function is

$$\begin{aligned} V_t(x_t, g, h_t, I, \zeta_t) = \max_{c_t, x_{t+1}} \{&u(c_t, h_t) + \beta s_{g,h,I,t} E_t V_{t+1}(x_{t+1}, g, h_{t+1}, I, \zeta_{t+1}) \\ &+\beta(1 - s_{g,h,I,t})\phi(e_t)\} \end{aligned} \tag{12}$$

subject to:

$$e_t = (x_t - c_t) - \max\{0, \tilde{\tau} \cdot (x_t - c_t - \tilde{x})\} \tag{13}$$

1.2 Benchmark Model

There is no bequest motive and utility is not a function of health. (Results are not better with these features).

$$V_t(x_t, g, h_t, I, \zeta_t) = \max_{c_t, x_{t+1}} \{\frac{c_t^{1-\nu}}{1-\nu} + \beta s_{g,h,I,t} E_t V_{t+1}(x_{t+1}, g, h_{t+1}, I, \zeta_{t+1})\} \tag{14}$$

subject to:

$$x_{t+1} = x_t - c_t + y_n(r(x_t - c_t) + y(g, I, t+1), \tau) + b_{t+1} - m_{t+1} \tag{15}$$

$$\ln m_t = m(g,h,I,t) + \sigma(g,h,I,t) \cdot \psi_t \tag{16}$$

$$\pi_{j,k,g,I,t} = \Pr(h_{t+1} = k | h_t = j, g, I, t),\ j,k \in \{1,0\}. \tag{17}$$

where

$$b_{t+1} = \max\{0, \underline{c} + m_{t+1} - [x_t - c_t + y_n(r(x_t - c_t) + y(g,I,t+1), \tau)]\} \tag{18}$$

and

$$\psi_t = \zeta_t + \xi_t,\ \xi_t \sim N(0, \sigma_\xi^2) \tag{19}$$
$$\zeta_t = \rho_m \zeta_{t-1} + \epsilon_t,\ \epsilon_t \sim N(0, \sigma_\epsilon^2) \tag{20}$$

1.3 Parameters

Benchmark calibration:

$\delta = 0$	$\beta = 0.97$	$\nu = 3.81$	$\theta = 0$	$\kappa = NA$
$\rho_m = .922$	$\sigma_\xi^2 = 0.05$	$\sigma_\epsilon^2 = 0.665$	$\underline{c} = 2663$	$r = 0.02$

From the author's code (in C), tax brackets are

$$\{6250, 40200, 68400, 93950, 148250, 284700\}$$

and corresponding marginal tax rates are

$$\{0.0765, 0.2616, 0.4119, 0.3499, 0.3834, 0.4360, 0.4761\}$$

The parameters for the survival probabilities, health transition probabilities, income and medical expenses depend on gender, income percentiles, health status, time and shocks (for medical expenses). They are recovered from regression coefficients (provided by the authors) and by plugging in the other variables.

2 Solving the Model

2.1 Algorithm

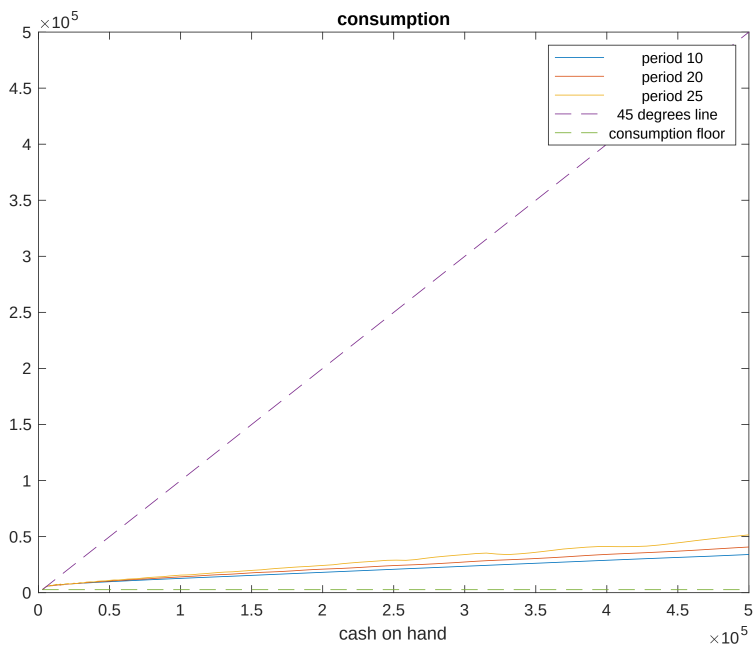
Solving the model consists in finding the decision rule $c_t(x_t, h_t, \zeta_t, t)$ for a given individual of gender g and permanent income percentile I. To obtain it,

1. create grid on cash on hand (tighter for small values: $\sqrt{x_i}$ are equally spaced) starting at $\underline{c}$
2. discretize persistent and transitory shocks ζ_t and ξ_t on medical expenses with associated transition matrices, as well as health shocks

3. solve the last period (trivial) problem: all the cash on hand is consumed
4. starting at $t = T - 1$, solve for consumption decision at t for all values of x, h, ζ using period $t + 1$ value function
5. iterate on 4. for $t = T - 2$, etc to $t = 1$.

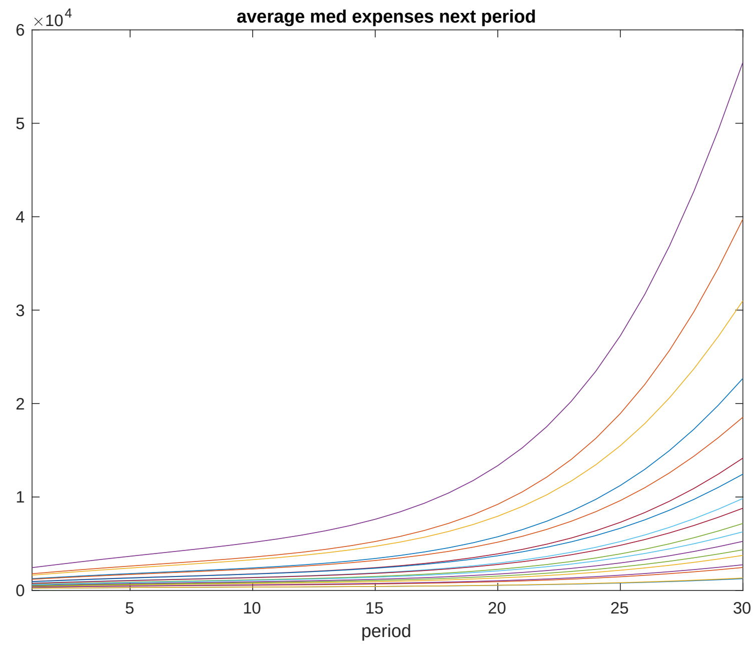
3 Results

Solving the model with a 100 pts grid for cash on hand, 9 pts for the persistent shock and 8 pts for the transitory shock takes about 44 sec. For a male with income at the 50th percentile, we get the following decision function when he is in good health and the persistent shock is at its mean value, 0:[1]



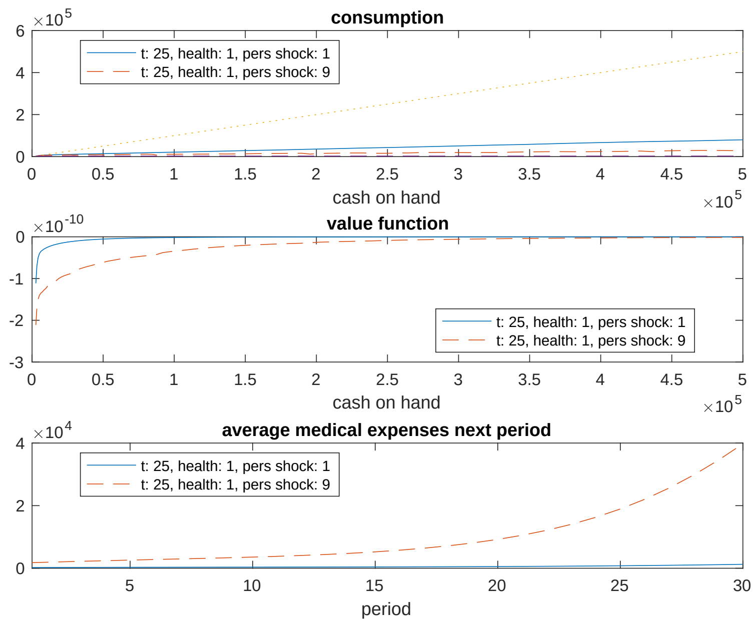
As shown in the following figure, end-of-life medical expenses can potentially be very large:

[1] The little spikes in the decision function at period 25 (and more pronounced spikes in later periods) come from the interaction between the medical expenses shock and the consumption floor. (If we use only 2 pts to discretize the persistent and transitory shocks on medical expenses, we have fewer spikes and the spikes are smoothed-out when we use 50 pts on both shocks; also, if we reduce medical expenses by 50%, the spikes are less pronounced).



However, this is only the expected medical expenses given a health status and a value of the persistent shock today. In the worst case, if the individual is sick and gets the worst realization of the persistent and transitory shocks to medical expenses, the latter will be around 450,000, in which case the consumption floor is binding even for a very wealthy individual. This is a low probability event: the probability that medical expenses exceed 100,000 in the last period is 4/144.

The persistent shock on medical expenses has a large effect on consumption in later years. The next figures compare two healthy individuals at period 25 who draw the lowest and highest realization of the persistent shock to medical expenses:



The effect of the persistent shock on consumption is very strong because of the difference in expected future medical expenses.

Agents at the lowest end of the permanent income distribution will draw down their wealth and take advantage of the consumption floor.

4 Simulations

4.1 Description

Using the file dataprep2.dta provided by the authors, I construct a .csv file that contains the initial distributions needed for the simulations. For every agent, I have the gender, the income percentile, the initial period number and the level of assets $\{g, I, t_0, h_{t_0}, a_{t_0}\}$, as well as the cohort. I use the average income percentile over the years that the agent is present in the data to obtain I and the age to deduce t_0. The persistent shock ζ_{t_0} is inferred from medical expenditures (assuming the transitory shock is null):

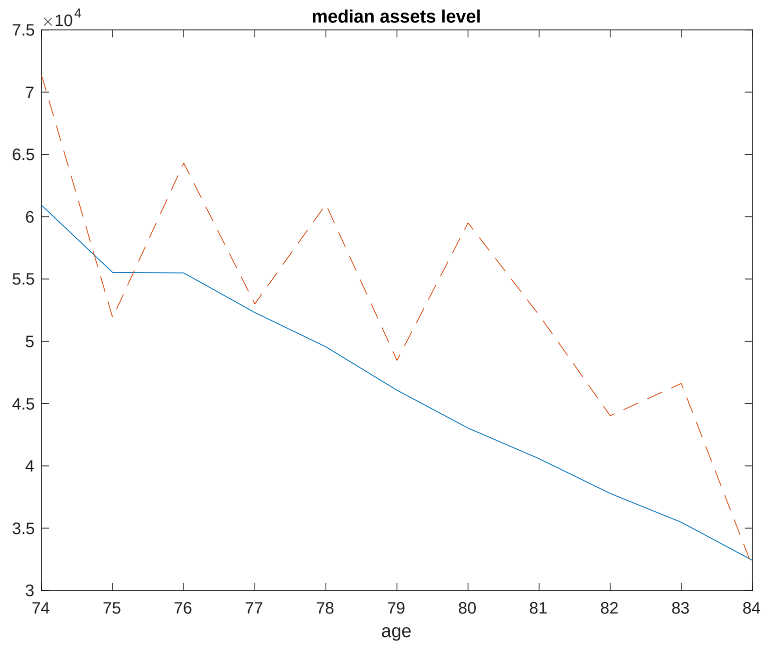
$$\hat{\zeta}_t = \frac{\ln m_t - m(g, h, I, t)}{\sigma(g, h, I, t)}. \tag{21}$$

Then, I compute x_{t_0} using data on assets, medical expenditures (as a function of ζ_{t_0}, g, I, t) and income (as a function of g, I, t). Simulations are run as follows:

1. solve the model for a specific g, I to obtain $c(x_t, h_t, \zeta_t, t)$,
2. for each individual with this specific g, I, infer ζ_{t_0} and draw $\{h_t\}_{t_0}^T$ and $\{\psi_t\}_{t_0}^T$ to compute a path for medical expenses $\{\hat{m_t}\}_{t_0}^T$,
3. using $\{\hat{m_t}\}_{t_0}^T$, $\{inc_t\}_{t_0}^T$ and $c(x_t, h_t, \zeta_t, t)$, compute the path for consumption $\{\hat{c_t}\}_{t_0}^T$ and assets $\{\hat{x_t}\}_{t_0}^T$ $(a_{t+1} = x_t - c_t)$,
4. using the function $s_{g,h,I,t}$, find the date of death t_d to keep only the part of $\{\hat{c_t}\}_{t_0}^T$ and $\{\hat{x_t}\}_{t_0}^T$ during which the agent is alive (unless computing paths conditional on survival),
5. compute the median path for $\{\hat{c_t}\}_{t_0}^T$ and $\{\hat{x_t}\}_{t_0}^T$ by taking the median for each date t.

