Comparing general equilibrium modeling in Julia and GAMS: An example using CSAVE

Y.-H. Henry Chen

November 18, 2025

Center for Sustainability Science and Strategy

Message from GAMS…

"The reason for GAMS superior performance in [having the shorter model generation time of] this example [IJKLM model] is the use of relational algebra… to process complex queries efficiently."

— Broihan (2023), GAMS Corporation

Response from Julia…

"… it is not difficult to address it [the bottle neck of Broihan's benchmark Julia model], given that general-purpose languages like Julia and Python have libraries specialized for this task."

— M. Lubin, O. Dowson, J. D. Garcia, J. Huchette, B. Legat (2023), JuMP developers

Questions

- What does a CGE written in Julia look like?
- Which language is faster in generating & solving CSAVE?

Scope

- A heuristic exercise based on two versions of CSAVE: Julia vs. GAMS
- Results may vary across models and computers
- Technical details/coding tricks are left for a workshop, if any interest

Progress

- Built a Julia package “CSVtoDIC”— converts CSV to dictionaries or vectors
- Built a Julia package “GTAPdata”— produces data in format CSAVE needs
- Built a multi-region CSAVE using Julia’s MPSGE.jl and GAMS/MPSGE

- All available in https://github.com/chenyhmitedu/

Coding in Julia

- Julia
  - Open source/free
  - General purpose w/ packages for extension
  - MPSGE.jl is a package of Julia
  - Free PATH license until 12/31/2025 for now
- MPSGE.jl
  - D. Anthoff, E. Lazarus, M. Phillipson

- GAMS
  - Proprietary
  - DSML for optimization problems
  - MPSGE is a sub-system of GAMS
  - Access PATH via GAMS license
- MPSGE
  - Tom Rutherford

Coding in Julia

- Besides the MPSGE package, other packages used include
  - JuMP — Julia for mathematical programming
  - Ipopt — Inter point optimizer
  - CSV — Reading & writing CSV files
  - Dataframes — Data manipulation
  - JLD2 — Save and load data in binary format
  - CSVtoDIC
  - GTAP9data

Coding in Julia

- Project environment

packages

  - Project.toml — names & identities of the direct dependencies of a project
  - Manifest.toml — the dependency graph, dependency version, where to load
  - Automatically generated following the programmer's input

Pin package "GTAPdata" to a specific commit

Project.toml

```
[deps]
CSV = "336ed68f-0bac-5ca0-87d4-7b16caf5d00b"
DataFrames = "a93c6f00-e57d-5684-b7b6-d8193f3e46c0"
GTAPdata = "130c15d4-ad42-4bd4-91d3-6787eb393e58"
Ipopt = "b6b21f68-93f8-5de0-b562-5493be1d77c9"
JLD2 = "033835bb-8acc-5ee8-8aae-3f567f8a3819"
JuMP = "4076af6c-e467-56ae-b986-b466b2749572"
MPSGE = "d5dc2f44-7ae2-49e9-bc77-b47b6bca565d"
```

Manifest.toml

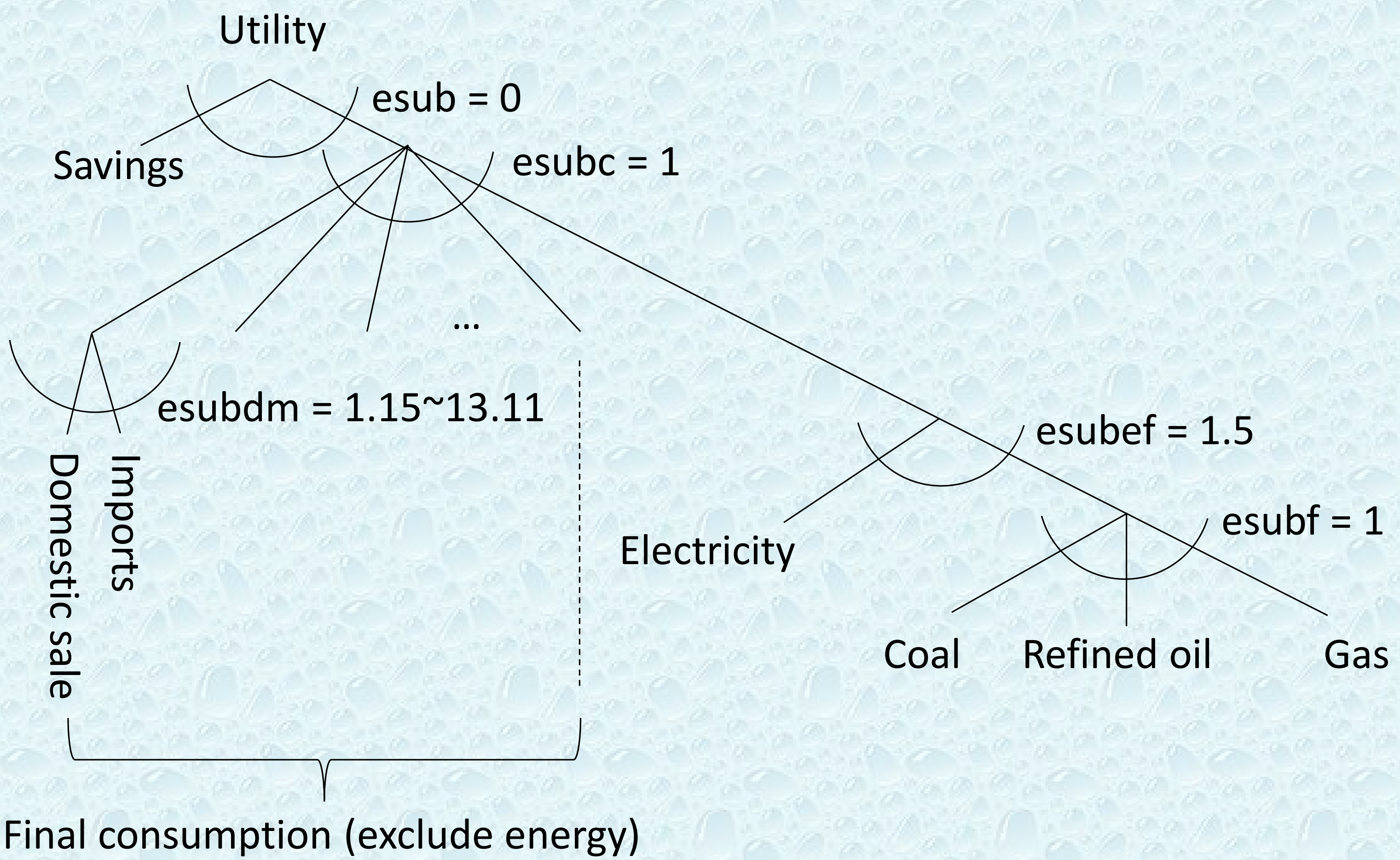
```
[[deps.GTAPdata]]
deps = ["CSV", "CSVtoDIC", "DataFrames", "JLD2"]
git-tree-sha1 = "d5cb8e98efe4a2d2de8cfabcb153f6ba5ac84558"
repo-rev = "master"
repo-url = "https://github.com/chenyhmitedu/GTAPdata"
uuid = "130c15d4-ad42-4bd4-91d3-6787eb393e58"
version = "0.1.0"

[[deps.HashArrayMappedTries]]
```

