Exploring Dynamic Properties of Non Linear Economic Models Through Asymptotic Linearization

Gary S. Anderson
and
Vi-Min Choong[*]

July 9, 1997

Abstract

Linearizing non linear models about their steady state makes it possible to apply the Anderson-Moore Algorithm(AIM) to investigate the asymptotic properties of the model.

AIM determines whether the non linear model has the saddle point property and provides diagnostics for models which do not. For models that have the saddle point property AIM provides a set of terminal conditions for solving the non linear models that improve upon the traditional approach of setting the end of the trajectory to the steady state values. A suite of Mathematica programs can manipulate nonlinear model equations to provide closed form solutions for many(sometimes all) of the linearization and AIM analysis. These calculations useful for estimating a restricted VAR. Also useful for getting starting values for estimating nonlinear certainty equivalence models. Employs functional programming paradigm to facilitate coherent interactions between numerical and symbolic characterizations of model features.

1 A Non-Linear Extension of the Anderson-Moore Technique

Economist interested in

- saddle point property
- rate of convergence

[*] The authors wish to thank ... We are responsible for any remaining errors. The views expressed herein do not necessarily represent the views of our employer.

- impulse response functions

how selection of parameters impact these features.

Also interested in how good and approximations the linearization provides for the non linear solution and whether or not quantitative and qualitative properties same.

1.1 General Model Specification

Consider the model

$$h(x_{t-\tau}, x_{t-\tau+1}, ..., x_{t+\theta-1}, x_{t+\theta}) = 0 \tag{1}$$

$$t = 0, \ldots, \infty \tag{2}$$

Where $x \in \Re^L \; and \; h : \Re^{L(\tau+1+\theta)} \rightarrow \Re^L$. We want to determine the solutions to equation system 1 with initial conditions

$$x_i = \bar{x}_i \; for \; i = -\tau, ..., -1$$

satisfying

$$\lim_{t\rightarrow\infty} x_t = x^*.$$

We can adapt the methods of (Anderson & Moore, 1985) to determine the existence, and local uniqueness of the solution to equation 1.

1.2 Asymptotic Linearization

If h were linear, we could immediately apply the methods of (Anderson & Moore, 1985) to determine the existence and uniqueness of a perfect foresight solution and to compute the solution. Since h is non-linear, we will compute approximate solutions to system 1 by using the nonlinear h constraints in system 1 for the initial part of the trajectory, and using a system of linear constraints which reflect the asymptotic properties of the system for the remainder of the trajectory.

This technique can be thought of as a generalization of the approach used by Fair-Taylor(Fair & Taylor, 1983). Although the Fair-Taylor technique could concievably accommodate cyclical behavior, most users apply it in such a way that the algorithm restricts the tail of the path to be a fixed point solution. We generalize Taylor's algorithm by allowing the solution to lie in the *stable subspace* of a linear system which characterizes the asymptotic properties of the nonlinear system. In addition, we can explicitly characterize convergence to a periodic point.

Compute the steady state value x^* satisfying

$$h(x^*, ..., x^*) = 0 \tag{3}$$

Near the steady state, the linear first-order Taylor expansion of h about x^* provides a good approximation to the function h. We can apply the techniques presented in (Anderson & Moore, 1985) to determine the existence and uniqueness of perfect foresight solutions near the steady state.

That stability analysis will produce a matrix Q which restricts values of the endogenous variables to the stable subspace of the linearized system. For trajectories which approach a steady state, we can ultimately replace the non-linear system with the constraints codified in the matrix Q.

$$Q \begin{bmatrix} x_{T-\tau} - x^* \\ \vdots \\ x_T - x^* \\ \vdots \\ x_{T+\theta} - x^* \end{bmatrix} = 0$$

Consequently, for solutions which converge to the steady state, we can in principal compute solutions to whatever accuracy required by increasing the magnitude of T.

This corresponds to increasing T in Fair-Taylor's approach. We differ in that we use equation 1.2 instead of imposing

$$I \begin{bmatrix} x_{T+1} - x^* \\ \vdots \\ x_{T+\theta} - x^* \end{bmatrix} = 0$$

Our approach more accurately characterizes the dynamics of the nonlinear system near the steady state.

Restricting the end of the trajectory to the asymptotic stable linear subspace provides a better approximation. This improvement in the approximation is reflected in the length of the trajectory needed to achieve a given level of accuracy for the values at the beginning of the trajectory. In order to achieve a specific number of significant digits in the computation of the points near the beginning of the trajectory, setting the end of the trajectory equal to a specific constant will force us to compute a longer solution path than adopting our approach of restricting the solution to the asymptotic linear space. These observations are even more important when the terminal behavior of the system is more complicated than a fixed point.

2 A Non-Linear Example

2.1 A Money Demand Model

Consider the three equation non-linear system

$$\ln \frac{m_t}{p_t} = \alpha + \beta \ln(\rho + (\frac{p_{t+1} - p_t}{p_t})) \tag{4}$$

$$m_t - m_{t-1} = \gamma(m_{t-1} - \mu) + \delta s_t$$
$$s_t = \lambda s_{t-1}(1 - s_{t-1})$$

Where $L = 3, \tau = 1, \theta = 1, and\ 0 \leq \lambda,\ \alpha < 0\ ,\ \beta < 0\ ,\ \rho > 0,\ \gamma < 0\ , and\ m^* > 0$ exogenously given.

This example augments a simple forward looking money demand model with a quadratic map. The quadratic map is a simple looking function which displays many of the important characteristic properties of non linear function. As we vary the parameter λ we can study a model with fixed points, limit cycles, and with more complicated invariant sets. The appendix contains a summary of some important properties of the quadratic map.

2.2 The Money Demand Model:Dynamics Near the Fixed Point

We linearize the system about this steady state value in order to investigate the dynamics of the system near the steady state.

We want to investigate the model with initial conditions

$$m_0 = \bar{m}_0$$

$$p_0 = \bar{p}_0$$

$$s_0 = \bar{s}_0$$

and terminal conditions

$$\lim_{t \to \infty} \begin{bmatrix} m_t \\ p_t \\ s_t \end{bmatrix} = \begin{bmatrix} m^* \\ p^* \\ s^* \end{bmatrix}$$

A Taylor expansion about an arbitrary sequence of points produces:

$$H_{-1} = \begin{bmatrix} 0 & 0 & 0 \\ -(1+\gamma) & 0 & 0 \\ 0 & 0 & \lambda(2s_{t-1} - 1) \end{bmatrix}$$

$$H_0 = \begin{bmatrix} \frac{1}{m_t} & \frac{\beta p_{t+1} - \rho p_t - (p_{t+1} - p_t)}{p_t(\rho p_t + (p_{t+1} - p_t))} & 0 \\ 1 & 0 & -\delta \\ 0 & 0 & 1 \end{bmatrix}$$

$$H_1 = \begin{bmatrix} 0 & \frac{-\beta}{\rho p_t + (p_{t+1} - p_t)} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

At the fixed point this simplifies to:

$$H_{-1}|_{x^*} = \begin{bmatrix} 0 & 0 & 0 \\ -(1+\gamma) & 0 & 0 \\ 0 & 0 & \lambda(2s^* - 1) \end{bmatrix}$$

$$H_0|_{x^*} = \begin{bmatrix} \frac{1}{m^*} & \frac{\beta-\rho}{\rho p^*} & 0 \\ 1 & 0 & -\delta \\ 0 & 0 & 1 \end{bmatrix}$$

$$H_1|_{x^*} = \begin{bmatrix} 0 & \frac{-\beta}{\rho p^*} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

so that applying the methods of (Anderson & Moore, 1985) near the fixed point, the state space transition matrix is

$$A = \begin{bmatrix} (1+\gamma) & 0 & \delta\lambda(2s^*-1) \\ \frac{\rho/\beta}{m^*/p^*} & \frac{\beta-\rho}{\beta} & 0 \\ 0 & 0 & -\lambda(2s^*-1) \end{bmatrix}$$

The Q matrix consists of two shifted equations and one unstable left eigenvector if $\lambda < 1$

$$Q = \begin{bmatrix} -(1+\gamma) & 0 & 0 & 1 & 0 & -\delta \\ 0 & 0 & \lambda(2s^*-1) & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{-\rho/\beta}{m^*/p^*} & \left(\frac{(\beta-\rho)}{\beta} + (1+\gamma)\right) & \frac{-\delta\frac{\rho/\beta}{m^*/p^*}}{(\frac{\beta-\rho}{\beta}+\delta)} \end{bmatrix}$$

