Parameterized Expectations Algorithm

Wouter J. Den Haan
London School of Economics

© 2011 by Wouter J. Den Haan

June 29, 2011
Overview

- Two PEA algorithms
- Explaining stochastic simulations PEA
- Advantages and disadvantages
- Improvements of Maliar, Maliar & Judd
- Extensions
  - learning
  - combining with perturbation
Model

\[ c_t^{-\nu} = \mathbb{E}_t \left[ \beta c_{t+1}^{-\nu} \left( \alpha z_{t+1} k_{t+1}^{\alpha-1} + 1 - \delta \right) \right] \]

\[ c_t + k_{t+1} = z_t k_t^\alpha + (1 - \delta) k_t \]

\[ \ln(z_{t+1}) = \rho \ln(z_t) + \varepsilon_{t+1} \]

\[ \varepsilon_{t+1} \sim N(0, \sigma^2) \]

\[ k_1, z_1 \text{ given} \]

\( k_t \) is beginning-of-period \( t \) capital stock
Two types of PEA

1. Standard projections algorithm:
   ① parameterize $E_t[\cdot]$ with $P_n(k_t, z_t; \eta_n)$
   ② solve $c_t$ from
   $$c_t = (P_n(k_t, z_t; \eta_n))^{-1/\nu}$$
   and $k_{t+1}$ from budget constraint

2. Simulations PEA
Stochastic simulations PEA

1. Simulate $\{z_t\}_{t=1}^T$
2. Let $\eta_{n1}$ be initial guess for $\eta_n$
Stochastic simulations PEA

Iterate until $\eta^i_n$ converges using following scheme

1. Generate $\{c_t, k_{t+1}\}^{T}_{t=1}$ using
   \[
   c^\nu_t = P_n(k_t, z_t; \eta^i_n),
   \]
   \[
   k_{t+1} = z_t k^\alpha_t + (1 - \delta) k_t - c_t
   \]

2. Generate $\{y_{t+1}\}^{T-1}_{t=1}$ using
   \[
   y_{t+1} = \beta c^\nu_{t+1} \left( \alpha z_{t+1} k^\alpha_{t+1} - 1 + \delta \right)
   \]

3. Let
   \[
   \hat{\eta}^i_n = \arg\min_{\eta} \sum_{t=T_{\text{begin}}}^{T} \left( \frac{y_{t+1} - P_n(k_t, z_t; \eta)}{T} \right)^2
   \]

4. Update upsing
   \[
   \eta^{i+1}_n = \omega \hat{\eta}^i_n + (1 - \omega) \eta^i_n \text{ with } 0 < \omega \leq 1
   \]
Stochastic simulations PEA

- $T_{\text{begin}} >> 1$ (say 500 or 1,000)
  - ensures possible bad period 1 values don’t matter
- $\omega < 1$ improves stability
Stochastic simulations PEA

• Idea of regression:

\[ y_{t+1} \approx P_n(k_t, z_t; \eta) + u_{t+1}, \]

• \( u_{t+1} \) is a prediction \(\implies u_{t+1} \) is orthogonal to regressors

• Suppose

\[ P_n(k_t, z_t; \eta) = \exp (a_0 + a_1 \ln k_t + a_2 \ln z_t). \]

• You are not allowed to run the linear regression

\[ \ln y_{t+1} = a_0 + a_1 \ln k_t + a_2 \ln z_t + \tilde{u}_{t+1} \]

Why not?
Suppose $\eta^*_n$ is the fixed point we are looking for

So $P_n(k_t, z_t; \eta^*_n)$ is best predictor of $E_t \cdot$

Does this mean that solution is a rational expectations equilibrium?
Disadvantages of stochastic simulation PEA

- The inverse of $X'X$ may be hard to calculate for higher-order approximations
- Regression points are clustered $\implies$ low precision
  - Recall that even equidistant nodes is not enough for uniform convergence
    "nodes" are even less spread out with simulations PEA)
Disadvantages of stoch. sim. PEA

- Projection step has sampling error
  - this disappears at slow rate (especially with serial correlation)
Advantages of stoch. sim. PEA