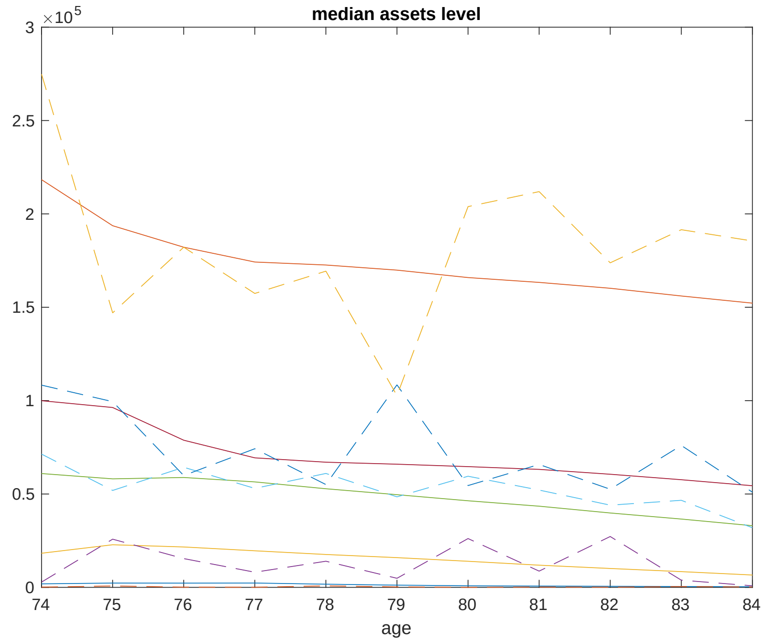
4.2 Results

For the first cohort (aged 72-76 in 1996) and the third quintile (only female), simulations show the following pattern (dotted line represents the data):



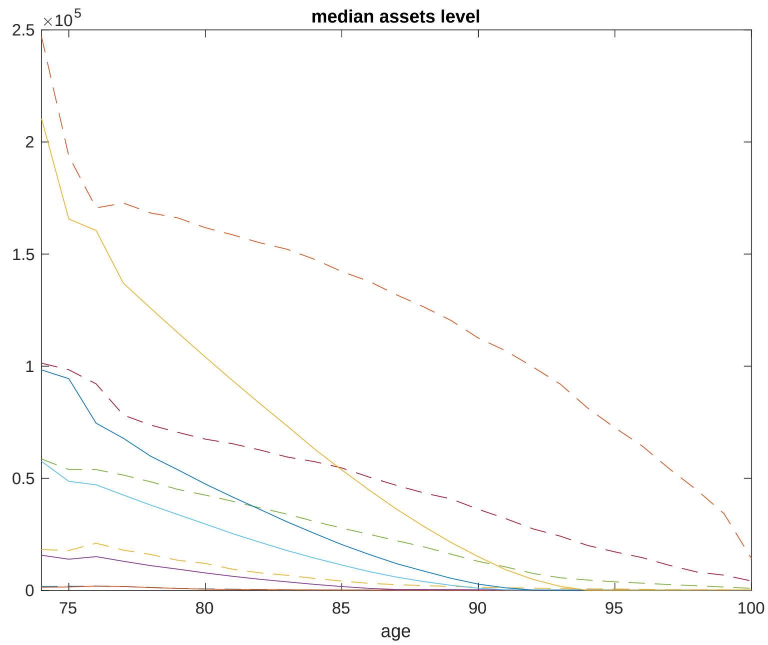
median assets level
×10^4
7.5
7
6.5
6
5.5
5
4.5
4
3.5
3
74
75
76
77
78
79
80
81
82
83
84
age

For all quintiles, median assets positions are as follows (dotted line for data):

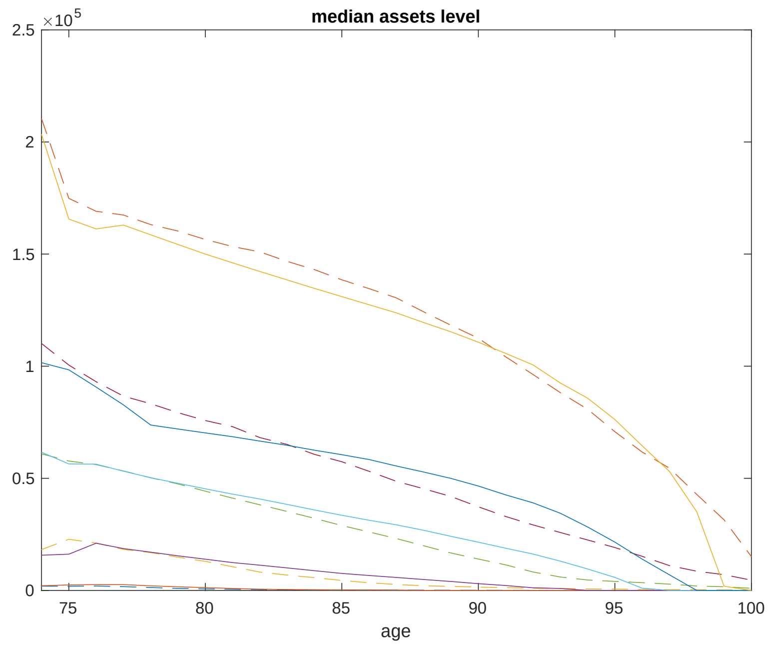


4.3 The Role of Medical Expenses

To understand the role of medical expenses, I compare asset decumulation with (dotted line) and without medical expenses (solid line):



In fact, it is the level of medical expenses and not their volatility that matters, as shown in the following graph, comparing asset decumulation in the benchmark model (dotted line) vs without medical expenses risk (solid line):



(The last two figures correspond to Figures 9 and 10 in the original article).

5 Comparing Speed: Matlab vs Python

I compare the time it takes to solve the model, i.e., to compute the decision function (consumption as a function of cash-on-hand), for different grid sizes for cash-on-hand and shocks to medical expenses. The codes are quite similar, the main difference being that some functions in the python code are compiled just-in-time (thanks to numba) and the matlab code runs in parallel (with 10 "workers"). The test is run on a Mac with a 3 GHz Intel Xeon W processor. Running times are rough approximations and only give an idea of the difference in computing time.

Speed Comparison			
Grids: N_x, N_ϵ, N_ζ	Matlab	Python	Ratio M/P
$(100, 9, 8)$	25 sec	6 sec	4.2
$(50, 3, 2)$	8 sec	2.3 sec	3.5
$(500, 9, 8)$	100 sec	30 sec	3.3
$(1000, 18, 16)$	500 sec	330 sec	1.5

6 Python Code

This section contains a jupyter notebook and the code of the module DFJ. The latter solves and simulates the model, and is called by the former.

DFJ

March 13, 2019

1 Replication DFJ 2010 (solving model and simulations)

rem: non-jitted version of code takes over 900 sec to run (compared to less than 5 for jitted version)

1.1 Decision function

Importing modules and solving for decision function for a male at the 50th percentile. Figure for a healthy individual at the mean value of the persistent shock

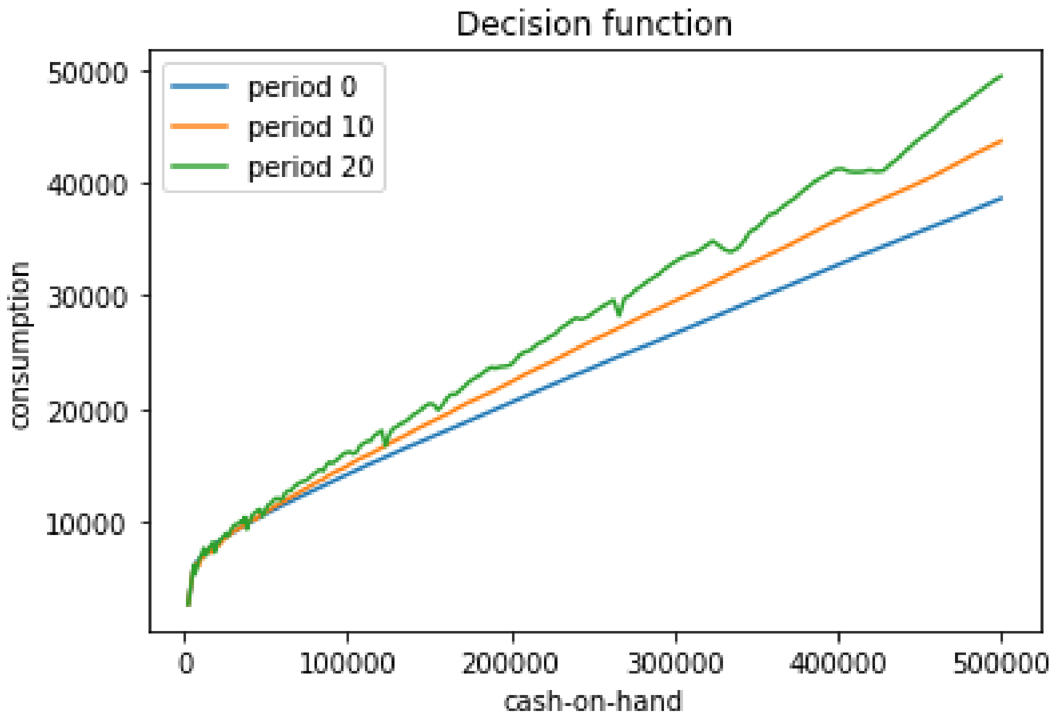
```
In [1]: %matplotlib inline

        import time
        import numpy as np
        import pandas as pd
        import importlib
        import DFJ
        importlib.reload(DFJ)
        import matplotlib.pyplot as plt

        pd.options.display.max_columns = None

        cp = DFJ.common_params(9, 8, 200, 500_000)
        ip = DFJ.indiv_params(1, 0.5)
        m_c, m_V = DFJ.solve_model(cp, ip)

        # figure: consumption
        ax = plt.subplot(1, 1, 1)
        for per in [0, 10, 20]:
            ax.plot(cp.grid_x, m_c[per, :, 5], label = f'period {str(per)}')
        ax.legend()
        ax.set_title('Decision function')
        ax.set_xlabel('cash-on-hand')
        ax.set_ylabel('consumption')
        plt.show()
```



2 Simulations

The median value of assets from the data and the model are compared.

2.1 Data preparation

```
In [2]: df = pd.read_stata('raw_data/dataprep2.dta')
        cohort_list = ['cohort' + str(x) for x in range(1,8)]
        time_list = ['time' + str(x) for x in range(1,8)]
        data = df.loc[df['realyear'] > 94, # data for 1994 underreported
                   ['HHID', 'age','male', 'PI', 'heal', 'assets', 'hhinc',
                    'medcost', 'realyear'] + time_list+ cohort_list].copy()

        # convert time1-time7 and cohort1-cohort6 to time and cohort vars
        def get_dum(row, col_names):
            for c in col_names:
                if row[c] == 1:
                    return int(c[-1])
        data['time'] = data.loc[:, time_list].apply(get_dum,
                                                    args = (time_list,),
                                                    axis=1)
        data['cohort'] = data.loc[:, cohort_list].apply(get_dum,
                                                        args = (cohort_list,),
```

```
                                                  axis=1)
data.drop(time_list+cohort_list, axis=1, inplace=True)

# compute average income percentile for each HHID
data['PI'] = data.groupby('HHID')['PI'].transform('mean')
# rename some cols
data.rename(inplace=True,
           columns={'male': 'g', 'heal': 'h', 'assets': 'a',
                    'hhinc': 'inc', 'PI': 'I', 'medcost': 'med'})
# income quintiles
bins = np.linspace(0, 1, num=6, endpoint=True)
names = [x for x in range(1,6)]
data['quintile'] = pd.cut(data['I'], bins, labels=names)
# group last two cohorts and start at 1
data['cohort'] = data['cohort'].replace({7:6}) - 1

# initial asset position
data_init = data.loc[data.realyear == 96,].dropna(subset=['a', 'med'])
data_init = data_init[[ 'g', 'I', 'h', 'a', 'inc', 'med',
                       'age', 'quintile', 'time', 'cohort']]
data_init = data_init[data_init['age']<100]
```

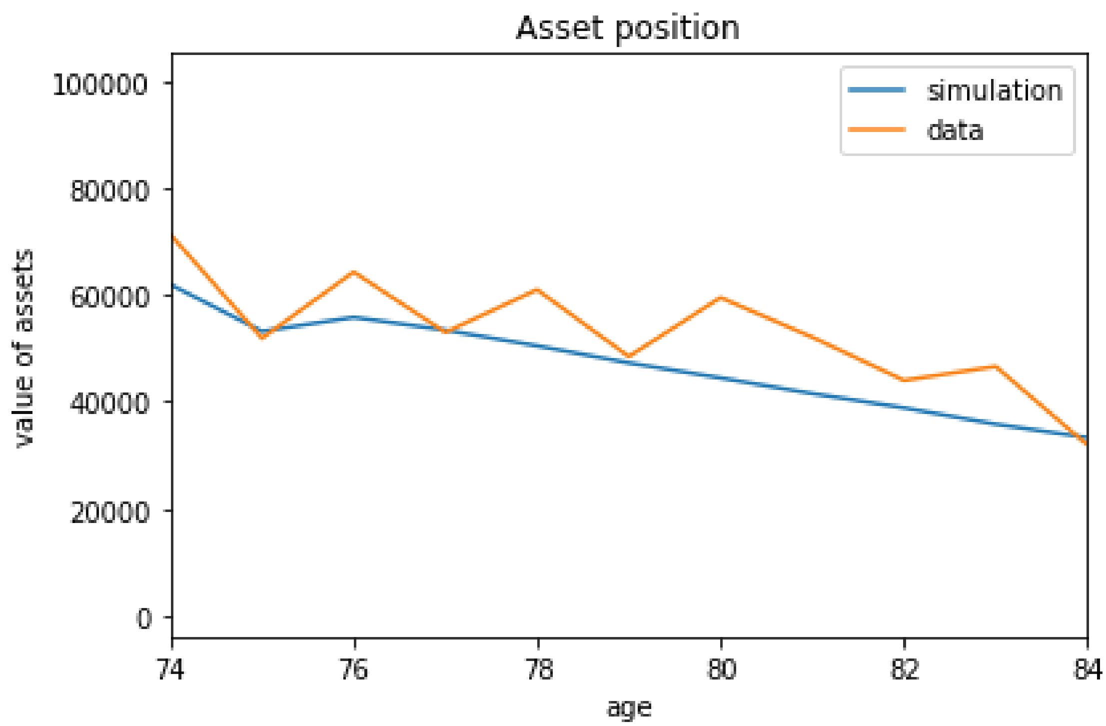
2.2 Simulation: females, 1st cohort, 3rd quintile

In [3]:
```
# ignore warning that all entries of col are nan
import warnings; warnings.simplefilter('ignore')
# data vs simulation: female, 3rd quintile, 1st cohort
cp = DFJ.common_params(9, 9, 500, 10_500_000)
g, quintile, cohort, N = 0, 3, 1, 2000
m_a, m_s = DFJ.simul(g, quintile, cohort, data_init, N, cp)

## simulation
med_simul = np.nanmedian(m_a, axis = 0)
index_simul = [70+t for t in range(len(med_simul))]
## data
mask = ((data.g == g) & (data.quintile == quintile)
               & (data.cohort == cohort))
med_data = data[mask].groupby('age').agg({'a': 'median'})

## figure
ax = plt.subplot(1, 1, 1)
ax.plot(index_simul, med_simul, label = 'simulation')
ax.plot(med_data.index, med_data, label = 'data')
ax.legend()
ax.set_title('Asset position')
ax.set_xlabel('age')
ax.set_ylabel('value of assets')
ax.set_xlim([74, 84])
```

```
plt.tight_layout()
plt.show()
```