So that

$$B = \begin{bmatrix} 1+\gamma & 0 & \delta\lambda - 2\delta\lambda s^* \\ \frac{(1+\gamma)\rho p^*}{(\beta\gamma+\rho)m^*} & 0 & \frac{\delta\lambda\rho(-\beta+\rho)p^*(-1+2s^*)}{(\beta\gamma+\rho)m^*(\beta-\beta\lambda-\rho+2\beta\lambda s^*)} \\ 0 & 0 & \lambda - 2\lambda s^* \end{bmatrix}$$

3 Programs

Nonlinear Extension of Anderson-Moore Algorithm: Initial Setup

```
begin
  if $\neg succeedsQ(xStar := computeFixedPoint(h, xGuess))$
    then $fail$ comment: unable to compute fixed point
    else $H := linearize(h, xStar)$
        if $\neg hasSaddlePointPropertyQ(Q := andersonMoore(H))$
          then $fail$ comment: no saddle point property at this fixed point
          else
              if $\neg hasConvergedQ(xPath :=$
                $convergentPath(xHistory, h, Q, T_{MIN}, T_{MAX}))$
                then $fail$ comment: path has yet to converge
                else $success(xPath)$
              fi
        fi
  fi
end
```

3.1 Flow Chart

3.2 Mathematica Programs

In accordance with the flow chart in Figure 1, we solve equation 3 for the steady state value x^*. The money demand model of section 2.1 has two fixed points: one associated with $s^* = 0$, the other $\frac{\lambda-1}{\lambda}$. To demonstrate how the procedures work, we analyze the model at the fixed point $s^* = 0$, $m^* = \mu$, and $p^* = \mu \exp^{-(\alpha+\beta \ln(\rho))}$

The procedure steadyState solves for x^*. It takes a model expression defined within a given head by the user, as input. Then x^* is returned by the procedure as a function of the model parameters. For example, if we use the Mathematica Procedure exmplFunc to characterize $h(x_{t-\tau}, x_{t-\tau+1}, ..., x_{t+\theta-1}, x_{t+\theta})$

```
exmplFunc:=mdModel[
{
Log[m[t]/p[t]] - alph - (bet* Log[rho + (p[t+1] -p[t])/p[t]]),
m[t] - m[t-1] - gam *(m[t-1] - mu) - del *s[t],
s[t] - lam * s[t-1] * ( 1 - s[t-1])
}]
```

we obtain:

```
In[2]:= {vars,ssFuncs}=steadyState[exmplFunc]

Out[2]= {{m, p, s}, {Function[{alph, bet, del, gam, lam, mu, rho},

             -alph - bet Log[rho]
>      {mu, E                     mu, 0}],

>     Function[{alph, bet, del, gam, lam, mu, rho},

        del - del lam + gam lam mu
>      {--------------------------,
                  gam lam

         -alph - bet Log[rho]
        E                     (del - del lam + gam lam mu)  -1 + lam
>       --------------------------------------------------, --------}]}}
                            gam lam                           lam
```

Analysis of the behavior of the models in the vicinity of the fixed points will determine whether or not either of the fixed points possess appropriate long run properties.

This analysis continues by linearizing about the steady state. This will result in the H matrix.

The procedure ssLinearizeAndMakeHMat, computes the the maximum number of lags and leads, and builds the H matrix at a given fixed point.

```
In[3]:= hmat1=ssLinearizeAndMakeHMat[exmplFunc,ssFuncs[[1]]]

Out[3]= Function[{alph1, bet1, del1, gam1, lam1, mu1, rho1},

                           alph1     -1 + bet1    alph1     bet1
                 1   bet1 E      rho1            E      rho1
>    {{0, 0, 0, ---, ------------------------- - ---------------, 0, 0,
                mu1             mu1                    mu1

               alph1     -1 + bet1
         bet1 E      rho1
>      -(-------------------------), 0},
                    mu1

>     {-1 - gam1, 0, 0, 1, 0, -del1, 0, 0, 0},

>     {0, 0, -lam1, 0, 0, 1, 0, 0, 0}}]
```

The non-singularity assumption in the methods of (Anderson & Moore, 1985) requires that the sum of coefficient matrices in the H matrix has full rank. As the leading coefficient matrix could be singular, the procedure checkHSumsNonSingularQ determines whether or not to proceed with the algorithm.

```
In[4]:= checkHSumsNonSingularQ[hmat1]

Out[4]= True
```

Given the H matrix, the procedure transitionMatrix constructs a list of two functions, both as a function of the parameters. The first computes the state space transition matrix, and the second provides the auxiliary initial conditions.

```
In[5]:= {trans1,aux1}=transitionMatrix[hmat1]//Simplify

Out[5]= {Function[{alph1, bet1, del1, gam1, lam1, mu1, rho1},

>     {{0, 0, 0, 1, 0, 0}, {0, 0, 0, 0, 1, 0}, {0, 0, 0, 0, 0, 1},

>      {0, 0, 0, 1 + gam1, 0, del1 lam1},

                     1 - bet1
                 rho1              rho1
>      {0, 0, 0, ------------, 1 - ----, 0}, {0, 0, 0, 0, 0, lam1}}],
                       alph1       bet1
                 bet1 E

>    Function[{alph1, bet1, del1, gam1, lam1, mu1, rho1},

>     {{-1 - gam1, 0, 0, 1, 0, -del1}, {0, 0, -lam1, 0, 0, 1}}]}
```

linearSystemEigenvalues computes the eigenvalues for the given transition matrix function as a function of the parameters.

```
In[6]:= evals1=linearSystemEigenvalues[trans1]

Out[6]= Function[{alph1, bet1, del1, gam1, lam1, mu1, rho1},

                               bet1 - rho1
>    {0, 0, 0, 1 + gam1, lam1, -----------}]
                                  bet1
```

The procedure `largeEVSpanningVectors` computes the vectors spanning the invariant space associated with roots outside the open disk, as a function of the parameters. However, the user is required to determine the “large” roots from the above function list, then enter, as input, the list of integers indexing the large roots.

```
In[7]:= evecs1=largeEVSpanningVectors[trans1,evals1,{6}]

Out[7]= Function[{alph1, bet1, del1, gam1, lam1, mu1, rho1},

                  -bet1 + bet1 lam1 + rho1
>    {{0, 0, 0, -(------------------------),
                       bet1 del1 lam1

        alph1     -1 + bet1
       E      rho1          (bet1 gam1 + rho1) (-bet1 + bet1 lam1 + rho1)
>      ------------------------------------------------------------------, 1}}
                                 bet1 del1 lam1

>     ]
```

`theQMatrix` computes the Q matrix given the function of auxiliary initial conditions and the function of selected eigenvectors. Given the Q matrix function as input, `theBMatrix` computes the B matrix. However, an intermediate checking procedure, `checkQRightNonSingularQ`, identifies to the user whether or not a B matrix exists.

```
In[8]:= q1=theQMatrix[aux1,evecs1]//Simplify

Out[8]= Function[{alph1, bet1, del1, gam1, lam1, mu1, rho1},

>    {{-1 - gam1, 0, 0, 1, 0, -del1}, {0, 0, -lam1, 0, 0, 1},

                bet1 - bet1 lam1 - rho1
>     {0, 0, 0, -----------------------,
                    bet1 del1 lam1

        alph1     -1 + bet1
       E      rho1          (bet1 gam1 + rho1) (-bet1 + bet1 lam1 + rho1)
>      ------------------------------------------------------------------, 1}}
                                 bet1 del1 lam1

>     ]
```

```
In[9]:= checkQRightNonSingularQ[q1]

Out[9]= True

In[10]:= b1=theBMatrix[q1]//Simplify

Out[10]= Function[{alph1, bet1, del1, gam1, lam1, mu1, rho1},

>    {{1 + gam1, 0, del1 lam1},

                       1 - bet1
        (1 + gam1) rho1
>      {------------------------, 0,
        alph1
       E      (bet1 gam1 + rho1)

                                          1 - bet1
             del1 lam1 (bet1 - rho1) rho1
>      ---------------------------------------------------}, {0, 0, lam1}}]
        alph1
       E      (bet1 - bet1 lam1 - rho1) (bet1 gam1 + rho1)
```

Fixed Points Provided $-2 < \gamma < 0$, $\rho > 0$ and $\beta < 0$, this model has two fixed points. For all values of $\lambda \neq 1$ one and only one of these fixed points has the saddle point property.

$$x^* = \begin{cases} x_1^* = \begin{bmatrix} \mu \\ \mu \exp^{-(\alpha+\beta\ln(\rho))} \\ 0 \end{bmatrix} & -2 < \lambda < 1 \\ \\ x_2^* = \begin{bmatrix} \mu - (\frac{\delta}{\gamma})\frac{\lambda-1}{\lambda} \\ \mu - (\frac{\delta}{\gamma})\frac{\lambda-1}{\lambda}\exp^{-(\alpha+\beta\ln(\rho))} \\ \frac{\lambda-1}{\lambda} \end{bmatrix} & 1 < \lambda < 3 \end{cases}$$

When $\lambda = 1$, the transition matrix has an eigenvalue of $+1$ and the system undergoes a transcritical bifurcation. The linearized system is not helpful for characterizing the dynamics near the steady state The analysis reveals that, when $\lambda \approx 1$, it may be worthwhile to set the terminal portion of the path equal to the steady state value.