Model

- CSAVE: Chen & Paltsev (2025)
    - Motivated by EPPA
    - Derived from GTAPinGAMS
    - In this exercise: multi-sector & multi-region — ↑ resolution for "stress testing"
    - Aggregated power sector for simplicity
    - GTAP9 database

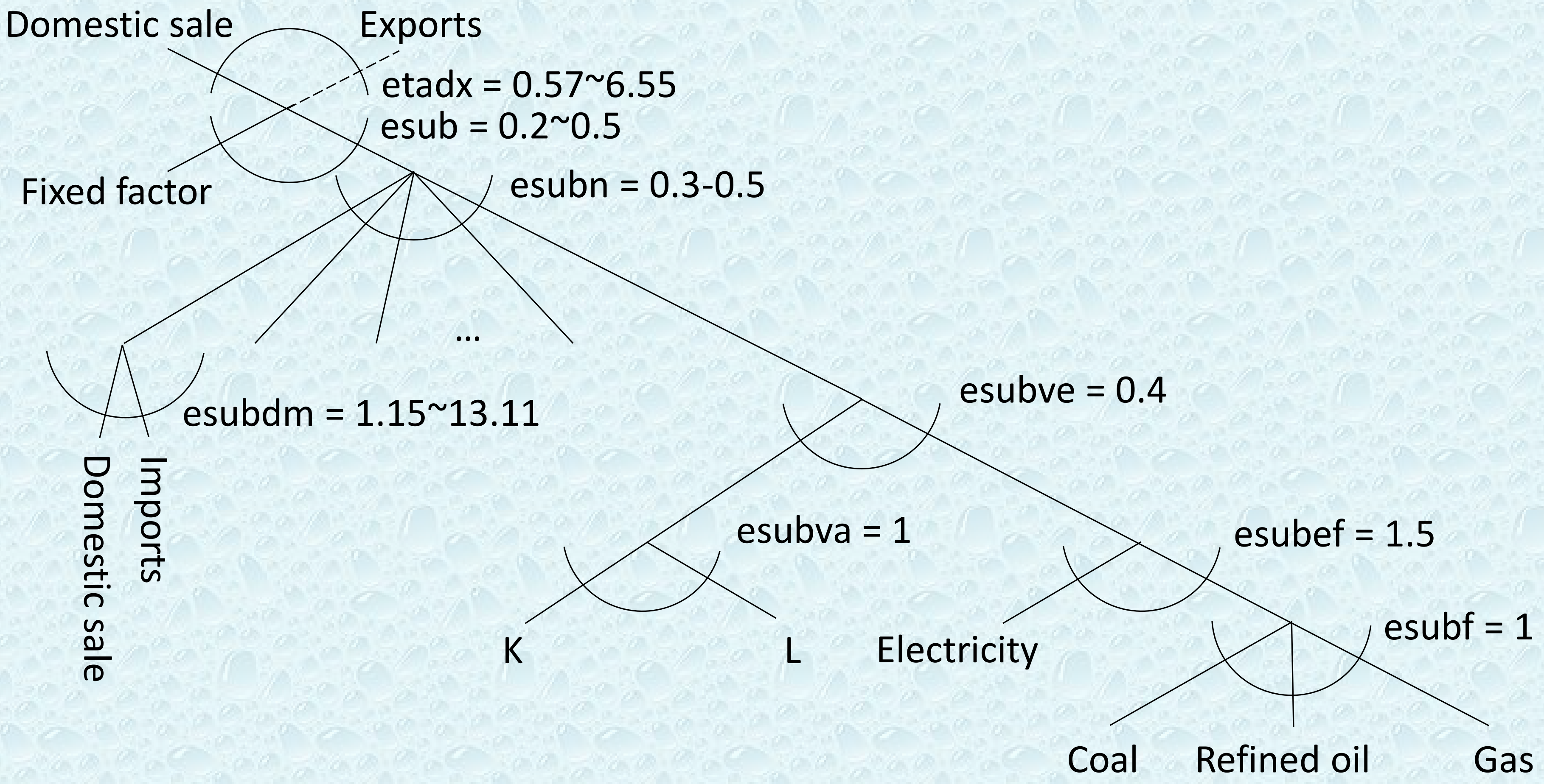
Model

Expenditure function



Model

Cost function for a production sector



Model

GAMS/MPSGE

```
$sectors:
y(g,r)$vom(g,r)                          ! Supply
   m(i,r)$vim(i,r)                       ! Imports
   yt(j)$vtw(j)                          ! Transportation services
   E(i,s,r)$vxmd(i,s,r)                  ! Exports
   A(i,g,r)$vafm(i,g,r)                  ! Armington good i used b


$commodities:
   p(g,r)$vom(g,r)                       ! Domestic output price
   pm(j,r)$vim(j,r)                      ! Import price
   pt(j)$vtw(j)                          ! Transportation services
   pf(f,r)$(evom(f,r)$mf(f))             ! Primary factors rent
   ps(f,g,r)$(sf(f) and vfm(f,g,r))      ! Sector-specific primary
   PX(i,s,r)$vxmd(i,s,r)                 ! Price index for exports
   PA(i,g,r)$vafm(i,g,r)                 ! Price for Armington goo
   PE(g,r)$vxm(g,r)                      ! Price index for exports

$consumers:
   ra(r)                                 ! Representative agent
```

Julia/MPSGE.jl

```
@sectors(MGE, begin
    Y[set_g, set_r],              (description = "Supply")
    M[set_i, set_r],              (description = "Imports")
    YT[set_i],                    (description = "Transportation services")
    E[set_i, set_r, set_r],       (description = "Subsidy and transport service
    A[set_i, set_g, set_r],       (description = "Armington good")
end)

@commodities(MGE, begin
    P[set_g, set_r],              (description = "Domestic output price")
    PM[set_i, set_r],             (description = "Import price")
    PT[set_i],                    (description = "Transportation services")
    PF[set_mf, set_r],            (description = "Non-sector-specific primary f
    PS[set_sf, set_g, set_r],     (description = "Sector-specific primary facto
    PX[set_i, set_r, set_r],      (description = "Price index for exports (incl