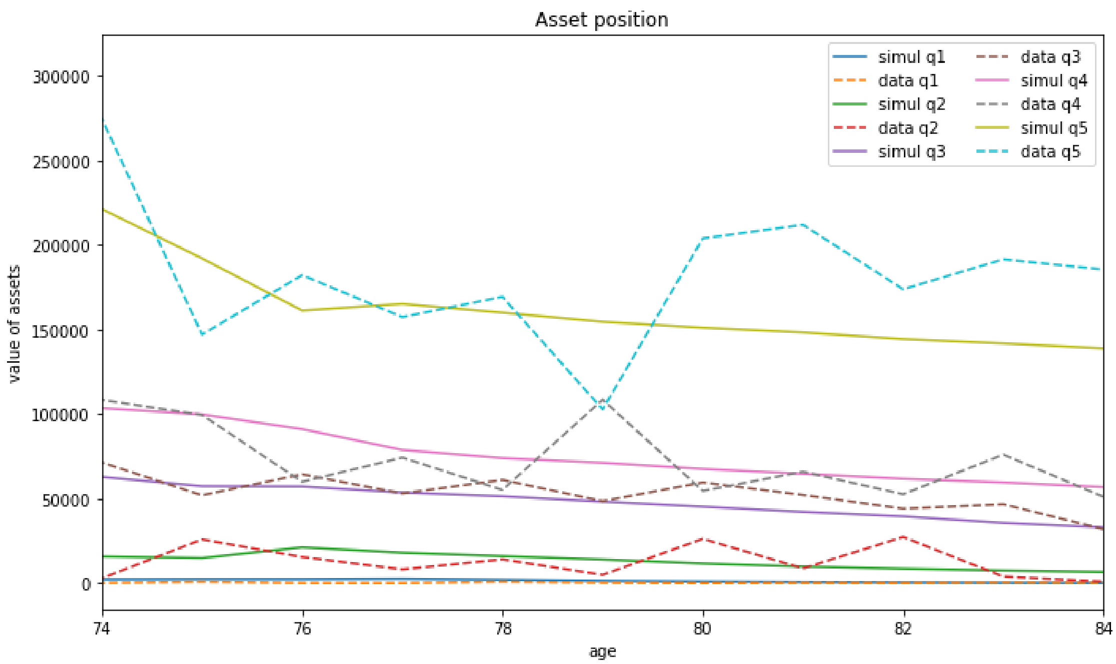


2.3 Simulation: females, 1st cohort

```
In [4]: # data vs simulation: female, all quintiles, 1st cohort
        cp = DFJ.common_params(9, 9, 500, 10_500_000)
        g, cohort, N = 0, 1, 2000

        plt.figure(figsize=(10,6))
        ax = plt.subplot(1, 1, 1)
        for quintile in range(1, 6):
            m_a, m_s = DFJ.simul(g, quintile, cohort, data_init, N, cp)
            # figure: asset position
            ## simulation
            med_simul = np.nanmedian(m_a, axis = 0)
            index_simul = [70+t for t in range(len(med_simul))]
            ## data
            mask = ((data.g == g) & (data.quintile == quintile)
                        & (data.cohort == cohort))
            med_data = data[mask].groupby('age').agg({'a': 'median'})
            ## plotting
            ax.plot(index_simul, med_simul,
                    label = f'simul q{quintile}')
```

```
    ax.plot(med_data.index, med_data, '--',
            label = f'data q{quintile}')
# finish plot
ax.legend(ncol=2)
ax.set_title('Asset position')
ax.set_xlabel('age')
ax.set_ylabel('value of assets')
ax.set_xlim([74, 84])
plt.tight_layout()
plt.show()
```



DFJ.py

```
import numpy as np
import pandas as pd
import quantecon as qe
from scipy import linalg
from quantecon.optimize.scalar_maximization import brent_max
from numba import njit

# fast code to solve DFJ


class common_params:
    """This class loads and computes parameters common to all agents"""

    def __init__(self, N_z, N_eps, N_x, upper_x,
                 c_ubar=2663, T=31, N_h=2, r=0.02,
                 nu=3.81, beta=0.97,
                 rho=0.922, sig_z=np.sqrt(0.05), sig_eps=np.sqrt(0.665)):

        self.N_x, self.N_z, self.N_eps, self.N_h = N_x, N_z, N_eps, N_h
        self.upper_x, self.c_ubar = upper_x, c_ubar
        self.T, self.r, self.beta = T, r, beta

        # utility function
        @njit
        def u(x):
            return x**(1-nu)/(1-nu)
        self.u = u

        # markov chain for medical expenses shocks
        z = qe.rouwenhorst(N_z, 0, sig_z, rho)
        eps = qe.rouwenhorst(N_eps, 0, sig_eps, 0)
        self.z, self.eps = z, eps
        self.grid_med = (np.kron(z.state_values, np.ones(N_eps))
                         + np.kron(np.ones(N_z), eps.state_values))
        self.Pi_med = np.kron(z.P, eps.P[1, :])

        # grid on cash on hand (more points for lower values)
        self.grid_x = np.linspace(np.sqrt(c_ubar), np.sqrt(upper_x), N_x)**2
```

```
        # tax function
        brackets = np.array([0, 6250, 40200, 68400, 93950,
                             148250, 284700, 2e7])
        tau = np.array([0.0765, 0.2616, 0.4119, 0.3499,
                        0.3834, 0.4360, 0.4761])
        tax = np.zeros(8)
        for i in range(7):
            tax[i+1] = tax[i] + (brackets[i+1]-brackets[i]) * tau[i]
        self.tax = tax
        self.brackets = brackets


class indiv_params:
    """This class loads and computes parameters for
    agents of gender g and income percentile q"""

    def __init__(self, g, inc_perc):
        # agent's characteristics for good (0) and bad health (1)
        m = np.array([[1, 0, g, inc_perc, inc_perc**2],
                      [1, 1, g, inc_perc, inc_perc**2]])
        # survival prob (sqrt() b/c probs for 2 years) by age and health status
        cols = ['constant', 'health', 'male', 'inc_perc', 'inc_perc_sq']
        df = pd.read_csv('raw_data/deathprof.out', delimiter=r"\s+",
                         header=None, index_col=0, names=cols)
        self.pr_s = np.sqrt(np.exp(df.loc[72:, :]@m.T)
                            / (1+np.exp(df.loc[72:, :]@m.T))).values

        # (2 years) prob of bad health by age and health status
        df = pd.read_csv('raw_data/healthprof.out', delimiter=r"\s+",
                         header=None, index_col=0, names=cols)
        _ = np.exp(df.loc[72:, :]@m.T).values
        self.pr_h = _ / (1 + _)

        # income by age and health status
        df = pd.read_csv('raw_data/incprof.out', delimiter=r"\s+", header=None,
                         index_col=0, names=cols)
        # same inc for good and bad health
        self.inc = np.exp(df.loc[:100, :]@m[0, :].T).values
```

```
        # mean and variance of medical expenses by age and health status
        cols_var = [x + '_var' for x in cols]
        df = pd.read_csv('raw_data/medexprof_adj.out', delimiter=r"\s+",
                         header=None, index_col=0, names=cols+cols_var)
        mean_med = (df.loc[:100, cols]@m.T).values
        var_med = (df.loc[:100, cols_var]@m.T).values
        self.med = [mean_med, var_med]


@njit
def interp(v_x, v_y, x0):
    """interpolation and extrapolation (using last 2 values)"""
    i = np.fmin(np.searchsorted(v_x, x0, 'left'), len(v_x)-1)
    return v_y[i-1] + (x0 - v_x[i-1])/(v_x[i] - v_x[i-1]) * (v_y[i] - v_y[i-1])


@njit
def interp_mat(v_x, mat_y, v_x0):
    """interpolation and extrapolation; nth value of v_x0 corresponds to
    nth col of mat_y"""
    v_y0 = np.empty_like(v_x0)
    for col in range(len(v_y0)):
        i = np.fmin(np.searchsorted(v_x, v_x0[col], 'left'), len(v_x)-1)
        v_y0[col] = (mat_y[i-1, col]
                     + (v_x0[col] - v_x[i-1])/(v_x[i] - v_x[i-1])
                     * (mat_y[i, col] - mat_y[i-1, col]))
    return v_y0


def solve_model(cp, i_par):
    """iterates backward on value function"""
    # load parameters
    T, r, u, beta = cp.T, cp.r, cp.u, cp.beta
    N_x, N_h, N_z, N_eps = cp.N_x, cp.N_h, cp.N_z, cp.N_eps
    brackets, tax, c_ubar = cp.brackets, cp.tax, cp.c_ubar
    grid_x, Pi_med, grid_med = cp.grid_x, cp.Pi_med, cp.grid_med
    inc, pr_h, med, pr_s = i_par.inc, i_par.pr_h, i_par.med, i_par.pr_s
    N_med = Pi_med.shape[1]

    # prepare matrices for results
    m_c = np.zeros((T, N_x, N_h*N_z))
    m_V = np.empty((T, N_x, N_h*N_z))
    # last period
    m_c[T-1, :, :] = np.column_stack([grid_x] * N_h*N_z)
    m_V[T-1, :, :] = np.column_stack([u(grid_x)] * N_h*N_z)
```

```
@njit
def objective(c, x, t, ind, med_ex, pr_tr, s_h, v_next):
    """objective function in period t"""
    tot_inc_f = r * (x-c) + inc[t+1]
    net_y = tot_inc_f - interp(brackets, tax, tot_inc_f)
    coh = np.fmax(x - c + net_y - med_ex, c_ubar*np.ones_like(med_ex))
    EV = np.dot(pr_tr[ind, :], interp_mat(grid_x, v_next, coh))
    return u(c) + beta * s_h * EV

# other periods
for t in reversed(range(T-1)):
    # expand val fn next period by # of transitory shocks
    v_next = np.repeat(m_V[t+1, :, :], N_eps, axis=1)
    # pr_h given for 2 periods:
    Pi_h = linalg.sqrtm(np.array([[1-pr_h[t+1, 0], pr_h[t+1, 0]],
                                  [1-pr_h[t+1, 1], pr_h[t+1, 1]]]))
    pr_tr = np.kron(Pi_h, Pi_med)
    med_ex = np.exp(np.kron(med[0][t+1, :], np.ones(N_med))
                    + np.kron(np.sqrt(med[1][t+1, :]), grid_med))
    v_cons = np.zeros((N_x, N_h*N_z))
    v_new = np.empty((N_x, N_h*N_z))

    for n_x in range(N_x):
        xi = grid_x[n_x]
        for n_h in range(N_h):
            for n_z in range(N_z):
                ind = n_h * N_z + n_z
                if n_x == 0:
                    val = objective(c_ubar, xi, t, ind,
                                    med_ex, pr_tr, pr_s[t+1, n_h], v_next)
                    cons = c_ubar
                else:
                    cons, val, _ = brent_max(
                        objective, c_ubar, xi,
                        args=(xi, t, ind, med_ex, pr_tr,
                              pr_s[t+1, n_h], v_next),
                        xtol=1)
                v_cons[n_x, ind] = cons
                v_new[n_x, ind] = val

    m_c[t, :, :], m_V[t, :, :] = v_cons, v_new

return m_c, m_V
```

```
def simul(g, quintile, cohort, data, N, cp):
    """simulates the model using initial asset position, med_expenses and
    income for N randomly drawn people from given g, quintile and cohort;
    returns survival and asset position for N people"""

    inc_perc = 0.2 * quintile - 0.1  # income percentile
    ip = indiv_params(g, inc_perc)
    m_c, _ = solve_model(cp, ip)
    # load parameters
    z, grid_z, eps = cp.z, cp.z.state_values, cp.eps  # pers and trans shocks
    T, r, brackets, tax = cp.T, cp.r, cp.brackets, cp.tax
    c_ubar, N_z, grid_x = cp.c_ubar, cp.N_z, cp.grid_x
    m_med, v_med, income = ip.med[0], ip.med[1], ip.inc
    pr_h, pr_s = ip.pr_h, ip.pr_s
    # create initial matrices
    m_a = np.empty((N, T)) * np.nan  # asset position
    m_s = np.zeros((N, T))  # present in data (1/0)

    def rand_indiv(data):
        """initial conditions and med shock for given individual"""
        mask = ((data.g == g) & (data.quintile == quintile)
                & (data.cohort == cohort))
        id = np.random.choice(data[mask].index)
        age0, a0, h0, inc0, med0 = data.loc[id, ['age', 'a', 'h',
                                                  'inc', 'med']]
        t0, h0 = int(age0 - 70), int(h0)  # initial period; int for indices
        # estimate initial state of persistent shock
        m_med0, v_med0 = m_med[t0, h0], v_med[t0, h0]  # mean, var med ex
        zeta_est = (np.log(med0) - m_med0) / np.sqrt(v_med0)
        n_z0 = np.argmin(np.abs(zeta_est - grid_z))
        zeta0 = grid_z[n_z0]  # initial persistent shock in grid
        # simulate index for zeta and medical shock
        t_len = T - t0  # simulation length
        zeta_index = z.simulate_indices(t_len, init=n_z0, random_state=12)
        med_shock = (z.simulate(t_len, init=zeta0, random_state=12)
                     + eps.simulate(t_len, init=0))
        return t0, h0, a0, inc0, med0, zeta_index, med_shock

    @njit
    def coh(a, inc, med):
        """computes cash-on-hand"""
        tot_inc_f = r * a + inc
        net_y = tot_inc_f - interp(brackets, tax, tot_inc_f)
        coh = np.fmax(a + net_y - med, c_ubar)
        return coh
```

```
for n in range(N):
    # initial values for loop
    t0, h, a, inc, med, zeta_index, med_shock = rand_indiv(data)
    s, n_z = 1, zeta_index[0]  # initial survival state and persist shock
    m_a[n, t0], m_s[n, t0] = a, 1
    for t in range(t0, T-1):
        x = coh(a, inc, med)
        ind = h * N_z + n_z
        c = interp(grid_x, m_c[t, :, ind], x)  # consumption
        Pi_h = linalg.sqrtm(np.array([[1 - pr_h[t+1, 0], pr_h[t+1, 0]],
                                      [1 - pr_h[t+1, 1], pr_h[t+1, 1]]]))
        # next period's variables
        h = int(np.random.rand() < Pi_h[h, 1])  # new draw for h
        s = np.where(s == 1, int(np.random.rand() < pr_s[t, h]), 0)
        a = x - c  # next period's asset position
        inc = income[t+1]
        med = np.exp(m_med[t+1, h]
                     + np.sqrt(v_med[t+1, h]) * med_shock[t+1-t0])
        n_z = zeta_index[t+1-t0]
        # save results
        m_a[n, t+1] = a
        m_s[n, t+1] = s

return m_a, m_s
```