Rate of Convergence to Steady State Since B has eigenvalues $0, 1+\gamma, \lambda(1-2s^*)$, the parameters γ and λ determine the asymptotic rate of convergence of the state variables to their steady state values.

Impulse Response Functions

- s_t Future values of s_t depend only on s_{t-1}. The parameter λ determines the magnitude of the impact of values of s_{t-1} on s_t. A unit change in s_{t-1} produces a λ unit change in s_t near steady states 1 when $0 < \lambda < 1$ or a $2 - \lambda$ unit change near steady state 2 when $1 < \lambda < 3$.
- m_t Future values of m_t depend only on s_{t-1} and m_{t-1}. The parameters δ and λ determine the magnitude of the impact of values of s_{t-1} on m_t. A unit change in s_{t-1} produces a $\delta\lambda$ unit change in m_t near steady states 1 when $0 < \lambda < 1$ and a $\delta(2 - \lambda)$ unit change near steady state 2 when $1 < \lambda < 3$. The parameter γ determines the magnitude of the impact of values of m_{t-1} on m_t. A unit change in m_{t-1} produces a $1+\gamma$ unit change in m_t near either steady state.
- p_t Future values of p_t depend only on s_{t-1} and m_{t-1}. The parameters $\alpha, \beta, \delta, \gamma, \lambda$, and ρ determine the magnitude of the impact of values of s_{t-1} on p_t. A unit change in s_{t-1} produces a $\frac{\delta\lambda\rho(\rho-\beta)e^{-(\alpha+\beta\log(\rho))}}{(\beta\gamma+\rho)(\beta\lambda-\beta+\rho)}$ unit change in p_t near the first steady state and a $\frac{\delta(2-\lambda)\rho(\rho-\beta)e^{-(\alpha+\beta\log(\rho))}}{(\beta\gamma+\rho)(\beta(2-\lambda)-\beta+\rho)}$ unit change in p_t near the second steady state. A unit change in m_{t-1} produces a $\frac{(1+\gamma)\rho e^{-(\alpha+\beta\log(\rho))}}{\beta\gamma+\rho}$ unit change in p_t for either steady state.

Deviations of lagged values of p_t from steady state do not affect the values of any state variables.

Quantitative Accuracy of Linearization Figure 7 compares the solutions for the model assuming linearity and non linearity. Figure 2 graphs $\frac{\Delta p_0}{\Delta m_{-1}}$ as a function of Δm_{-1}. When Δm_{-1} is near zero, the impulse response to a shock in m_{-1} is close to the value provided by the linearization about the steady state. The impulse response diverges from the asymptotic value as the size of the shock to m_{-1} increases. For these particular values of the parameters, a 50% shock to m_{-1} changes the impulse respone by about 10%.

Qualitative Accuracy of Linearization Figure 7 compares the solutions for the model assuming linearity and non linearity.

3.3 Other Models

3.3.1 Opportunistic Disinflation

$$i_t^s = \begin{cases} \alpha_{50} + \pi_{t-1} + \alpha_{51} y_{t-1} + \alpha_{52}(\pi_{t-1} - \pi^*) & \text{conventional policy rule} \\ \alpha_{50} + \pi_{t-1} + \alpha_{51} y_{t-1} + \alpha_{52}(\pi_{t-1} - \tilde{\pi}^*) & \text{gradualist version of conventional policy rule} \\ \alpha_{50} + \pi_{t-1} + \alpha_{51} y_{t-1} + \alpha_{52} f(\pi_{t-1} - \tilde{\pi}^*) & \text{opportunistic policy rule} \end{cases}$$

where

$$\tilde{\pi}_t = \alpha_{53}\pi^* + \alpha_{54}\pi^\dagger$$

and

$$f(\pi - \tilde{\pi}) = \begin{cases} 2(\pi - \tilde{\pi} - \delta) & \pi - \tilde{\pi} > \delta \\ 0 & -\delta \geq \pi - \tilde{\pi} \geq \delta \\ 2(\pi - \tilde{\pi} + \delta) & -\delta > \pi - \tilde{\pi} \end{cases}$$

$$i_t^l = \alpha 40 + \frac{1 - \alpha_{41}}{1 - \alpha_{41}^1 7} E_t(\sum_{j=0}^{1} 6\alpha_{41}^j i_{t+j}^s r_t^l = i_t^l - E_t(p_{t+4} - p_t)$$

$$c_t = \alpha_0 + \alpha_1 c_{t-1} + \alpha_2 y_t^p - \alpha_3 r_t^l$$

$$y_t^p = \frac{(1 - \alpha_4)}{(1 - \alpha_4^9)} \sum_{j=0}^{8} \alpha_4^j y_{t+j}$$

$$v_t = \alpha_5 + \alpha_6 v_{t-1} - \alpha_7 v_{t-2} + \alpha_8 y_t - \alpha_9 y_{t-1} - \alpha_{10} r_t^l$$

$$z_t = \alpha_{11} + \alpha_{12} z_{t-1} + \alpha_{13} y_t + \alpha_{14} y_{t-1} - \alpha_{15} r_t^l$$

$$n_t = \alpha_{16} n_{t-1} - \alpha_{17} y_t + \alpha_{18} y_t^w - \alpha_{19} e_t$$

$$g_t = \alpha_{20} + \alpha_{21} g_{t-1}$$

$$y_t = c_t + z_t + v_t + n_t + g_t - 1$$

$$v_t = \alpha_{22}(x_t - p_t) + \alpha_{23}(x_{t-1} - p_{t-1}) + \alpha_{24}(x_{t-2} - p_{t-2}) + \alpha_{25}(x_{t-3} - p_{t-3})$$

$$x_t - p_t = E_t(\alpha_{26} v_t + \alpha_{27} v_{t+1} + \alpha_{28} v_{t+2} + \alpha_{29} v_{t+3}) + \alpha_{30} E_t(\alpha_{31} y_t + \alpha_{32} y_{t+1} + \alpha_{33} y_{t+2} + \alpha_{34} y_{t+3})$$

$$p_t = \alpha_{35} x_t + \alpha_{36} x_{t-1} + \alpha_{37} x_{t-2} + \alpha_{38} x_{t-3}$$