    PA[set_i, set_g, set_r],      (description = "Price index for Armington goo
    PE[set_i, set_r],             (description = "Price index for exports (excl
end)

@consumers(MGE, begin
    RA[set_r],                    (description = "Representative agent")
end)
```

Model

GAMS/MPSGE

```
$prod:y(g,r)$vom(g,r)   t:etadx(g)   s:esub(g)   sn(s):esubn(g)   sve(sn):esubve(g)   sva(sve):esubva(g)   sef(sve):esubef(g)   sf(sef):esubf(g)
        o:P(g,r)                q:(vom(g,r)-vxm(g,r))                             a:RA(r)  t:rto(g,r)
        o:PE(g,r)               q:vxm(g,r)                                        a:RA(r)  t:rto(g,r)
        i:PA(i,g,r)$fe(i)       q:vafm(i,g,r)                                                                 sf:
        i:PA(i,g,r)$elec(i)     q:vafm(i,g,r)                                                                 sef:
        i:PA(i,g,r)$ne(i)       q:vafm(i,g,r)                                                                 sn:
        i:ps(sf,g,r)            q:vfm(sf,g,r)    p:(1+rtf0(sf,g,r))               a:ra(r)  t:rtf(sf,g,r)
        i:pf(mf,r)              q:vfm(mf,g,r)    p:(1+rtf0(mf,g,r))               a:ra(r)  t:rtf(mf,g,r)      sva:
```

Julia/MPSGE.jl

```
for g ∈ set_i, r ∈ set_r
    @production(MGE, Y[g, r], [t = etadx[g], s = esub[g], sn => s = esubn[g], sve => sn = esubve[g], sva => sve = esubva[g], sef => sve = esubef[g], sf =
        @output(P[g, r],          vhm[g, r], t, taxes = [Tax(RA[r], rto[g, r])], reference_price = 1-rto0[g, r])
        @output(PE[g, r],         vxm[g, r], t, taxes = [Tax(RA[r], rto[g, r])], reference_price = 1-rto0[g, r])
        [@input(PA[i, g, r],      vafm[i, g, r], sf) for i ∈ set_fe]...
        [@input(PA[i, g, r],      vafm[i, g, r], sef) for i ∈ set_elec]...
        [@input(PA[i, g, r],      vafm[i, g, r], sn) for i ∈ set_ne]...
        [@input(PS[sf, g, r],     vfm[sf, g, r],  s, taxes = [Tax(RA[r], rtf[sf, g, r])],   reference_price = 1 + rtf0[sf, g, r])  for sf ∈ set_sf]...
        [@input(PF[mf, r],        vfm[mf, g, r],  sva, taxes = [Tax(RA[r], rtf[mf, g, r])],   reference_price = 1 + rtf0[mf, g, r])  for mf ∈ set_mf]...
    end)
end
```

