Price sensitivity experiments for the supply models of CAPRI with the GAMS scenario solver

Wolfgang Britz, University Bonn, Summer 2011

Background

The supply side of CAPRI consists of programming models for which no closed form representation of the changes in activity levels with regard to prices exist (yet). Existing approaches such as the one described in Heckelei 2002 or Merel & Bucaram 2010 cannot take into account a change in the set of binding constraints and are not yet available for more complex functional forms. In order to derive point elasticities, it is therefore necessary to perform simulation experiments with the model itself. Even if 2000 farm models can be solved in 30 secs on a powerful 8 core desktop if the shock is relatively small, running ~50 price experiments would still require a sizeable amount of time. The paper discusses an alternative approach based on the new scenario solver available in GAMS since version 27.3 might.

Knowledge about the implicit price elasticities from the CAPRI supply model might be valuable in a number of contexts. Firstly, in CAPRI itself, the point price elasticities might be used in future to parameterize the supply functions in the market model which are currently based on a derivation based on Heckelei 2002. Secondly, when experiencing problems with the convergence behavior of CAPRI, it is often useful to check if the supply reactions to prices are very large. The iteration tracking tool (http://www.capri-model.org/docs/tracking.pdf) delivers typically the hint in which Member State the problem roots. Thirdly, it might be also useful to compare the estimated elasticities from Janssonn & Hecekelei 2011 against the ones from sensitivity experiments to learn if the calibration process which assigns PMP terms works properly. And fourthly, the supply elasticities might be used in other studies e.g. when linking CAPRI to other models.

Scenario solver

When executing a normal model solve statement in GAMS, the GAMS base engine will convert the equation definitions into non-linear instructions for the solver based on the current status of the GAMS symbols. Depending on model size and complexity, that process of model generation can require a sizeable amount of the whole time needed for the solution process. The scenario solver (http://www.gams.com/dd/docs/solvers/guss.pdf) available since version 23.7 in GAMS termed Gather-Update-Solve-Scatter (GUSS) solver basically skips that conversion step and instead only updates constants in the non-linear instructions used in the solver. That is however only feasible if the structure of the equations is not affected by the requested update. Loading e.g. a new region in the template as in the case of the CAPRI supply model template is not possible with the scenario solver, as dynamic sets and dollar controls will change the number and actual structure of the equations and not only constants entering them.

The application of the solver consists of three steps:

1. Definition of the experiments to run in a so-called dictionary
2. Executing a solve passing the dictionary
3. Collecting and processing the results

Definition and population of the dictionary

The following screen shots shows the dictionary used for the price sensitivity experiments (see *supply\supply_model.gms*).

```
    parameter p_scenOptions / SkipBaseCase 0, UpdateType 0, logoption 1
$ife  %system.gamsversion%>237 NoMatchLimit 5000
     /;

    set uvagDict(*,*,*)
                /
                    om_obje_scen.scenario. ''
$ife  %system.gamsversion%>236 p_scenOptions.   opt     .''
                    p_uvagScen.    param.   p_uvagScens
                    v_netPutQuant. level  .p_netPutScens
                    v_actLevl     . level  .p_actLevlScens
                /;

     option kill=p_netPutScens,kill=p_uvagScen,kill=p_uvagScens,kill=p_actLevlScens;

     set headers / objval /;
     parameter scenrep(om_obje_scen,headers);
```

It defines that the scenarios (".scenario.") will comprise the elements of the set *om_obje_scen*, i.e. the loop which will be executed to update the model. That set comprises netputs entering with their prices the objective function of the supply model. The dictionary will send in updated data, i.e. new prices, via the parameter *p_uvagScen* (UVAG for unit value gross production is the mnemonics for "farm gate price for outputs" used in CAPRI). That parameter is picked up by the supply model in the linear part of the objective function when *p_useUvagScen* is set. *p_uvagScen* replaces the price information stored otherwise on the data parameter :

```
*
   LINEAR_(RUNR)..    v_linObjePart(RUNR)  =E=
*
*      --- sales/purchases valued by "unit value" price
*          from gross Economic Accounts for Agriculture

*
      SUM(  RUNR_OMOBJE(RUNR,OM_OBJE),
            v_netPutQuant(RUNR,OM_OBJE) * SUM(R_RAGG(RUNR,MSACT), (  %data%(MSACT,"UVAG",OM_OBJE,"Y") $ (not p_useUvagScen)
                                                                 + p_uvagScen(MSACT,OM_OBJE) $ p_useUvagScen )))
```

For each element of the set *om_obje_scen*, the model will hence executed one time, with the price stored for that specific scenario.

The ".level." entries in the dictionary describe the variable levels to be reported back. They are stored on two parameters, *p_netPutScens* for the netput quantities and *p_actLevlScens* for the activity levels. Finally, we also pass specific options for the scenario solver under ".opt.".

