FINAL – ECON 416

MICHAEL WOOLEY

This paper shows how to solve for the equilibrium path out of a bank run for the model in Gertler and Kiyotaki (2015). The solution method developed here can be described as "shooting backwards": beginning from a point near the steady state a candidate eqilibrium path can be solved for by working backwards in a recursive manner. A satisfactory path for a given run state asset price (Q^*) will be one that hits certain pre-determined equilibrium values at period 2, the period following the bank run. A complete solution will consist of a path and a Q^* that additionally satisfies a condition at the run period.

Section 1 reviews the model. The presentation tracks the exposition in Christiano, Dalgic, and Wen (2015) quite closely; readers familiar with the model can skip ahead without missing anything. Section 2 finds and discusses the features of the set of possible steady states, which are indexed by Q^*. The strategy for finding the equilibrium path is described in Section 3.1. Results are presented and discussed in Section 4.

1. Model

1.1. **Bankers.** There is a unit measure of bankers. At the start of each period $1-\sigma$ bankers are randomly chosen to exit. The remaining σ bankers survive and operate in period t. At the beginning of each period a measure $1-\sigma$ of new bankers enter. Therefore, the number of bankers is constant across periods.

1.1.1. *Banking Problem in No-Bank Run Period.* Individual bankers have net worth n_t and issue deposits d_t. They buy capital k_t^b at a price Q_t subject to the balance sheet constraint

$$Q_t k_t^b = n_t + d_t. \tag{1.1}$$

Banks pay a gross interest rate $\bar{R}_t$ on deposits d_t to households in period $t+1$.

At each period a banker may choose either to default or not to default.

Non-Defaulting Banks. Consider a banker who chooses not to default at t. Then his net worth at $t+1$ is

$$n_{t+1} = \begin{cases} Q_t k_t^b (Z_{t+1} + Q_{t+1}) - \bar{R}_t d_t & \text{If no bank run in } t+1. \\ Q_t k_t^b \left(Z_{t+1} + Q_{t+1}^*\right) - x_{t+1} \bar{R}_t d_t & \text{If bank run in } t+1. \end{cases}$$

where $Q_{t+1}^* < Q_{t+1}$ and 1.1 hold. In a bank run bank assets are wiped out. The term x_{t+1} represents the *recovery rate* of depositors in a bank run state.

Participation Constraint. A banker who chooses not to default at period t has a recursive franchise value of

$$V_t = \max_{d_t} \mathbb{E}_t \left[\beta(1-\sigma) n_{t+1} + \beta\sigma \max\left(V_{t+1}, \theta Q_{t+1} k_{t+1}^b\right)\right]. \tag{1.2}$$

A banker that defaults can abscond with θ of his assets:

$$\theta Q_t k_t^b \tag{1.3}$$

which leaves $(1-\theta) Q_t k_t^b$ for the depositors. A banker therefore chooses to *not* to default if and only if the value of not defaulting (1.2) dominates the value of defaulting (1.3):

$$\theta Q_t k_t^b \leq V_t. \tag{1.4}$$

Any bank that does not satisfy (1.4) will not receive deposits–households will choose to deposit elsewhere.[1]

1.1.2. *Bankers' Problem.* Focusing on an equilibrium where bankers choose not to abscond, we get the following maximization problem,

$$\begin{aligned} V_t &= \max_{d_t} \mathbb{E}_t [\beta(1-\sigma) n_{t+1} + \beta\sigma V_{t+1}] \\ &\text{s.t.} \\ \theta Q_t k_t^b &\leq V_t \forall t. \end{aligned} \tag{1.5}$$

Notice that banks only differ according to their net worth, n_t. Consider the following scaling:

$$\psi_t \equiv \frac{V_t}{n_t} \qquad \phi_t \equiv \frac{Q_t k_t^b}{n_t} \ .$$

Date: January 4, 2016.

[1] It is the households that are choosing to "participate" or not in the participation constraint.

Then (1.5) can be rewritten as

$$\psi_t = \max_{\phi_t} \mathbb{E}_t \left\{ [\beta(1-\sigma) + \beta\sigma\psi_{t+1}] \frac{n_{t+1}}{n_t} \right\} \tag{1.6}$$
$$\text{s.t.}$$
$$\theta\phi_t \leq \psi_t \forall t.$$

Notice that the bankers are now choosing the leverage multiplier, ϕ_t, rather than deposits, d_t.

1.1.3. *Law of Motion of Individual Banker Net Worth.* A banker's net worth at $t+1$ is

$$n_{t+1} = \begin{cases} k_t^b(Z_{t+1} + Q_{t+1}) - \bar{R}_t d_t & \text{If no bank run in } t+1. \\ k_t^b\left(Z_{t+1} + Q_{t+1}^*\right) - x_{t+1}\bar{R}_t d_t & \text{If bank run in } t+1. \end{cases} \tag{1.7}$$

In the no run case a banker's revenues may be rewritten as

$$k_t^b(Z_{t+1} + Q_{t+1}) = n_t \phi_t \frac{Z_{t+1} + Q_{t+1}}{Q_t}.$$

Recalling from (1.1) that $d_t = Q_t k_t^b - n_t$ and $\phi_t = Q_t k_t^b / n_t$ we get

$$\begin{aligned} \bar{R}_t d_t &= \bar{R}_t \left[Q_t k_t^b - n_t \right] \\ &= \bar{R}_t n_t [\phi_t - 1]. \end{aligned}$$

If a bank run occurs the banker is completely wiped out. Thus, the law of motion for a banker's net worth (1.7) may be rewritten as

$$n_{t+1} = \begin{cases} n_t \left[\phi_t \frac{Z_{t+1} + Q_{t+1}}{Q_t} - \bar{R}_t(\phi_t - 1) \right] & \text{If no bank run in } t+1. \\ 0 & \text{If bank run in } t+1. \end{cases} \tag{1.8}$$

Furthermore, the fact that a banker is wiped out in a run implies that the *recovery rate* on deposits x_t in the event of a run is

$$x_{t+1} = \frac{k_t^b\left(Z_{t+1} + Q_{t+1}^*\right)}{\bar{R}_t d_t} < 1.$$

1.1.4. *Banker's Problem, Again.* Denote the probability of a bank run at period $t+1$ by P_t. Then using (1.8) the banker's problem (1.6) may be rewritten as

$$\psi_t = \max_{\phi_t} \mathbb{E}_t (1-P_t)[\beta(1-\sigma) + \beta\sigma\psi_{t+1}] \left[\phi_t \frac{Z_{t+1} + Q_{t+1}}{Q_t} - \bar{R}_t(\phi_t - 1) \right] \tag{1.9}$$
$$\text{s.t.}$$
$$\theta\phi_t \leq \psi_t \forall t.$$

Since n_t is absent from (1.9) this implies that all bankers choose the same leverage multiplier, ϕ_t.

Notice that because

$$V_t = \psi_t n_t$$

and we now know that ψ_t is not a function of n_t, it follows that $n_t = 0$ implies $V_t = 0$. That is, the banker's franchise value is zero if they have no net worth.

A further constraint is needed to ensure that bankers will choose to have maximum leverage:

$$\theta > \frac{Z_{t+1} + Q_{t+1}}{Q_t} - \bar{R}_t > 0. \tag{1.10}$$

That is, the excess marginal value from non-default must be positive but less than the marginal gain from default, θ. This ensures that banks will always accept deposits and, in particular, that they will maximize their leverage,

$$\phi_t = \frac{\psi_t}{\theta} = \frac{1}{\theta}(1-P_t)[\beta(1-\sigma) + \beta\sigma\psi_{t+1}] \left[\phi_t \frac{Z_{t+1} + Q_{t+1}}{Q_t} - \bar{R}_t(\phi_t - 1) \right]. \tag{1.11}$$

This inequality should be verified numerically.

1.1.5. *Aggregation.* Since all bankers face the same problem, aggregate leverage Φ_t equals each individual banker's leverage, ϕ_t. Thus, we have the first equilibrium condition,[2]

$$\Phi_t = \frac{\beta}{\theta}(1-P_t)[(1-\sigma)+\sigma\theta\Phi_{t+1}]\left[\Phi_t\frac{Z_{t+1}+Q_{t+1}}{Q_t}-\bar{R}_t(\Phi_t-1)\right] \tag{1.12}$$

The law of motion for aggregate banker net worth may also be found by making use of the scaling factors. Let $\zeta_t(n)$ be the period t distribution for banker net worth. Then total net worth at t is given by

$$N_t \equiv \int_0^\infty n\cdot\zeta_t(n)dn.$$

When there is no bank run at period $t+1$ total end-of-period net worth for bankers operating in t is:[3]

$$\tilde{N}_{t+1} = N_t\left[\Phi_t\frac{Z_{t+1}+Q_{t+1}}{Q_t}-\bar{R}_t(\Phi_t-1)\right].$$

Measure $1-\sigma$ of bankers are randomly chosen to exit at $t+1$. Their aggregate consumption is

$$C^b_{t+1} = (1-\sigma)\tilde{N}_{t+1} = (1-\sigma)N_t\left[\Phi_t\frac{Z_{t+1}+Q_{t+1}}{Q_t}-\bar{R}_t(\Phi_t-1)\right]$$

A measure σ of bankers from period t survive while a measure $1-\sigma$ of bankers enter with endowment w^b_{t+1}. Denote the total endowment of new bankers by $W^b_{t+1} = (1-\sigma)w^b_{t+1}$. Then the law of motion for the aggregate beginning-of-period net worth of bankers is

$$N_{t+1} = \sigma N_t\left[\Phi_t\frac{Z_{t+1}+Q_{t+1}}{Q_t}-\bar{R}_t(\Phi_t-1)\right]+W^b_{t+1} \tag{1.13}$$

1.2. **Bank Run.** In this model a bank run can only occur via an economic shock or decline in asset prices Q_{t+1}. It must, therefore, affect the *entire* banking system. In fact, under "normal" conditions a bank can repay all of its depositors in full.

