

Computer Programming in Econometrics

Introduction, structure, and advanced programming techniques

Charles S. Bos

VU University Amsterdam

Tinbergen Institute

`cbos@feweb.vu.nl`

11 October 2010, Tinbergen Institute

Don't use

Just checking for myself; these slides only confuse the students for now, I think...

Transforming parameters – extended

Problem of transforming parameters: Very general, very repetitive:

- ▶ Optimization common
- ▶ Often restricted
- ▶ Linear simple restrictions of one of four types:
 $[L, \infty)$, $[L, U]$, $(-\infty, U]$, $(-\infty, \infty)$

Re-inventing the routine for each problem at hand: Waste of time...

Transforming parameters – extended

Problem of transforming parameters: Very general, very repetitive:

- ▶ Optimization common
- ▶ Often restricted
- ▶ Linear simple restrictions of one of four types:
 $[L, \infty)$, $[L, U]$, $(-\infty, U]$, $(-\infty, \infty)$

Re-inventing the routine for each problem at hand: Waste of time... So look at general problem:

Constraint	θ^*	θ
$[0, \infty)$	$\log(\theta)$	$\exp(\theta^*)$
$[0, 1]$	$\log\left(\frac{\theta}{1-\theta}\right)$	$\frac{\exp(\theta^*)}{1+\exp(\theta^*)}$

Of course, to get a range of $[L, U]$, use a rescaled $[0, 1]$ transformation.

Transformation: Rescale

Rescale $\theta \in [L, U]$ to range $\theta^\dagger \in [0, 1]$:

Constraint θ	$\theta^\dagger(\theta)$	$\theta(\theta^\dagger)$	Constraining θ^\dagger
$[L, \infty)$	$\theta - L$	$\theta^\dagger + L$	$[0, \infty]$
$[L, U]$	$\frac{\theta - L}{U - L}$	$(U - L)\theta^\dagger + L$	$[0, 1]$
$(-\infty, U]$	$U - \theta$	$U - \theta^\dagger$	$[0, \infty)$
$(-\infty, \infty)$	θ	θ^\dagger	$(-\infty, \infty)$

Then use aforementioned transformation to get to $(-\infty, \infty)$, going $\theta \leftrightarrow \theta^\dagger \leftrightarrow \theta^*$

TransPar usage

Target setup:

```
main()
{
    mLU= <-5,          7;
           M_INF_NEG, 3;
           8,          M_INF;
           M_INF_NEG, M_INF>;
    vP= <-2; 2; 9; 4>;

    TransPar(&vPtr, vP, mLU);           // Transform parameters
    TransBackPar(&vP2, vPtr, mLU);      // Transform back
    print ("%c", {"Par", "Trans", "Par v2", "L", "U"},
           vP~vPtr~vP2~mLU);
}
```

Move back and forth, with general limits.

TransPar

See live coding...

TransPar limits: Go global?

TransPar/TransBackPar: Needed from AvgLnLiklRegr(). No space for mLU...

Options:

- ▶ Go global, leave mLU there.
- ▶ Call once with mLU, store in 'semi-global', only touched in TransPar routines. Put the mLU in that 'semi-global' at the first call, afterwards read it from the 'semi-global'. Advantage: Main program does not have to deal with the global at all...