Minimum Consumption and Risky Investments.

Following Bridge Loan, Uncertainty and Risky Investment, we now incorporate minimum consumption into the three period risky asset choice problem.

Table of Contents

Risky Asset Three Period Model with Rollover Loans

We have again the standard budget constraint (with standard rollover budgets), but now we include minimum consumption. Without minimum consumption, the model is the same as the one in Bridge Loan, Uncertainty and Risky Investment, see that file for details. The modified problem is:

$$\max_{K\geq 0, b_2} \left\{ \log(c_1) + \sum_{s\in\{l,h\}} p_s \cdot \log(\widehat{c_{2,s}}) + \sum_{s\in\{l,h\}} \sum_{\hat{s}\in\{l,h\}} p_s \cdot p_{\hat{s}} \cdot \log(\widehat{c_{3,s}}) \right\}$$

$$c_1 + b_2 + K = e_1$$

$$c_2 + b_3(b_2, s) = Y_{2,s}(K) + b_2$$

$$\text{and, } c_3 = Y_{3,s}(K) + \widehat{b_3}(b_2, s)$$

$$\text{and } \textit{Minimum Consumption}\text{: } \widehat{c} = \begin{cases} c, \text{ if } c \geq c_{\min} \\ c_{min}, \text{ if } c < c_{\min} \end{cases}$$

$$\textit{Borrowing Constraint}\text{: } b \geq \overline{b}$$

$$\text{and } \textit{Debt Continuation Condition}\text{: } \widehat{b} = \begin{cases} b, \text{ if } c \geq c_{\min} \\ b + (c - c_{min}), \text{ if } c < c_{\min} \end{cases}$$

We modify the production function a little bit so that it is now decreasing return to scale.

Note that in this example:

- Consumption in the first period is just c_1, there is no endowment in the first period, so this consumption is financed by borrowing. Borrowing allows for consumption as well as investments in the first period.

- Default is now possible in the second and third periods with $c_{\min}$.
- The borrowing constraint and debt continuation condition restricts b_3 so that one does not borrow or save infinitely. See Minimum Consumption and Borrowing. The debt continuation condition has no impact on borrowing given the rollover possibility here.

Solving Prep 1--Define Parameters

We can solve the model as we did before in Bridge Loan, Uncertainty and Risky Investment.

First define parameters, the addition is the borrowing bound and savings bounds, which are required now.

```
% 1. Define Parameters

% 1.1 Parmaeters
e23 = 1;
[e1, e2, e3] = deal(0, e23, e23);
[A, alpha] = deal(3, 0.36);
% p=0.5; % Probability of the Low State

% 1.2 Min Max Vectors
[cmin_min, cmin_max, cmin_n] = deal(0, 10e-3, 5);
% cmin_vector = [0 logspace(log10(cmin_min), log10(cmin_max), cmin_n)];
cmin_vector = linspace(cmin_min, cmin_max, cmin_n);
% [0 logspace(log10(cmin_min), log10(cmin_max), cmin_n)]
[p_min, p_max, p_n] = deal(0.05, 0.50, 5);
p_vector = linspace(p_min, p_max, p_n);

% 1.3 Borrow Bounds for Cmin
[borr_bd, save_bd, k_bd] = deal(-6, 6, 6);
```

Solving Prep 2--Define Functions

Define Various Functions. For the simple model here, we can explicitly define a consumption function that is an indicator function. This indicator function is what distinguishes the problem here from the problem in Bridge Loan, Uncertainty and Risky Investment.

Now, the maximization problem is no longer constrained by the natural borrowing constraint, households are willing to borrow/invest more when the chance of the low shock states happening are low.

The utility function is parameterized by x, p and cmin. We will maximize over four choices:

- x(1) = k, optimal risky capital choice
- x(2) = b2, first period borrowing
- x(3) = second period borrowing if second period shock is low
- x(4) = second period borrowing if second period shock is high

```
% 2. Define Various Functions

% 2.1.a Consumption function with cmin
fc = @(c, cmin) (c).*(c >= cmin) + (cmin).*(c < cmin);
% 2.1.b Debt, unpaid
funpaid = @(c, cmin) 0 + (c - cmin).*(c < cmin);
```

```
% 2.2 Utility function with Cmin Indicator Function
U_roll_xpcmin = @(x, p, cmin) ...
     (-1)*(log(fc(e1 - x(2) - x(1), cmin)) + ...
           p*log(fc(e2 + x(2) - x(3), cmin)) + ...
           (1-p)*log(fc(A*x(1)^alpha + e2 + x(2) - x(4), cmin)) + ...
           (p)*(p)*log(fc(e3 + x(3), cmin)) +...
           (p)*(1-p)*log(fc(A*x(1)^alpha + e3 + x(3), cmin)) +...
           (1-p)*(p)*log(fc(e3 + x(4), cmin)) +...
           (1-p)*(1-p)*log(fc(A*x(1)^alpha + e3 + x(4), cmin)));

% 2.3. Consumption Functions given optimal choices: not c but c_hat
% out: c1_o;c2_l_o;c2_h_o;c3_ll_o;c3_lh_o;c3_hl_o;c3_hh_o
% input: b3L_B_o, b3H_B_o are included but they are zero here
fchat_vec = @(K_o, b2_o, b3L_o, b3H_o, b3L_B_o, b3H_B_o) ...
            [e1 - b2_o - K_o;
             e2 + b2_o - b3L_o;
             e2 + A*K_o^alpha + b2_o - b3H_o;
             e3 + b3L_o;
             e3 + A*K_o^alpha + b3L_o;
             e3 + b3H_o;
             e3 + A*K_o^alpha + b3H_o;];
```

Solving Prep 3--Constraints and Bounds

```
% 3. Define Constraints

% 3.1. Cmin requires both savings and borrowing bounds
% Four columns are for K, b2, b3H, b3L
% First row is K >= 0, next 3, borrow bound, next 3 save bounds
LIN_CON = [-1, 0, 0, 0;
            0,-1, 0, 0;
            0, 0,-1, 0;
            0, 0, 0,-1;
            1, 0, 0, 0;
            0, 1, 0, 0;
            0, 0, 1, 0;
            0, 0, 0, 1;];

% 3.2 Bounds
lb = [0, borr_bd, borr_bd, borr_bd];
ub = [k_bd, save_bd, save_bd, save_bd];
q = [-lb, ub]';

% 3.3. Starting Values
b0 = [0.1,-0.15,-0.15,-0.15];
```

Solving Set A--MultiStart with FminCon

An important point for solving problems with $c_{\min}$ as demonstrated in , is that $c_{\min}$ introduces multiple local minimizers as shown in Minimum Consumption and Borrowing. The optimal choice when $c_{\min}$ was 0 is still a local optimal choice, but there is now an additional optimal choice at the higher range of borrowing levels that are now possible because $c_{\min} > 0$. Furthermore, as p changes, the global optimal choice might corner solutions at the borrowing/savings bounds.