A Run on an Individual Bank. Recall that in the no-bank run case we had

$$\frac{Z_{t+1}+Q_{t+1}}{Q_t}-\bar{R}_t > 0. \tag{1.14}$$

If a bank is called on to pay back all of its deposits under normal conditions (i.e. when this condition holds) it could do so. To see this use the aggregate analog to the balance sheet condition, $Q_tK^b_t = D_t+N_t$, and multiply both sides of (1.18) by $Q_tK^b_t$:

$$K^b_t(Z_{t+1}+Q_{t+1}) > Q_tK^b_t\bar{R}_t = [D_t+N_t]\bar{R}_t > D_t\bar{R}_t.$$

That is, so long as $Z_{t+1}+Q_{t+1}$ is great enough, the bank can repay all of its depositors with interest $D_t\bar{R}_t$.

The Possibility of a Run. Consider a case where (1.14) does not hold:

$$K^b_t\left(Z_{t+1}+Q^*_{t+1}\right) < D_t\bar{R}_t. \tag{1.15}$$

Here, Q^*_{t+1} denotes the price that banks could sell their assets for *if they were forced to do so.* It is quite possible that households could set $D_{t+1} = D_t$, which means that banks do not have to pay out any deposits and the banking system continues to operate as usual.

If, however, households demanded all of their deposits back (i.e. $D_{t+1} = 0$), then the banks could not pay back the deposits in full. In this case they would be *insolvent.*

Condition (1.15) relates not only to the solvency of the banks but also the agency problem. The individual bank analog to (1.15) implies that the participation constraint is not met; if a household were to deposit savings with a bank in these conditions, the bank would default. This is why $D_{t+1} = 0$ in equilibrium during the bank run.[4]

---

[2]Notice that, besides substituting the ϕ_t's for Φ_t, the leverage condition 1.11 has been used to substitute $\theta\Phi_t$ for ψ_t.

[3]The use of the $\tilde{N}_{t+1}$ is meant to distinguish this *end-of-period* net worth from the *beginning of period* net worth that will be of interest.

[4]If this reasonsing appears to be circular then I think that's just becuase it is. The banks are only insolvent because $D_{t+1} = 0$ in equilibrium and $D_{t+1} = 0$ because the banks are insolvent. The actual occurence (though not th probability) of a bank run is not coming from the fundamentals; its a sunspot shock.

The Probability of a Bank Run. The above discussion shows that there is nothing within the model that dictates whether a bank run will, in fact, occur or not. This will have to be specified exogenously. Recall that the probability of a bank run was P_t. We will now specify that

$$P_t = 1 - \min\{x_{t+1}, 1\} \tag{1.16}$$

In terms of aggregate variables, the recovery rate on deposits, x_{t+1}, is

$$x_{t+1} = \frac{\left(Z_{t+1} + Q^*_{t+1}\right) K^b_t}{(\Phi_t - 1) N_t \bar{R}_t}.$$

Aside: Entrant Banks During a Run. When (1.15) holds then $D_{t+1} = 0$. This implies that the $1-\sigma$ banks that would have entered in $t+1$ do not accept deposits until $t+2$.

This is technically convenient because it ensures that banks that are run on can only sell to households. It also ensures that $D_{t+1} = 0$.

1.3. **Households.** Households have log utility in consumption and a per-period endowment $W^h_t = Z_t \omega^h_t$. The representative household has a budget constraint

$$C^h_t + D_t + Q_t K^h_t + f\left(K^h_t\right) = W^h_t + R_t D_{t-1} + (Z_t + Q_t) K^h_{t-1}.$$

where $f\left(K^h_t\right) := \frac{\alpha}{2}\left(K^h_t\right)^2$ is the price paid by a household that manages its own capital.

Households choose deposits and capital at each period to maximize utility. The FOC for deposits is,

$$(1 - P_t)\bar{R}_t \beta \frac{C^h_t}{C^h_{t+1}} + P_t \bar{R}_t x_{t+1} \beta \frac{C^h_t}{C^*_{t+1}} = 1 \tag{1.17}$$

while that for capital is

$$(1 - P_t)\beta \frac{C^h_t}{C^h_{t+1}} \frac{Z_{t+1} + Q_{t+1}}{Q_t + \alpha K^h_t} + P_t \beta \frac{C^h_t}{C^*_{t+1}} \frac{Z_{t+1} + Q^*_{t+1}}{Q_t + \alpha K^h_t} \leq 1. \tag{1.18}$$

The inequality in (1.18) is an equality whenever $K^h_t > 0$.

Resource Constraint in a No-Run Equilibrium. The resource constraint in a no-run equilibrium is

$$C^h_t + C^b_t + \frac{\alpha}{2}\left(K^h_t\right)^2 \leq Z_t + W^h_t + W^b_t. \tag{1.19}$$

1.4. **No-Run Equilibrium Conditions.** The equilibrium variables of interest are

$$K^h_t, D_t, Q_t, \bar{R}_t, P_t, C^h_t, \Phi_t, N_t.$$

1.4.1. *Case $t > 2$.* At each $t > 2$ an equilibrium path must satisfy:

$$\Phi_t = \frac{\beta}{\theta}(1 - P_t)[(1-\sigma) + \sigma\theta\Phi_{t+1}]\left[\Phi_t \frac{Z_{t+1} + Q_{t+1}}{Q_t} - \bar{R}_t(\Phi_t - 1)\right] \tag{1.20}$$

$$N_{t+1} = \sigma N_t \left[\Phi_t \frac{Z_{t+1} + Q_{t+1}}{Q_t} - \bar{R}_t(\Phi_t - 1)\right] + W^b_{t+1} \tag{1.21}$$

$$P_t = 1 - x_{t+1} \tag{1.22}$$

$$1 = (1 - P_t)\bar{R}_t \beta \frac{C^h_t}{C^h_{t+1}} + P_t \bar{R}_t x_{t+1} \beta \frac{C^h_t}{C^*_{t+1}} \tag{1.23}$$

$$1 = (1 - P_t)\beta \frac{C^h_t}{C^h_{t+1}} \frac{Z_{t+1} + Q_{t+1}}{Q_t + \alpha K^h_t} + P_t \beta \frac{C^h_t}{C^*_{t+1}} \frac{Z_{t+1} + Q^*_{t+1}}{Q_t + \alpha K^h_t} \tag{1.24}$$

$$C^h_t + C^b_t = Z_t + W^h_t + W^b_t - \frac{\alpha}{2}\left(K^h_t\right)^2 \tag{1.25}$$

$$\Phi_t = \frac{Q_t\left(1 - K^h_t\right)}{N_t} \tag{1.26}$$

$$N_t = Q_t\left(1 - K^h_t\right) - D_t \tag{1.27}$$

Also recall the following auxiliary equations:

$$x_{t+1} = \min\left\{\frac{\left(Z_{t+1} + Q^*_{t+1}\right)\left(1 - K^h_t\right)}{(\Phi_t - 1) N_t \bar{R}_t}, 1\right\}$$

$$C^b_t = \frac{1-\sigma}{\sigma}\left[N_t - W^b_t\right]$$

$$W^h_t = Z_t \omega^h_t.$$

1.4.2. *Case* $t=2$. When $t=2$ the net worth of bankers is pinned down by

$$N_2=W_2^b+\sigma W_1^b.$$

Thus, in the "usual" condition (1.27) (i.e. $N_2=Q_2\left(1-K_2^h\right)-D_2$), one of Q_2,K_2^h,or D_2 is pinned down given the other two variables. In practice I will use this to determine D_2 given Q_2 and K_2^h. This means that the state vector at $t=2$ is $s_2=\begin{bmatrix}Q_2 & K_2^h & \bar{R}_2\end{bmatrix}$.

Moreover, we say that the bankers that "entered" during the bank run period in fact only enter at $t=2$. This ensures that $N_1=0$. It implies that the resource constraint at $t=2$ is

$$C_2^h+C_2^b=Z_2+W_2^h+W_2^b+W_1^b-\frac{\alpha}{2}\left(K_2^h\right)^2,$$

where, again, $W_1^b+W_2^b=(1-\sigma)\left(w_1^b+w_2^b\right)$.