Model

GAMS/MPSGE

```
$demand:ra(r)
        d:p("c",r)                q:vom("c",r)
        e:p("c","USA")            q:vb(r)
        e:p("g",r)                q:(-vom("g",r))
        e:p("i",r)                q:(-vom("i",r))
        e:ps(sf,j,r)              q:vfm(sf,j,r)
        e:pf(mf,r)                q:evom(mf,r)
```

Julia/MPSGE.jl

```
for r ∈ set_r
    @demand(MGE, RA[r], begin
        @final_demand(P[:c, r],     vom[:c, r])
        @endowment(P[:c, :USA],     vb[r])
        @endowment(P[:g, r],        -vom[:g, r])
        @endowment(P[:i, r],        -vom[:i, r])
        [@endowment(PF[f, r],       evom[f, r]) for f ∈ set_mf]...
        [@endowment(PS[f, j, r],    vfm[f, j, r]) for f ∈ set_sf, j ∈ set_i]...
    end)
end
```

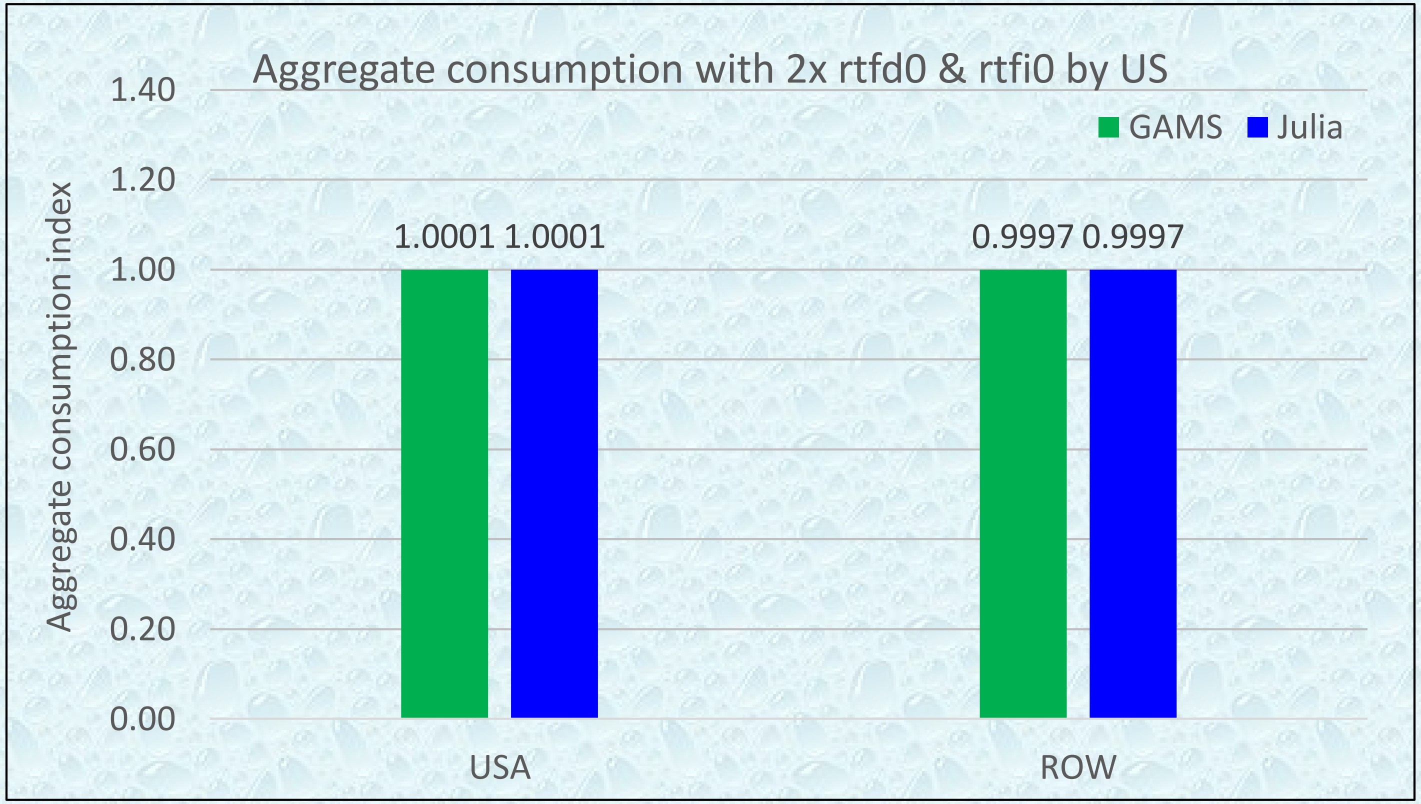
Model

- Data resolution
  - 56x2 — 56-sector; 2-region: USA, ROW
  - 56x4 — 4-region: USA, EUR, CHN, ROW
  - 56x8 — 8-region: USA, EUR, ROE, JPN, IND, CHN, ANZ, ROW
  - 56x16 — 16-region: USA, …, ANZ, TWN, CAN, MEX, BRA, RUS, KOR, IDZ, ASI, ROW
  - 56x32 — 32-region: USA, [*disaggregated EUR*: DEU, FRA, GBR, ITA, ESP, …], …, ROW