3.3.2 Blanchard's Social Security and Capital Accumulation Example Model

Blanchard(Blanchard & Fischer, 1989) describes a model that has the saddle point property:

$$u'(w_t - s_t) = \frac{1 + r_{t+1}}{1 + \theta} u'((1 + r_{t+1}) s_t)$$

$$s_t = (1 + n) k_t$$

$$w_t = f(k_{t-1}) - k_{t-1} f'(k_{t-1})$$

$$r_t = f'(k_{t-1})$$

3.3.3 Endogenous Growth Model

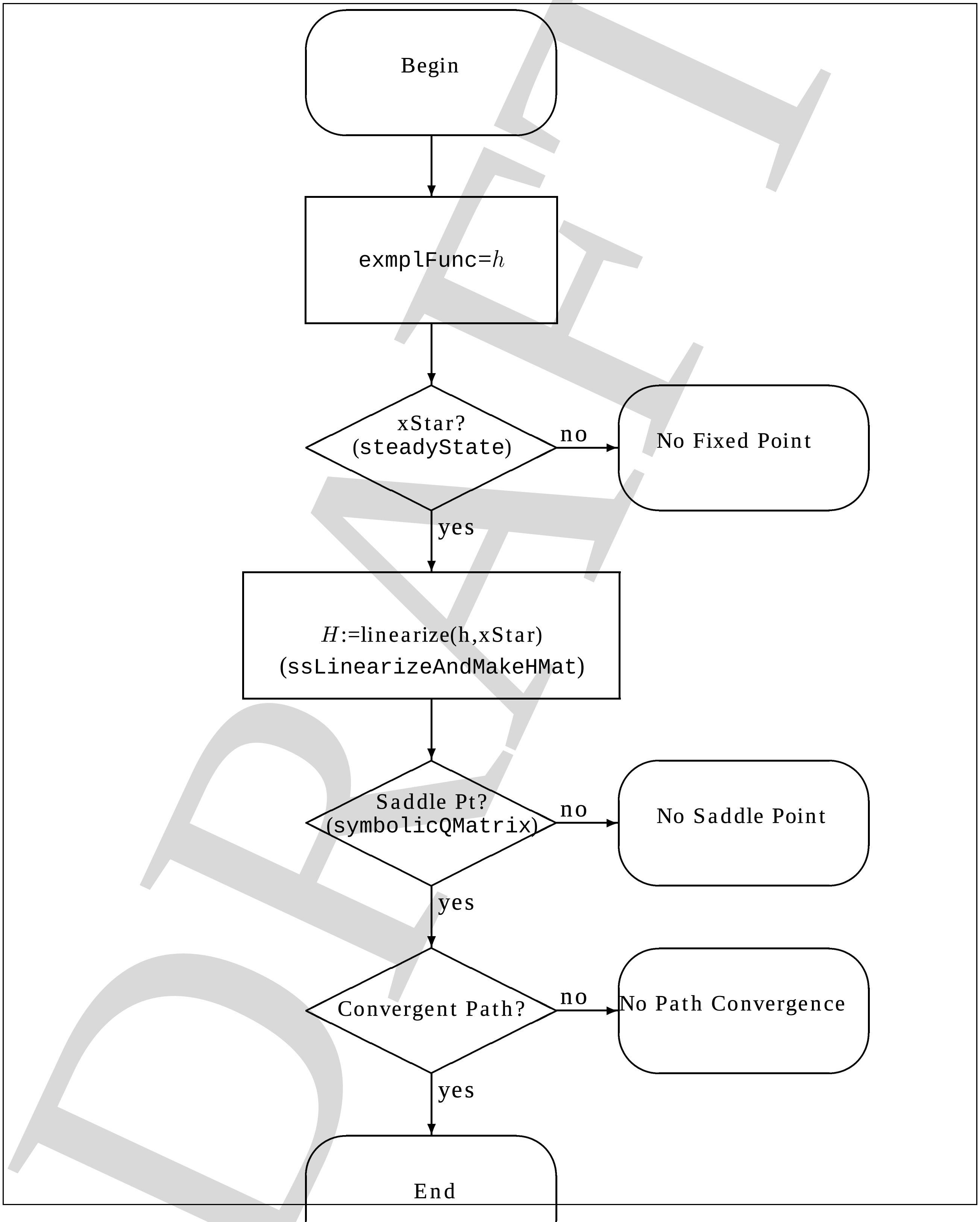

Figure 1: Flow Chart

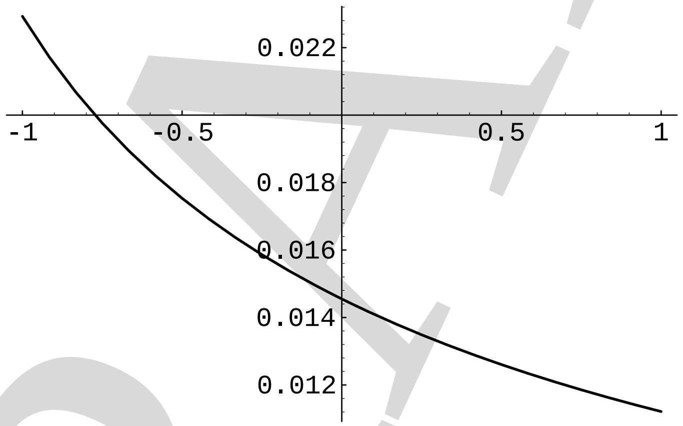