1.5. **Equilibrium in a Bank Run.** Key to solving the above set of equation are the values Q_{t+1}^* and C_t^*. In order to make the bank run state the same at all periods, *assume* that

$$Z_t=Z,W_t^h=W^h,W_t^b=W^b\,\forall t.$$

The main exercise here considers a case when a run *actually* occurs at $t=1$ but agents are aware of the possibility of a run at later periods (that never actually precipitates).

Thus, suppose there is a bank run at $t=1$. Then $D_1=0$ and $K_1^h=1$. Thus, the resource constraint simplifies to

$$C_1^{h*}=Z_1+W_1^h-\frac{\alpha}{2}.$$

Further, since bankers have no deposits to default on in period 2, the probability of a default in period 1 is zero: $P_1=0$. From condition (1.24)–the household deposit FOC–this implies

$$1=\beta\frac{C^{h*}}{C_2^h}\frac{Z+Q_2}{Q_1^*+\alpha}. \tag{1.28}$$

Condition (1.28) will determine whether we have a numeric solution or not.

1.6. **Parameter Values.** All that follows is based on the following set of parameter values:

$$\alpha=0.008,\theta=0.193,\sigma=.95,\beta=0.99$$
$$W^h=0.045,W^b=0.00148/10,Z=0.0126.$$

2.Steady State Solution

There are two cases of interest for the steady state: $x\geq 1$ and $x<1$. Which of these cases applies depends in turn on Q^*, which will take on a single value in equilibrium but may take on many different values as we iterate to a solution.

The script `ssEqnSet` sets up the steady state equations for both cases. Script `xGt1SS` finds and confirms the uniqueness of the steady state for when $x\geq 1$. For the case with $x<1$ the function `ssFindLt` finds the steady state vector for a given Q^*. The main script for investigating the steady states is `ssMapExplore`.

2.1. **Case $x>1$[5].** When $x>1$ we have $P_{ss}=0$. From (1.20)-(1.27) one can see that the asset price in a run Q^* only enters the system of equilibrium equations in cases where $P_{ss}\neq 0$. Thus, the steady state for this case will only indirectly depend on the choice of Q^*.

Solving the system yields the following steady state vector of state variables:

$$\begin{bmatrix}Q_{gt} & K_{gt}^h & D_{gt} & \bar{R}_{gt}\end{bmatrix}:=\begin{bmatrix}1.0497, & 0.2480, & 0.7512, & 1.0101\end{bmatrix}. \tag{2.1}$$

As alluded to above, this steady state is not entirely independent of Q^*. This is because x itself depends on both Q^* and the state vector. In order for $x\geq 1$ we must therefore have

$$x=\frac{(Z+Q^*)\left(1-K_{gt}^h\right)}{D_{gt}\cdot\bar{R}_{gt}}\geq 1. \tag{2.2}$$

Plugging the values from (2.1) into (2.2) and solving for Q^* we find a value for Q^* such that this inequality holds:

$$\underline{Q}_{gt}^*=0.9965. \tag{2.3}$$

Thus, for all $Q^*<\underline{Q}_{gt}^*$ we only need to worry about one steady state. In examining the case $x<1$ we will also find that (2.1) is the unique feasible steady state for all $Q^*\geq\underline{Q}_{gt}^*$.

[5]See `xGtSS`.

2.2. **Case $x<1$[6].** When $x<1$ the solution will depend in a direct manner on Q^* because $P_{ss}>0$. Thus, this case is slightly more complicated than the previous one. To find a steady state for this case we will make use of `fsolve`. One potential problem with blindly using `fsolve` here is that the solution must satisfy several non-linear inequalities. One way to increase the probability that the solution returned by the solver satisfies these constraints is by submitting an initial guess that itself satisfies the inequalities. A procedure for doing just this is outlined first. Then we discuss the solution.

2.2.1. *Choosing an `s0`.* In order to find a solution that is feasible it seems advisable to submit an initial guess that is itself feasible. That is, an `s0` that satisfies:

$$\begin{aligned}
\frac{(Z+Q^*)\left(1-K^h\right)}{D\cdot\bar{R}} &< 1 && [x]\\
\frac{1-\sigma}{\sigma}\left[Q\left(1-K^h\right)-D-W^b\right] &\geq 0 && \left[C^b\right]\\
Z+W^h+W^b-\frac{\alpha}{2}\left(K^h\right)^2-\frac{1-\sigma}{\sigma}\left[Q\left(1-K^h\right)-D-W^b\right] &\geq 0 && \left[C^h\right]\\
\frac{Z+Q}{Q}-\bar{R} &> 0 && [\text{Leverage}]\\
\frac{Z+Q}{Q}-\bar{R} &< \theta && [\text{No-default}]\\
Q,K^h,D,\bar{R} &\geq 0 && [s]\\
K^h &\leq 1
\end{aligned}$$

Non-negativity constraints for N and Φ are excluded because these are superseded by $\left[C^b\right]$.[7]

I will begin by drawing a $K_0^h\in\left[\underline{K}^h,\bar{K}^h\right]\subset[0,1]$. Then I will set $x_0\in(0,1)$. This will $\bar{R}$ in terms of D:

$$\bar{R}(D)=\frac{(Z+Q^*)\left(1-K_0^h\right)}{D\cdot x_0}.$$

Set $C_0^b:=\kappa_1\left[Z+W^h+W^b-\frac{\alpha}{2}\left(K_0^h\right)^2\right]$ where $\kappa_1\in[0,1]$ is an arbitrary constant. Then get Q in terms of D and K^h:

$$Q(D)=\frac{D+W^b+\frac{\sigma}{1-\sigma}C_0^b}{1-K_0^h}.$$

Finally, see if there is a D such that $\frac{Z+Q(D)}{Q(D)}-\bar{R}(D)=\kappa_2\theta$, where $\kappa_2\in(0,1)$ is an arbitrary constant. The roots of this problem can be shown to be[8]

$$D\in\frac{-[\mathcal{A}-\mathcal{E}-\kappa_2\theta\mathcal{B}]\pm\left\{[\mathcal{A}-\mathcal{E}-\kappa_2\theta\mathcal{B}]^2+4x_0[1-\kappa_2\theta]\mathcal{E}\mathcal{B}\right\}^{\frac{1}{2}}}{2x_0[1-\kappa_2\theta]}.$$

---

[6] Prototyped in `ssFindLtProto.m`. General function in `ssFindLt.m`. General function tested in `ssFindLtTest.m`.

[7] These are $Q\left(1-K^h\right)-D\geq 0$ for N and $\frac{Q\left(1-K^h\right)}{Q(1-K^h)-D}\geq 0$ for Φ.

[8]
$$\begin{aligned}
\frac{Z+Q(D)}{Q(D)}-\bar{R}(D)&=\kappa_2\theta\\
\frac{\overbrace{Z\cdot\left(1-K_0^h\right)+W^b+\frac{\sigma}{1-\sigma}C_0^b}^{:=\mathcal{A}}+D}{\underbrace{D+W^b+\frac{\sigma}{1-\sigma}C_0^b}_{:=\mathcal{B}}}-\frac{\overbrace{(Z+Q^*)\left(1-K_0^h\right)}^{:=\mathcal{E}}}{D\cdot x_0}&=\kappa_2\theta\\
\frac{\mathcal{A}+D}{D+\mathcal{B}}-\frac{\mathcal{E}}{D\cdot x_0}&=\kappa_2\theta\\
Dx_0[\mathcal{A}+D]-\mathcal{E}[D+\mathcal{B}]&=D\cdot x_0\kappa_2\theta[D+\mathcal{B}]\\
Dx_0\mathcal{A}+D^2x_0-D\mathcal{E}-\mathcal{B}\mathcal{E}&=D^2\cdot x_0\kappa_2\theta+D\cdot x_0\kappa_2\theta\mathcal{B}\\
D^2\cdot x_0[1-\kappa_2\theta]+D\cdot x_0[\mathcal{A}-\mathcal{E}-\kappa_2\theta\mathcal{B}]-\mathcal{E}\mathcal{B}&=0
\end{aligned}$$