Model

- Scenario
  - US changes tax rates on fossil fuels use
- 5 runs for each data resolution
  - rtfd0[i, g, :USA] x 0.0 & rtfi0[i, g, r] x 0.0 ; r ∈ USA, i ∈ coa, p_c, gas
  - rtfd0[i, g, :USA] x 0.5 & rtfi0[i, g, r] x 0.5 ; r ∈ USA, i ∈ coa, p_c, gas
  - rtfd0[i, g, :USA] x 1.0 & rtfi0[i, g, r] x 1.0 ; r ∈ USA, i ∈ coa, p_c, gas → Base year levels
  - rtfd0[i, g, :USA] x 1.5 & rtfi0[i, g, r] x 1.5 ; r ∈ USA, i ∈ coa, p_c, gas
  - rtfd0[i, g, :USA] x 2.0 & rtfi0[i, g, r] x 2.0 ; r ∈ USA, i ∈ coa, p_c, gas

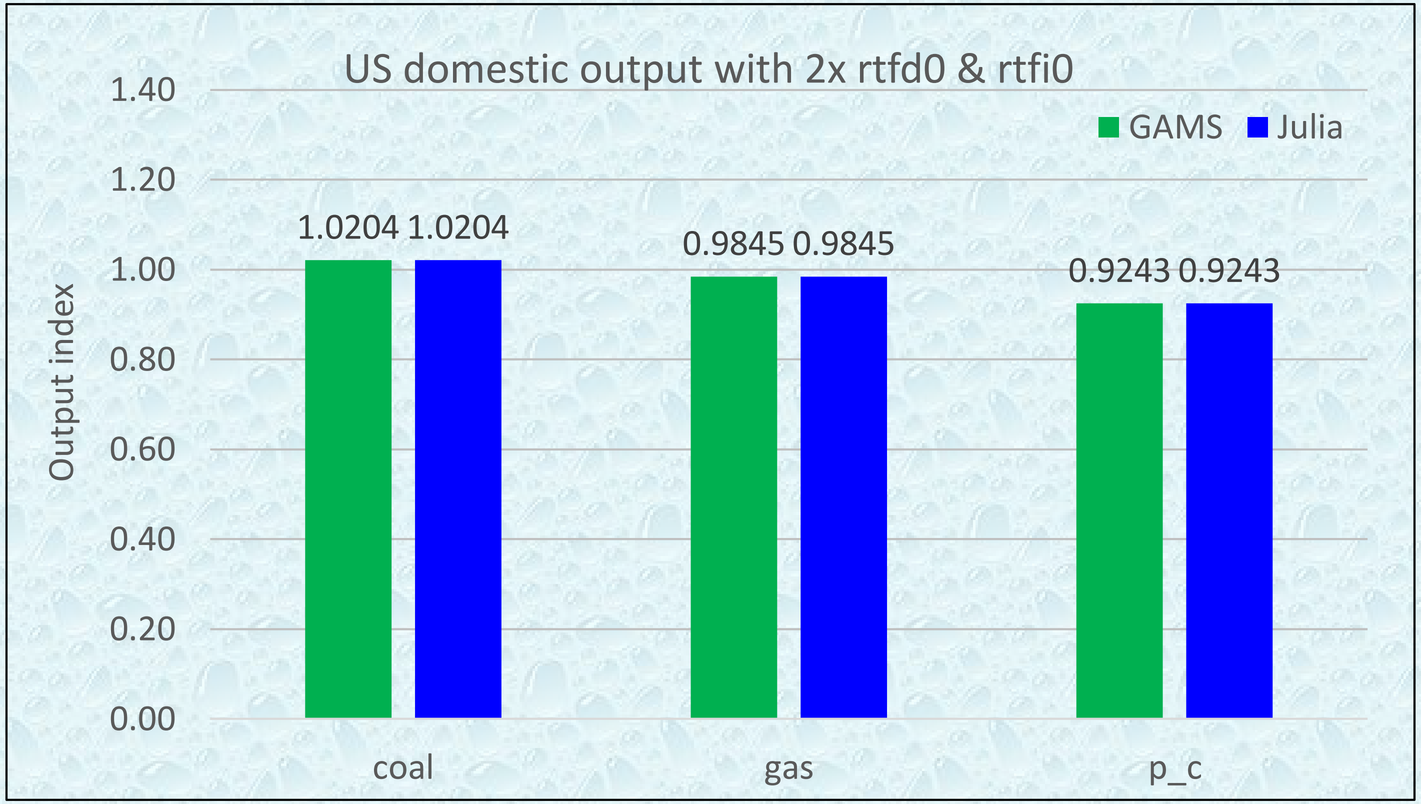
Simulation



The two versions of CSAVE should produce exactly the same results!