Figure 2: $\frac{\Delta p_0}{\Delta m_{-1}}$ versus Δm_{-1}

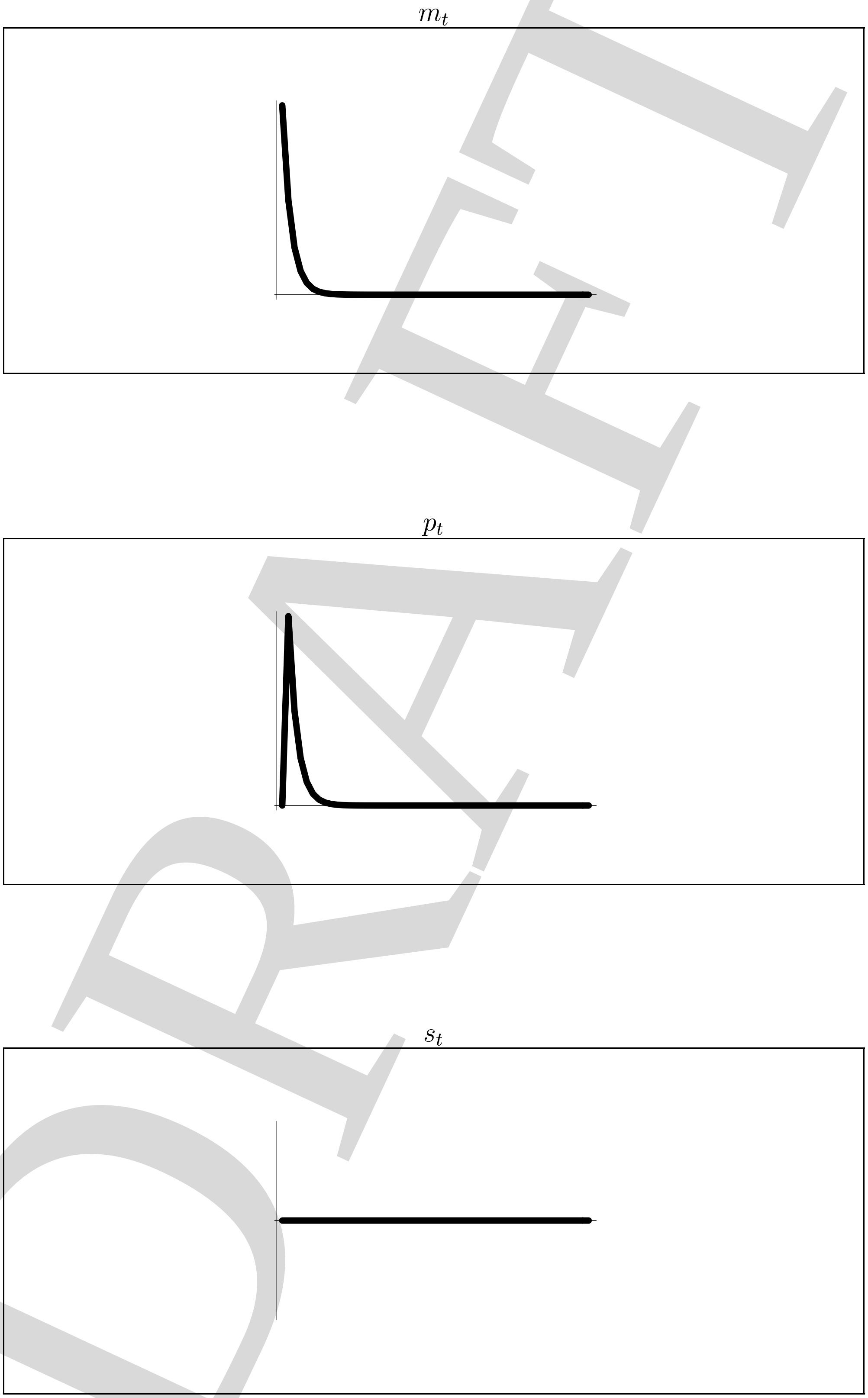

Figure 3: Impulse Response Functions for Deviations of m_{-1} from Steady State

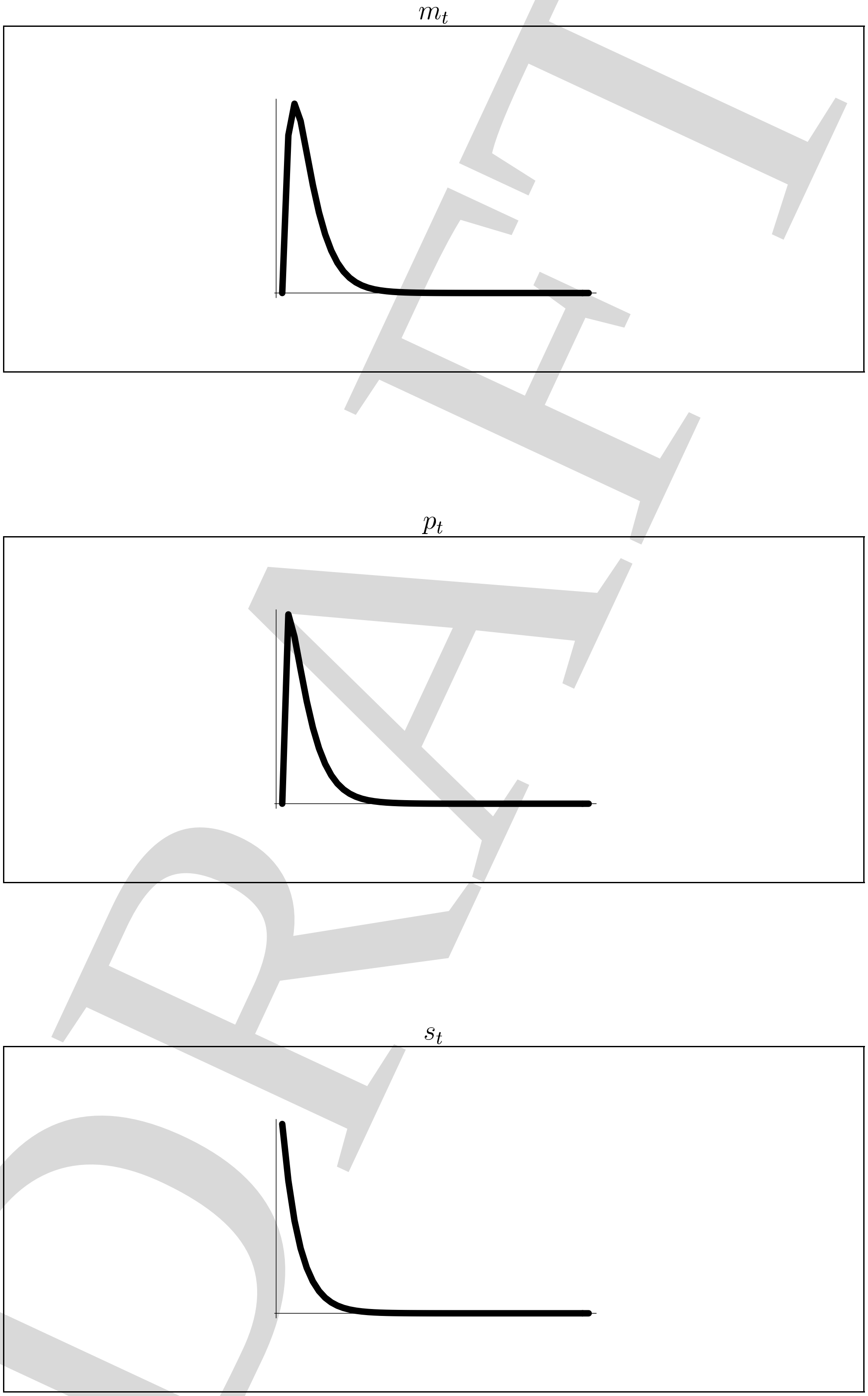

Figure 4: Impulse Response Functions for Deviations of s_{-1} from Steady State, with λ small.

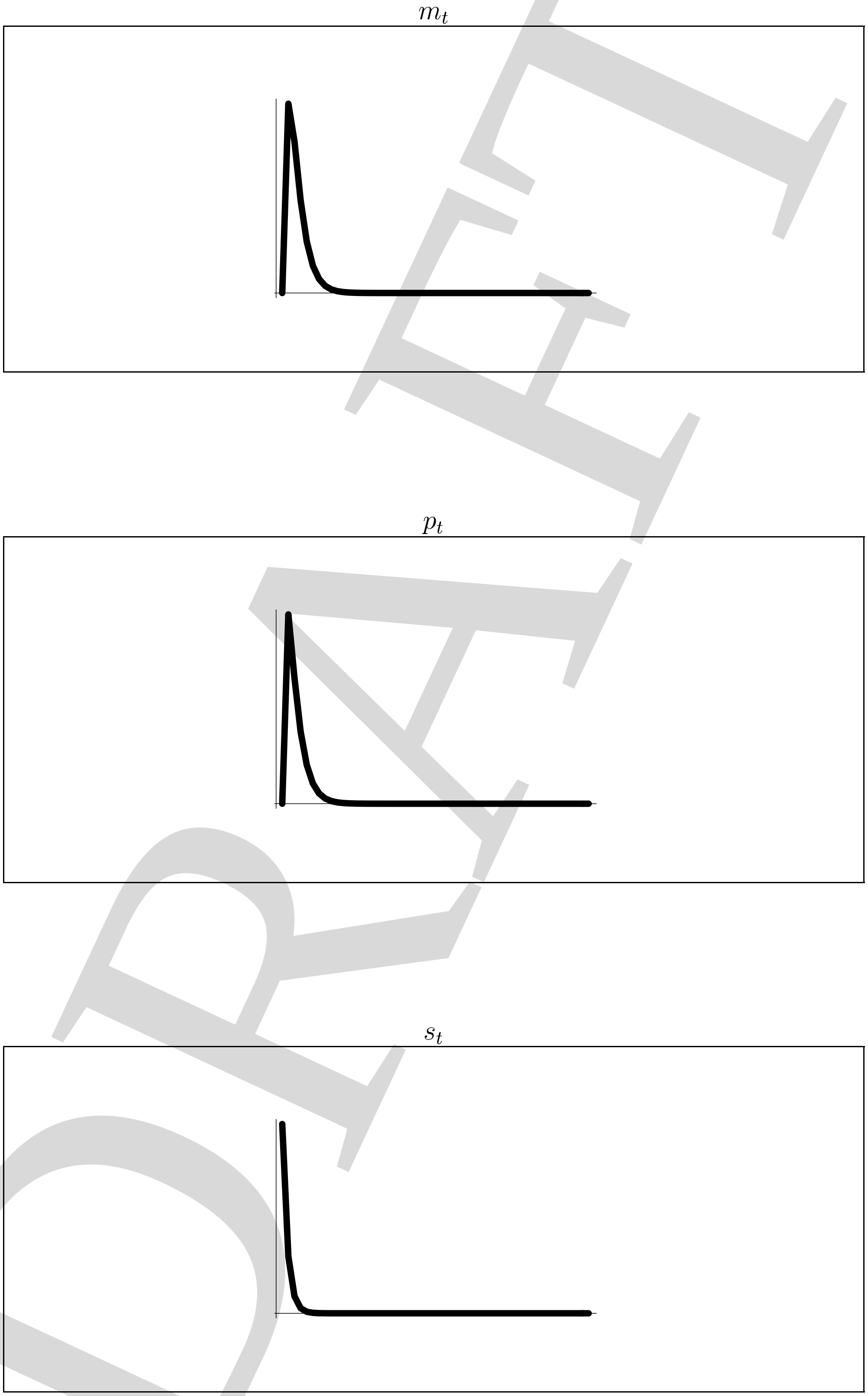

Figure 5: Impulse Response Functions for Deviations of s_{-1} from Steady State, with λ intermediate.