7 Matlab Code

7.1 DFJ.m: Solving the Benchmark Model

```
%%%%%%%%%%%%%%%%%%%%%%%
% DFJ BENCHMARK MODEL %
%%%%%%%%%%%%%%%%%%%%%%%

clc;
clear all;
path_data = '../data/';
path_graphs = '../graphs/';

% PARAMETERS

% agent specific
g            = 0;    % gender: 1 is male
I            = 0.5;  % income percentile (in paper, quintiles: .1, .3, ...)
m_agent      = [1 0 g I I^2;     % agent specific characteristics
                1 1 g I I^2];    % for good and bad health (second row)
% for all agents
nu           = 3.81; % curvature on period utility function
beta         = 0.97; % discount factor
c_ubar       = 2663; % consumption floor
age_min      = 70;   % starting age
age_max      = 100;  % max age
T            = age_max-age_min+1; % number of periods
r            = 0.02;     % interest rate
rho          = 0.922;    % rho medical shock; zeta(t) = rho*zeta(t-1)+eps(t)
sig_z        = sqrt(0.05); % sd persistent med shock; eps ~ N(0,sig_zeta^2)
sig_eps      = sqrt(0.665); % sd transitory shock medical expenses
N_x          = 100;  % number of points on grid cash on hand (coh)
N_h          = 2;    % number of health states
N_z          = 9;   % number grid points medical expenses permanent shock
N_eps        = 8;   % number grid points medical expenses transitory shock
tol          = 1;    % tol on golden section search algorithm

% tax schedule (brackets and marginal rates tau
brackets     = [0, 6250, 40200, 68400, 93950, 148250, 284700, 1e10];
tau          = [0.0765, 0.2616, 0.4119, 0.3499, 0.3834, 0.4360, 0.4761];
tax          = zeros(8, 1);
for i = 1:7
    tax(i+1)    =  tax(i) + (brackets(i+1)-brackets(i)) * tau(i);
end
```

```
% upload coefficient matrices for survival, health shock, income and
% medical expenses and convert them into vector indexed by age 70 to 100

fileID      = fopen(strcat(path_data,'deathprof.out'),'r');
s_coef      = fscanf(fileID, '%f', [6 33]);        % survival logit coefs
Xb_s        = (m_agent * s_coef(2:6, 3:32))';      % age 72 to 102 for probs
s           = sqrt(exp(Xb_s) ./ (1+exp(Xb_s)));    % survival probabilities
                                                    % sqrt() b/c 2 years prob
fileID      = fopen(strcat(path_data,'healthprof.out'),'r');
h_coef      = fscanf(fileID, '%f', [6 33]);        % health logit coefs
Xb_h        = (m_agent * h_coef(2:6, 3:32))';      % age 72 to 102 for probs
p_h         = exp(Xb_h) ./ (1+exp(Xb_h));          % health transition probs

fileID      = fopen(strcat(path_data,'incprof.out'),'r');
inc_coef    = fscanf(fileID, '%f', [6 33]);        % income coefs
Xb_inc      = (m_agent * inc_coef(2:6, 1:31))';    % using age 70 to 100
inc         = exp(Xb_inc(:,1));                    % income indep of h

fileID      = fopen(strcat(path_data,'medexprof_adj.out'),'r');
med_coef    = fscanf(fileID, '%f', [11 33]);       % medical expenses coefs
Xb_med      = (m_agent * med_coef(2:6, 1:31))';    % average (age 70-100)
Xb_var_med  = (m_agent * med_coef(7:11, 1:31))';   % volatility (age 70-100)


% SOLVING MODEL
% grid on cash in hand x
lower_x     = c_ubar;
upper_x     = 500000;
v_x         = linspace(sqrt(lower_x), sqrt(upper_x), N_x)'.^2;
                % tighter grid for smaller values
d           = (sqrt(upper_x) - sqrt(lower_x)) / (N_x-1);
                % distance between gridpoints

% approximate shocks on medical expenses
[Pi_z, eps, v_z] = tauchen(N_z, 0, rho, sig_z);
                    % Pi_z: transition matrix: v_z: vector of shocks
[Pi_eps, eps, v_eps] = tauchen(N_eps, 0, 0, sig_eps);
                        % Pi_eps: transition matrix; v_eps: vector of shocks
% grid on combined (persistent and transitory) med shocks
v_med   = kron(v_z, ones(N_eps, 1)) + kron(ones(N_z,1), v_eps);
% transition matrix for combined med shocks
Pi_med  = kron(Pi_z, Pi_eps(1,:));
N_med   = N_z * N_eps;

% create matrices to store value functions, consumption and others
% index: periods in good health, periods in bad health
```

```
m_V_f        = zeros(N_x, N_h * N_z);     % future value
m_V          = zeros(N_x, N_h * N_z, T);  % value function
m_c          = zeros(N_x, N_h * N_z, T);  % consumption choice
m_a          = zeros(N_x, N_h * N_z, T);  % end-of-period assets
p_bh         = zeros(T, N_h);             % prob of bad health (graph)
next_med     = zeros(T-1, N_h * N_z);     % med expenses next period (graph)

% LAST PERIOD (T)
utility          = @(c) c.^(1-nu) / (1-nu);  % period utility function
m_c(:,:,T)       = repmat(v_x, 1, N_h * N_z);
m_a(:,:,T)       = zeros(N_x, N_h * N_z);
m_V(:,:,T)       = repmat(utility(v_x), 1, N_h * N_z);

% ITERATIONS ON VALUE FUNCTION
for t = T-1:-1:1

    disp(sprintf('t: %d', t))
    m_V_f        = m_V(:, :, t+1); % parallel comput does not work with m_V
    Pi_h         = [1-p_h(t,1) p_h(t,1); 1-p_h(t,2) p_h(t,2)]^(1/2); % Pi(h)
    prob_tr      = kron(Pi_h, Pi_med);
    p_bh(t,:)    = Pi_h(:,2)'; % prob of bad health (graph below)
    med          = exp(repmat(Xb_med(t+1,:), 1, N_med) ...
                        + kron(Xb_var_med(t+1,:).^(1/2), v_med'));
    next_med(t, :) = prob_tr * med';  % average med exp next period (graph)

    parfor n_x = 1:N_x

        v_cons = zeros(N_h * N_z, 1);
        v_x_p = zeros(N_h * N_z, 1);
        v_V = zeros(N_h * N_z, 1);

        for n_h = 1:N_h

            for n_z = 1:N_z

                ind = (n_h-1) * N_z + n_z;

            if n_x == 1 % consumption floor is reached

                V_s = objective(c_ubar, r, v_x(n_x), inc(t+1), ...
                      brackets, tax, med, c_ubar, prob_tr(ind,:), v_x, ...
                      m_V_f, d, lower_x, nu, beta, s(t, n_h));
                cons = c_ubar;
            else
```

```
                f = @(c) objective(c, r, v_x(n_x), inc(t+1), ...
                    brackets, tax, med, c_ubar, prob_tr(ind,:), v_x, ...
                    m_V_f, d, lower_x, nu, beta, s(t, n_h));

                [cons, V_s] = gss(f, c_ubar, v_x(n_x), tol);
            end

                v_cons(ind, 1)       = cons;
                v_x_p(ind,1)         = v_x(n_x) - cons;
                v_V(ind,1)           = V_s;
            end
        end

        m_c(n_x,:,t)       = v_cons;
        m_a(n_x,:,t)       = v_x_p;
        m_V(n_x,:,t)       = v_V;
    end
end

% FIGURES

figure(1)
for period = [10, 20, 25]
    plot(v_x, m_c(:, 5, period))
    hold on
end
plot(v_x, v_x, '--', v_x, c_ubar*ones(N_x,1), '--')
hold off
legend(['        period 10'; '        period 20'; '
period 25'; ...
       ' 45 degrees line'; 'consumption floor'])
title('consumption')
xlabel('cash on hand')

figure(2)
plot(1:T-1, next_med)
title('average med expenses next period')
xlabel('period')
xlim([1 T-1])

% parameters: t (1 to 31), health (1/2 for healthy/sick),
%             persistent shocks (1 to N_z)

t   = [25 25];
n_h = [1 1];
n_z = [1 9];
```

```
ind1 = (n_h(1)-1) * N_z + n_z(1);
ind2 = (n_h(2)-1) * N_z + n_z(2);
text = [sprintf('t: %d, health: %d, pers shock: %d', t(1), n_h(1), n_z(1));
        sprintf('t: %d, health: %d, pers shock: %d', t(2), n_h(2), n_z(2))];

figure(3)
ax1 = subplot(3, 1, 1);
plot(v_x, m_c(:, ind1, t(1)), '-', v_x, m_c(:, ind2, t(2)), '--', ...
     v_x, v_x, ':', v_x, c_ubar*ones(N_x,1), '--')
title('consumption')
xlabel('cash on hand')
xlim([0,upper_x])
legend(text(1,:), text(2,:), 'location', 'best')

ax2 = subplot(3, 1, 2);
plot(v_x, m_V(:, ind1, t(1)), '-', v_x, m_V(:, ind2, t(2)), '--')
title('value function')
xlabel('cash on hand')
legend(text(1,:), text(2,:), 'location', 'best')

subplot(3, 1, 3);
plot(1:T-1, next_med(:, ind1), '-', 1:T-1, next_med(:, ind2), '--');
title('average medical expenses next period')
xlabel('period')
legend(text(1,:), text(2,:), 'location', 'best')
xlim([1 T-1])

figure(4)
ax3 = subplot(3, 1, 1);
plot(1:T-1, s(:,1), '-', 1:T-1, s(:,2), '--');
title('survival probability')
xlabel('period')
xlim([1 T-1])
legend(ax3, {'good health', 'bad health'}, 'location', 'best')

ax4 = subplot(3, 1, 2);
plot(1:T, p_bh(:,1), '-', 1:T, p_bh(:,2), '--');
title('prob of bad health next period')
xlabel('period')
xlim([1 T-1])
legend(ax4, {'good health', 'bad health'}, 'location', 'best')

subplot(3, 1, 3);
plot(1:T, inc(:));
title('gross income (not from assets)')
```

```
xlim([1 T-1])
xlabel('period')

% save figures (with right size)
h = figure(1);
set(h,'Units','Inches');
pos = get(h,'Position');
set(h,'PaperPositionMode','Auto','PaperUnits','Inches','PaperSize', ...
    [pos(3), pos(4)])
print(h, strcat(path_graphs, 'cons.pdf'), '-dpdf', '-r0')

h = figure(2);
set(h,'Units','Inches');
pos = get(h,'Position');
set(h,'PaperPositionMode','Auto','PaperUnits','Inches','PaperSize', ...
    [pos(3), pos(4)])
print(h, strcat(path_graphs, 'med.pdf'), '-dpdf', '-r0')

h = figure(3);
set(h,'Units','Inches');
pos = get(h,'Position');
set(h,'PaperPositionMode','Auto','PaperUnits','Inches','PaperSize', ...
    [pos(3), pos(4)])
print(h, strcat(path_graphs, 'decisions.pdf'), '-dpdf', '-r0')

h = figure(4);
set(h,'Units','Inches');
pos = get(h,'Position');
set(h,'PaperPositionMode','Auto','PaperUnits','Inches','PaperSize', ...
    [pos(3), pos(4)])
print(h,strcat(path_graphs, 'params.pdf'),'-dpdf','-r0')

% FUNCTIONS

% objective function
function V = objective(c, r, x, inc, brackets, tax, med, c_ubar, ...
                        prob_tr, v_x, m_V_f, d, lower_x, nu, beta, s)

y       = r * (x - c)  + inc; % earnings next period
net_y   = y - interp1(brackets, tax, y); % earnings net of taxes
cih     = max(x - c + net_y - med, c_ubar)'; % cih next period
EV      = prob_tr * interpy(v_x, m_V_f, cih, d, lower_x); % E[value(t+1)]
V       = c.^(1-nu) / (1-nu) + beta * s * EV; % value(t)
end

% fast interpolation function (updated to work with matrix m_V
```

```
function y0 = interpy(x, m_V, x0, d, lower_x) % interpolation

[M, N] = size(m_V);
ind1 = min(floor((sqrt(x0)-sqrt(lower_x))/d)+1, M-1); % index for x
ind2 = kron([0:N-1]', ones(length(x0)/N,1)*M); % ind to adj for col of m_V
ind = ind1 + ind2; % ind to use linear index in m_V

y0 = m_V(ind) + (x0-x(ind1))./(x(ind1+1)-x(ind1)) .* (m_V(ind+1)-m_V(ind));
end


% golden section search function; searches for a MAX
function [argmax_gss, max_gss] = gss(f, a, b, tol)

psi = (1+sqrt(5))/2;
c   = b - (b-a)/psi;
d   = a + (b-a)/psi;
f_c = f(c);
f_d = f(d);

while b - a > tol

    if f_c > f_d

        b   = d;
        d   = c;
        c   = b - (b-a)/psi;
        f_d = f_c;
        f_c = f(c);
    else

        a   = c;
        c   = d;
        d   = a + (b-a)/psi;
        f_c = f_d;
        f_d = f(d);
    end
end

argmax_gss  = (a+b)/2;
max_gss     = f(argmax_gss);
end
```