- Regression points are clustered
  \[\implies\text{better fit where it matters}\]  \textbf{IF} functional form is poor
  (with good functional form it is better to spread out points)
Advantages of stoch. sim. PEA

- Grid: you may include impossible points
  Simulation: model itself tells you which nodes to include
  - (approximation also important and away from fixed point you may still get in weird places of the state space)
Odd shapes ergodic set in matching model
Improvements proposed by Maliar, Maliar & Judd

1. Use flexibility given to you
2. Use $\hat{E}[y_{t+1}]$ instead of $y_{t+1}$ as regressand
   - $\hat{E}[y_{t+1}]$ is numerical approximation of $E[y_{t+1}]$
   - even with poor approximation the results improve !!!
3. Improve regression step
Use flexibility

1. Many \(E[]\)'s to approximate.

   1. Standard approach:

\[
c_t^{-\nu} = E_t \left[ \beta c_{t+1}^{-\nu} \alpha \beta c_{t+1}^{-\nu} \left( \alpha z_{t+1} k_{t+1}^{\alpha-1} + 1 - \delta \right) \right]
\]

2. Alternative:

\[
k_{t+1} = E_t \left[ k_{t+1} \beta \alpha \beta \left( \frac{c_{t+1}}{c_t} \right)^{-\nu} \left( \alpha z_{t+1} k_{t+1}^{\alpha-1} + 1 - \delta \right) \right]
\]

- Such transformations can make computations easier but can also affect stability of algorithm (for better or worse)

2. \(P_n(k, z; \eta)\) could be linear (before or after transformation)
**E[y] instead of y as regressor**

- \( E[y_{t+1}] = E[f(\varepsilon_{t+1})] \) with \( \varepsilon_{t+1} \sim N(0, \sigma^2) \)
  \( \implies \) Hermite Gaussian quadrature can be used
  (MMJ: using \( \hat{E}[y_{t+1}] \) calculated using one node is better than using \( y_{t+1} \))

- Key thing to remember: sampling uncertainty is hard to get rid of
E[y] instead of y as regressor

- Suppose:
  \[
  y_{t+1} = \exp\left( a_0 + a_1 \ln k_t + a_2 \ln z_t \right) + u_{t+1}
  \]
  \[
  u_{t+1} = \text{prediction error}
  \]

- Then you cannot estimate coefficients using LS based on
  \[
  \ln (y_{t+1}) = a_0 + a_1 \ln k_t + a_2 \ln z_t + u_{t+1}^*
  \]

- You have to use non-linear least squares
E[y] instead of y as regressor

• Suppose:

\[
E[y_{t+1}] = \exp(a_0 + a_1 \ln k_t + a_2 \ln z_t) + \tilde{u}_{t+1}
\]

\[
\tilde{u}_{t+1} = \text{numerical error}
\]

• Then you can estimate coefficients using LS based on

\[
E[\ln (y_{t+1})] = a_0 + a_1 \ln k_t + a_2 \ln z_t + \tilde{u}_{t+1}^*
\]

• Big practical advantage
Simple ways to improve regression

1. Hermite polynomials and scaling
2. LS-Singular Value Decomposition
3. Principle components
Simple ways to improve regression

- The main underlying problem is that $X'X$ is ill conditioned which makes it difficult to calculate $X'X$
- This problem is reduced by
  1. Scaling so that each variable has zero mean and unit variance
  2. Hermite polynomials
Hermite polynomials; Definition

\[ P_n(x) = \sum_{j=0}^{n} a_j H_j(x) \]

where the basis functions, \( H_j(x) \), satisfy

\[ E \left[ H_i(x) H_j(x) \right] = 0 \text{ for } i \neq j \]

if \( x \sim N(0,1) \)
Hermite polynomials; Construction

\[ H_0(x) = 1 \]
\[ H_1(x) = x \]
\[ H_{m+1}(x) = xH_m(x) - mH_{m-1}(x) \text{ for } j > 1 \]