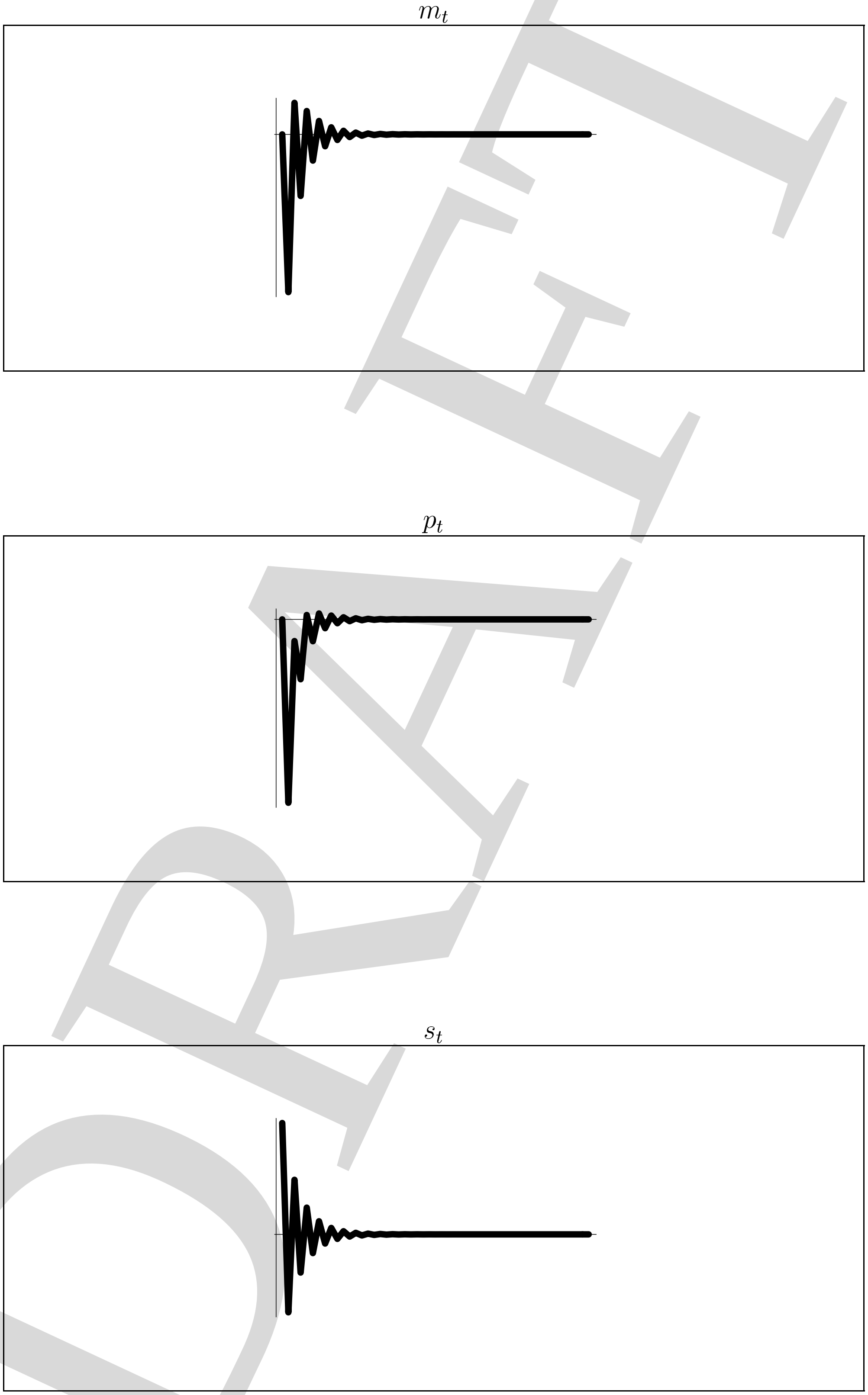

Figure 6: Impulse Response Functions Deviations of m_t, and s_{-1} from Steady State, with λ large.

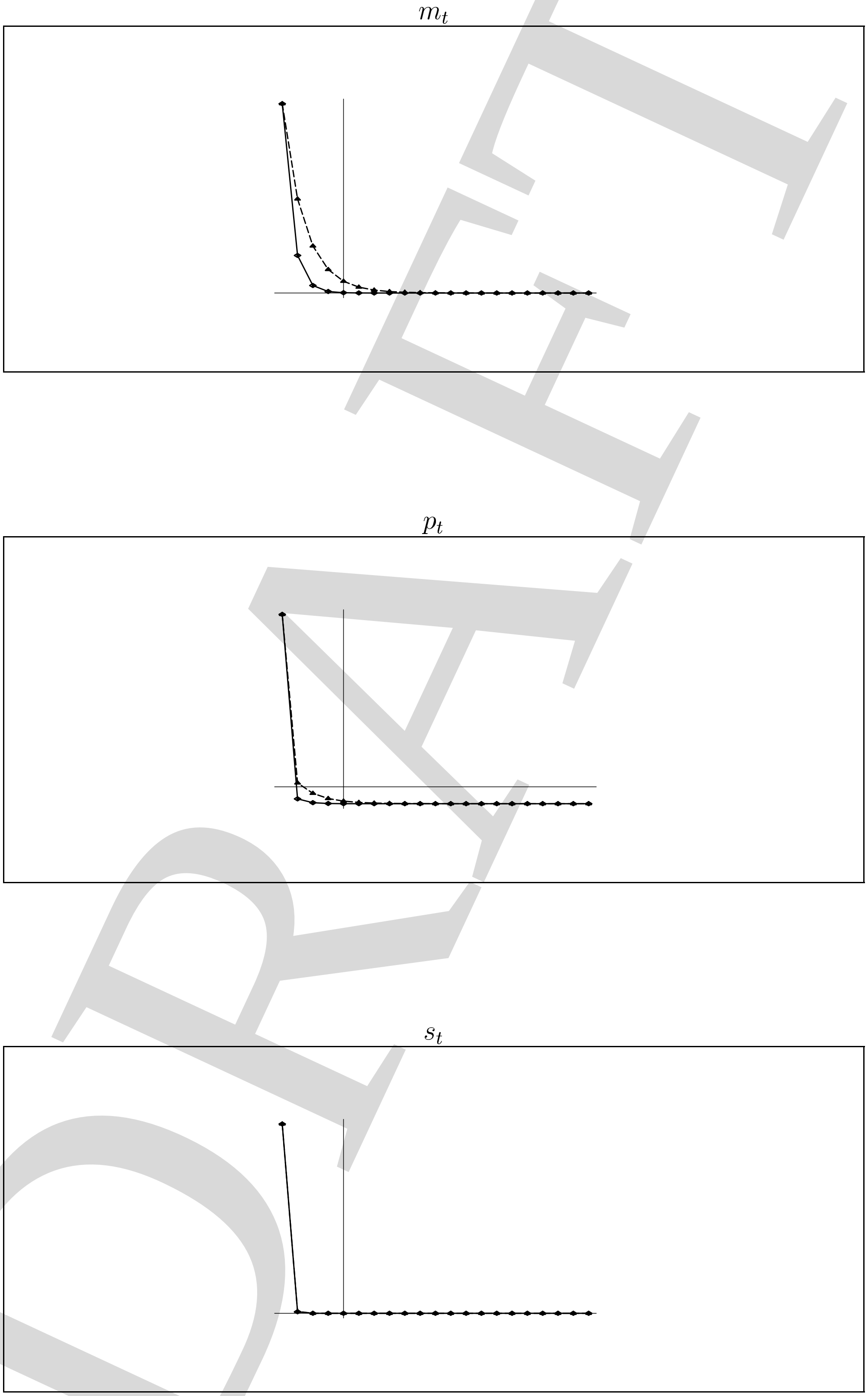

Figure 7: Linear vs. Non Linear Response Functions for Deviations of s_{-1} from Steady State, with λ small.