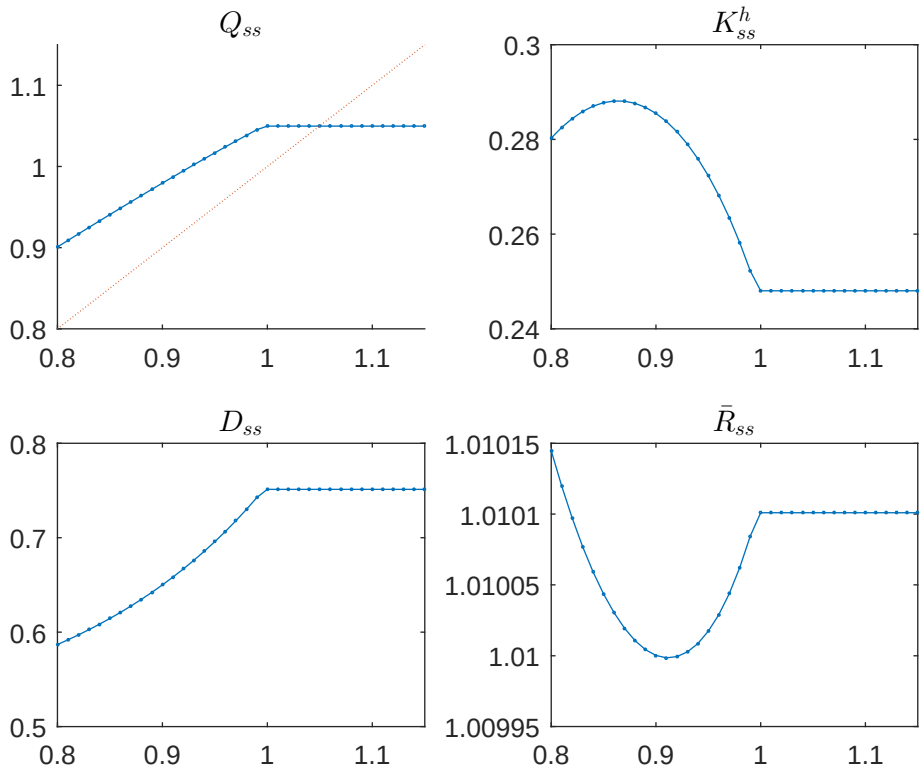

FIGURE 2.1. Mapping from Q^* to the Steady State Variables

The horizontal axis for all subfigures is Q^*. The dots represent individual points at which the steady state was found. The mapping is constant for all $Q^* > 0.995$, the point after which $x \geq 1$. The 45 degree line in the Q_{ss} subfigure demonstrates that, for all Q^* in a "reasonable" range, $Q_{ss} > Q^*$. This is a necessary requirement of a solution because Q^*must reflects the return to households from holding capital while Q_{ss} predominantly reflects the return to banks from holding capital. By assumption the return on capital to banks is greater than that to households.

From this one can see that sufficient conditions for D to have a positive root is that $\mathcal{A}-\mathcal{E}-\kappa_2\theta\mathcal{B}<0$ and $\kappa_2<1/\theta$. But since $\kappa_2<1<1/\theta$ by assumption, this latter condition is automatically satisfied.

The function `ssLtInitDraw` carries out a brute force guess-and-check version of the above procedure.

2.3. **Solving for the Steady State.** To find the actual steady state we will first use `ssLtInitDraw` to find a good initial guess, plug the guess into `fsolve`, then check to see if the solution satisfies the constraints. If it doesn't the process will be repeated. This turns out to be a fairly quick process: a steady state can be found in about 0.10 seconds.

Figure 2.1 shows the mapping from Q^* to the steady state variables. I find that the steady state at each $Q^* < \underline{Q}^*_{gt}$ is unique. Moreover, for all $Q^* > \underline{Q}^*_{gt}$ I find that the proposed solution fails to satisfy the requirement that $x<1$. Therefore, moving forward it is only necessary to consider one steady state for each Q^*.

3.Solving for the Equilibrium Path

The method I employ to find the equilibrium path can be thought of as "shooting backwards". The basic idea is to begin at a point near the steady state and solve the system backwards analytically until banker net worth equals the period 2 banker net worth, $N_2=(1+\sigma)W^b$. The period 1 condition (1.28) is then checked and we iterate to a solution.

Two key problems must be solved to make the method work. First, a way of iterating backwards must be found. In Subsection 3.1 I show that the problem can be reduced to finding the roots of a polynomial in K^h_t whose coefficients are determined by $\{Q_{t+1},K^h_{t+1},D_{t+1},\bar{R}_{t+1}\}$ and the run price Q^*. Second, one has to choose an initial state vector to ensure that the backwards solution will terminate with $N_2=(1+\sigma)W^b$. In Subsection 3.2 I show that this problem can be reduced to a one-dimensional interpolation problem.

3.1. **Solving Backwards.** Suppose that we have a vector of state variables $\{Q_{t+1},K^h_{t+1},D_{t+1},\bar{R}_{t+1}\}$ for $t>2$ in addition to a run price Q^*. This section will show how to find the vector of state variables in the *previous* period, $\{Q_t,K^h_t,D_t,\bar{R}_t\}$.

3.1.1. *Simplifying the System.* Return to considering the system of equations 1.20-1.27. Equations 1.22, 1.25, 1.26, and 1.27 yield solutions for $P_t,C^h_t,\Phi_t,$and N_t that can be substituted into 1.20, 1.21, 1.23, and 1.24. In a like manner the next period $(t+1)$ counterparts of 1.25, 1.26, and 1.27 can be used to substitute $Q_{t+1},K^h_{t+1},D_{t+1},\bar{R}_{t+1}$ for $C^h_{t+1},\Phi_{t+1},$and N_{t+1} in 1.20, 1.21, 1.23, and 1.24. We now have a set of four equations (1.20, 1.21, 1.23, and 1.24) parametrized by $Q_{t+1},K^h_{t+1},D_{t+1},\bar{R}_{t+1}$ and Q^*. This set can be used to solve for the four unknowns $\{Q_t,K^h_t,D_t,\bar{R}_t\}$.

While the above set of equations can be solved with, e.g., `fsolve`, time savings can be achieved with a further simplification. Using `MATLAB`'s symbolic toolbox one can solve 1.20, 1.21, and 1.23 for $Q_t,D_t,\bar{R}_t$ as a function of K^h_t and the parameters $Q_{t+1},K^h_{t+1},D_{t+1},\bar{R}_{t+1}$ and Q^*. Once this is done the solutions for $Q_t,D_t,\bar{R}_t$ may be substituted into 1.24. We now have one equation 1.24 in one unknown–K^h_t. This is now a rootfinding problem. That is, denoting the error function form of 1.24 by $H\big(k;Q_{t+1},K^h_{t+1},D_{t+1},\bar{R}_{t+1},Q^*\big)$, a solution K^h_t must satisfy

$$H\big(k;Q_{t+1},K^h_{t+1},D_{t+1},\bar{R}_{t+1},Q^*\big)=0.$$

3.1.2. *Finding and Excluding Roots.* The general expression for $H\big(k;Q_{t+1},K^h_{t+1},D_{t+1},\bar{R}_{t+1},Q^*\big)$ is unwieldy.[9] However, plugging in numbers for the parameters one finds that H has a rather tractable form. In particular, H is a fraction with fourth-order polynomials of k in the numerator and denominator, i.e.

$$H\big(k;Q_{t+1},K^h_{t+1},D_{t+1},\bar{R}_{t+1},Q^*\big)=\frac{\eta_4k^4+\eta_3k^3+\eta_2k^2+\eta_1k+\eta_0}{\delta_4k^4+\delta_3k^3+\delta_2k^2+\delta_1k+\delta_0}, \tag{3.1}$$

where the η_i's and δ_j's are functions of $Q_{t+1},K^h_{t+1},D_{t+1},\bar{R}_{t+1},$and Q^*.

With the exception of one special but pertinent case, the roots of H will therefore just be the roots of the numerator polynomial of 3.1. These roots may be solved for analytically using the `roots` function.[10] The special but pertinent case mentioned above occurs when the numerator and denominator have roots at the same point. While it isn't immediately clear to me how we ought to evaluate 0/0 I think it is clear that $0/0\neq0$. Thus, this isn't a legitimate root.

Figure 3.1 shows a prototypical example of the H function. One can see that the numerator polynomial has two roots. However, the H function only has one root. This is explained by the fact that the denominator polynomial has a root that coincides with the numerator polynomial.

3.1.3. *Files and Pseudocode.* The main function for the solving backwards step is `backSolve`. It has three principle child functions: `kRootsBack`, `SvGt2[G,L]t1`, and `NlCon[Gt,Eq]2[GL]t1`.[11] Function `kRootsBack` solves the rootfinding problem discussed in Subsection 3.1.2. The key child functions in kRootsBack return the expressions for the coefficients in 3.1. These functions follow the naming convention `Kc[Gt,Eq]t2[G,L]t1` for the numerator and `Kdc[Gt,Eq]2[GL]t1` for the denominator. Function `SvGt2[G,L]t1` returns the full state vector $\{Q_t,K^h_t,D_t,\bar{R}_t\}$ once the root K^h_t is found with `kRootsBack`. Functions of the form `NlCon[Gt,Eq]2[GL]t1` checks the set of constraints .

The functions `SvGt2[G,L]t1` and `Kc[d][Gt,Eq]2[GL]t1` are generated automatically using `matlabFunction`. The scripts in which the system is solved, substituted, and simplified for each case follow the naming convention `solve[Gt,Eq]2[G,L]1.`

Algorithm 1 demonstrates the basic method for solving backwards. `kRootsBack` is straightforward: it takes $\big[Q_{t+1},K^h_{t+1},D_{t+1},\bar{R}_{t+1},Q^*\big]$ and returns the coefficients from the polynomials in 3.1. It then uses the `roots` function to retrieve the set of roots and find the appropriate root. We then get the full period t state vector by calling `SvGt2[G,L]t1.` Next we check that all of the constraints on the set of period t equilibrium values

[9]How unwieldy? If you want to see the whole thing you'll need to allow `MATLAB` to print more than 25,000 characters in one go. The numerical counterpart to it created using `matlabFunction` is 373kb (compared to maybe 6kb for a more-involved human-generated function). Also, if you convert it into a function using `matlabFunction` be sure to set the option `optimize` to `false`.