To deal with this, rather than using fmincon as before, we will use fmincon with multi-start, otherwise the optimization routine might not find the global optimal solution.

Invoke FMINCON via ***MULTISTART***. Calling functions written at the end of this file.

```
% 4. Minimization
model_name = 'rollover_cmin';
argmin_store = multistart_argmin(model_name, p_vector, cmin_vector, ...
                                 U_roll_xpcmin, LIN_CON, ...
                                 q, lb, ub, b0, 1000);
```

Solving Set A--Results

Below, I show optimal choices, looping over 5 values of p and 5 values of $c_{\min}$. The results show:

1. As p decreases, optimal risky capital investment increases and borrowing increases.
2. Households borrow significantly more when $c_{\min} > 0$ then when $c_{\min} = 0$.
3. At the lowest level of p with $c_{\min} > 0$, households default after receiving two bad shocks (in the second and third period), or after receiving a good shock in the first period and then receiving a bad shock in the second period.
4. At higher levels of p, with reduced levels of borrowing, households only default after two bad shocks in a row.
5. For each p, there is a treshold level of $c_{\min}$. If $c_{\min} \leq c_{\min}^{*\text{threshold}}(p)$ below this level, then there is no difference in optimal choices compared to when $c_{\min} = 0$. If $c_{\min}^{*\text{corner}}(p) \geq c_{\min} > c_{\min}^{*\text{threshold}}(p)$, however, households switch to default when hit by bad shock in the third period. If $c_{\min} > c_{\min}^{*\text{corner}}(p)$, household borrowing choices go up to corner solutions, where the borrowing/savings constraint bounds are, and default.

For the Tables below:

1. **K_o**: Optimal capital choice (made in t=1)
2. **E_b:** Expected total borrowing over two periods (see *Function--Table Making Tool*)
3. **E_owe_f:** Expected fraction of debt paid back (see *Function--Table Making Tool*)
4. **b2_o**: Optimal 1st period borrowing
5. **b3L_o**: Optimal 2nd period, when shock is low.
6. **b3H_o**: Optimal 2nd period, when shock is high.
7. **EV**: expected value given optimal choices.

```
% 5. Output Results To Table
[asset_choices, consumption_choices] = ...
    argmin2table(model_name, p_vector, cmin_vector,...
                 fchat_vec, funpaid, argmin_store);
disp(asset_choices);
```

	K_o	E_b	E_owe_f	b2_o	b3L_o	b3H_o	b3L_B_o	b3H_B
rollover_cmin p:0.050000 cmin:0	0.77	-1.89	0	-1.89	-0.99	-0.57	0	0
rollover_cmin p:0.050000 cmin:0.0025	2.82	-5.2	0.02	-5.2	-4.8	-2.67	0	0
rollover_cmin p:0.050000 cmin:0.005	2.82	-5.2	0.02	-5.2	-4.8	-2.67	0	0
rollover_cmin p:0.050000 cmin:0.0075	2.82	-5.2	0.02	-5.2	-4.8	-2.67	0	0
rollover_cmin p:0.050000 cmin:0.01	2.82	-5.2	0.02	-5.2	-4.8	-2.67	0	0
rollover_cmin p:0.162500 cmin:0	0.63	-1.66	0	-1.66	-0.95	-0.25	0	0
rollover_cmin p:0.162500 cmin:0.0025	1.67	-3.39	0.02	-3.35	-3.58	-0.54	0	0
rollover_cmin p:0.162500 cmin:0.005	1.67	-3.39	0.02	-3.35	-3.58	-0.54	0	0
rollover_cmin p:0.162500 cmin:0.0075	1.67	-3.39	0.02	-3.35	-3.58	-0.54	0	0

rollover_cmin p:0.162500 cmin:0.01	1.67	-3.39	0.02	-3.35	-3.58	-0.54	0	0
rollover_cmin p:0.275000 cmin:0	0.5	-1.46	0	-1.46	-0.87	-0.06	0	0
rollover_cmin p:0.275000 cmin:0.0025	1.51	-3.21	0.06	-3.1	-3.48	-0.33	0	0
rollover_cmin p:0.275000 cmin:0.005	1.51	-3.21	0.06	-3.1	-3.48	-0.33	0	0
rollover_cmin p:0.275000 cmin:0.0075	1.51	-3.21	0.06	-3.1	-3.48	-0.33	0	0
rollover_cmin p:0.275000 cmin:0.01	1.51	-3.21	0.06	-3.1	-3.48	-0.33	0	0
rollover_cmin p:0.387500 cmin:0	0.38	-1.28	0	-1.28	-0.78	0.06	0	0
rollover_cmin p:0.387500 cmin:0.0025	1.38	-3.14	0.12	-2.95	-3.45	-0.19	0	0
rollover_cmin p:0.387500 cmin:0.005	1.38	-3.14	0.12	-2.95	-3.45	-0.19	0	0
rollover_cmin p:0.387500 cmin:0.0075	1.38	-3.14	0.12	-2.95	-3.45	-0.19	0	0
rollover_cmin p:0.387500 cmin:0.01	1.38	-3.14	0.12	-2.95	-3.45	-0.19	0	0
rollover_cmin p:0.500000 cmin:0	0.29	-1.12	0	-1.12	-0.68	0.14	0	0
rollover_cmin p:0.500000 cmin:0.0025	0.29	-1.12	0	-1.12	-0.68	0.14	0	0
rollover_cmin p:0.500000 cmin:0.005	1.3	-3.17	0.2	-2.86	-3.48	-0.08	0	0
rollover_cmin p:0.500000 cmin:0.0075	0.29	-1.12	0	-1.12	-0.68	0.14	0	0
rollover_cmin p:0.500000 cmin:0.01	1.3	-3.17	0.2	-2.86	-3.48	-0.08	0	0

```
disp(consumption_choices);
```