7.2 Simulate_model.m: Calling the Simulations

This script calls the scripts that simulate the benchmark model (simulation.m), the model without medical expenses (`simulation_no_med.m`) and the model without medical expenses risk (`simulation_mean_med.m`).

```
% computes decision functions for both genders and all quintiles

clc
clear all

path_graphs = '../graphs/';
M_C = cell(1,5); % store decision functions
M_C_no_med = cell(1,5);
M_C_mean_med = cell(1,5);

params.g       = 0;      % gender: female
params.N_x     = 500;  % number of points on grid cash on hand (coh)
params.upper_x = 10500000; % upper bound on x grid adjusted to max wealth
params.N_z     = 9;  % number grid points medical expenses permanent shock
params.N_eps   = 8;  % number grid points medical expenses transitory shock

params.nu      = 3.81; % curvature on period utility function
params.beta    = 0.97; % discount factor
params.c_ubar  = 2663; % consumption floor
age_min        = 70;   % starting age
age_max        = 100;  % max age
params.T       = age_max-age_min+1; % number of periods
params.r       = 0.02;      % interest rate
params.rho     = 0.922;   % rho medical shock; zeta(t)=rho*zeta(t-1)+eps(t)
params.sig_z   = sqrt(0.05); % sd persistent med shock; eps~N(0,sig_zeta^2)
params.sig_eps = sqrt(0.665); % sd transitory shock medical expenses
params.N_h     = 2;  % number of health states

params.path_data = '../data/'; % path for data
params.path_output = '../output/'; % path for output

% CREATE DECISION FUNCTIONS


for quintile = 1:5

    quintile

    ind = quintile;

    M_C{ind} = DFJ_cons(quintile, params);
```

```
    M_C_no_med{ind} = DFJ_cons_no_med(quintile, params);
    M_C_mean_med{ind} = DFJ_cons_mean_med(quintile, params);

end


save('../output/decision_fns.mat', 'M_C', 'M_C_no_med', 'M_C_mean_med')


% SIMULATIONS USING DECISION FUNCTIONS ABOVE

% simulations for cohort 1, all quintiles and female
N = 2000; % number of simulations
cohort = 1;

% load data
cohort1_female = readtable(strcat(params.path_data,'cohort1_female.csv'));

% FEMALE, COHORT 1 AND QUINTILE 3

figure(1)
quintile=3
[c_sim, x_sim, a_sim, s_sim] = simulation(quintile, cohort, N, params);
plot(70:100, median(a_sim, 'omitnan'), ...
cohort1_female{:,'age'}, cohort1_female{:,sprintf('q%d',quintile)},'--')

title('median assets level')
xlabel('age')
xlim([74, 84])

% FIGURE 5 IN PAPER (only for females of cohort 1)

figure(2)
for quintile = 1:5

    [c_sim, x_sim, a_sim, s_sim] = simulation(quintile, cohort, N, params);
    plot(70:100, median(a_sim, 'omitnan'), ...
    cohort1_female{:,'age'}, cohort1_female{:,sprintf('q%d',quintile)},'--')

    hold on
end

    title('median assets level')
    xlabel('age')
    xlim([74, 84])
hold off
```

```
% FIGURE 9 IN PAPER (only for females of cohort 1)

figure(3)

for quintile = 1:5

    [c_sim, x_sim, a_sim, s_sim] = simulation(quintile, cohort, N, params);
    plot(70:100, median(a_sim, 'omitnan'), '--')
    hold on
    [c_sim, x_sim, a_sim, s_sim] = simulation_no_med(quintile, cohort, ...
                                                       N, params);
    plot(70:100, median(a_sim, 'omitnan'))
    hold on
end

    title('median assets level')
    xlabel('age')
    xlim([74, 100])
hold off

% FIGURE 10 IN PAPER (only for females of cohort 1)

figure(4)

for quintile = 1:5

    [c_sim, x_sim, a_sim, s_sim] = simulation(quintile, cohort, N, params);
    plot(70:100, median(a_sim, 'omitnan'), '--')
    hold on
    [c_sim, x_sim, a_sim, s_sim] = simulation_mean_med(quintile, ...
                                                        cohort, N, params);
    plot(70:100, median(a_sim, 'omitnan'))
    hold on
end

    title('median assets level')
    xlabel('age')
    xlim([74, 100])
hold off

% save figures (with right size)

h = figure(1);
set(h,'Units','Inches');
pos = get(h,'Position');
```

```
set(h,'PaperPositionMode','Auto','PaperUnits','Inches','PaperSize', ...
    [pos(3), pos(4)])
print(h, strcat(path_graphs, 'simul_q3.pdf'), '-dpdf', '-r0')

h = figure(2);
set(h,'Units','Inches');
pos = get(h,'Position');
set(h,'PaperPositionMode','Auto','PaperUnits','Inches','PaperSize', ...
    [pos(3), pos(4)])
print(h, strcat(path_graphs, 'assets.pdf'), '-dpdf', '-r0')

h = figure(3);
set(h,'Units','Inches');
pos = get(h,'Position');
set(h,'PaperPositionMode','Auto','PaperUnits','Inches','PaperSize', ...
    [pos(3), pos(4)])
print(h, strcat(path_graphs, 'no_med.pdf'), '-dpdf', '-r0')

% save figures (with right size)
h = figure(4);
set(h,'Units','Inches');
pos = get(h,'Position');
set(h,'PaperPositionMode','Auto','PaperUnits','Inches','PaperSize',[...
    pos(3), pos(4)])
print(h, strcat(path_graphs, 'mean_med.pdf'), '-dpdf', '-r0')
```

7.2.1 simulation.m

This script simulates the benchmark model. It uses `DFJ_cons.m` (below).

```
%%%%%%%%%%%%%%
% SIMULATION %
%%%%%%%%%%%%%%

% simulation for N agents given g, quintile, cohort
% uses consumption functions computed w/ DFJ_cons.m
% creates artificial data for c, x and s

function [c_sim, x_sim, a_sim, s_sim] = simulation(quintile, cohort, ...
                                            N , params)

g           = params.g; % gender, female
c_ubar      = params.c_ubar; % consumption floor
r           = params.r;     % real interest rate
T           = params.T; % number of periods
```

```
rho         = params.rho;    % rho med shock; zeta(t)=rho*zeta(t-1)+eps(t)
sig_z       = params.sig_z;   % sd persist med shock; eps ~ N(0,sig_zeta^2)
sig_eps     = params.sig_eps;  % sd transitory shock medical expenses
N_x         = params.N_x;  % number of points on grid cash on hand (coh)
upper_x     = params.upper_x; % upper bound on x grid adj to max wealth
N_h         = params.N_h;    % number of health states
N_z         = params.N_z;   % numb grid points med expenses permanent shock
N_eps       = params.N_eps; % numb grid pts med expenses transitory shock
path_data   = params.path_data; % path for data
path_output = params.path_output; % path for output

% open decision function
ind = g*5 + quintile; % M_C: g=0,1 and quintile=1,...,5
decision_fns = load(strcat(path_output,'decision_fns.mat'));
m_c = decision_fns.M_C{ind};

% upload coefficient matrices for survival, health shock, income and
% medical expenses and convert them into vector indexed by age 70 to 100

I = 0.1 + 0.2*(quintile-1);     % middle range of quintile
m_agent     = [1 0 g I I^2;     % agent specific characteristics
               1 1 g I I^2];    % for good and bad health (second row)
fileID      = fopen(strcat(path_data,'deathprof.out'),'r');
s_coef      = fscanf(fileID, '%f', [6 33]);     % survival logit coefs
Xb_s        = (m_agent * s_coef(2:6, 3:33))';   % age 72 to 102 for probs
p_s           = sqrt(exp(Xb_s) ./ (1+exp(Xb_s))); % survival probabilities
                                                % sqrt() b/c 2 years prob
fileID      = fopen(strcat(path_data,'healthprof.out'),'r');
h_coef      = fscanf(fileID, '%f', [6 33]);     % health logit coefficients
Xb_h        = (m_agent * h_coef(2:6, 3:33))';   % age 72 to 102 for probs
p_h         = exp(Xb_h) ./ (1+exp(Xb_h));       % health transition prob

fileID      = fopen(strcat(path_data,'incprof.out'),'r');
inc_coef    = fscanf(fileID, '%f', [6 33]);         % income coefficients
Xb_inc      = (m_agent * inc_coef(2:6, 1:32))';     % using age 70 to 100
inc         = exp(Xb_inc(:,1));                     % income indep of h

fileID      = fopen(strcat(path_data,'medexprof_adj.out'),'r');
med_coef    = fscanf(fileID, '%f', [11 33]);       % medical expenses coefs
Xb_med      = (m_agent * med_coef(2:6, 1:32))';   % average (age 70-100)
Xb_var_med  = (m_agent * med_coef(7:11, 1:32))';  % volatility (age 70-100)

% markov chains med expenses shock
[Pi_z, eps, z_grid] = tauchen(N_z, 0, rho, sig_z); % zeta shock
[Pi_eps, eps, eps_grid] = tauchen(N_eps, 0, 0, sig_eps); % eps shock
```

```
% tax schedule
brackets    = [0, 6250, 40200, 68400, 93950, 148250, 284700, 1e10];
% income brackets (upper bound 1e6)
tau         = [0.0765, 0.2616, 0.4119, 0.3499, 0.3834, 0.4360, 0.4761];
% marginal rates
tax         = zeros(8, 1);
for i = 1:7
    tax(i+1)    =  tax(i) + (brackets(i+1)-brackets(i)) * tau(i);
end

% create grid on cash in hand x
lower_x     = c_ubar;
v_x         = linspace(sqrt(lower_x), sqrt(upper_x), N_x)'.^2;
                         % tighter grid for smaller values
d           = (sqrt(upper_x) - sqrt(lower_x)) / (N_x-1); % dist b/w gridpts

% initial distributions
tab_init = readtable(strcat(path_data,'data_init.csv'));
tab_gI = tab_init((tab_init.g == g) & (tab_init.quintile == quintile) ...
                   & (tab_init.cohort == cohort), ...
                  {'h', 'a', 'med', 'inc', 'age'});

% random draws in inital distribution
vec_n = randi(size(tab_gI,1), N,1);

% empty matrices for survival paths, consumption paths, cash on hand paths
s_sim = NaN(N, T);
c_sim = NaN(N, T);
x_sim = NaN(N, T);
a_sim = NaN(N, T);

% loop on simulations
for i = 1:N

n = vec_n(i); % pick initial conditions for an agent g, I

a0 = tab_gI{n, 'a'}; % assets
inc0 = tab_gI{n, 'inc'}; % income
med0 = tab_gI{n, 'med'}; % med expenses
t0 = tab_gI{n, 'age'} - 69; % initial period (age 70: 1)
h0 = tab_gI{n, 'h'} + 1; % good health: 1, bad health: 2

% initial cash on hand x0:
y0          = r*a0  + inc0; % earnings next period
net_y0      = y0 - interp1(brackets, tax, y0); % earnings net of taxes
x0          = max(a0 + net_y0 - med0, c_ubar); % initial cash on hand
```

```
% simulating medical shock
z0_est = (log(med0) - Xb_med(t0, h0))/Xb_var_med(t0, h0); %estimates z0
[z0, n_z0] = min(abs(z_grid - z0_est)); % closest value in z grid

theta_z = simul(n_z0, Pi_z, T-t0); % path for persistent shock index
theta_eps = simul(1, Pi_eps, T-t0); % path for transitory shock index
v_z = z_grid(theta_z); % persistent shock
v_eps = eps_grid(theta_eps); % transitory shock
v_psi = v_z + v_eps; % combined shocks

h = h0; % initial health
ind_z = n_z0; % initial persistent shock index
x = x0; % initial coh
x_sim(i,t0) = x;
s_sim(i,t0) = 1; % initial survival state (1: alive)
a_sim(i,t0) = a0; % initial asset position
s = 1; % survival in first period



% simulate path for every draw
for t = t0:T-1

    if (s == 1) & (rand < p_s(t, h))  % s=1 if survival, s=NaN otherwise
        s = 1;
    else
        s = NaN; % dead nest period
    end

    s_sim(i, t+1) = s;

    ind = (h-1)*N_z + ind_z; % index agent w/ health h and persis shock n_z
    v_c = m_c(:, ind, t); % decision fn for agent w/ h, n_z
    c = interpy1(v_x, v_c, x, d, lower_x); % consumption choice
    c_sim(i, t) = c;
    a_sim(i, t+1) = x - c; % assets at beginning of t+1 (and end of t)

    h = 1 + (rand < p_h(t, h)); % h next period
    ind_z = theta_z(t-t0+1); % z ind next period
    med = exp(Xb_med(t+1, h) + ...
              Xb_var_med(t+1, h).^(1/2)*v_psi(t-t0+1)); % med next period
    y       = r*(x-c)  + inc(t+1); % earnings next period
    net_y   = y - interp1(brackets, tax, y); % earnings net of taxes
    x       = max(x-c + net_y - med, c_ubar); % cash on hand next period
    x_sim(i, t+1) = x;
```

```
    a_sim(i, t+1) = x - c; % assets at beginning of t+1 (and end of t)
end

c_sim(i, t+1) = x;

end

% fast interpolation function (vectors x, y and point x0)
function y0 = interpy1(x, y, x0, d, lower_x) % interpolation
N = length(x);
ind = min(floor((sqrt(x0)-sqrt(lower_x))/d)+1, N-1);
y0 = y(ind) + (x0-x(ind)) .* (y(ind+1)-y(ind)) ./ (x(ind+1)-x(ind));

% simulation of markov chain
function theta = simul(theta0, Pi, T)

Pi_cum = cumsum(Pi, 2);
theta = [theta0; zeros(T-1, 1)];

for t = 1:T-1

    theta(t+1) = find(rand < Pi_cum(theta(t), :), 1, 'first');
end
```

7.2.2 DFJ_cons.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DFJ BENCHMARK MODEL ONLY CONS %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% used in simulations

function m_c = benchmark_decision(quintile, params)


% PARAMETERS

% for all agents
g          = params.g;        % gender female
nu         = params.nu;       % curvature on period utility function
beta       = params.beta;     % discount factor
c_ubar     = params.c_ubar;   % consumption floor
r          = params.r;        % real interest rate
T          = params.T;        % number of periods
```

```
rho          = params.rho;      % rho med shock; zeta(t)=rho*zeta(t-1)+eps(t)
sig_z        = params.sig_z;    % sd persist med shock; eps ~ N(0,sig_zeta^2)
sig_eps      = params.sig_eps;   % sd transitory shock medical expenses
N_x          = params.N_x;      % number of points on grid cash-in-hand (cih)
upper_x      = params.upper_x; % upper bound on grid;
N_h          = params.N_h;     % number of health states
N_z          = params.N_z;   % numb grid points med expenses permanent shock
N_eps        = params.N_eps; % numb grid pts med expenses transitory shock
path_data    = params.path_data; % path for data

% gender: 1 is male, income quintile 1 to 5
I = quintile*0.2-0.1; % middle percentile for each quintile (0.1-0.9)
m_agent      = [1 0 g I I^2;      % agent specific characteristics
                1 1 g I I^2];     % for good and bad health (second row)

tol          = 1;            % tol on golden section search algorithm

% tax schedule (brackets and marginal rates tau
brackets     = [0, 6250, 40200, 68400, 93950, 148250, 284700, 1e10];
tau          = [0.0765, 0.2616, 0.4119, 0.3499, 0.3834, 0.4360, 0.4761];
tax          = zeros(8, 1);
for i = 1:7
    tax(i+1)    =  tax(i) + (brackets(i+1)-brackets(i)) * tau(i);
end