[10]The `roots` function takes as arguments the coefficients of the polynomial. It turns out that there are a few nice functions for extracting this information from 3.1 when the coefficients are still in their general, symbolic form. First, use the `numden` function to separate the numerator and denominator of H. Then use the `coeffs` function to extract the coefficients from k. In this last step it is necessary to specify that the argument of the polynomial is k (i.e. `coeffs(H,k)` rather than just `coeffs(H)`). This will yield a 4x1 symbolic expression in terms of $\{Q_{t+1},K^h_{t+1},D_{t+1},\bar{R}_{t+1},Q^*\}$. Finally, use `matlabFunction` to save the coefficients as a numerical function with arguments $\{Q_{t+1},K^h_{t+1},D_{t+1},\bar{R}_{t+1},Q^*\}$.

[11]That is, there are two separate functions called `SvGt2Lt1` and `SvGt2Gt1`. Each deals with the case when x_{t+1} is *Less than* and *Greater than 1*, respectively.

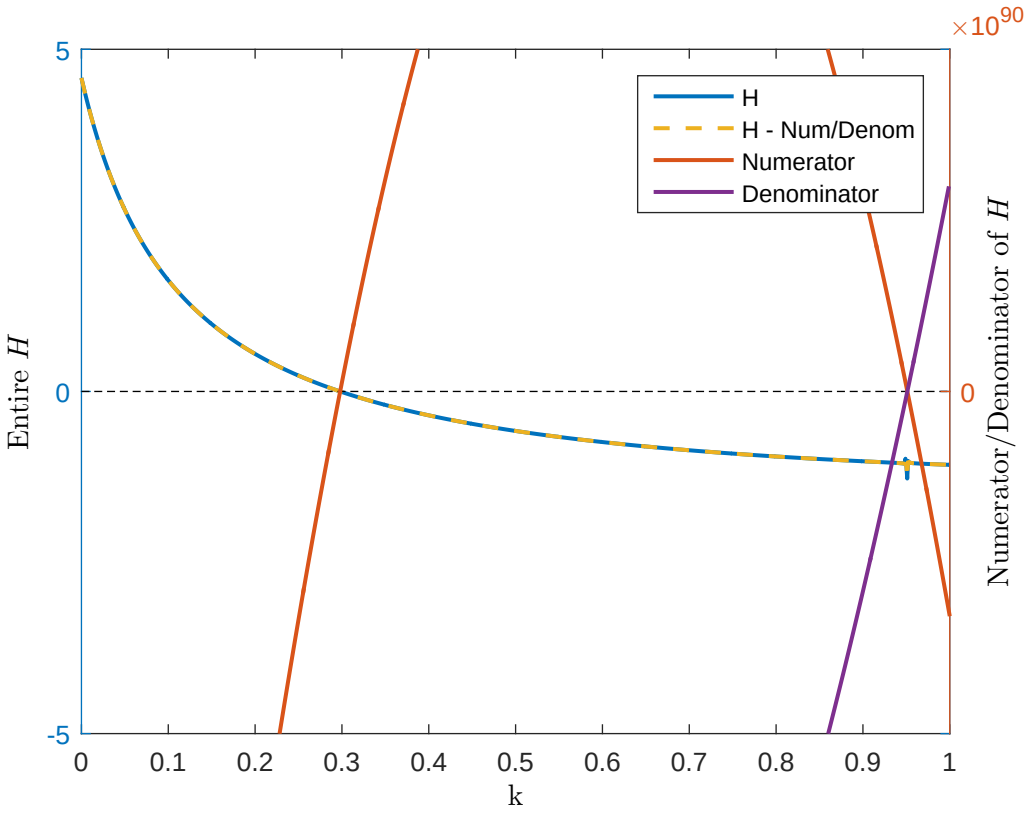

FIGURE 3.1. Example of $H\left(k{:}Q_{t+1},K^h_{t+1},D_{t+1},\bar{R}_{t+1},Q^*\right)$

The true root is at approximately $k=0.2979$ while the phony root is at $k=0.952$. The "squiggle" in the blue/yellow lines at $k=0.952$ reflects a combination of numerical imprecision (as the denominator goes to zero) and the discontinuity at approximately $k=0.952$, the shared, phony root. The solid blue line is the H function while the dashed yellow line is the red line (numerator of H) divided by the purple line (denominator of H). The point of plotting these two lines is to show that the decomposition of H into numerator and denominator parts is, in fact, correct. We plot $k\in[0,1]$ w.l.o.g. because total capital in the economy is 1 and $K^h_t\geq 0$.

are satisfied by the proposed solution. This step should be a formality unless one goes out of one's way to find a pathological case.[12]

The final step is the only new part and the convention established here will be useful in the next path. `backSolve` is agnostic about how many periods will elapse between the initial position on the path and a state vector that will [hopefully] correspond to the period 2 state vector. The key to determining if the solution path has hit or overshot the period 2 state vector is to find the banker net worth at each period. We know that at $t=2$ banker net worth is $N_2=(1+\sigma)W^b$ and, in general, $N_t=Q_t\left(1-K^h_t\right)-D_t$. Under the conjecture that banker net worth is greater than $(1+\sigma)W^b$ for all $t>2$ along the equilibrium path, we can say that period 2 has been overshot whenever the $Q_t\left(1-K^h_t\right)-D_t<(1+\sigma)W^b$. We then cut off all remaining rows in the `sp` matrix of state variables and return the matrix.

The key to finding a *useful* backwards solution path for a given Q^* is to find a way to make the period 2 banker net worth actually hit the correct net worth: $Q_2\left(1-K^h_2\right)-D_2=N_2=(1+\sigma)W^b$. This can be done by choosing an appropriate initial state vector $[Q_0,K^h_0,D_0,\bar{R}_0]$ as an input to `backSolve`. This is the process that will be described in Subsection 3.2.

3.2. **Hitting $(1+\sigma)W^b$.** The last subsection demonstrated an economical method for finding the path of the system *given an initial state vector and* Q^*. We now discuss how to find an initial state vector *given* Q^*. Ideal initial state vectors will generate state vector paths that a) appear to converge to the relevant steady state

[12] An earlier draft of this function was in some ways more robust because it accommodated cases where $x_{t+1}>1$ for some periods and $x_{t'+1}<1$ for others without returning an error. In the current exercise finding that x_{t+1} switched cases is probably more indicative of a bug in the code rather than a part of a legitimate solution so the strategy of returning an error when this case arises then determining whether it is, in fact, an error is probably a safe way of doing things here. In any case, this doesn't seem to be an issue in the final iteration of the code.

Algorithm 1 Solving Backwards

```
INPUT: Initial state vector [Q_0,K_0^h,D_0,R̄_0], Run Price Q*
% Initialize output and time variables
sp = -100*ones(1000,4);  % Matrix of state values
sp(1000,:) = [Q_0,K_0^h,D_0,R̄_0]; % Begin at input vector
t = 999;                  % Period to be solved
endLoop = 0;              % Stop while loop
while t > 2 & endLoop == 0
    % Find the state vector at t
    root = kRootsBack(sp(t+1,:),Q*,t);     % Find root
    sp(t,:) = SvGt2Lt1(sp(t+1,:),root,Q*); % Find state vector at t
    % Check that constraints hold
    if NlConGt2(sp(t,:),Q*) == 0
        return;
    endif
    % Check if banker net worth greater than banker net worth at t = 2
    if Q_t(1−K_t^h)−D_t>(1+σ)W^b            % If not, continue.
        t = t - 1;
    else                                     % If is, stop.
        sp = sp(t:end,:);
        endLoop = 1;
    endif
endwhile
Output: sp
```

and b) satisfy the banker net worth requirement at period 2: $Q_2\left(1-K_2^h\right)-D_2=N_2=(1+\sigma)W^b$. The first requirement provides guidance on the neighborhood around which solutions ought to be sought out–within some (ideally arbitrarily small) ε-ball of the steady state. The second can be used to evaluate whether the path generated by a particular initial state vector is a good candidate solution path. This subsection first discusses the set of initial state vectors that are searched. It then discusses a fairly quick[13] method for finding an initial state vector that satisfies the period 2 requirement.

3.2.1. *The Set of Initial State Vectors.* An initial state vector will ideally be as close to the steady state associated with the given Q^* as possible.[14] One would think that the ideal would be to make the initial value of a state variable just below the steady state if it approaches from below and likewise if it approaches from above. But we don't know this information unless we already know the solution paths. Even if we did know the trajectories of the solution paths–as was the case here–we don't know if the trajectories are the same for all paths associated with all the Q^* that will have to be found en route to a solution.