During the last iteration in a simulation run with CAPRI, the parameter *p_uvagScen* is populated as shown below. As seen, in a first step, the currently active prices on DATA are copied over to the parameter for all products entering the objective function according to the set *OM_OBJE*. Next, for each experiment, the related price is increased by 10%.

```
$iftheni %runPriceScens%==on
   if ( p_isLastStep,

      MSACT(MS) = YES;
      p_uvagScens(OM_OBJE_SCEN,MSACT,OM_OBJE) = DATA(MSACT,"UVAG",OM_OBJE,"Y");
      p_uvagScens(OM_OBJE_SCEN,MSACT,OM_OBJE) $ sameas(OM_OBJE_SCEN,OM_OBJE) = DATA(MSACT,"UVAG",OM_OBJE,"Y") * 1.10;
      option kill=p_uvagScen;
      option kill=p_netPutScens;

      option kill=MS_LARGE;
      RU_SMALL(RU) = YES;

   );
$endif
```

Executing the solve

As indicated above, the solver must be applied for each region separately as the structure of the equations is not allowed to change. The screenshot below shows the code snippet which is executed inside a loop over the regional units:

```
*
$iftheni %runPriceScens%==on
*
         if ( p_isLastStep,
$batinclude 'util\title.gms' "'Price sensitivity analysis with supply model for '" RU_SMALL.tl
$setglobal capmod m_capmodQ
            p_useUvagScen = 1;
*
*           --- use scenario solver (= hot model updates) to perform
*               sensitivity analysis for prices
            SOLVE %capMod% USING NLP MAXIMIZING v_obje SCENARIO uvagDict;
*
*           --- collect results
*
            execute_load 'scenrep.gdx', scenRep;
*
            p_useUvagScen = 0;

            SOLVE %capMod% USING NLP MAXIMIZING v_obje;

          );
$endif
```

It calls the solver using the dictionary, collects the results in the parameter *scenRep* (a fixed name) and afterwards solves the model again at the normal price to restore the normal results.

Processing the results

After that point, the results generate by the scenario solver can be processed as seen below. In a first step, we are aggregating over the technology variants *A*, both the results returned from the scenario solver and the final simulation results. Equally, we aggregate activity levels and net put quantities from regional to Member State level. We then calculate relative changes between the experiments and the final solution and convert them into elasticities. Elasticities below a certain thresholds are set to zero and the results are copied into the listing.

```
$iftheni %runPriceScens%==on

      if ( p_isLastStep,
         v_netPutQuant.l(MS,ROWS) = sum( map_rr(MS,RU), v_netPutQuant.l(RU,ROWS));


         p_actLevlScens(OM_OBJE_SCEN,RU,MPACT,"T")
          = sum( a $ p_technFact(RU,MPACT,"levl",a), p_actLevlScens(OM_OBJE_SCEN,RU,MPACT,a));

         p_actLevlScens(OM_OBJE_SCEN,MS,COLS,"T")
          = sum( map_rr(MS,RU), p_actLevlScens(OM_OBJE_SCEN,RU,COLS,"T"));

         v_actLevl.l(RU,MPACT,"T")
          = sum( a $ p_technFact(RU,MPACT,"levl",a), v_actLevl.l(RU,MPACT,A));

         v_actLevl.l(MS,COLS,"T")
          = sum( map_rr(MS,RU), v_actLevl.l(RU,COLS,"T"));

         p_netPutScens(OM_OBJE_SCEN,MS,ROWS)
          = sum( map_rr(MS,RU), p_netPutScens(OM_OBJE_SCEN,RU,ROWS));


         p_netPutElas(MS,ROWS,OM_OBJE_SCEN) $ (v_netPutQuant.l(MS,ROWS) and p_netPutScens(OM_OBJE_SCEN,MS,ROWS))
          = (p_netPutScens(OM_OBJE_SCEN,MS,ROWS)/v_netPutQuant.l(MS,ROWS) -1 ) * 10;

         p_netPutElas(MS,ROWS,OM_OBJE_SCEN) $ ( abs(p_netPutElas(MS,ROWS,OM_OBJE_SCEN)) le 0.001) = 0;

         p_levlElas(RU,COLS,OM_OBJE_SCEN) $ (v_actLevl.l(RU,COLS,"T") and p_actLevlScens(OM_OBJE_SCEN,RU,COLS,"T"))
          = (p_actLevlScens(OM_OBJE_SCEN,RU,COLS,"T")/v_actLevl.l(RU,COLS,"T")-1) * 10;

         p_levlElas(MS,COLS,OM_OBJE_SCEN) $ (v_actLevl.l(MS,COLS,"T") and p_actLevlScens(OM_OBJE_SCEN,MS,COLS,"T"))
          = (p_actLevlScens(OM_OBJE_SCEN,MS,COLS,"T")/v_actLevl.l(MS,COLS,"T")-1) * 10;

         p_levlElas(MS,COLS,OM_OBJE_SCEN) $ ( abs(p_levlElas(MS,COLS,OM_OBJE_SCEN)) le 0.001) = 0;
         p_levlElas(RU,COLS,OM_OBJE_SCEN) $ ( abs(p_levlElas(RU,COLS,OM_OBJE_SCEN)) le 0.001) = 0;

         display p_netPutElas,p_levlElas,v_actLevl.l,p_actLevlScens;

      );

$endif
```

Currently, the results are only reported in the listing and not yet not stored to disk. Equally, the price sensitivity experiments need to be switched on manually.

Summary and outlook

The scenario solver provides an efficient way to solve models if only updates of constant in equations are required, and not a full update of the equation structure. It is much faster compared to a normal solve. It allows, on demand, to determine which sensitivity experiments should be executed. In CAPRI, it is currently used to derive point point own- and cross price elasticities. In future, these elasticities might be used to parameterize the supply functions in the market model for those countries which are also represented by programming models.

References

Heckelei, T. (2002): Calibration and Estimation of Programming Models for Agricultural Supply Analysis. Habilitation Thesis, University of Bonn, Germany

Merel, P. R. and Bucaram, S. (2010). Exact Calibration of Programming Models of Agricultural Supply against Exogenous Sets of Supply Elasticities. *European Review of Agricultural Economics* 37: 395-418.

Jansson, T. and T. Heckelei (2011): Estimating a Primal Model of Regional Crop Supply in the European Union. *Journal of Agricultural Economics* 62(1): 137-152