% upload coefficient matrices for survival, health shock, income and
% medical expenses and convert them into vector indexed by age 70 to 100

fileID       = fopen(strcat(path_data,'deathprof.out'),'r');
s_coef       = fscanf(fileID, '%f', [6 33]);        % survival logit coefs
Xb_s         = (m_agent * s_coef(2:6, 3:32))';     % age 72 to 102 for probs
s            = sqrt(exp(Xb_s) ./ (1+exp(Xb_s)));   % survival probabilities
                                                    % sqrt() b/c 2 years prob
fileID       = fopen(strcat(path_data,'healthprof.out'),'r');
h_coef       = fscanf(fileID, '%f', [6 33]);        % health logit coefs
Xb_h         = (m_agent * h_coef(2:6, 3:32))';     % age 72 to 102 for probs
p_h          = exp(Xb_h) ./ (1+exp(Xb_h));         % health transition probs

fileID       = fopen(strcat(path_data,'incprof.out'),'r');
inc_coef     = fscanf(fileID, '%f', [6 33]);        % income coefs
Xb_inc       = (m_agent * inc_coef(2:6, 1:31))';   % using age 70 to 100
inc          = exp(Xb_inc(:,1));                    % income indep of h

fileID       = fopen(strcat(path_data,'medexprof_adj.out'),'r');
med_coef     = fscanf(fileID, '%f', [11 33]);       % medical expenses coefs
Xb_med       = (m_agent * med_coef(2:6, 1:31))';   % average (age 70-100)
```

```
Xb_var_med  = (m_agent * med_coef(7:11, 1:31))';  % volatility (age 70-100)


% SOLVING MODEL
% grid on cash in hand x
lower_x     = c_ubar;
v_x         = linspace(sqrt(lower_x), sqrt(upper_x), N_x)'.^2;
                % tighter grid for smaller values
d           = (sqrt(upper_x) - sqrt(lower_x)) / (N_x-1);
                % distance between gridpoints

% approximate shocks on medical expenses
[Pi_z, eps, v_z] = tauchen(N_z, 0, rho, sig_z);
                    % Pi_z: transition matrix: v_z: vector of shocks
[Pi_eps, eps, v_eps] = tauchen(N_eps, 0, 0, sig_eps);
                       % Pi_eps: transition matrix; v_eps: vector of shocks
% grid on combined (persistent and transitory) med shocks
v_med   = kron(v_z, ones(N_eps, 1)) + kron(ones(N_z,1), v_eps);
% transition matrix for combined med shocks
Pi_med  = kron(Pi_z, Pi_eps(1,:));
N_med   = N_z * N_eps;

% create matrices to store value functions, consumption and others
% index: periods in good health, periods in bad health

m_V_f       = zeros(N_x, N_h * N_z);     % future value
m_V         = zeros(N_x, N_h * N_z, T);  % value function
m_c         = zeros(N_x, N_h * N_z, T);  % consumption choice

% LAST PERIOD (T)
utility = @(c) c.^(1-nu) / (1-nu);  % period utility function
m_c(:,:,T)      = repmat(v_x, 1, N_h * N_z);
m_V(:,:,T)      = repmat(utility(v_x), 1, N_h * N_z);

% ITERATIONS ON VALUE FUNCTION
for t = T-1:-1:1

    disp(sprintf('t: %d', t))
    m_V_f       = m_V(:, :, t+1); % parallel comput does not work with m_V
    Pi_h        = [1-p_h(t,1) p_h(t,1); 1-p_h(t,2) p_h(t,2)]^(1/2); % Pi(h)
    prob_tr     = kron(Pi_h, Pi_med);
    med         = exp(repmat(Xb_med(t+1,:), 1, N_med) ...
                          + kron(Xb_var_med(t+1,:).^(1/2), v_med'));

    parfor n_x = 1:N_x
```

```
        v_cons = zeros(N_h * N_z, 1);
        v_V = zeros(N_h * N_z, 1);

        for n_h = 1:N_h

            for n_z = 1:N_z

                ind = (n_h-1) * N_z + n_z;

            if n_x == 1 % consumption floor is reached

                V_s = objective(c_ubar, r, v_x(n_x), inc(t+1), ...
                      brackets, tax, med, c_ubar, prob_tr(ind,:), v_x, ...
                      m_V_f, d, lower_x, nu, beta, s(t, n_h));
                cons = c_ubar;
            else

                f = @(c) objective(c, r, v_x(n_x), inc(t+1), ...
                         brackets, tax, med, c_ubar, prob_tr(ind,:), v_x, ...
                         m_V_f, d, lower_x, nu, beta, s(t, n_h));

                [cons, V_s] = gss(f, c_ubar, v_x(n_x), tol);
            end

                v_cons(ind, 1)       = cons;
                v_V(ind,1)           = V_s;
            end
        end

        m_c(n_x,:,t)        = v_cons;
        m_V(n_x,:,t)        = v_V;
    end
end


% FUNCTIONS

% objective function
function V = objective(c, r, x, inc, brackets, tax, med, c_ubar, ...
                       prob_tr, v_x, m_V_f, d, lower_x, nu, beta, s)

y           = r * (x - c)  + inc; % earnings next period
net_y       = y - interp1(brackets, tax, y); % earnings net of taxes
cih         = max(x - c + net_y - med, c_ubar)'; % cih next period
EV          = prob_tr * interpy(v_x, m_V_f, cih, d, lower_x); % E[value(t+1)]
V           = c.^(1-nu) / (1-nu) + beta * s * EV; % value(t)
```

```
% fast interpolation function (updated to work with matrix m_V
function y0 = interpy(x, m_V, x0, d, lower_x) % interpolation

[M, N] = size(m_V);
ind1 = min(floor((sqrt(x0)-sqrt(lower_x))/d)+1, M-1); % index for x
ind2 = kron([0:N-1]', ones(length(x0)/N,1)*M); % ind to adj for col of m_V
ind = ind1 + ind2; % ind to use linear index in m_V

y0 = m_V(ind) + (x0-x(ind1))./(x(ind1+1)-x(ind1)) .* (m_V(ind+1)-m_V(ind)) ;


% golden section search function; searches for a MAX
function [argmax_gss, max_gss] = gss(f, a, b, tol)

psi = (1+sqrt(5))/2;
c   = b - (b-a)/psi;
d   = a + (b-a)/psi;
f_c = f(c);
f_d = f(d);

while b - a > tol

    if f_c > f_d

        b   = d;
        d   = c;
        c   = b - (b-a)/psi;
        f_d = f_c;
        f_c = f(c);
    else

        a   = c;
        c   = d;
        d   = a + (b-a)/psi;
        f_c = f_d;
        f_d = f(d);
    end
end

argmax_gss  = (a+b)/2;
max_gss     = f(argmax_gss);
```

7.2.3 `simulation_no_med.m`

This script simulates the model without medical expenditures. It uses `DFJ_cons_no_med.m`

```
%%%%%%%%%%%%%%
% SIMULATION %
%%%%%%%%%%%%%%

% simulation for N agents given g, quintile, cohort
% uses consumption functions computed w/ DFJ_cons_no_med.m
% creates artificial data for c, x and s

function [c_sim, x_sim, a_sim, s_sim] = simulation_no_med(quintile, ...
                                        cohort, N , params)

g           = params.g; % gender, female
c_ubar      = params.c_ubar; % consumption floor
r           = params.r;     % real interest rate
T           = params.T; % number of periods
rho         = params.rho;     % rho med shock; zeta(t)=rho*zeta(t-1)+eps(t)
sig_z       = params.sig_z;    % sd persist med shock; eps ~ N(0,sig_zeta^2)
sig_eps     = params.sig_eps;  % sd transitory shock medical expenses
N_x         = params.N_x;  % number of points on grid cash on hand (coh)
upper_x     = params.upper_x; % upper bound on x grid adjusted to max wealth
N_h         = params.N_h;     % number of health states
N_z         = params.N_z;    % numb grid points med expenses permanent shock
N_eps       = params.N_eps; % numb grid pts med expenses transitory shock
path_data   = params.path_data; % path for data
path_output = params.path_output; % path for output

% open decision function
ind = g*5 + quintile; % M_C: g=0,1 and quintile=1,...,5
decision_fns = load(strcat(path_output,'decision_fns.mat'));
m_c = decision_fns.M_C_no_med{ind};

% upload coefficient matrices for survival, health shock, income and
% medical expenses and convert them into vector indexed by age 70 to 100

I = 0.1 + 0.2*(quintile-1);     % middle range of quintile
m_agent     = [1 0 g I I^2;     % agent specific characteristics
               1 1 g I I^2];    % for good and bad health (second row)
fileID      = fopen(strcat(path_data,'deathprof.out'),'r');
s_coef      = fscanf(fileID, '%f', [6 33]);     % survival logit coeffs
Xb_s        = (m_agent * s_coef(2:6, 3:33))';   % uage 72 to 102 for probs
p_s           = sqrt(exp(Xb_s) ./ (1+exp(Xb_s))); % survival probabilities
                                                % sqrt() b/c 2 years prob
fileID      = fopen(strcat(path_data,'healthprof.out'),'r');
h_coef      = fscanf(fileID, '%f', [6 33]);     % health logit coefficients
Xb_h        = (m_agent * h_coef(2:6, 3:33))';   % age 72 to 102 for probs
```

```
p_h         = exp(Xb_h) ./ (1+exp(Xb_h));       % health transition probs

fileID      = fopen(strcat(path_data,'incprof.out'),'r');
inc_coef    = fscanf(fileID, '%f', [6 33]);          % income coefficients
Xb_inc      = (m_agent * inc_coef(2:6, 1:32))';      % using age 70 to 100
inc         = exp(Xb_inc(:,1));                      % income indep of h

fileID      = fopen(strcat(path_data,'medexprof_adj.out'),'r');
med_coef    = fscanf(fileID, '%f', [11 33]);      % medical expenses coeffs
Xb_med      = (m_agent * med_coef(2:6, 1:32))';   % average (age 70-100)
Xb_var_med  = (m_agent * med_coef(7:11, 1:32))';  % volatility (age 70-100)

% markov chains med expenses shock
[Pi_z, eps, z_grid] = tauchen(N_z, 0, rho, sig_z); % zeta shock
[Pi_eps, eps, eps_grid] = tauchen(N_eps, 0, 0, sig_eps); % eps shock

% tax schedule
brackets    = [0, 6250, 40200, 68400, 93950, 148250, 284700, 1e10];
                % income brackets (upper bound 1e6)
tau         = [0.0765, 0.2616, 0.4119, 0.3499, 0.3834, 0.4360, 0.4761];
                % marginal rates
tax         = zeros(8, 1);
for i = 1:7
    tax(i+1)    =  tax(i) + (brackets(i+1)-brackets(i)) * tau(i);
end

% create grid on cash in hand x
lower_x     = c_ubar;
v_x         = linspace(sqrt(lower_x), sqrt(upper_x), N_x)'.^2;
                        % tighter grid for smaller values
d           = (sqrt(upper_x) - sqrt(lower_x)) / (N_x-1); % dist b/w gridpts

% initial distributions
tab_init = readtable(strcat(path_data,'data_init.csv'));
tab_gI = tab_init((tab_init.g == g) & (tab_init.quintile == quintile) ...
                  & (tab_init.cohort == cohort), ...
                {'h', 'a', 'med', 'inc', 'age'});

% random draws in inital distribution
vec_n = randi(size(tab_gI,1), N,1);

% empty matrices for survival paths, consumption paths, cash on hand paths
s_sim = NaN(N, T);
c_sim = NaN(N, T);
x_sim = NaN(N, T);
a_sim = NaN(N, T);
```

```
% loop on simulations
for i = 1:N

n = vec_n(i); % pick initial conditions for an agent g, I

a0 = tab_gI{n, 'a'}; % assets
inc0 = tab_gI{n, 'inc'}; % income
med0 = tab_gI{n, 'med'}; % med expenses
t0 = tab_gI{n, 'age'} - 69; % initial period (age 70: 1)
h0 = tab_gI{n, 'h'} + 1; % good health: 1, bad health: 2

% initial cash on hand x0:
y0          = r*a0  + inc0; % earnings next period
net_y0      = y0 - interp1(brackets, tax, y0); % earnings net of taxes
x0          = max(a0 + net_y0 - med0, c_ubar); % initial cash on hand

% simulating medical shock
z0_est = (log(med0) - Xb_med(t0, h0))/Xb_var_med(t0, h0); %estimates z0
[z0, n_z0] = min(abs(z_grid - z0_est)); % closest value in z grid

theta_z = simul(n_z0, Pi_z, T-t0); % path for persistent shock index
theta_eps = simul(1, Pi_eps, T-t0); % path for transitory shock index
v_z = z_grid(theta_z); % persistent shock
v_eps = eps_grid(theta_eps); % transitory shock
v_psi = v_z + v_eps; % combined shocks

h = h0; % initial health
ind_z = n_z0; % initial persistent shock index
x = x0; % initial coh
x_sim(i,t0) = x;
s_sim(i,t0) = 1; % initial survival state (1: alive)
a_sim(i,t0) = a0; % initial asset position
s = 1; % survival in first period



% simulate path for every draw
for t = t0:T-1

    if (s == 1) & (rand < p_s(t, h))  % s=1 if survival, s=NaN otherwise
        s = 1;
    else
        s = NaN; % dead nest period
    end
```

```
    s_sim(i, t+1) = s;

    ind = (h-1)*N_z + ind_z; % index agent w/ health h and persis shock n_z
    v_c = m_c(:, ind, t); % decision fn for agent w/ h, n_z
    c = interpy1(v_x, v_c, x, d, lower_x); % consumption choice
    c_sim(i, t) = c;
    a_sim(i, t+1) = x - c; % assets at beginning of t+1 (and end of t)

    h       = 1 + (rand < p_h(t, h)); % h next period
    ind_z   = theta_z(t-t0+1); % z ind next period
    med     = 0; % med next period
    y       = r*(x-c)  + inc(t+1); % earnings next period
    net_y   = y - interp1(brackets, tax, y); % earnings net of taxes
    x       = max(x-c + net_y - med, c_ubar); % cash on hand next period
    x_sim(i, t+1) = x;
    a_sim(i, t+1) = x - c; % assets at beginning of t+1 (and end of t)
end

c_sim(i, t+1) = x;

end

% fast interpolation function (vectors x, y and point x0)
function y0 = interpy1(x, y, x0, d, lower_x) % interpolation
N = length(x);
ind = min(floor((sqrt(x0)-sqrt(lower_x))/d)+1, N-1);
y0 = y(ind) + (x0-x(ind)) .* (y(ind+1)-y(ind)) ./ (x(ind+1)-x(ind));