Program	Domain $\longrightarrow$ Range
steadyState	$f \longrightarrow \{ss_1(\beta), \ldots, ss_k(\beta)\}$
Computes a list of functions of the parameters. Each function returns a list of steady state values.	
ssLinearizeAndMakeHMat	$\{f, ss_i(\beta)\} \longrightarrow h_i(\beta)$
Given the function and a function which computes a particular steady state value for given parameters, this routine returns a function providing the linearization at the particular fixed point.	
transitionMatrix	$h_i(\beta) \longrightarrow \{A_i(\beta), Z_i(\beta)\}$
Constructs a list consisting of two functions. The first computes the transition matrix as a function of the parameters, the second provides the auxiliary initial conditions as a function of the parameters.	
linearSystemEigenvalues	$A_i(\beta) \longrightarrow \Lambda_i(\beta)$
Computes a function that computes the eigenvalues for the given transition matrix function.	
largeEVSpanningVectors	$\{A_i(\beta), \Lambda_i, \{e_1, \ldots, e_k\}\} \longrightarrow V_{i,\{e_1,\ldots,e_k\}}(\beta)$
Computes a function that computes the vectors spanning the invariant space characterized by the selected eigenvalues as a function of the parameters.	
theQMatrix	$\{Z_i(\beta), V_{i,\{e_1,\ldots,e_k\}}(\beta), \} \longrightarrow Q_{i,\{e_1,\ldots,e_k\}}(\beta)$
Computes a function returning the asymptotic linear constraints as a function of the parameters.	
theBMatrix	$Q_{i,\{e_1,\ldots,e_k\}}(\beta) \longrightarrow B_{i,\{e_1,\ldots,e_k\}}(\beta)$
Computes a function returning the asymptotic linear constraints normalized to provide a vector auto regression as a function of the parameters.	

Table 1: Functional Programming Characterization of Model Features

4 Conclusion

The asymptotic linearization facilitates the computation of the non linear path converging to the steady state. Applying the techniques of this section to compute the path of money and prices for any set of initial conditions, we found that for any given horizon, the procedure required only 6 Newton steps to compute the path. In addition, we found that a trajectory of length 6 was sufficient to obtain the values of m and p to 10 decimal places.