	c1_o	c2_l_o	c3_ll_o	c3_lh_o	c2_h_o	c3_hl_o	c3_hh_o
rollover_cmin p:0.050000 cmin:0	1.12	0.1	0.01	2.74	2.4	0.43	3.16
rollover_cmin p:0.050000 cmin:0.0025	2.38	0.59	-3.8	0.56	2.83	-1.67	2.69
rollover_cmin p:0.050000 cmin:0.005	2.38	0.59	-3.8	0.56	2.83	-1.67	2.69
rollover_cmin p:0.050000 cmin:0.0075	2.38	0.59	-3.8	0.56	2.83	-1.67	2.69
rollover_cmin p:0.050000 cmin:0.01	2.38	0.59	-3.8	0.56	2.83	-1.67	2.69
rollover_cmin p:0.162500 cmin:0	1.04	0.29	0.05	2.59	2.12	0.75	3.28
rollover_cmin p:0.162500 cmin:0.0025	1.67	1.23	-2.58	1.03	1.8	0.46	4.08
rollover_cmin p:0.162500 cmin:0.005	1.67	1.23	-2.58	1.03	1.8	0.46	4.08
rollover_cmin p:0.162500 cmin:0.0075	1.67	1.23	-2.58	1.03	1.8	0.46	4.08
rollover_cmin p:0.162500 cmin:0.01	1.67	1.23	-2.58	1.03	1.8	0.46	4.08
rollover_cmin p:0.275000 cmin:0	0.96	0.41	0.13	2.46	1.94	0.94	3.27
rollover_cmin p:0.275000 cmin:0.0025	1.6	1.37	-2.48	1	1.7	0.67	4.14
rollover_cmin p:0.275000 cmin:0.005	1.6	1.37	-2.48	1	1.7	0.67	4.14
rollover_cmin p:0.275000 cmin:0.0075	1.6	1.37	-2.48	1	1.7	0.67	4.14
rollover_cmin p:0.275000 cmin:0.01	1.6	1.37	-2.48	1	1.7	0.67	4.14
rollover_cmin p:0.387500 cmin:0	0.89	0.5	0.22	2.35	1.79	1.06	3.18
rollover_cmin p:0.387500 cmin:0.0025	1.57	1.5	-2.45	0.92	1.61	0.81	4.19
rollover_cmin p:0.387500 cmin:0.005	1.57	1.5	-2.45	0.92	1.61	0.81	4.19
rollover_cmin p:0.387500 cmin:0.0075	1.57	1.5	-2.45	0.92	1.61	0.81	4.19
rollover_cmin p:0.387500 cmin:0.01	1.57	1.5	-2.45	0.92	1.61	0.81	4.19
rollover_cmin p:0.500000 cmin:0	0.84	0.56	0.32	2.23	1.65	1.14	3.05
rollover_cmin p:0.500000 cmin:0.0025	0.84	0.56	0.32	2.23	1.65	1.14	3.05
rollover_cmin p:0.500000 cmin:0.005	1.56	1.62	-2.48	0.81	1.51	0.92	4.21
rollover_cmin p:0.500000 cmin:0.0075	0.84	0.56	0.32	2.23	1.65	1.14	3.05
rollover_cmin p:0.500000 cmin:0.01	1.56	1.62	-2.48	0.81	1.51	0.92	4.21

Solving Set B--Solve at many P, for cmin=0 and cmin=0.01

To graphically show the effect of cmin, we will now solve the model along a vector of p. Given the parameters here, at $c_{\min} = 0.01$, how do optimal choices change as the chance for the bad state happening decreases.

The pattern here is the same as what was summarized above for Set A. We just have a more finely solved grid of p, and can show the results visually below.

One caveat here is there for higher p values many multi-start points are needed to correctly capture the optimal choices using the code included below. This code is for demonstration purposes. In the fully implementation of the model, we will not be using fmincon.

The benefit of the code here is that we can solve the problem here exactly without any approximations.

```
cmin_vector = [0, 0.01];
[p_min, p_max, p_n] = deal(0.509, 0.511, 6);
p_vector = linspace(p_min, p_max, p_n);
% p_vector = [p_vector, 0.60, 0.55, 0.45];
model_name = 'rollover_cmin';
tic;

% Found threshold where default go from 0 to 1 to in 2 states
p_default_1_ll_lower = 0.509;
% p_default_1_ll = 0.5105;
p_default_1_ll = p_default_1_ll_lower;
p_default_2_ll_hl = 0.0837;

p_vector_one = linspace(0.01, p_default_2_ll_hl-0.0001, 50);
p_vector_two = linspace(p_default_2_ll_hl+0.0001, 0.510, 50);
p_vector_three = linspace(0.511, 0.99, 40);

p_vector_seg1 = [p_vector_one, p_vector_two];
% ms_count = 5;
ms_count = 10000;
argmin_store_seg1 = multistart_argmin(model_name, p_vector_seg1, cmin_vector, ...
                                U_roll_xpcmin, LIN_CON, ...
                                q, lb, ub, b0, ms_count);

p_vector_seg2 = [p_vector_three];
% ms_count = 5;
ms_count = 100000;
argmin_store_seg2 = multistart_argmin(model_name, p_vector_seg2, cmin_vector, ...
                                U_roll_xpcmin, LIN_CON, ...
                                q, lb, ub, b0, ms_count);

% Combine Tables
argmin_store = vertcat(argmin_store_seg1, argmin_store_seg2);

toc;
```

```
Elapsed time is 23397.129165 seconds.
```