% simulation of markov chain
function theta = simul(theta0, Pi, T)

Pi_cum = cumsum(Pi, 2);
theta = [theta0; zeros(T-1, 1)];

for t = 1:T-1

    theta(t+1) = find(rand < Pi_cum(theta(t), :), 1, 'first');
end
```

7.2.4 DFJ_cons_no_med.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DFJ BENCHMARK MODEL ONLY CONS %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% used in simulations

function m_c = benchmark_decision(quintile, params)


% PARAMETERS

% for all agents
g           = params.g;          % gender female
nu          = params.nu;         % curvature on period utility function
beta        = params.beta;       % discount factor
c_ubar      = params.c_ubar;     % consumption floor
r           = params.r;          % real interest rate
T           = params.T;          % number of periods
rho         = params.rho;      % rho med shock; zeta(t)=rho*zeta(t-1)+eps(t)
sig_z       = params.sig_z;    % sd persist med shock; eps ~ N(0,sig_zeta^2)
sig_eps     = params.sig_eps;    % sd transitory shock medical expenses
N_x         = params.N_x;      % number of points on grid cash-in-hand (cih)
upper_x     = params.upper_x; % upper bound on grid;
N_h         = params.N_h;    % number of health states
N_z         = params.N_z;   % numb grid points med expenses permanent shock
N_eps       = params.N_eps; % numb grid pts med expenses transitory shock
path_data   = params.path_data; % path for data

% gender: 1 is male, income quintile 1 to 5
I = quintile*0.2-0.1; % middle percentile for each quintile (0.1-0.9)
m_agent     = [1 0 g I I^2;      % agent specific characteristics
               1 1 g I I^2];     % for good and bad health (second row)

tol         = 1;             % tol on golden section search algorithm

% tax schedule (brackets and marginal rates tau
brackets    = [0, 6250, 40200, 68400, 93950, 148250, 284700, 1e10];
tau         = [0.0765, 0.2616, 0.4119, 0.3499, 0.3834, 0.4360, 0.4761];
tax         = zeros(8, 1);
for i = 1:7
    tax(i+1)    =  tax(i) + (brackets(i+1)-brackets(i)) * tau(i);
end

% upload coefficient matrices for survival, health shock, income and
% medical expenses and convert them into vector indexed by age 70 to 100

fileID      = fopen(strcat(path_data,'deathprof.out'),'r');
s_coef      = fscanf(fileID, '%f', [6 33]);       % survival logit coefs
Xb_s        = (m_agent * s_coef(2:6, 3:32))';     % age 72 to 102 for probs
```

```
s            = sqrt(exp(Xb_s) ./ (1+exp(Xb_s)));   % survival probabilities
                                                    % sqrt() b/c 2 years prob
fileID       = fopen(strcat(path_data,'healthprof.out'),'r');
h_coef       = fscanf(fileID, '%f', [6 33]);        % health logit coefs
Xb_h         = (m_agent * h_coef(2:6, 3:32))';      % age 72 to 102 for probs
p_h          = exp(Xb_h) ./ (1+exp(Xb_h));          % health transition probs

fileID       = fopen(strcat(path_data,'incprof.out'),'r');
inc_coef     = fscanf(fileID, '%f', [6 33]);        % income coefs
Xb_inc       = (m_agent * inc_coef(2:6, 1:31))';    % using age 70 to 100
inc          = exp(Xb_inc(:,1));                    % income indep of h

fileID       = fopen(strcat(path_data,'medexprof_adj.out'),'r');
med_coef     = fscanf(fileID, '%f', [11 33]);       % medical expenses coefs
Xb_med       = (m_agent * med_coef(2:6, 1:31))';    % average (age 70-100)
Xb_var_med   = (m_agent * med_coef(7:11, 1:31))';   % volatility (age 70-100)


% SOLVING MODEL
% grid on cash in hand x
lower_x      = c_ubar;
v_x          = linspace(sqrt(lower_x), sqrt(upper_x), N_x)'.^2;
                 % tighter grid for smaller values
d            = (sqrt(upper_x) - sqrt(lower_x)) / (N_x-1);
                 % distance between gridpoints

% approximate shocks on medical expenses
[Pi_z, eps, v_z] = tauchen(N_z, 0, rho, sig_z);
                      % Pi_z: transition matrix: v_z: vector of shocks
[Pi_eps, eps, v_eps] = tauchen(N_eps, 0, 0, sig_eps);
                        % Pi_eps: transition matrix; v_eps: vector of shocks
% grid on combined (persistent and transitory) med shocks
v_med    = kron(v_z, ones(N_eps, 1)) + kron(ones(N_z,1), v_eps);
% transition matrix for combined med shocks
Pi_med   = kron(Pi_z, Pi_eps(1,:));
N_med    = N_z * N_eps;

% create matrices to store value functions, consumption and others
% index: periods in good health, periods in bad health

m_V_f        = zeros(N_x, N_h * N_z);      % future value
m_V          = zeros(N_x, N_h * N_z, T);  % value function
m_c          = zeros(N_x, N_h * N_z, T);  % consumption choice

% LAST PERIOD (T)
utility = @(c) c.^(1-nu) / (1-nu);  % period utility function
```

```
m_c(:,:,T)          = repmat(v_x, 1, N_h * N_z);
m_V(:,:,T)          = repmat(utility(v_x), 1, N_h * N_z);

% ITERATIONS ON VALUE FUNCTION
for t = T-1:-1:1

    disp(sprintf('t: %d', t))
    m_V_f          = m_V(:, :, t+1); % parallel comput does not work with m_V
    Pi_h           = [1-p_h(t,1) p_h(t,1); 1-p_h(t,2) p_h(t,2)]^(1/2); % Pi(h)
    prob_tr        = kron(Pi_h, Pi_med);
    med            = zeros(1, N_h*N_med);

    parfor n_x = 1:N_x

        v_cons = zeros(N_h * N_z, 1);
        v_V = zeros(N_h * N_z, 1);

        for n_h = 1:N_h

            for n_z = 1:N_z

                ind = (n_h-1) * N_z + n_z;

            if n_x == 1 % consumption floor is reached

                V_s = objective(c_ubar, r, v_x(n_x), inc(t+1), ...
                      brackets, tax, med, c_ubar, prob_tr(ind,:), v_x, ...
                      m_V_f, d, lower_x, nu, beta, s(t, n_h));
                cons = c_ubar;
            else

                f = @(c) objective(c, r, v_x(n_x), inc(t+1), ...
                         brackets, tax, med, c_ubar, prob_tr(ind,:), v_x,...
                         m_V_f, d, lower_x, nu, beta, s(t, n_h));

                [cons, V_s] = gss(f, c_ubar, v_x(n_x), tol);
            end

                v_cons(ind, 1)          = cons;
                v_V(ind,1)              = V_s;
            end
        end

        m_c(n_x,:,t)          = v_cons;
        m_V(n_x,:,t)          = v_V;
    end
```

```
end


% FUNCTIONS

% objective function
function V = objective(c, r, x, inc, brackets, tax, med, c_ubar, ...
                        prob_tr, v_x, m_V_f, d, lower_x, nu, beta, s)

y         = r * (x - c)  + inc; % earnings next period
net_y     = y - interp1(brackets, tax, y); % earnings net of taxes
cih       = max(x - c + net_y - med, c_ubar)'; % cih next period
EV        = prob_tr * interpy(v_x, m_V_f, cih, d, lower_x); % E[value(t+1)]
V         = c.^(1-nu) / (1-nu) + beta * s * EV; % value(t)

% fast interpolation function (updated to work with matrix m_V
function y0 = interpy(x, m_V, x0, d, lower_x) % interpolation

[M, N] = size(m_V);
ind1 = min(floor((sqrt(x0)-sqrt(lower_x))/d)+1, M-1); % index for x
ind2 = kron([0:N-1]', ones(length(x0)/N,1)*M); % ind to adj for col of m_V
ind = ind1 + ind2; % ind to use linear index in m_V

y0 = m_V(ind) + (x0-x(ind1))./(x(ind1+1)-x(ind1)) .* (m_V(ind+1)-m_V(ind));


% golden section search function; searches for a MAX
function [argmax_gss, max_gss] = gss(f, a, b, tol)

psi = (1+sqrt(5))/2;
c   = b - (b-a)/psi;
d   = a + (b-a)/psi;
f_c = f(c);
f_d = f(d);

while b - a > tol

    if f_c > f_d

        b   = d;
        d   = c;
        c   = b - (b-a)/psi;
        f_d = f_c;
        f_c = f(c);
    else
```

```
        a   = c;
        c   = d;
        d   = a + (b-a)/psi;
        f_c = f_d;
        f_d = f(d);
    end
end

argmax_gss  = (a+b)/2;
max_gss     = f(argmax_gss);
```

7.2.5 simulation_mean_med.m

```
%%%%%%%%%%%%%%
% SIMULATION %
%%%%%%%%%%%%%%

% simulation for N agents given g, quintile, cohort
% uses consumption functions computed w/ DFJ_cons_mean_med.m
% creates artificial data for c, x and s

function [c_sim, x_sim, a_sim, s_sim] = simulation_mean_med(quintile, ...
                                    cohort, N , params)

g           = params.g; % gender, female
c_ubar      = params.c_ubar; % consumption floor
r           = params.r;     % real interest rate
T           = params.T; % number of periods
rho         = params.rho;    % rho med shock; zeta(t)=rho*zeta(t-1)+eps(t)
sig_z       = params.sig_z;   % sd persist med shock; eps ~ N(0,sig_zeta^2)
sig_eps     = params.sig_eps;  % sd transitory shock medical expenses
N_x         = params.N_x;  % number of points on grid cash on hand (coh)
upper_x     = params.upper_x; % upper bound on x grid adjusted to max wealth
N_h         = params.N_h;    % number of health states
N_z         = params.N_z;    % numb grid points med expenses permanent shock
N_eps       = params.N_eps; % numb grid pts med expenses transitory shock
path_data   = params.path_data; % path for data
path_output = params.path_output; % path for output

% open decision function
ind = g*5 + quintile; % M_C: g=0,1 and quintile=1,...,5
decision_fns = load(strcat(path_output,'decision_fns.mat'));
m_c = decision_fns.M_C_mean_med{ind};

% upload coefficient matrices for survival, health shock, income and
```

```
% medical expenses and convert them into vector indexed by age 70 to 100

I = 0.1 + 0.2*(quintile -1);      % middle range of quintile
m_agent     = [1 0 g I I^2;       % agent specific characteristics
               1 1 g I I^2];      % for good and bad health (second row)
fileID      = fopen(strcat(path_data,'deathprof.out'),'r');
s_coef      = fscanf(fileID, '%f', [6 33]);      % survival logit coeffs
Xb_s        = (m_agent * s_coef(2:6, 3:33))';    % age 72 to 102 for probs
p_s           = sqrt(exp(Xb_s) ./ (1+exp(Xb_s))); % survival probabilities
                                                 % sqrt() b/c 2 years prob
fileID      = fopen(strcat(path_data,'healthprof.out'),'r');
h_coef      = fscanf(fileID, '%f', [6 33]);      % health logit coefficients
Xb_h        = (m_agent * h_coef(2:6, 3:33))';    % age 72 to 102 for probs
p_h         = exp(Xb_h) ./ (1+exp(Xb_h));        % health transition prob

fileID      = fopen(strcat(path_data,'incprof.out'),'r');
inc_coef    = fscanf(fileID, '%f', [6 33]);          % income coefficients
Xb_inc      = (m_agent * inc_coef(2:6, 1:32))';      % using age 70 to 100
inc         = exp(Xb_inc(:,1));                      % income indep of h

fileID      = fopen(strcat(path_data,'medexprof_adj.out'),'r');
med_coef    = fscanf(fileID, '%f', [11 33]);       % medical expenses coeffs
Xb_med      = (m_agent * med_coef(2:6, 1:32))';    % average (age 70-100)
Xb_var_med  = (m_agent * med_coef(7:11, 1:32))';  % volatility (age 70-100)

% markov chains med expenses shock
[Pi_z, eps, z_grid] = tauchen(N_z, 0, rho, sig_z); % zeta shock
[Pi_eps, eps, eps_grid] = tauchen(N_eps, 0, 0, sig_eps); % eps shock
med_grid   = kron(z_grid, ones(N_eps, 1)) + kron(ones(N_z,1), eps_grid);

% tax schedule
brackets    = [0, 6250, 40200, 68400, 93950, 148250, 284700, 1e10];
                % income brackets (upper bound 1e6)
tau         = [0.0765, 0.2616, 0.4119, 0.3499, 0.3834, 0.4360, 0.4761];
                % marginal rates
tax         = zeros(8, 1);
for i = 1:7
    tax(i+1)    =  tax(i) + (brackets(i+1)-brackets(i)) * tau(i);
end

% create grid on cash in hand x
lower_x     = c_ubar;
v_x         = linspace(sqrt(lower_x), sqrt(upper_x), N_x)'.^2;
                        % tighter grid for smaller values
d           = (sqrt(upper_x) - sqrt(lower_x)) / (N_x-1); % dist b/w gridpts
```

```
% initial distributions
tab_init = readtable(strcat(path_data,'data_init.csv'));
tab_gI = tab_init((tab_init.g == g) & (tab_init.quintile == quintile) ...
                  & (tab_init.cohort == cohort), ...
                 {'h', 'a', 'med', 'inc', 'age'});

% random draws in inital distribution
vec_n = randi(size(tab_gI,1), N,1);

% empty matrices for survival paths, consumption paths, cash on hand paths
s_sim = NaN(N, T);
c_sim = NaN(N, T);
x_sim = NaN(N, T);
a_sim = NaN(N, T);

% loop on simulations
for i = 1:N

n = vec_n(i); % pick initial conditions for an agent g, I

a0 = tab_gI{n, 'a'}; % assets
inc0 = tab_gI{n, 'inc'}; % income
med0 = tab_gI{n, 'med'}; % med expenses
t0 = tab_gI{n, 'age'} - 69; % initial period (age 70: 1)
h0 = tab_gI{n, 'h'} + 1; % good health: 1, bad health: 2

% initial cash on hand x0:
y0          = r*a0  + inc0; % earnings next period
net_y0      = y0 - interp1(brackets, tax, y0); % earnings net of taxes
x0          = max(a0 + net_y0, c_ubar); % initial cash on hand

% simulating medical shock
z0_est = (log(med0) - Xb_med(t0, h0))/Xb_var_med(t0, h0); %estimates z0
[z0, n_z0] = min(abs(z_grid - z0_est)); % closest value in z grid

theta_z = simul(n_z0, Pi_z, T-t0); % path for persistent shock index
theta_eps = simul(1, Pi_eps, T-t0); % path for transitory shock index
v_z = z_grid(theta_z); % persistent shock
v_eps = eps_grid(theta_eps); % transitory shock
v_psi = v_z + v_eps; % combined shocks

h = h0; % initial health
ind_z = n_z0; % initial persistent shock index
x = x0; % initial coh
x_sim(i,t0) = x;
s_sim(i,t0) = 1; % initial survival state (1: alive)
```

```
a_sim(i,t0) = a0; % initial asset position
s = 1; % survival in first period



% simulate path for every draw
for t = t0:T-1

    if (s == 1) & (rand < p_s(t, h))  % s=1 if survival, s=NaN otherwise
        s = 1;
    else
        s = NaN; % dead next period
    end

    s_sim(i, t+1) = s;

    ind = (h-1)*N_z + ind_z; % index agent w/ health h and pers shock n_z
    v_c = m_c(:, ind, t); % decision fn for agent fn of h, n_z
    c = interpy1(v_x, v_c, x, d, lower_x); % consumption choice
    c_sim(i, t) = c;
    a_sim(i, t+1) = x - c; % assets at beginning of t+1 (and end of t)

    h       = 1 + (rand < p_h(t, h)); % h next period
    ind_z   = theta_z(t-t0+1); % z ind next period
    med     = mean(exp(Xb_med(t+1, h) + ...
                Xb_var_med(t+1, h).^(1/2) * med_grid)); %mean med next period
    y       = r*(x-c)  + inc(t+1); % earnings next period
    net_y   = y - interp1(brackets, tax, y); % earnings net of taxes
    x       = max(x-c + net_y - med, c_ubar); % cash on hand next period
    x_sim(i, t+1) = x;
    a_sim(i, t+1) = x - c; % assets at beginning of t+1 (and end of t)
end

c_sim(i, t+1) = x;

end

% fast interpolation function (vectors x, y and point x0)
function y0 = interpy1(x, y, x0, d, lower_x) % interpolation
N = length(x);
ind = min(floor((sqrt(x0)-sqrt(lower_x))/d)+1, N-1);
y0 = y(ind) + (x0-x(ind)) .* (y(ind+1)-y(ind)) ./ (x(ind+1)-x(ind));

% simulation of markov chain
function theta = simul(theta0, Pi, T)
```

```
Pi_cum = cumsum(Pi, 2);
theta = [theta0; zeros(T-1, 1)];

for t = 1:T-1

    theta(t+1) = find(rand < Pi_cum(theta(t), :), 1, 'first');
end
```