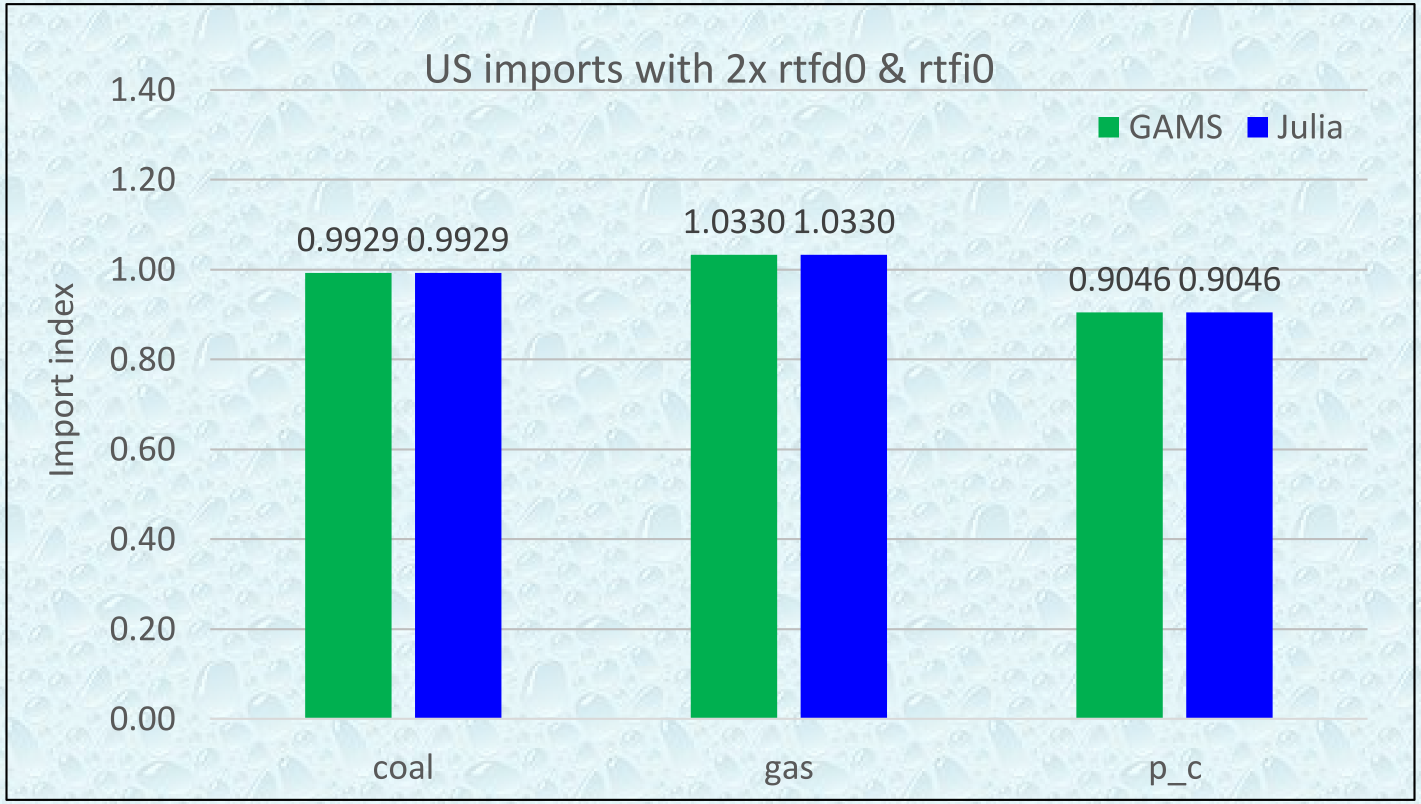
Center for Sustainability Science and Strategy