The approach taken here is to do a first order expansion of the state vector solution about the steady state. Consider a function G that maps $\left\{\left[Q_{t+1},K_{t+1}^h,D_{t+1},\bar{R}_{t+1}\right],K_t^h,Q^*\right\}$ to the period t state vector $\left[Q_t,K_t^h,D_t,\bar{R}_t\right]$.[15] If $\left[Q_{ss},K_{ss}^h,D_{ss},\bar{R}_{ss};Q^*\right]$ is the steady state vector associated with Q^*, the set of initial state vectors will be of the form

$$G\left(\left[Q_{ss}+\gamma,K_{ss}^h+\gamma,D_{ss}+\gamma,\bar{R}_{ss}+\gamma\right],K_{ss}^h+\gamma,Q^*+\gamma\right)=\left[Q_{ss},K_{ss}^h,D_{ss},\bar{R}_{ss};Q^*\right]+\nabla G\Big|_{ss}\cdot\gamma \tag{3.2}$$

where γ is a deviation from the steady state to be determined and $\nabla G\Big|_{ss}$ is the jacobian of G at the steady state.

The first thing to be said about 3.2 is that it seems to work. The second thing that ought to be mentioned is that it works despite the fact that the initial state vector will be a) on the "wrong side" of the steady state for D and b) discontinuous for $\bar{R}$. What seems happen is that–while the initial state vector is a "bad" guess–it generates a path that is in all subsequent periods continuous and altogether "nice". Unfortunately, I can't offer much of an explanation as to why this works out so nicely right now. This is one of the weaker points of the solution.

In any case if we are ready to say that initial state vectors of the form 3.2 will be a good enough, then the problem of choosing an initial state vector is now one of choosing the right γ.

[13] I think.

[14] And it can't be equal to the steady state because the backwards solution to the steady state vector is...the steady state.

[15] The functions `Sv[Gt,Eq]2[G,L]t1` do this.

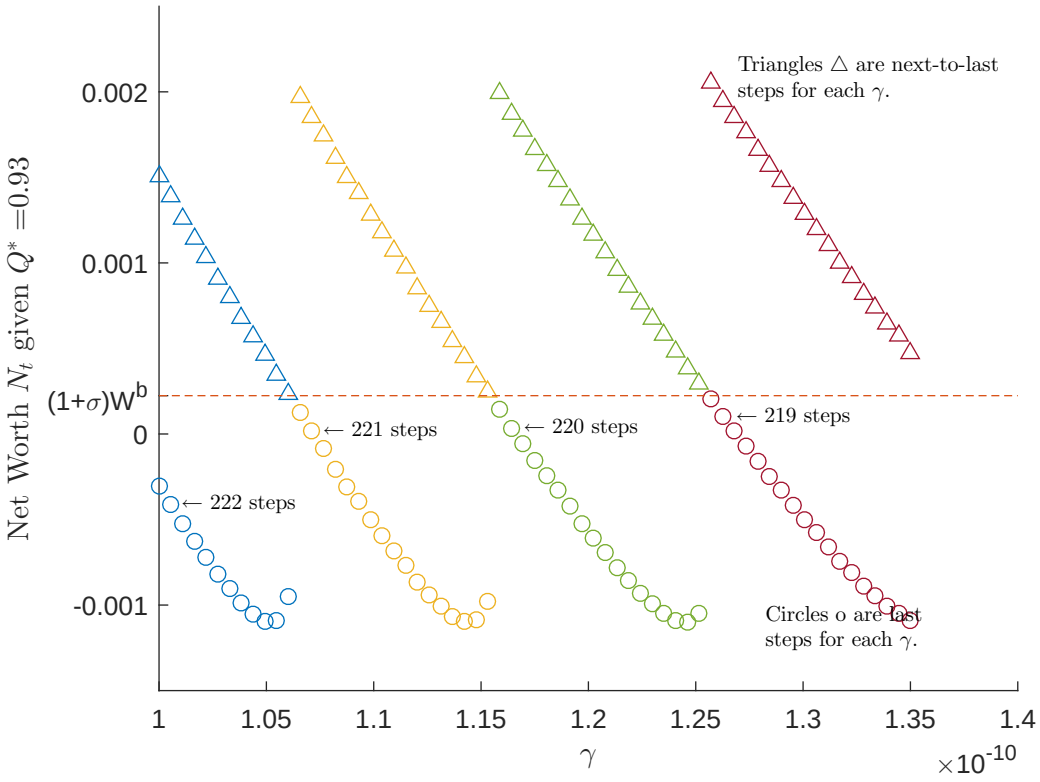

FIGURE 3.2. Final and Penultimate Banker Net Worth Values as a Function of γ

The triangles are for the penultimate period while the squares are for the final period. Each color represents the number of steps from the initial point to the final point, which is determined by the fact that $N_t < (1+\sigma)W^b$ at that point. The different colors represent the number of steps on the path. For this example with $Q^* = 0.93$ the blue is associated with 222 steps, the yellow with 221, the green with 220, and the maroon with 219.

3.2.2. *Choosing γ to Satisfy $N_2 = (1+\sigma)W^b$.* The basic strategy used to choose γ will be to solve backwards for a few values of γ, save the resulting N_2, then interpolate from $N_2 = (1+\sigma)W^b$ to the correct γ. The backwards solution procedure will complicate this basic idea a bit because–as can be seen from the final `if` condition in Algorithm 1–`backSolve` only returns a path once the net worth N implied by a given state vector is less than $(1+\sigma)W^b$.

Figure 3.2 displays the mapping between γ and banker net worth N_t for the last two steps of the path. The triangles are the second to last step while the circles are the last steps, a relationship that is determined by the fact that $N_t < (1+\sigma)W^b$. The different colors represent the number of steps on the path. For this example with $Q^* = 0.93$ the blue is associated with 222 steps, the yellow with 221, the green with 220, and the maroon with 219.

From Figure 3.2 one can see that for any Q^* there will be numerous γ that will yield a path that terminates with $N_2 = (1+\sigma)W^b$.[16] Moreover, one can see that looking at only the last steps will be less-than-ideal at best and possibly misleading whenever a new step is added. It will instead be ideal to keep track of the number of steps in each path then look at a mix of the final and penultimate steps when interpolating. Finally, one can see that we are interpolating a one-armed quadratic with slight but non-zero curvature. In order to get an accurate result via interpolation it will be necessary to either a) do a highly-local linear interpolation or b) form a spline. In the present coding configuration these methods will be about equal in terms of computational burden; the spline requires the creation of four paths but it will generally require at least as many paths to zero in on a sufficiently local linear approximation. Both methods will yield interpolations with errors on the same order of magnitude and solution Q^*'s that differ by about $1e{-}4$. The current code interpolates with `interp1` using method `pchip`.[17]

[16]It will turn out that the solution will be robust to the particular choice of γ used. I would conjecture that each possible solution (i.e. the initial state vector corresponding to each γ for which $N_2 = (1+\sigma)W^b$) is, in fact, a step on the full solution path.

[17]One can look at the results of a local linear approximation by changing the method in line 104 of `qRunIterQ` to `linear`.

Algorithm 2 Finding the Right γ–`qRunIterQ`

```
INPUT: Run Price Q*, initial γ grad0m, γ adjustment multiplier gradAdj
Find the steady state - ssRoute
Find Jacobian of state vectors at steady state - SvJLt1
First path:
    Form initial state vector - ss + jacobian*grad0m
    Get path - sp{1} = backSolve
    Adjust γ - grad0m = grad0m*gradAdj
Second path: (Same substeps as first path)
Check to make sure number of steps is okay
Adjust γ - grad0m = (1/3)*grad0m*gradAdj
Third path: (Same substeps as first path)
Adjust γ - grad0m = (2/3)*grad0m*gradAdj
Fourth path:
(Same as first path)
Interpolation:
    x = Cor-
    rect N's for each path (depending on number of steps in each path)
    v = set of γ's
    Interpolate to find γ cor-
    resonding to N=(1+σ)W^b - interp1(x,v,(1+σ)W^b,'method','pchip')
Get path with interpolated γ. (Same substeps as first path.)
OUTPUT: Path with interpolated γ
```

3.2.3. *Files and Pseudocode.* The funtion `qRunIterQ` carries out this step.[18] The steps in this function are outlined in Algorithm 2. The user supplies Q^* (`Qri`: Q-run-iteration), an initial γ (here called `grad0m` for reasons that are murky at this point) and an adjustment multiplier called `gradAdj`. The function first finds the steady state corresponding to the supplied Q^* and the jacobian of the state equations at the steady state.[19] It then runs `backSolve` with initial state vector formed according to 3.2 with γ=`grad0m`. A second run of `backSolve` with γ=`grad0m*gradAdj` yields the second interpolation point.[20] If the first path returned by backSolve has M steps in total, we would like the second `backSolve` path to have $M-1$ steps.[21] Assuming that this requirement is met, two more paths are formed that use γ that are equally spaced between the first two γ. So in the normal case the interpolations will be based on paths with $\gamma = \{$`grad0m`, $\frac{1}{3}\cdot$`grad0m*gradAdj`, $\frac{2}{3}\cdot$`grad0m*gradAdj`, `grad0m*gradAdj`$\}$. Next, we need to form the set of banker net worths that will be used in interpolation. In a normal case this will be N_{end-1} for the path with γ=`grad0m`, N_{end} for γ=`grad0m*gradAdj`, and one of either N_{end} or N_{end-1} for the paths with intermediate γ depending on whether there are M or $M-1$ total steps.