Solving Set B--Results into Tables and Matrixes

```
% Generate Statistics from Results
p_vector = [p_vector_seg1, p_vector_seg2];
[asset_choices, consumption_choices] = ...
    argmin2table(model_name, p_vector, cmin_vector,...
                 fchat_vec, funpaid, argmin_store);

% Show Tabular
% disp(asset_choices);
% disp(consumption_choices);

% Graph out results
K_o_choices =  asset_choices{:,'K_o'};
b2_o_choices =  asset_choices{:,'b2_o'};
E_b_choices =  asset_choices{:,'E_b'};
E_owe_f_choices =  asset_choices{:,'E_owe_f'};
```

```
% K and b2
K_o_mat = reshape(K_o_choices, length(cmin_vector), length(p_vector))';
b2_o_mat = reshape(b2_o_choices, length(cmin_vector), length(p_vector))';
E_b_o_mat = reshape(E_b_choices, length(cmin_vector), length(p_vector))';
E_owe_f_o_mat = reshape(E_owe_f_choices, length(cmin_vector), length(p_vector))';

% Subseting
p_vector_max_threshold = 0.8;
```

Solving Set B--Graphs for K, E(B), E(unpaid debt fraction)

Below we generate graphs for optimal capital choices, expected total borrowing, and expected default unpaid fraction due to default

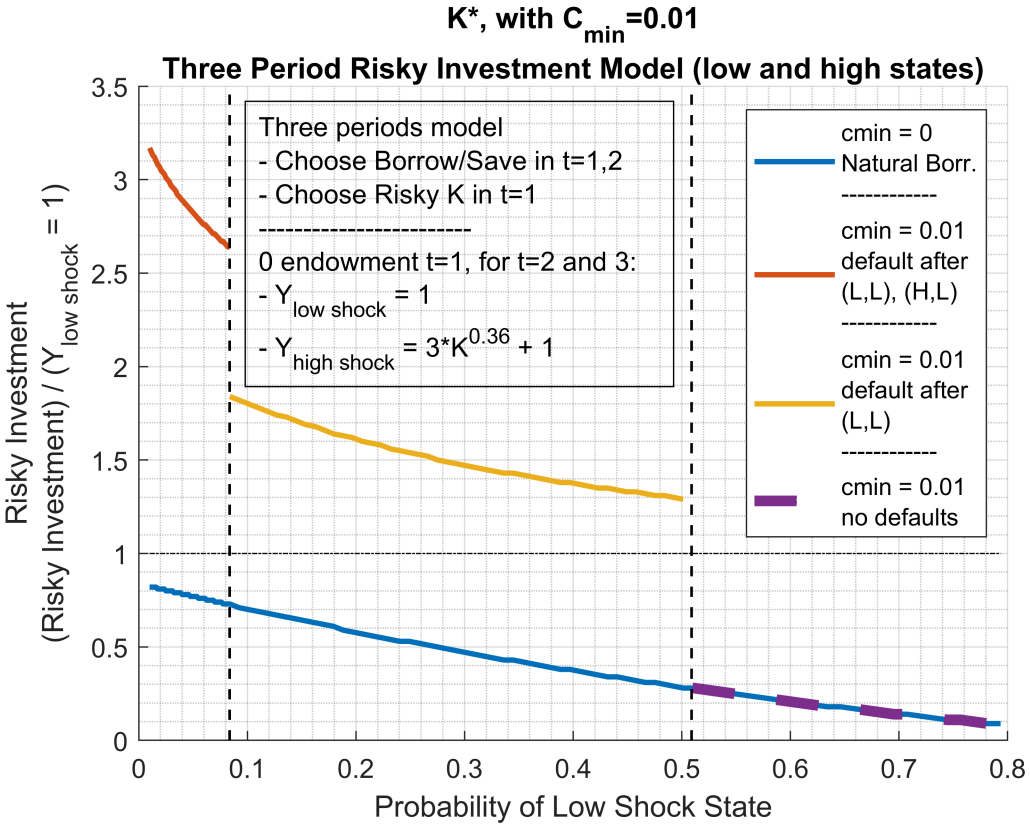
Each plot has:

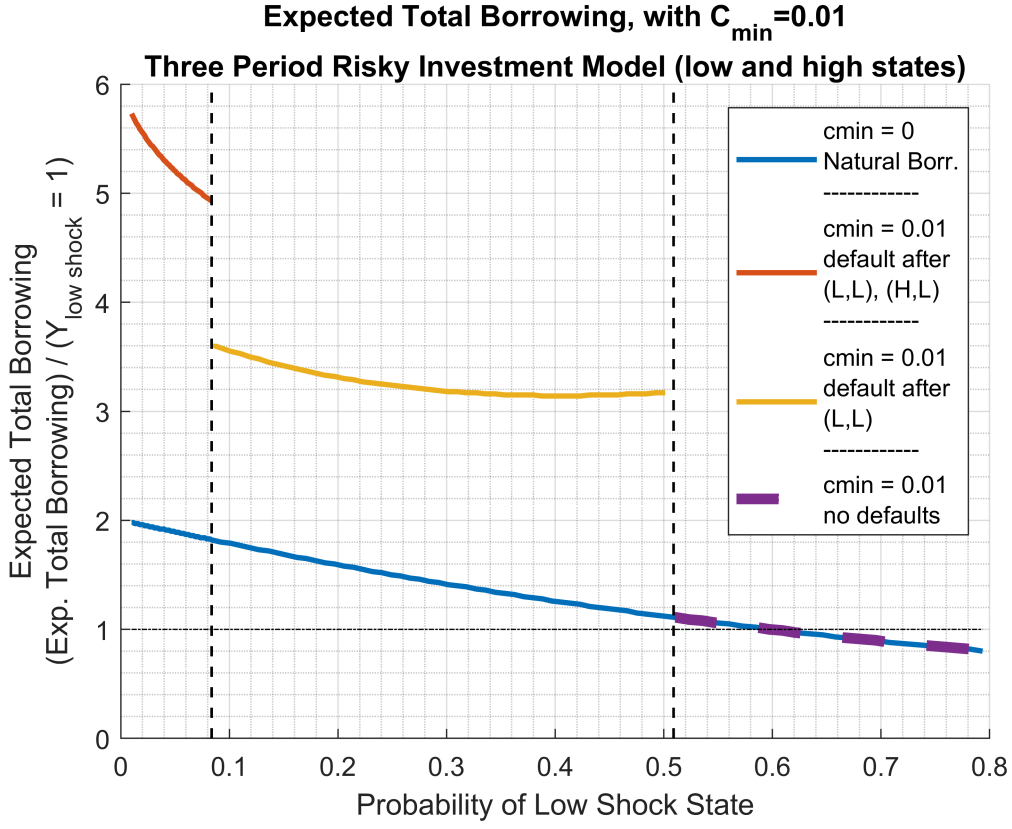
- **Blue** line: marking out result when $c_{\min} = 0$, the results here are constrained by the non-binding natural borrowing constraint.
- **Red** segment line: marking out result when $c_{\min} = 0.01$ and $p < 0.083$. Households see the chance of low state happening as very low and are willing to borrow a lot to finance more risky investment. Due to the high level of borrowing, for the low chance event that the bad state happens, they will default in the 3rd period, both when the second period had high shock as well as when the 2nd period had low shock. The overall expected debt that will be defaulted on, despite defaulting in both bad shock 3rd period states, is low because p is low.
- **Orange** segment line: marking out results when $c_{\min} = 0.01$ and $0.51 < p < 0.083$. Given our parameters here, this is the range of p value when households also borrow more than what they would have borrowed given the blue line (with the natural borrowing constraint), but now p is sufficiently high that they are worried about the low utility when they have to consume $c_{\min}$. Given the more moderate amount of borrowing, households here only default after getting two bad shocks in the third period. They will not default in the bad state in the third period if they faced a good shock in the second period.
- **Purple Dashed** semgent line: marking out results when $c_{\min} = 0.01$ and $0.51 > p$. At these ranges, the chance of the bad state happening is pretty high. Households want to avoid the very high possibility of consuming only $c_{\min}$ if they borrow too much. The borrowing here, even though $c_{\min} = 0.01$ is the same as borrowing when $c_{\min} = 0$.