A Mathematica Package

```
BeginPackage["AsymptoticLinearization`",
        {"LinearAlgebra`MatrixManipulation`", "DiscreteMath`Combinatorica`"}]
DeclarePackage["newAim`",
        {"transitionMatrixAndAuxiliaryInitialConditions",
                "unobviousShiftRights","obviousShiftRights","obviousShifts"}]

(*
Routines to Manipulate NonLinear Models for AIM Extended Path Algorithm

Gary S. Anderson
    &
Vi-Min Choong

August 13, 1996


*)

steadyState::usage=
"     steadyState[modelFunction]
        Returns {List of functions of the parameters
                with each function returning steady state values}"
makeFuncList::usage=
"     makeFuncList[modelFunction]
        Returns {List of functions of the parameters with each
                function returning symbolic form steady state as a
                        function of parameters}"
ssLinearizeAndMakeHMat::usage=
"     ssLinearizeAndMakeHMat[modelFunction,selectedFixedPointFunction]
        Returns {Function providing the linearization
                        at the selected fixed point}"
checkHSumsNonSingularQ::usage=
"     checkHSumsNonSingularQ[HMatrixFunction]
        Returns {Decision on whether to proceed or not with algorithm}"
transitionMatrix::usage=
"     transitionMatrix[HMatrixFunction]
        Returns {List of the stateSpaceTransitionMatrixFunction
                        and the auxiliaryInitialConditionsFunction}"
linearSystemEigenvalues::usage=
"     linearSystemEigenvalues[stateSpaceTransitionMatrixFunction]
        Returns {Function that computes the eigenvalues given the
                        stateSpaceTransitionMatrixFunction}"
largeEVSpanningVectors::usage=
"     largeEVSpanningVectors[stateSpaceTransitionMatrixFunction,
                linearSystemEigenvaluesFunction,largeRootsList]
        Returns {Function that computes the vectors spanning
                the invariant space characterized by the large
                        eigenvalues}"
theQMatrix::usage=
"     theQMatrix[auxiliaryInitialConditionsFunction,
                largeEVSpanningVectorsFunction]
        Returns {Function returning aymptotic linear constraints}"
checkQRightNonSingularQ::usage=
"     checkQRightNonSingularQ[QMatrixFunction]
        Returns {Decision on whether to proceed or not with algorithm,
                that is, whether or not there is a B matrix}"
theBMatrix::usage=
"     theBMatrix[theQMatrix]
        Returns {Function returning aymptotic linear constraints
                normalized to provide a vector auto regression}"



(* Enter NonLinear model Function here *)

exmplFunc:=mdModel[
{
Log[m[t]/p[t]] - alph - (bet* Log[rho + (p[t+1] -p[t])/p[t]]),
m[t] - m[t-1] - gam *(m[t-1] - mu) - del *s[t],
s[t] - lam * s[t-1] * ( 1 - s[t-1])
}]


Begin["private`"]

(* Routine steadyState *)

steadyState[func_]:=
steadyState[func]=
Module[{eqns,vars,mathematicaFuncList,params,ss},
With[{fn=func //. t -> $timeMarker},
With[{vars=Union[(Head /@ Union[(Part @@ Prepend[#,fn]) & /@
                Position[fn,x_[$timeMarker]]]),
(Head /@ Union[(Part @@ Prepend[#,fn]) & /@
                Position[fn,x_[$timeMarker + y_]]]])]},
With[{mathematicaFuncList={Sqrt,Exp,Log,Sin,Cos,Tan,Csc,Sec,Cot,ArcSin,
        ArcCos,ArcTan,ArcCsc,ArcSec,ArcCot,Sinh,Cosh,Tanh,Csch,Sech,
                Coth,ArcSinh,ArcCosh,ArcTanh,ArcCsch,ArcSech,ArcCoth,
                        Abs,Pi,E,Exp,Re,Im,Plus,Times,Divide,Power,List,
                                Sqrt,Derivative,D,Integrate,Head[func],
                                        $timeMarker}},
With[{params=Complement[Union[
        (Part @@ Prepend[#,fn]) & /@ Position[fn,x_Symbol]],
                mathematicaFuncList,vars]},
        With[{eqns=(#==0 & /@ Level[fn,{2}])
                        /. {$timeMarker + _ -> $timeMarker}},
                With[{ss=Partition[Flatten[
                        Solve[eqns,(#[$timeMarker]& /@ vars)]],
                                Count[vars,x_Symbol]]},
{vars, Function @@ {params, #}& /@
        If[ss == {}, func,((#[$timeMarker]& /@ vars) /. ss)]}]
]]]]]



(* Routine makeFuncList *)

makeFuncList[func_]:=
makeFuncList[func]=
Module[{fn,vars,paramsSpec,params},
With[{fn=func //. t -> $timeMarker},
With[{vars=Union[(Head /@ Union[(Part @@ Prepend[#,fn]) & /@
                Position[fn,x_[$timeMarker]]]]),
(Head /@ Union[(Part @@ Prepend[#,fn]) & /@
                Position[fn,x_[$timeMarker + y_]]]])]},
With[{mathematicaFuncList={Sqrt,Exp,Log,Sin,Cos,Tan,Csc,Sec,Cot,ArcSin,
        ArcCos,ArcTan,ArcCsc,ArcSec,ArcCot,Sinh,Cosh,Tanh,Csch,Sech,
                Coth,ArcSinh,ArcCosh,ArcTanh,ArcCsch,ArcSech,ArcCoth,
                        Abs,Pi,E,Exp,Re,Im,Plus,Times,Divide,Power,List,
                                Sqrt,Derivative,D,Integrate,Head[func],
                                        $timeMarker}},
With[{paramsSpec=Complement[Union[
        (Part @@ Prepend[#,fn]) & /@ Position[fn,x_Symbol]],
                mathematicaFuncList,vars]},
With[{params=Unique[ToString[#]]&/@paramsSpec},
        Function @@ {params, (# @@ params)& /@ vars}
]]]]]]



(* Routine ssLinearizeAndMakeHMat *)

ssLinearizeAndMakeHMat[func_,fixedPtFunc_]:=
ssLinearizeAndMakeHMat[func,fixedPtFunc]=
Module[{fn,vars,paramsSpec,ssPrelim,fixedPt,sStates,tSpec,tAppearing,
                fns,vblsPresent,tRange,tVal,ss,
                params},
With[{fn=func //. t -> $timeMarker},
With[{vars=Union[(Head /@ Union[(Part @@ Prepend[#,fn]) & /@
                Position[fn,x_[$timeMarker]]]),
(Head /@ Union[(Part @@ Prepend[#,fn]) & /@
                Position[fn,x_[$timeMarker + y_]]]])]},
With[{mathematicaFuncList={Sqrt,Exp,Log,Sin,Cos,Tan,Csc,Sec,Cot,ArcSin,
        ArcCos,ArcTan,ArcCsc,ArcSec,ArcCot,Sinh,Cosh,Tanh,Csch,Sech,
                Coth,ArcSinh,ArcCosh,ArcTanh,ArcCsch,ArcSech,ArcCoth,
                        Abs,Pi,E,Exp,Re,Im,Plus,Times,Divide,Power,List,
                                Sqrt,Derivative,D,Integrate,Head[func],
                                        $timeMarker}},
With[{paramsSpec=Complement[Union[
        (Part @@ Prepend[#,fn]) & /@ Position[fn,x_Symbol]],
                mathematicaFuncList,vars]},
With[{params=Unique[ToString[#]]&/@paramsSpec},
With[{ssPrelim= fixedPtFunc /.
                Thread[Rule[paramsSpec,params]],
        fns=(func[[1]] /.
                Flatten[{t -> tSpec,Thread[Rule[paramsSpec,params]]}])},
With[{sStates={MapThread[#1[$steadyStateT] -> #2 &,
                {vars,ssPrelim[[2]]}]}},
With[{ssSubs=sStates/.$steadyStateT->Blank[],
                tAppearing=Union[#[[1]]& /@ Flatten[Cases[
                        fns,(Alternatives @@ vars)[#],Infinity]& /@
                                {tSpec, tSpec + _}]] - tSpec},
With[{tRange=Prepend[{Min[tAppearing],Max[tAppearing]},tVal]},
With[{vblsPresent=Flatten[Table[#[tSpec+tVal],tRange]& /@ vars]},
With[{varSubs=Thread[#1->#2&[vblsPresent,Unique[ToString[#[[0]]]&
                /@ vblsPresent]]]},
With[{sslin= Part[Function[ss,
Function @@ {params,
                CoefficientList[
                 (Plus @@#&/@ Outer[(D[#1,#2]/.ss)(#2-(#2/.ss))&,
                                fns,vblsPresent]),
                Flatten[
                Transpose[Partition[vblsPresent,
                                Max[tAppearing]-Min[tAppearing]+1]]]]]}] /@
                ssSubs,1]},
With[{maxLag=Min[tAppearing],
        maxLead=Max[tAppearing],
                neq=Length[fns],
                        coeffLists=Part[sslin,2]},
With[{dpth=(maxLead-maxLag+1)neq},
With[{vals=((Level[#,{dpth}]&) /@ # &)[coeffLists],
                zeroRow={Table[0,{dpth}]}},
With[{further=
                MapThread[Function[{x,y},
                {#,Select[Position[y,#],Length[#]==dpth&]}& /@ x],
                {vals,coeffLists}]},
Function @@ {params,
AppendColumns @@
(If[Head[#]==Integer,zeroRow,#,#]& /@
Map[Function[x,Plus @@(Take[FoldList[Plus,0,#],-1]&/@
                (#[[2]]/.{1->0,2->#[[1]]}& /@ x))],further])}
]]]]]]]]]]]]]]]]



(* Routine checkHSumsNonSingularQ *)

checkHSumsNonSingularQ[hMatFunc_]:=
checkHSumsNonSingularQ[hMatFunc]=
With[{hMat=Part[hMatFunc,2]},
With[{blksum=Sum[SubMatrix[hMat,
                {1,i},{Length[hMat],Length[hMat]}],
                {i,1,Length[hMat[[1]]] - Length[hMat] +1 ,Length[hMat]}]},
        If[NullSpace[blksum] == {}, True, False]
]]



(* Routine transitionMatrix *)

transitionMatrix[hMatFunc_]:=
transitionMatrix[hMatFunc]=
With[{hMat=Part[hMatFunc,2]},
        With[{es=transitionMatrixAndAuxiliaryInitialConditions[hMat]},
                With[{params=Part[hMatFunc,1]},
Function @@ {params,#}& /@
        {Part[es,3],Part[es,2]}]]]



(* Routine linearSystemEigenvalues *)

linearSystemEigenvalues[transMatFunc_]:=
linearSystemEigenvalues[transMatFunc]=
Module[{transMat,ev,params},
With[{transMat=Part[transMatFunc,2]},
        With[{ev=Eigenvalues[Transpose[transMat]]},
                With[{params=Part[transMatFunc,1]},
Function @@ {params,ev}]]]]



(* Routine largeEVSpanningVectors *)
```

```
largeEVSpanningVectors[transMatFunc_,eigenValFunc_,largeRoots_List]:=
largeEVSpanningVectors[transMatFunc,eigenValFunc,largeRoots]=
Module[{transMat,eigenValues,eVecs,params},
With[{transMat=Part[transMatFunc,2]},
        With[{eigenValues=Part[eigenValFunc,2]},
        With[{params=Part[transMatFunc,1]},
Function @@ {params,
                Flatten[NullSpace[Transpose[transMat] -
                #(IdentityMatrix[Length[transMat]])]& /@
                        (eigenValues[[#]]& /@ largeRoots),1]
}]]]]



(* Routine theQMatrix *)

theQMatrix[auxCond_,spanningEV_]:=
theQMatrix[auxCond,spanningEV]=
Module[{params,moreRows},
With[{params=Part[auxCond,1]},
Block[{moreRows=Part[spanningEV,2]},
Function @@ {params,AppendColumns[auxCond[[2]],moreRows]}
]]]



(* Routine checkQRightNonSingularQ *)

checkQRightNonSingularQ[qMatFunc_]:=
checkQRightNonSingularQ[qMatFunc]=
With[{qMat=Part[qMatFunc,2]},
With[{qRight=SubMatrix[qMat,
                {1,Length[qMat[[1]]]-Length[qMat]+1},
                        {Length[qMat],Length[qMat]}]},
If[NullSpace[qRight] == {},True, False]
]]



(* theBMatrix *)

theBMatrix[qMatrix_]:=
theBMatrix[qMatrix]=
Module[{qmat,qRight,bmat,params},
With[{qmat=Part[qMatrix,2]},
        With[{qRight=SubMatrix[qmat,
                {1,1+Length[qmat[[1]]]-Length[qmat]},
                        {Length[qmat],Length[qmat]}]},
        With[{bmat=-Inverse[qRight] . qmat},
                With[{params=Part[qMatrix,1]},
If[NullSpace[qRight] == {},
        Function @@ {params,
                TakeColumns[bmat,Length[bmat[[1]]]-Length[bmat]]},
                        Print["right block singular: no b matrix"]]
]]]]]



End[]
EndPackage[]


(* The following sequence of commands will execute the
        routines in the package given the money demand model,

{vars,ssFuncs}=steadyState[exmplFunc]
hmat1=ssLinearizeAndMakeHMat[exmplFunc,ssFuncs[[1]]]
hmat2=ssLinearizeAndMakeHMat[exmplFunc,ssFuncs[[2]]]
checkHSumsNonSingularQ[hmat1]
checkHSumsNonSingularQ[hmat2]
{trans1,aux1}=transitionMatrix[hmat1]
{trans2,aux2}=transitionMatrix[hmat2]
evals1=linearSystemEigenvalues[trans1]
evals2=linearSystemEigenvalues[trans2]
evecs1=largeEVSpanningVectors[trans1,evals1,{6}]
evecs2=largeEVSpanningVectors[trans2,evals2,{6}]
q1=theQMatrix[aux1,evecs1]
q2=theQMatrix[aux2,evecs2]
checkQRightNonSingularQ[q1]
checkQRightNonSingularQ[q2]
b1=theBMatrix[q1]
b2=theBMatrix[q2]
# @@ {1,2,3,4,5,6,7} & /@
{ssFuncs[[1]],ssFuncs[[2]],
hmat1,trans1,aux1,evals1,evecs1,q1,b1,
hmat2,trans2,aux2,evals2,evecs2,q2,b2}//N
*)
```

References

Anderson, Gary, & Moore, George. 1985. A Linear Algebraic Procedure For Solving Linear Perfect Foresight Models. *Economics Letters*, **17**.

Blanchard, Olivier Jean, & Fischer, Stanley. 1989. *Lectures on Macroeconomics*. The MIT Press. Chap. The Overlapping Generations Model.

Fair, Ray, & Taylor, John. 1983. Solution and Maximum Likelihood Estimation of Dynamic Rational Expectations Models. *Econometrica*, **51**.

Guckenheimer, John, & Holmes, Philip. 1983. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer-Verlag.