3.3. **Finding Q^*.** The last subsection showed how to find an adequate candidate solution path given a Q^*. This subsection shows how to find the correct Q^*. In equilibrium Q^* must satisfy 1.28:

$$1=\beta\frac{C^{h*}}{C_2^h}\frac{Z+Q_2}{Q^*+\alpha}.$$

Function `qRunIterQ` will return a solution path that will include Q_2 and from which C_2^h may be derived. If the Q^* implied by 1.28 is the same Q^* used to derive the solution path, then we have an equilibrium. If the Q^* implied by 1.28 is not the same as the Q^* used to compute the path, then a new Q^* must be tried. Following the suggestion in the handout, I use the Q^* implied by 1.28 as the new guess to be tried.

[18] A closely related but deprecated function, qRunIter, is identical to qRunIterQ except that it carries out linear interpolation based on two (generally distantly separated) choices of γ. It can be called by setting the `M` argument in the parent function backUmbrella to the appropriate value. At this point it mostly exists to demonstrate that the curvature of the mapping to be interpolated is too great to be ignored.

[19] The jacobian function `SvJLt1` takes a lot of time to get into the cache–21% of the 4.147 second runtime of the test script `qRunIterQTest`. Once this is done, though, the whole of `qRunIterQ` runs in 0.45 seconds.

[20] From examining figures like 3.2 for different Q^* seems that setting gradAdj between 1.05 and 1.1 will work for most cases. The full code can accommodate cases where `gradAdj` is either to big or small by making further adjustments and getting those paths. From a speed perspective, though, it would be ideal to find an adjustment multiplier that works for almost all cases the first time around.

[21] Assuming that `gradAdj` >1.

Algorithm 3 Finding Q^*–backUmbrella

```
INPUT: Initial Q* guess (Q0), Required convergence tol-
erance (Tol), maximum number of Q* guess iterations to complete, ini-
tial γ (grad0m), Adjustment to γ (fradAdj), interpolation method (M)
Qri = Q0;
iter = 1;
WHILE iter < maxIter
    path = qRunIterQ(Qri, grad0m, gradAdj);
    Period-2-Equilibrium values = objEq2(path);
    QrunP = β(C^{h*}/C_2^h)(Z+Q_2)−α
    if abs((QrunP - Qri)/Qri) < Tol
         Equilibrium Found
         Exit;
    else
         Qri = QrunP
         iter = iter + 1;
    end
ENDWHILE
OUTPUT: Equilibrium path
```

backUmbrella is the function that carries out this final step. Pseudocode for this function can be found in Algorithm 3. The actual code is a bit more crowded in order to print out statuses concerning convergence and the like but the idea is simple; run a function, check a condition, update if it doesn't converge and exit if it does.

4.Results[22]

4.1. Equilibrium Values in a Run.

4.1.1. *Price of Assets.* In equilibrium $Q^*=0.90087$. This result is invariant to the initial guess of the equilibrium Q^* and the choice of γ.[23]

Figure 4.1 shows the mapping of Q^* from iteration to iteration. In practice, I find that convergence is only essentially monotone. That is, it is possible to have non-monotone Q^* updates but only after the process has already essentially converged.[24]

4.1.2. *Other Equilibrium Values.* The values of other equilibrium variables during a run are I think either trivial or debatable. First, the trivial. Households hold all of the capital because banks have to sell all of it to them to pay out depositors in the run; $K^{h*}=1$. Households remove all of their deposits in the run; $D^*=0$. The probability of a run next period is zero because banks have no deposits to steal; $P^*=0$. Banks are bankrupted by the run; $N^*=0$. Banks that exit are bankrupt and therefore consume nothing; $C^{b*}=0$. Households therefore have consumption $C^{h*}=Z+W^h-\frac{\alpha}{2}$.

Bank leverage is $\Phi=D/(N+D)$. In the run both N and D are zero. From a conceptual and linguistic point of view, I think that it makes the most sense to say that bank leverage is zero during the run. Leverage is all about how much assets an entity has to spend over and above its own equity. To say that leverage in the run is positive infinity makes it sound as if the banks have money to spend. This clearly is not the case, though.

Finally, there is the question of deposit interest rates during the run. One way to answer this would be to use 1.23 and the fact that $P^*=0$ to find

$$\bar{R}^*=\frac{1}{\beta}\frac{C_2^h}{C^{h*}}=1.0523.$$

I don't like this answer, though. The reason is that it seems to suggest that the market for deposits is clearing because $\bar{R}^*=1.0523$. However, the quantity of deposits during the run is not a function of the interest rate; it's a sunspot phenomenon. Conditional on there being a run it does not matter if $\bar{R}^*=1.5$ or 1.0005. It is

[22] All results can be called from the script `final.m`.

[23] There does, however, seem to be a trade-off between γ and the level of convergence achieved. In particular, to achieve closer convergence of Q^* one must use a *larger* γ (i.e. 1e-5 as opposed to 1e-8 or 1e-10).

[24] For example, the specification used to create Figure 4.1 specified $\gamma=1e-8$ and a convergence level of $1e-8$ ($1e-06\%$). Starting with $Q^*_{guess}=0.98$ `backUmbrella` runs for a total of 90 iterations before converging. Convergence is monotone and negative for the first 66 iterations. At iteration 66 $Q^*_{guess}=0.90087077$ and the *percentage* difference from the previous period is $-2.41e-5\%$; i.e. the process has already essentially converged. From iterations 67-90 the largest percentage difference between periods is $+2.19e-05\%$. The Q^* returned by `backUmbrella` is 0.9008705918.

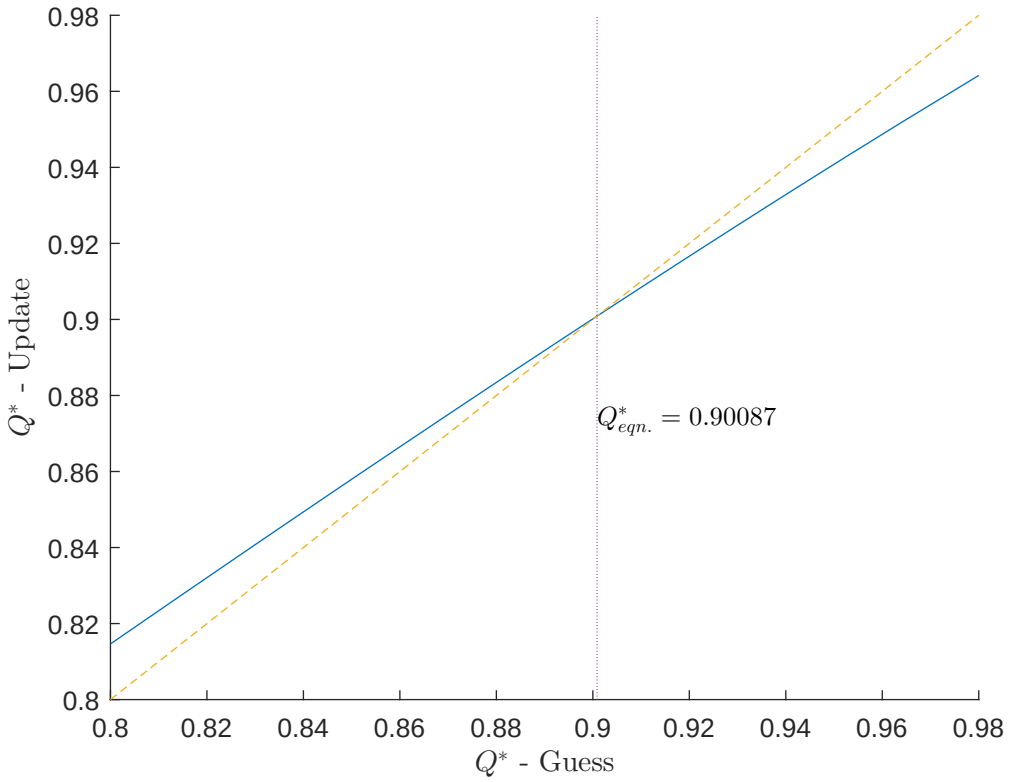

FIGURE 4.1. Mapping from a Guess of Q^* to an Updated Guess of Q^*

This figure–created in the script `qRunMapping`–was formed by running `backUmbrella` twice with initial Q^* of 0.98 in the first run and 0.8 in the second.

non-optimal for an individual household to deposit in a bank at any interest rate because the bankrupt bank will always run away with the deposit. One might then want to say that, in order for the market for deposits to clear we must have $\bar{R}$"="$+\infty$; otherwise banks will always be willing to accept some sort of deposit. Setting aside the question of whether infinite prices make any sense it is enough to show that this line of argument is flawed. Suppose that interest rates were arbitrarily high. How much deposits would a bank be willing to supply in this environment? I think that the answer is, however much households are willing to provide. Why? The bankrupt bank has no intention of paying the interest rate, it will always default and run away. The interest rate has no bearing on this decision. Thus, I think it is wrong to go out looking for an interest rate that will clear the market for deposits during the run period. Instead, I think it makes the most sense to just say that interest rate during the run period is indeterminate.