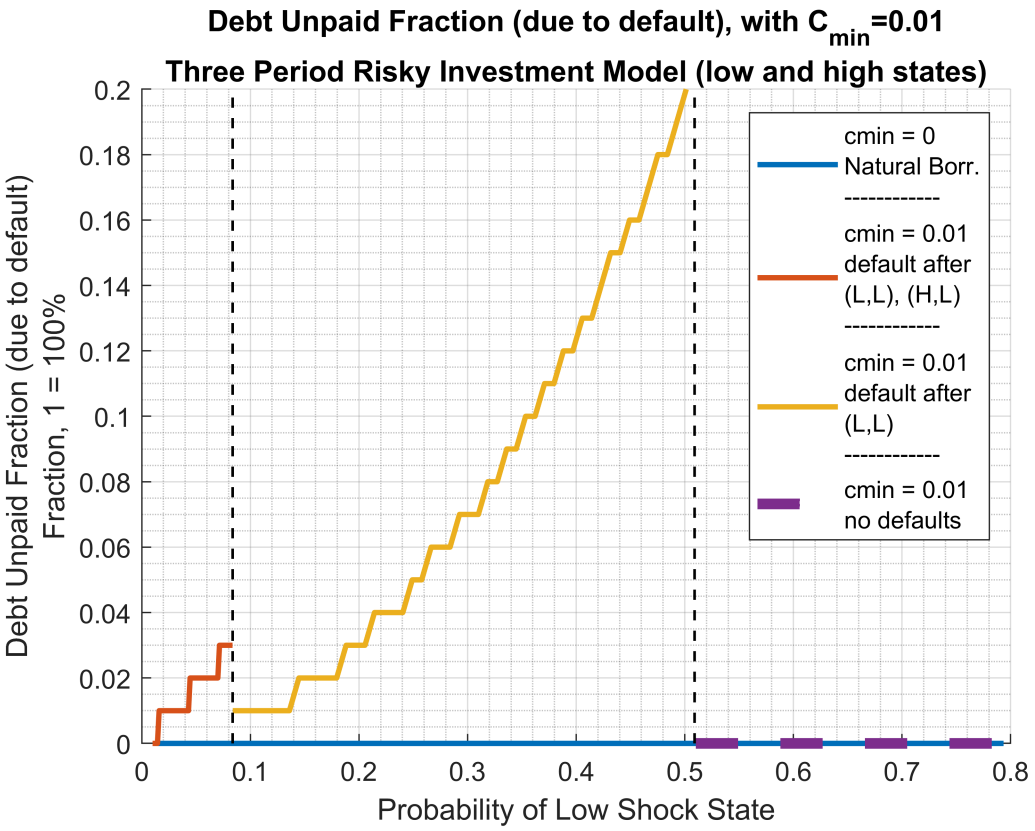
```
subtitle = 'Three Period Risky Investment Model (low and high states)';
xlabeltext = 'Probability of Low Shock State';
[E_b_o_y_title, E_owe_f_y_title] = ...
                  deal('Expected Total Borrowing',...
                       'Exp. Debt Unpaid Fraction (Due to Default)');

for mat_i=[1,3,4]

    % Process each matrix separately
    if (mat_i == 1)
        cur_mat = K_o_mat;
        title_var_short = 'K*';
        title_var_long = 'Risky Investment';
        title_var_long_L2 = '(Risky Investment) / (Y_{low shock} = 1)';
        legend_position = 'northeast';
    elseif (mat_i == 2)
        cur_mat = b2_o_mat;
        title_var_short = 'B2*';
        title_var_long = 'T1 Borrowing';
        title_var_long_L2 = '(T1 Borrowing) / (Y_{low shock} = 1)';
```

```
        legend_position = 'southeast';
    elseif (mat_i == 3)
        cur_mat = E_b_o_mat*(-1);
        title_var_short = E_b_o_y_title;
        title_var_long = E_b_o_y_title;
        title_var_long_L2 = '(Exp. Total Borrowing) / (Y_{low shock} = 1)';
        legend_position = 'northeast';
    elseif (mat_i == 4)
        cur_mat = E_owe_f_o_mat;
        title_var_short = E_owe_f_y_title;
        title_var_long = E_owe_f_y_title;
        title_var_long_L2 = 'Fraction, 1 = 100%';
        legend_position = 'northeast';
    end

    % Figure
    figure();
    hold on;
    % Main Plot
    legends = {};
    legend_ctr = 0;
    for cmin_i=1:length(cmin_vector)
        % cmin
        cmin = cmin_vector(cmin_i);
        % Graph main
        p_vec_use_idx = (p_vector < p_vector_max_threshold);
        if (cmin == 0)
           plot(p_vector(p_vec_use_idx) , ...
                cur_mat(p_vec_use_idx, cmin_i)/e23, ...
                '-', 'LineWidth', 2);
            % Collect Legends
            legend_ctr = legend_ctr + 1;
            legends{legend_ctr} = [sprintf(['cmin = %s\n' ...
                'Natural Borr.\n------------'],num2str(cmin))];
        else

           d_ll_hl_idx = (p_vector < p_default_2_ll_hl);
           d_ll_idx = ((p_vector > p_default_2_ll_hl) & ...
                          (p_vector < p_default_1_ll_lower));
           d_none_idx = ((p_vector > p_default_1_ll) & ...
                          (p_vector < p_vector_max_threshold));

           plot(p_vector(d_ll_hl_idx), cur_mat(d_ll_hl_idx, cmin_i)/e23, ...
                '-', 'LineWidth', 2);
           legend_ctr = legend_ctr + 1;
           legends{legend_ctr} = [sprintf(['cmin = %s\n' ...
               'default after\n(L,L), (H,L)\n------------'],num2str(cmin))];

           plot(p_vector(d_ll_idx), cur_mat(d_ll_idx, cmin_i)/e23, ...
                '-', 'LineWidth', 2);
           legend_ctr = legend_ctr + 1;
           legends{legend_ctr} = [sprintf(['cmin = %s\n' ...
               'default after\n(L,L)\n------------'],num2str(cmin))];

           plot(p_vector(d_none_idx), cur_mat(d_none_idx, cmin_i)/e23, ...
                '--', 'LineWidth', 4);
           legend_ctr = legend_ctr + 1;
           legends{legend_ctr} = [sprintf(['cmin = %s\n' ...
```

```
                'no defaults'], num2str(cmin))];

        end
    end

    % Horizontal Endowment Line
    if (mat_i <= 3)
        plot(p_vector(p_vec_use_idx), ones(size(p_vector(p_vec_use_idx))) * 1, 'k-.');
    end
    if (mat_i == 1)
        dim = [0.225 0.87 0.1 0];
        str = [sprintf(['Three periods model\n' ...
                        '- Choose Borrow/Save in t=1,2\n' ...
                        '- Choose Risky K in t=1\n' ...
                        '-----------------------\n' ...
                        '0 endowment t=1, for t=2 and 3: \n' ...
                        '- Y_{low shock} = 1\n' ...
                        '- Y_{high shock} = %s*K^{%s} + 1'], ...
                         num2str(A), num2str(alpha))];
        annotation('textbox',dim,'String',str,'FitBoxToText','on');
    end

    % Vertical Shift in Default Lines: LL
    y_lim = get(gca,'YLim');
    plot(p_default_1_ll*ones(1,10), ...
        linspace(y_lim(1), y_lim(2),10), ...
        'k--', 'LineWidth', 1);

    % Vertical Shift in Default Lines: LL and HL
    y_lim = get(gca,'YLim');
    plot(p_default_2_ll_hl*ones(1,10), ...
        linspace(y_lim(1), y_lim(2),10), ...
        'k--', 'LineWidth', 1);

    % Titling
    title({[title_var_short ', with C_{min}=' num2str(cmin)], [subtitle]});
    ylabel({[title_var_long],...
            [title_var_long_L2]});
    xlabel(xlabeltext);
    legend(legends, 'Location', legend_position);
    % Grids Axis etc
    grid on;
    grid minor;
end
```


K*, with C min =0.01
Three Period Risky Investment Model (low and high states)
Three periods model
- Choose Borrow/Save in t=1,2
- Choose Risky K in t=1
0 endowment t=1, for t=2 and 3:
- Y low shock = 1
- Y high shock = 3*K^0.36 + 1
cmin = 0
Natural Borr.
cmin = 0.01
default after
(L,L), (H,L)
cmin = 0.01
default after
(L,L)
cmin = 0.01
no defaults
Risky Investment
(Risky Investment) / (Y low shock = 1)
Probability of Low Shock State