7.2.6 DFJ_cons_mean_med.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DFJ BENCHMARK MODEL ONLY CONS %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% used in simulations

function m_c = benchmark_decision(quintile, params)


% PARAMETERS

% for all agents
g           = params.g;         % gender female
nu          = params.nu;        % curvature on period utility function
beta        = params.beta;      % discount factor
c_ubar      = params.c_ubar;    % consumption floor
r           = params.r;         % real interest rate
T           = params.T;         % number of periods
rho         = params.rho;     % rho med shock; zeta(t)=rho*zeta(t-1)+eps(t)
sig_z       = params.sig_z;   % sd persist med shock; eps ~ N(0,sig_zeta^2)
sig_eps     = params.sig_eps;   % sd transitory shock medical expenses
N_x         = params.N_x;     % number of points on grid cash-in-hand (cih)
upper_x     = params.upper_x; % upper bound on grid;
N_h         = params.N_h;    % number of health states
N_z         = params.N_z;   % numb grid points med expenses permanent shock
N_eps       = params.N_eps; % numb grid pts med expenses transitory shock
path_data   = params.path_data; % path for data

% gender: 1 is male, income quintile 1 to 5
I = quintile*0.2-0.1; % middle percentile for each quintile (0.1-0.9)
m_agent     = [1 0 g I I^2;     % agent specific characteristics
               1 1 g I I^2];    % for good and bad health (second row)

tol         = 1;              % tol on golden section search algorithm
```

```
% tax schedule (brackets and marginal rates tau
brackets    = [0, 6250, 40200, 68400, 93950, 148250, 284700, 1e10];
tau         = [0.0765, 0.2616, 0.4119, 0.3499, 0.3834, 0.4360, 0.4761];
tax         = zeros(8, 1);
for i = 1:7
    tax(i+1)    =  tax(i) + (brackets(i+1)-brackets(i)) * tau(i);
end

% upload coefficient matrices for survival, health shock, income and
% medical expenses and convert them into vector indexed by age 70 to 100

fileID      = fopen(strcat(path_data,'deathprof.out'),'r');
s_coef      = fscanf(fileID, '%f', [6 33]);         % survival logit coefs
Xb_s        = (m_agent * s_coef(2:6, 3:32))';       % age 72 to 102 for probs
s           = sqrt(exp(Xb_s) ./ (1+exp(Xb_s)));    % survival probabilities
                                                    % sqrt() b/c 2 years prob
fileID      = fopen(strcat(path_data,'healthprof.out'),'r');
h_coef      = fscanf(fileID, '%f', [6 33]);         % health logit coefs
Xb_h        = (m_agent * h_coef(2:6, 3:32))';       % age 72 to 102 for probs
p_h         = exp(Xb_h) ./ (1+exp(Xb_h));           % health transition probs

fileID      = fopen(strcat(path_data,'incprof.out'),'r');
inc_coef    = fscanf(fileID, '%f', [6 33]);         % income coefs
Xb_inc      = (m_agent * inc_coef(2:6, 1:31))';     % using age 70 to 100
inc         = exp(Xb_inc(:,1));                     % income indep of h

fileID      = fopen(strcat(path_data,'medexprof_adj.out'),'r');
med_coef    = fscanf(fileID, '%f', [11 33]);        % medical expenses coefs
Xb_med      = (m_agent * med_coef(2:6, 1:31))';     % average (age 70-100)
Xb_var_med  = (m_agent * med_coef(7:11, 1:31))';    % volatility (age 70-100)

% SOLVING MODEL
% grid on cash in hand x
lower_x     = c_ubar;
v_x         = linspace(sqrt(lower_x), sqrt(upper_x), N_x)'.^2;
                % tighter grid for smaller values
d           = (sqrt(upper_x) - sqrt(lower_x)) / (N_x-1);
                % distance between gridpoints

% approximate shocks on medical expenses
[Pi_z, eps, v_z] = tauchen(N_z, 0, rho, sig_z);
                    % Pi_z: transition matrix: v_z: vector of shocks
[Pi_eps, eps, v_eps] = tauchen(N_eps, 0, 0, sig_eps);
                        % Pi_eps: transition matrix; v_eps: vector of shocks
% grid on combined (persistent and transitory) med shocks
v_med   = kron(v_z, ones(N_eps, 1)) + kron(ones(N_z,1), v_eps);
```

```
% transition matrix for combined med shocks
Pi_med   = kron(Pi_z, Pi_eps(1,:));
N_med    = N_z * N_eps;

% create matrices to store value functions, consumption and others
% index: periods in good health, periods in bad health

m_V_f        = zeros(N_x, N_h * N_z);     % future value
m_V          = zeros(N_x, N_h * N_z, T);  % value function
m_c          = zeros(N_x, N_h * N_z, T);  % consumption choice

% LAST PERIOD (T)
utility = @(c) c.^(1-nu) / (1-nu);  % period utility function
m_c(:,:,T)       = repmat(v_x, 1, N_h * N_z);
m_V(:,:,T)       = repmat(utility(v_x), 1, N_h * N_z);

% ITERATIONS ON VALUE FUNCTION
for t = T-1:-1:1

    disp(sprintf('t: %d', t))
    m_V_f        = m_V(:, :, t+1); % parallel comput does not work with m_V
    Pi_h         = [1-p_h(t,1) p_h(t,1); 1-p_h(t,2) p_h(t,2)]^(1/2); % Pi(h)
    prob_tr      = kron(Pi_h, Pi_med);
    med_         = exp(repmat(Xb_med(t+1,:), 1, N_med) ...
                          + kron(Xb_var_med(t+1,:).^(1/2), v_med'));
    med          = kron([mean(med_(1:N_med)), mean(med_(N_med+1:2*N_med))],...
                       ones(1,N_med)); % mean med expenses by health status

    parfor n_x = 1:N_x

        v_cons = zeros(N_h * N_z, 1);
        v_V = zeros(N_h * N_z, 1);

        for n_h = 1:N_h

            for n_z = 1:N_z

                ind = (n_h-1) * N_z + n_z;

            if n_x == 1 % consumption floor is reached

                V_s = objective(c_ubar, r, v_x(n_x), inc(t+1), ...
                      brackets, tax, med, c_ubar, prob_tr(ind,:), v_x, ...
                      m_V_f, d, lower_x, nu, beta, s(t, n_h));
                cons = c_ubar;
            else
```

```
                f = @(c) objective(c, r, v_x(n_x), inc(t+1), ...
                         brackets, tax, med, c_ubar, prob_tr(ind,:), v_x, ...
                         m_V_f, d, lower_x, nu, beta, s(t, n_h));

                [cons, V_s] = gss(f, c_ubar, v_x(n_x), tol);
            end

                v_cons(ind, 1)          = cons;
                v_V(ind,1)              = V_s;
            end
        end

        m_c(n_x,:,t)          = v_cons;
        m_V(n_x,:,t)          = v_V;
    end
end


% FUNCTIONS

% objective function
function V = objective(c, r, x, inc, brackets, tax, med, c_ubar, ...
                       prob_tr, v_x, m_V_f, d, lower_x, nu, beta, s)

y           = r * (x - c)  + inc; % earnings next period
net_y       = y - interp1(brackets, tax, y); % earnings net of taxes
cih         = max(x - c + net_y - med, c_ubar)'; % cih next period
EV          = prob_tr * interpy(v_x, m_V_f, cih, d, lower_x); %E[value(t+1)]
V           = c.^(1-nu) / (1-nu) + beta * s * EV; % value(t)

% fast interpolation function (updated to work with matrix m_V
function y0 = interpy(x, m_V, x0, d, lower_x) % interpolation

[M, N] = size(m_V);
ind1 = min(floor((sqrt(x0)-sqrt(lower_x))/d)+1, M-1); % index for x
ind2 = kron([0:N-1]', ones(length(x0)/N,1)*M); % ind to adj for col of m_V
ind = ind1 + ind2; % ind to use linear index in m_V

y0 = m_V(ind) + (x0-x(ind1))./(x(ind1+1)-x(ind1)) .* (m_V(ind+1)-m_V(ind));


% golden section search function; searches for a MAX
function [argmax_gss, max_gss] = gss(f, a, b, tol)

psi = (1+sqrt(5))/2;
```

```
c    = b - (b-a)/psi;
d    = a + (b-a)/psi;
f_c = f(c);
f_d = f(d);

while b - a > tol

    if f_c > f_d

        b    = d;
        d    = c;
        c    = b - (b-a)/psi;
        f_d = f_c;
        f_c = f(c);
    else

        a    = c;
        c    = d;
        d    = a + (b-a)/psi;
        f_c = f_d;
        f_d = f(d);
    end
end

argmax_gss  = (a+b)/2;
max_gss     = f(argmax_gss);
```

7.2.7 tauchen.m: Approximate Stochastic Processes

(written by J. Adda)

```
function [prob,eps,z]=tauchen(N,mu,ro,sig);
% function written down by Adda
% Discretizes an AR(1) process into a Markov chain. Determines the optimal grid
% and transition matrix. Based on Tauchen (1991).
%
%  y(t)= mu(1-ro) + ro*y(t-1) + u(t)       with    V(u) = sig^2
%
% syntax:
%
% [prob,eps,z]=tauchen(N,mu,ro,sig)
%
% N is the number of points on the grid.
% prob is the transition matrix
% eps contains the cut-off points from - infty to + infty
% z are the grid points, i.e. the conditional mean within [eps(i),eps(i+1)].
```

```
global mu_ ro_ sigEps_ sig_ eps_ jindx_

if N==1;prob=1; eps=mu;z=mu;
else;
    if ro==0;
        sigEps=sig;
        eps=repmat(sigEps,[1 N+1]).*repmat(norminv((0:N)/N),size(sigEps))+mu;
        eps(:,1)=-20*sigEps+mu;
        eps(:,N+1)=20*sigEps+mu;
        aux=(eps-mu)./repmat(sigEps,[1 N+1]);
        aux1=aux(:,1:end-1);
        aux2=aux(:,2:end);
        z=N*repmat(sigEps,[1 N]).*(normpdf(aux1)-normpdf(aux2))+mu;
        prob=ones(N,N)/N;
    else;
        sigEps=sig/sqrt(1-ro^2);
        eps=sigEps*norminv((0:N)/N)+mu;
        eps(1)=-20*sigEps+mu;
        eps(N+1)=20*sigEps+mu;
        aux=(eps-mu)/sigEps;
        aux1=aux(1:end-1);
        aux2=aux(2:end);
        z=N*sigEps*(normpdf(aux1)-normpdf(aux2))+mu;
        mu_=mu;ro_=ro;sigEps_=sigEps;eps_=eps;sig_=sig;
        prob=zeros(N,N);

        for i=1:N
            for jindx_=1:N
                prob(i,jindx_)=quadl(@integ3,eps_(i),eps_(i+1),1e-6)*N;
            end
        end

    end
 z=z';
 eps=eps';
end


function F=integ3(u);

global mu_ ro_ sigEps_ eps_ jindx_ sig_
aux1=(eps_(jindx_)-mu_*(1-ro_)-ro_*u)/sig_;
aux2=(eps_(jindx_+1)-mu_*(1-ro_)-ro_*u)/sig_;
F=(normcdf(aux2)-normcdf(aux1));
F=F.*exp(-0.5*(u-mu_).*(u-mu_)/sigEps_^2);
```

```
pi=4*atan(1);
F=F/sqrt(2*pi*sigEps_^2);
```