4.2. **The Equilibrium Path After a Run.** Figure 4.2 shows the paths of equilibrium variables beginning from period 2.

One noteworthy feature of this figure is that the net interest rate $\bar{R}_t$ is below the steady state $\bar{R}_{ss}$ at $t=2$ and then jumps above $\bar{R}_{ss}$ at $t=3$. At this point I do not think that this reflects an error in my code or method but is instead a part of the true solution. In short, I think that households are willing to accept a lower interest rate in order to deposit more funds with the banks. Households want to deposit with them to reap the higher returns. However, the low net worth of banks limits their ability to take on leverage without violating the incentive compatibility constraint. A lower interest rate makes the non-default option optimal for banks despite their high leverage. From period 2 to period 3 bank net worth increases by 219%. This windfall means that the banking sector can take on more deposits while lowering leverage.

In period 2 there is a relatively large probability of a run; a force that should push up interest rates. However, I have argued that the desire of households to deposit in spite of low bank net worth will push down interest rates and that, in fact, this opposite force will dominate. In period 3 the probability of a bank run is still quite high. The ability of banks to take on deposits while respecting the incentive constraint increased substantially relative to the previous period, though. Therefore, the bank run force will dominate and interest rates will be relatively high.

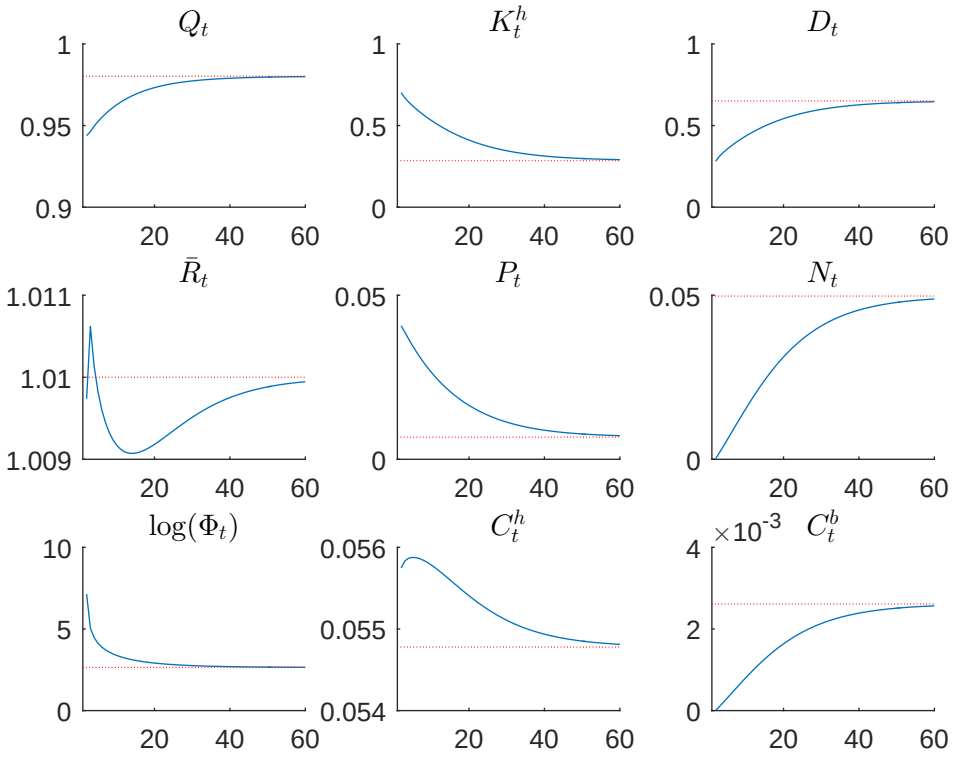

FIGURE 4.2. Equilibrium Paths

	t				
	2	60	120	160	Steady State
Q_t	0.9438	0.9799	0.9802	0.9802	0.9802
K_t^h	0.7018	0.2911	0.2854	0.2854	0.2854
D_t	0.2812	0.6458	0.6507	0.6507	0.6507
$\bar{R}_t$	1.0097	1.0099	1.0100	1.0100	1.0100
P_t	0.0407	0.0072	0.0068	0.0068	0.0068
N_t	0.0002	0.0489	0.0497	0.0497	0.0497
Φ_t	1256.4600	14.2176	14.0833	14.0823	14.0822
C_t^h	0.0557	0.0548	0.0548	0.0548	0.0548
C_t^b	0.0000	0.0026	0.0026	0.0026	0.0026

TABLE 1. Equilibrium Values at Select Periods

4.3. **Time To Reach The Steady State.** Table 1 displays the values of equilibrium variables at select periods after the steady state and the values of the variables in steady state. With the exception of Φ_t the variables are all within $1e{-}4$ of the steady state after 120 periods (approximately 30 years).

The method used here makes it relatively straightforward to consider later periods by choosing a smaller γ. Looking at the column for $t{=}160$ one can see that Φ_t has almost converged to the steady state value by that time.

4.4. **Comparing Steady States.** Table 2 shows the steady state values side-by-side for the two parameterizations of the model. The biggest differences between the two are found in the probability of a run, P_{ss} followed by the banker net worth N_{ss} and leverage Φ_{ss}.

Recall that the probability of a run is determined by

$$P_{ss} = 1 - \frac{[Z+Q^*]\left[1-K_{ss}^h\right]}{D_{ss}\cdot\bar{R}_{ss}}.$$

	Current	Handout	%Δ
W^b	0.0011487/10	0.0011487/50	+160.93
Q^*	0.9007	0.91	-1.0272
Q_{ss}	0.9802	0.98	+0.0204
K^h_{ss}	0.2854	0.28	+1.9102
D_{ss}	0.6507	0.66	-1.4191
$\bar{R}_{ss}$	1.0100	1.0101	-0.0100
P_{ss}	0.0068	0.0075	-9.7980
N_{ss}	0.0497	0.047	+5.5857
Φ_{ss}	14.0822	14.93	-5.8461
C^h_{ss}	0.0548	0.055	-0.3643
C^b_{ss}	0.0026	0.0025	+3.9221

TABLE 2. Equilibrium Values at Select Periods

Minimum Number of Steps From Run to Steady State	170	120
Mean Periods in Steady State	147.47	148.09
	(45.12)	(39.24)
Mean Periods to Return to Steady State after Run	345.60	202.40
	(73.21)	(30.76)

TABLE 3. Summary Statistics from Simulation

Table displays means and standard deviations (in parentheses) of summary statistics from 1,000 simulations carried out by function `simuPath` for 5,000 periods. The path used in the simulation was generated by `backUmbrella`. The path used is identical across simulations and is stored in `simuData.mat`.

The decreases in Q^* and D_{ss} and increase in K^h_{ss} when $W^b = 0.0011487/10$ relative to when $W^b = 0.0011487/50$ all contribute to the decrease in P_{ss}.

4.5. **Simulation.** The function `simuPath` simulates the economy. Table 3 presents summary statistics from the simulation. One important deviation of the method used here from that suggested in the handout is that the number of periods to the steady state is not set as a parameter. In my benchmark solution path there are 178 periods from the run period to the initial point near the steady state. I then cut off the last eight periods to account for any odd results arising from the semi-arbitrary initial point.[25] This longer path will make it appear that it takes longer to return to the steady state. To account for this I include in `simuPath` the option to jump to the steady state after a fixed number of periods. The two columns in Table 3 include simulation results for the cases in which the minimum number of steps from the run period to the steady state are 170 and 120, respectively.

As expected the mean periods in steady state is similar across columns. Once the economy is in the steady state the amount of time it stays in steady state is only a function of P_{ss}=0.0068, which in turn only depends on current and future variables, not past periods. The theoretical expectation of the mean periods in steady state is $1/P_{ss}$=147.0588. This is quite close to the mean of mean periods in steady state across simulations.

The difference in mean periods to return to the steady state after a run are quite different across columns. As mentioned above, this feature partly arises from the fact that, when the solution path is longer, it will simply take more periods to reach the path. From this effect alone we might expect that the periods to return to the steady state will be fifty periods greater in the first column relative to the second. Another factor contributing to the longer time to return is that a run may occur while the economy is taking the last fifty steps. In column 2 such a run would be counted as ending a rather short stretch in the steady state. In column 1 the economy is never counted as entering the steady state and there will be a minimum of 170 more periods until the economy could be counted as being in steady state. A third effect increasing the time to return in column 1 relative to column 2 is that the probability of a run is decreasing to the steady state value. However, after 120 periods the probability of a run is almost identical to that in the steady state. Therefore, I do not think that this factor plays a significant role in explaining the difference between columns.

[25] On this point see Subsection 3.2.1.

References

Christiano, L. J., H. Dalgic, and X. Wen (2015). Gertler-kiyotaki: 'banking, liquidity, and bank runs in an infinite horizon economy', aer (2015). Technical report.

Gertler, M. and N. Kiyotaki (2015). Banking, liquidity, and bank runs in an infinite horizon economy. *American Economic Review 105*(7), 2011–43.