Simulation



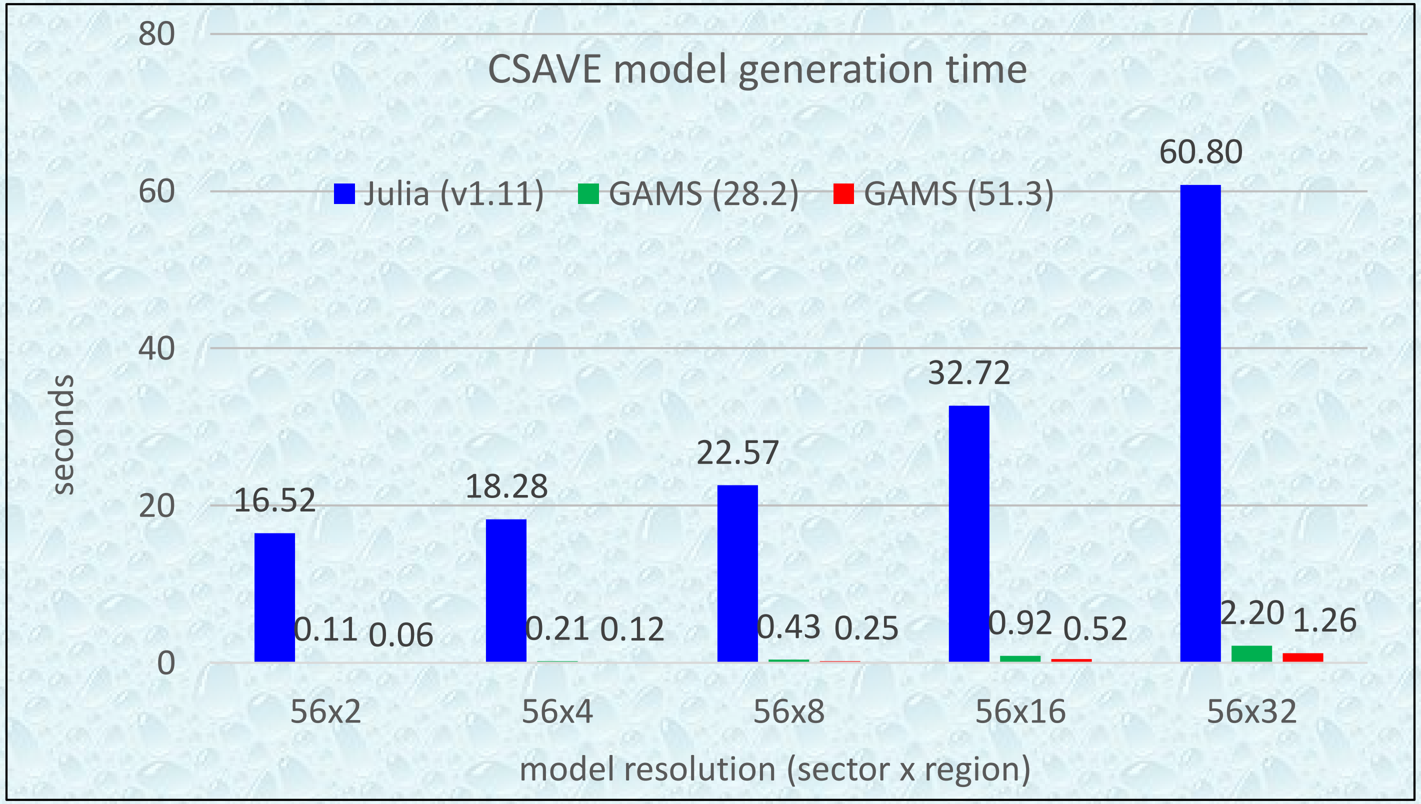
The two versions of CSAVE should produce exactly the same results!

Simulation



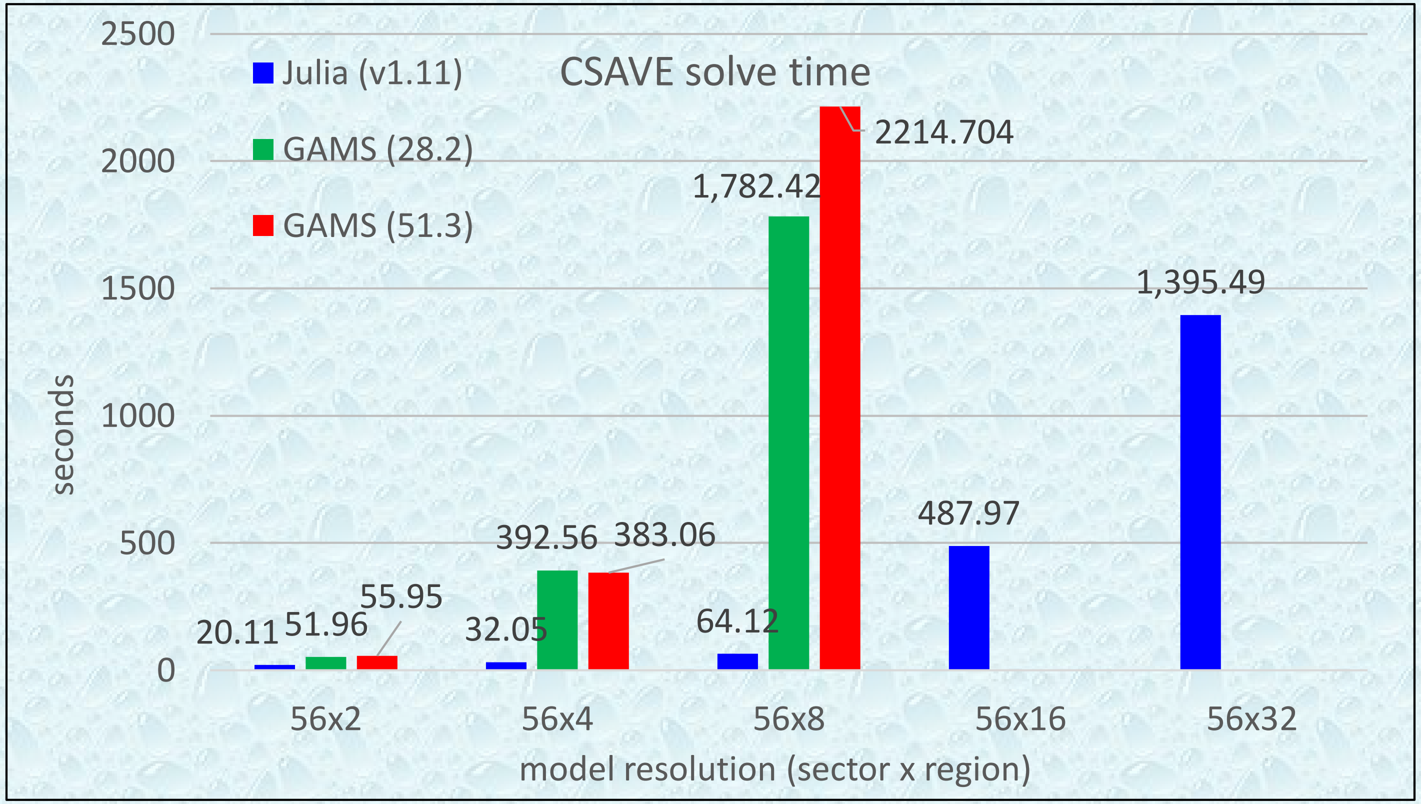
The two versions of CSAVE should produce exactly the same results!