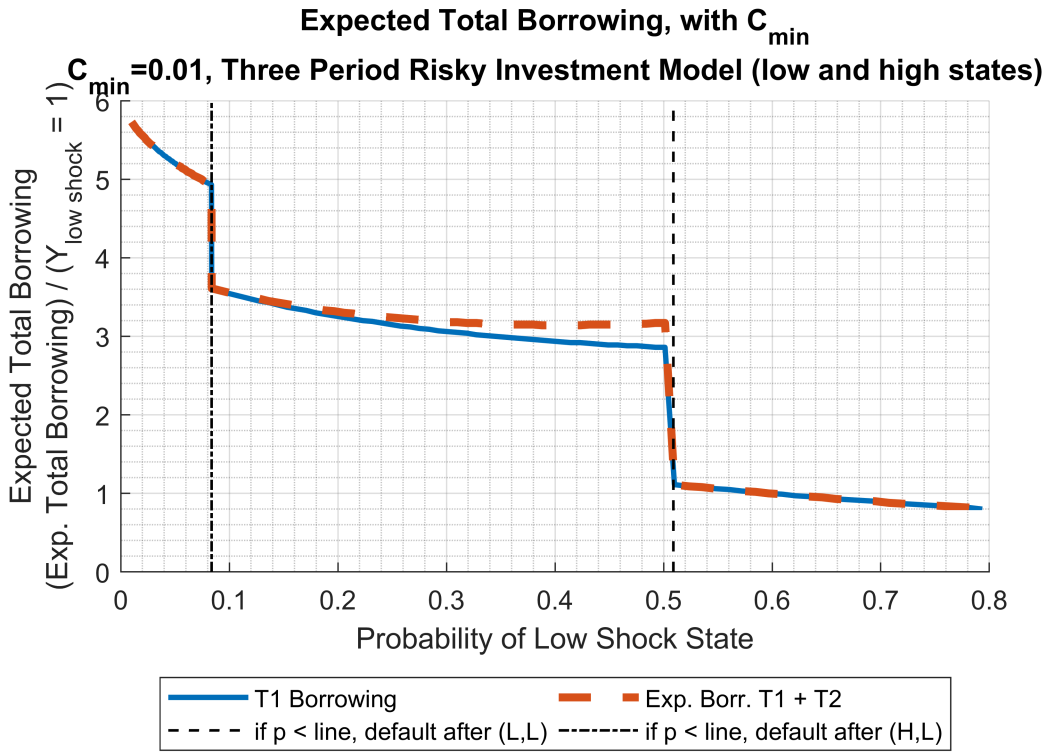

Expected Total Borrowing, with C min =0.01
Three Period Risky Investment Model (low and high states)
cmin = 0
Natural Borr.
cmin = 0.01
default after
(L,L), (H,L)
cmin = 0.01
default after
(L,L)
cmin = 0.01
no defaults
Expected Total Borrowing
(Exp. Total Borrowing) / (Y low shock = 1)
Probability of Low Shock State



Solving Set B--Two Joint Plots

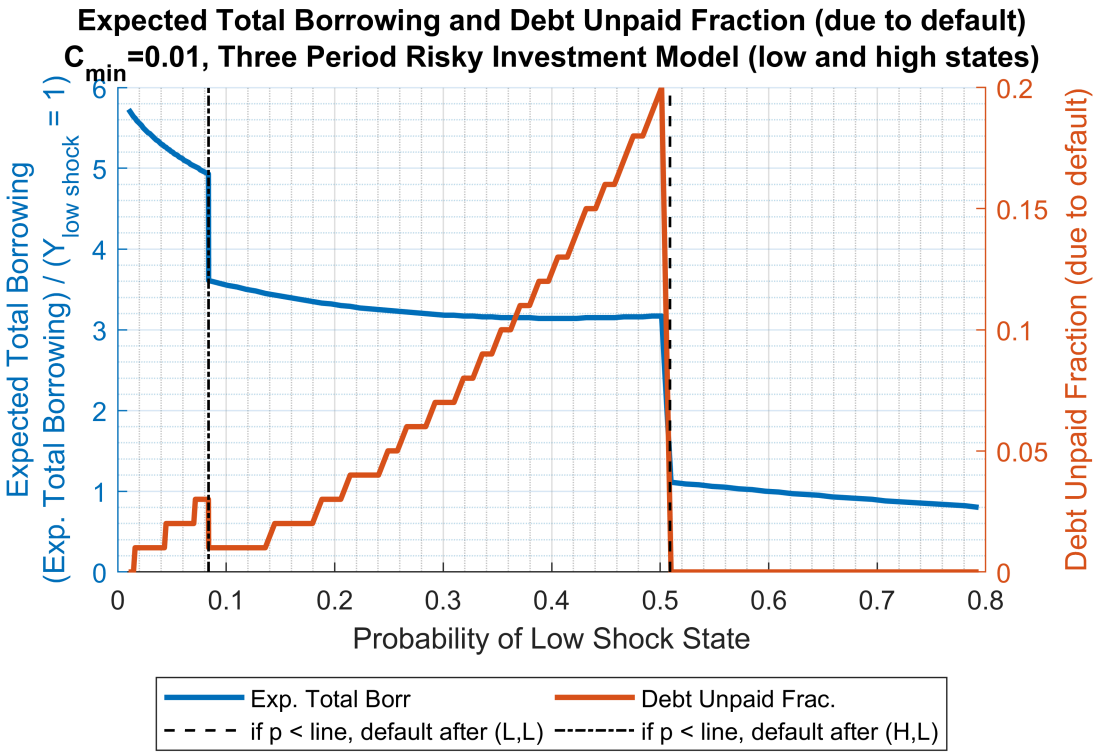
Below, we plot two joint plots.

```
figure();
hold on;
% Main Lines
plot(p_vector(p_vec_use_idx), -1*b2_o_mat(p_vec_use_idx,2), '-', 'LineWidth', 2);
plot(p_vector(p_vec_use_idx), -1*E_b_o_mat(p_vec_use_idx,2), '--', 'LineWidth', 3);
% Default Lines
y_lim = get(gca,'YLim');
plot(p_default_1_ll*ones(1,10), ...
    linspace(y_lim(1), y_lim(2),10), ...
    'k--', 'LineWidth', 1);
y_lim = get(gca,'YLim');
plot(p_default_2_ll_hl*ones(1,10), ...
    linspace(y_lim(1), y_lim(2),10), ...
    'k-.', 'LineWidth', 1);
% Legends Etc
title({[E_b_o_y_title ', with C_{min}'], ['C_{min}=' num2str(cmin) ', ' subtitle]});
ylabel({[E_b_o_y_title], ['(Exp. Total Borrowing) / (Y_{low shock} = 1)']});
xlabel(xlabeltext);
legend({['T1 Borrowing'],['Exp. Borr. T1 + T2'],...
        ['if p < line, default after (L,L)'],...
        ['if p < line, default after (H,L)']},...
        'Location', 'southoutside', 'Orientation','horizontal', 'NumColumns', 2);
grid on;
grid minor;
```




```
figure();
hold on

% Main Lines
yyaxis left
plot(p_vector(p_vec_use_idx), (-1)*E_b_o_mat(p_vec_use_idx, 2), '-', 'LineWidth', 2);
ylabel({[E_b_o_y_title], ['(Exp. Total Borrowing) / (Y_{low shock} = 1)']});

% Main Lines
yyaxis right
plot(p_vector(p_vec_use_idx), E_owe_f_o_mat(p_vec_use_idx, 2), '-', 'LineWidth', 2);

% Default Lines
y_lim = get(gca,'YLim');
plot(p_default_1_ll*ones(1,10), ...
    linspace(y_lim(1), y_lim(2),10), ...
    'k--', 'LineWidth', 1);
y_lim = get(gca,'YLim');
plot(p_default_2_ll_hl*ones(1,10), ...
    linspace(y_lim(1), y_lim(2),10), ...
    'k-.', 'LineWidth', 1);

% More Legends
title({[E_b_o_y_title ' and ' E_owe_f_y_title], ['C_{min}=' num2str(cmin) ', ' subtitle]});
ylabel([E_owe_f_y_title]);
xlabel(xlabeltext);
legend({['Exp. Total Borr'], ['Debt Unpaid Frac.'],...
        ['if p < line, default after (L,L)'],...
        ['if p < line, default after (H,L)']}, ...
        'Location', 'southoutside', 'Orientation','horizontal', 'NumColumns', 2);
```

```
grid on;
grid minor;
```