This gives

\[ H_0(x) = 1 \]
\[ H_1(x) = x \]
\[ H_2(x) = x^2 - 1 \]
\[ H_3(x) = x^3 - 3x \]
\[ H_4(x) = x^4 - 6x^2 + 3 \]
\[ H_5(x) = x^5 - 10x^3 + 15x \]
One tricky aspect about scaling

Suppose one of the explanatory variables is

\[ x_t = \frac{k_t - M_T}{S_T} \]

\[ M_T = \sum_{t=1}^{T} k_t / T \] \& \[ S_T = \left( \sum_{t=1}^{T} (k_t - M(k_t)^2 / T) \right)^{1/2} \]
One tricky aspect about scaling

- Each iteration the explanatory variables change (since $M$ and $S$ change)

- Taking a weighted average of old and new coefficient is odd

- I found that convergence properties can be quite bad actually better without taking a weighted average, but that only works for well behaved models

- In principle you can avoid problem by rewriting polynomial, but that is tedious for higher-order

- So better to keep $M_T$ and $S_T$ fixed across iterations
Two graphs say it all; regular polynomials
Two graphs say it all; Hermite polynomials
LS-Singular Values Decomposition

- Goal: avoid calculating $X'X$ explicitly
- SVD of the $(T \times n)$ matrix $X$

\[ X = USV' \]

- $U$: $(T \times n)$ orthogonal matrix
- $S$: $(n \times n)$ diagonal matrix with singular values $s_1 \geq s_2 \geq \cdots$
- $V$: $(n \times n)$ orthogonal matrix

- $s_i$ is the sqrt of $i^{th}$ eigen value
LS-Singular Values Decomposition

\[ \hat{\beta} = (X'X)^{-1} X'Y = VS^{-1}U'Y \]

- Goal: avoid calculating \( X'X \) explicitly
- SVD of the \((T \times n)\) matrix \( X \):
  \[ X = USV' \]
  - \( U \): \((T \times n)\) orthogonal matrix
  - \( S \): \((n \times n)\) diagonal matrix with singular values \( s_1 \geq s_2 \geq \cdots \)
  - \( V \): \((n \times n)\) orthogonal matrix
- \( s_i \) is the sqrt of \( i^{\text{th}} \) eigen value
LS-Singular Values Decomposition

In Matlab

\[ [U, S, V] = \text{svd}(X, 0); \]
**Principle components**

- With many explanatory variables use principle components
  - SVD: $X = USV'$ where $X$ is demeaned
  - Principle components: $Z = XV$
  - Properties $Z_i$: mean zero and variance $s_i^2$
- Idea: exclude principle components corresponding to lower eigenvalues
- But check with how much $R^2$ drops
PEA and learning

• Traditional algorithm:
  • simulate an economy using belief $\eta_n^i$
  • formulate new belief $\eta_n^{i+1}$
  • simulate same economy using belief $\eta_n^{i+1}$
PEA and learning

- Alternative algorithm to find *fixed point*
  - simulate $T$ observations using belief $\eta_n^{T-1}$
  - formulate new belief $\eta_n^T$
  - generate 1 more observation
  - use $T + 1$ observations to formulate new belief $\eta^{T+1}$
  - continue

- Convergence properties can be problematic
PEA and learning

- Modification of alternative algorithm is economically interesting
  - simulate $T$ observations using belief $\eta^{T-1}_n$
  - use $\tau$ observations to formulate new belief $\eta^T_n$
  - generate 1 more observation
  - use last $\tau$ observations to formulate new belief $\eta^{T+1}_n$
  - continue

- Beliefs are based on limited past $\rightarrow$ time-varying beliefs
Suppose the model has different regimes

- e.g. high productivity and low productivity regime
- agents do not observe regime \(\Rightarrow\) it makes sense to use limited number of past observations

- With the above algorithm agents gradually learn new law of motion
PEA and perturbation

- True in many macroeconomic models:
  - perturbation generates accurate solution of real side of the economy
  - perturbation does not generate accurate solution of asset prices
  - real side does not at all or not much depend on asset prices

- Then solve for real economy using perturbation and for asset prices using PEA
  - one-step algorithm (no iteration needed)
References

- Den Haan, W.J., Parameterized expectations, lecture notes.