Asset Prices in General Equilibrium with Transactions Costs and Recursive Utility

Adrian Buss\textsuperscript{1}, Raman Uppal,\textsuperscript{2}, Grigory Vilkov\textsuperscript{1}

\textsuperscript{1}Goethe University Frankfurt
\textsuperscript{2}EDHEC Business School

January 26, 2011
NAG Quant Day in HoF, Frankfurt
Research Questions/ Objective

**Study Asset Prices in an Economy with:**
- Multiple agents
- Heterogeneous recursive (Epstein-Zin) preferences
- Heterogeneous beliefs
- Exchange economy
- Multiple assets (trees)
- Proportional transaction costs (to value and number of shares traded)
- Possibility of other frictions (shortsale, leverage constraints, etc)

**Consequences of introducing TC for:**
- Interest rate
- Stock price
- Expected return on the stock
- Volatility of stock and bond returns
Past Research/ Literature Review I

- **General Equilibrium with TC**

- **General Equilibrium with multiple agents and recursive utility**
  - Dumas, Uppal and Wang 2000, ...

- **Partial Equilibrium with TC**
  - Davis and Norman 1990, Duffie and Sun 1990, Dumas and Luciano 1991, Muthuraman and Kumar 2006, ...

- **Solution methods**
  - Dumas and Lyasoff 2010, ...
Helicopter View

The Task and its Treatment

- Discrete time and space setup: recombining tree
- GE problems: backward-forward system of equations
- Use Dumas and Lyasoff 2010 time shift: backward system
- Transaction costs: additional path-dependency
- Enhanced numerical scheme to deal with TC

Tools and NAG benefits

- Each Node of a tree/ state variables grid: system of equations
- Each system of equations: switching system depending on solver value
- Each system of equations: multivariate interpolation
- Use Matlab with NAG Toolbox: solver and interpolation
- Benefits compared to Matlab alone: > 1,500x speedup
Problem Formulation

Two agents, $l = 1, 2$ with utility function $u_{l,t}(c_{l,t,s})$, each solving:

$$\max_{c_{l,t,s}, \theta_{l,t,s}} u_{l,t}(c_{l,t,s}) + E_t \left[ \sum_{\tau=1}^{T-t} u_{l,t+\tau}(c_{l,t+\tau}) \right]$$

subject to the floating budget constraint:

$$c_{l,t,s} + \theta_{l,t,s}^B B_{t,s} + \theta_{l,t,s}^S S_{l,t,s} + \tau(\theta_{l,t,s}^S, \theta_{l,t-1,s}^S, S_{l,t,s}, k) = \theta_{l,t-1,s}^B B_{t,s} + \theta_{l,t-1,s}^S (S_{l,t,s} + d_{t,s}),$$

where $\tau(\theta_{l,t,s}^S, \theta_{l,t-1,s}^S, S_{l,t,s}, k)$ — transaction costs function

++GE: market clearing conditions, kernel conditions
Optimality Conditions (FONC)

For each time $t$, each state (node) $s$, and each agent $l$, we have

1. $u'_{l,t} = \lambda_{l,t,s}$

2. $E_t\left[u'_{l,t+1}B_{t+1,s+}\right] = \lambda_{l,t,s}B_{t,s}$

3. $E_t\left[u'_{l,t+1}\left(S_{l,t+1,s+} + d_{t+1,s+} - \frac{\partial \tau}{\partial \theta^S_{l,t,s}} (\theta^S_{l,t+1,s+}, \theta^S_{l,t,s}, S_{l,t+1,s+}, k)\right)\right] = \lambda_{l,t,s}\left(S_{l,t,s} + \frac{\partial \tau}{\partial \theta^S_{l,t,s}} (\theta^S_{l,t,s}, \theta^S_{l,t-1,s-}, S_{l,t,s}, k)\right)$

4. $c_{l,t,s} + \theta^B_{l,t,s}B_{t,s} + \theta^S_{l,t,s}S_{l,t,s} + \tau(\theta^S_{l,t,s}, \theta^S_{l,t-1,s-}, S_{l,t,s}, k) = \theta^B_{l,t-1,s-}B_{t,s} + \theta^S_{l,t-1,s-}(S_{l,t,s} + d_{t,s})$

Assuming proportional Transaction costs:

$$\frac{\partial \tau}{\partial \theta^S_{l,t,s}}(\theta^S_{l,t,s}, \theta^S_{l,t-1,s-}, S_{l,t,s}, k) = \frac{\partial \text{abs}(\theta^S_{l,t,s} - \theta^S_{l,t-1,s-})}{\partial \theta^S_{l,t,s}} \cdot S_{l,t,s} \cdot k$$
Optimality Conditions: System of Equations

Proportional Transaction Costs → No-Trade and Trade Regions

For each value of the state variables on a grid, solve the following:

No-Trade Region (NTR) – no trading takes place
- Budget equations
- Market clearing conditions
- Supply equations

Trade Region (TR) – agents trade
- Budget equations
- Market clearing conditions
- Supply equations
- Kernel conditions
Traditional Numerical Scheme: Formulation (w/o TC)

State variables (grid over):
- (traditionally) entering wealth $\theta_{l,t-1,s-}^B B_{t,s} + \theta_{l,t-1,s-}^S (S_{l,t,s} + d_{t,s})$
- (if with TC path-dependency) past portfolio holdings $\theta_{l,t-1,s-}^S$

Algorithm
- Work backwards
- Solve for current consumption $c_{l,t,s}$ and portfolio holdings $\theta_{l,t,s}^{S,B}$
- For each grid value interpolate $u'_{l,t+1} (c_{l,t+1,s+}, S_{l,t+1,s+}, B_{l,t+1,s+}$
- Create interpolating function for the earlier step
Traditional Numerical Scheme: Problems

The system is backward-forward!

For $c_t$ and $\theta_t$ one needs:

- Future marginal utility $u'_{l,t+1}(c_{l,t+1},s_+)$
- Future asset prices $S_{l,t+1,s+}, B_{l,t+1,s+}$
- Past portfolio holdings $\theta_{l,t-1,s-}^{B,S}$

$c_t$ and $\theta_t$ affects:

- Future wealth, and hence $u'_{l,t+1}(c_{l,t+1},s_+), S_{l,t+1,s+}, B_{l,t+1,s+}$

One needs the convergence of the solution via expectation step

- Takes time..
- No guarantee of a positive outcome...
- Interpolation of a marginal utility is quite unstable...
Adjusting the Numerical Scheme: Principle

**Major change:**

Shift all equations (except for kernel condition) one time period forward!

**State variables (grid over):**

- Current consumption $c_{l,t,s}$
- (if with TC path-dependency) split the solution into two steps:
  - ▶ $\frac{\partial \tau}{\partial \theta^S_{l,t,s}}(\theta^S_{l,t,s}, \theta^S_{l,t-1,s-}, S_{l,t,s}, k) = +1/ - 1 \times S_t k$ to get NTR
  - ▶ Past portfolio holdings $\theta^S_{l,t-1,s-}$ inside NTR

**Idea of the Solution**

- Work backwards
- Split the solution at each time point into two steps
  - ▶ Solve for the NTR boundaries with 2-point grid $=$ {Trade, No Trade}
  - ▶ Solve the system inside of the NTR with full asset holdings grid
- Create distinct interpolating functions for each case
- Change the system of equations depending on where you are today
Adjusting the Numerical Scheme: Principle

**Algorithm**

- Work backwards
  
  Solve for the boundaries of the NTR
- Solve for future consumption $c_{l,t+1,s+}$ and current portfolio $\theta_{l,t,s}^{S,B}$
- For each grid value interpolate $S_{l,t+1,s+}, B_{l,t+1,s+}, \theta_{l,t+1,s+}^{S,B}$
- Create interpolating function for the earlier step

Solve inside of NTR

- Solve for future consumption $c_{l,t+1,s+}$ and current bond holdings $\theta_{l,t,s}^B$
- NTR $\rightarrow$ stock holdings $\theta_{l,t,s}^S$ are carried forward from $t - 1$
- For each grid value interpolate $S_{l,t+1,s+}, B_{l,t+1,s+}, \theta_{l,t+1,s+}^{S,B}$
- Create interpolating function for the earlier step
Using NAG: Functions Used

**Multidimensional Interpolation**

- One asset with TC
  - `e01sg` – modified Shepard’s method, two variables
  - `e01sh` – evaluate interpolant computed by `e01sg`, two variables
- Two assets with TC
  - `e01sg` – modified Shepard’s method, two variables
  - `e01sh` – evaluate interpolant computed by `e01sg`, two variables
  - `e01tg` – modified Shepard’s method, three variables
  - `e01th` – evaluate interpolant computed by `e01tg`, three variables

**Solution of the System of Equations**

- `c05nc` – system of nonlinear equations using function values only
Using NAG: Pros and Cons

**Multidimensional Interpolation**

**NAG**
- allows to save the interpolant in a structure, and use it later
- takes input table as vector, and can deal with non-square grids
- uses quadratic splines
- is limited by 3-dim interpolations
- has some “collinearity problems” when the step goes down

**Matlab**
- computes the interpolant on the fly
- only takes matrices as inputs, hence grid limitations
- uses cubic splines
- can go to any dimension

→ 50 – 100 times speed improvement by using NAG

**Caution:** Matlab may be more accurate due to higher dimension
Using NAG: Pros and Cons

Solution of the System of Equations

- NAG needs global variables: dangerous (e.g., parallel execution)
- Matblab can take parameters in the function
- NAG is 3 – 10 times faster