Function--Multi-Start Minimization with Fmincon

This is the shared minimization Code

```
function [argmin_store] = ...
    multistart_argmin(model_name, p_vector, cmin_vector, ...
                      U_xpcmin, LIN_CON, q, lb, ub, b0, ms_count)

    argmin_store = zeros(length(cmin_vector)*length(p_vector), 7);
    store_ctr = 0;
    for p_i=1:length(p_vector)
        for cmin_i=1:length(cmin_vector)
            % A.1 Current CMIN value and P value
            cmin = cmin_vector(cmin_i);
            p = p_vector(p_i);

            % A.2 Evaluate Function
            U_roll_x = @(x) U_xpcmin(x, p, cmin);

            % A.3 Optimization have to use multi-start, because of multiple local minimums
            opts = optimoptions(@fmincon,'Algorithm','interior-point');
            m_problem = createOptimProblem('fmincon','objective', ...
                                           U_roll_x, 'x0', b0, ...
                                          'Aineq', LIN_CON, 'bineq', q, ...
                                          'lb', lb, 'ub', ub, ...
                                          'options',opts);
```

```
            if (ms_count >= 100)
                ms_f = MultiStart('Display', 'off', 'UseParallel', true);
            else
                ms_f = MultiStart('Display', 'off', 'UseParallel', false);
            end

            % A.4 Optimization Results
            [x_argmin, obj] = run(ms_f, m_problem, ms_count);
%             [x,fval,exitflag,output,solutions] = run(ms_f, m_problem, 20);
            % A.6 Store Results
            store_ctr = store_ctr + 1;

            if (strcmp(model_name, 'rollover_cmin') || ...
                strcmp(model_name, 'no_rlov_cmin'))
                [b3L_B_o, b3H_B_o] = deal(false, false);
                x_argmin = [x_argmin, b3L_B_o, b3H_B_o];
            end

            argmin_store(store_ctr, :) = [x_argmin, (-1)*obj];
        end
    end

end
```

Function--Table Making Tool

The function below takes in optimization results and saves them to a matlab table. This is used by functions above

```
function [asset_choices, consumption_choices] =...
    argmin2table(model_name, p_vector, cmin_vector, ...
                 fchat_vec, funpaid, argmin_store)

    % B.1 Output Results To Table
    asset_choices = [];
    consumption_choices = [];
    table_rows_all = {};
    graph_counter_all = 0;

    % B.2 Loop Over Results
    store_ctr = 0;
    for p_i=1:length(p_vector)
        for cmin_i=1:length(cmin_vector)

            % B.3 Current CMIN value
            cmin = cmin_vector(cmin_i);
            p = p_vector(p_i);

            % B.4 Deal Stored Optimal Choices
            store_ctr = store_ctr + 1;
            x_argmin_cells = num2cell(argmin_store(store_ctr, :));
            [K_o, b2_o, b3L_o, b3H_o, b3L_B_o, b3H_B_o, EV] = deal(x_argmin_cells{:});

            % B.5 Optimal choices to consumption choices
            c_out_cells = num2cell(fchat_vec(K_o, b2_o, b3L_o, b3H_o, b3L_B_o, b3H_B_o)');
            [c1_o, c2_l_o, c2_h_o, c3_ll_o, c3_lh_o, c3_hl_o, c3_hh_o] = deal(c_out_cells{:});

            % Total Debt
```

```
            borrow_L_path = min(b2_o, b3L_o);
            borrow_H_path = min(b2_o, b3H_o);
            % Borrowing choices made in T1 and T2, two possible paths
            E_b = p*borrow_L_path + (1-p)*borrow_H_path;

            % Total Debt not Paid
            % Four possible Path, Realizations are in the 3rd period
            E_unpaid = p*p*funpaid(c3_ll_o, cmin) +  ...
                       p*(1-p)*funpaid(c3_lh_o, cmin) + ...
                       (1-p)*p*funpaid(c3_hl_o, cmin) + ...
                       (1-p)*(1-p)*funpaid(c3_hh_o, cmin);
            E_owe_f = E_unpaid/E_b;

            % B.7 Optimal Assets and Consumptions Tables
            o_a = table(K_o, E_b, E_owe_f, b2_o, b3L_o, b3H_o, b3L_B_o, b3H_B_o, EV);
            o_c = table(c1_o, c2_l_o, c3_ll_o, c3_lh_o, c2_h_o, c3_hl_o, c3_hh_o);

            % B.8 Add Results to Tables
            graph_counter_all = graph_counter_all + 1;
            asset_choices = [asset_choices; o_a];
            consumption_choices = [consumption_choices; o_c];
            table_row_names_all{graph_counter_all} =...
                [model_name ' p:'  sprintf('%7.6f', p) ' cmin:'  num2str(cmin)];
        end
    end

    % Table Row Names and Decimal Setting
    asset_choices.Properties.RowNames = table_row_names_all;
    asset_choices.Variables =  round(asset_choices.Variables, 2);
    consumption_choices.Properties.RowNames = table_row_names_all;
    consumption_choices.Variables =  round(consumption_choices.Variables, 2);
end
```