Generation time



Center for Sustainability Science and Strategy

Solve time

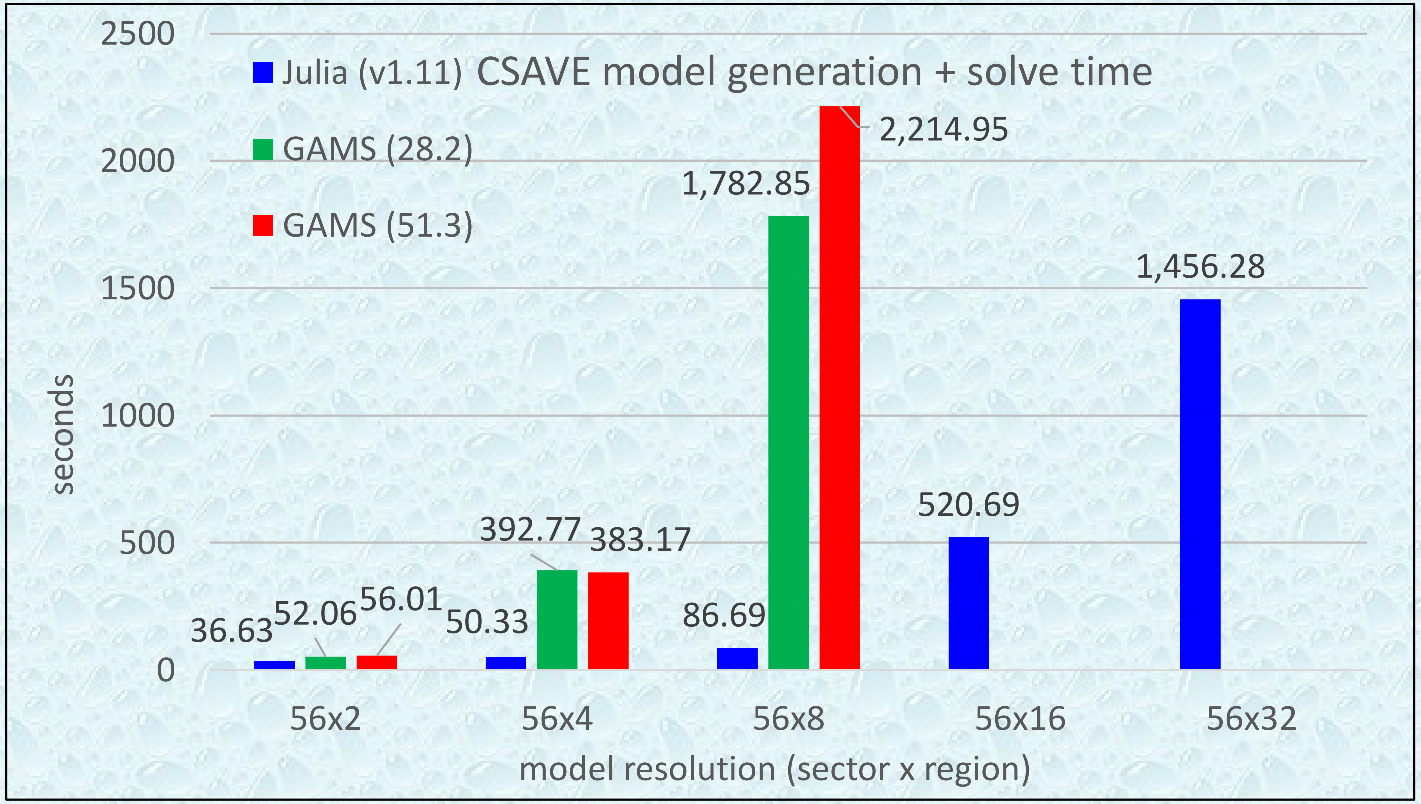


Note: the time scale is much bigger here.

GAMS is time out before finding a solution.

Center for Sustainability Science and Strategy

Generation + Solve time



Center for Sustainability Science and Strategy

Summary

- Julia
  - CSAVEinJulia: slower in model generation
  - CSAVEinJulia: faster in solve
  - More involved in coding
- GAMS
  - CSAVEinGAMS: faster in model generation
  - CSAVEinGAMS: slower in solve
  - Easier to implement

Next steps

- Apply statistical methods
- Extend the GTAPdata package
- Build a web interface

Acknowledgement

I am thankful for Mitch Phillipson for sharing his knowledge on CGE modeling in Julia, and for Angelo Gurgel who provides me Julia course materials. All errors are my own.

Thank you!

Questions?

Y.-H. Henry Chen
chenyh@mit.edu