Solving PDEs with Backward Implicit Time Steps

Matthieu Gomez *

February 15, 2023

This package EconPDEs.jl introduces a fast and robust way to solve systems of PDEs + algebraic equations (i.e. DAEs) associated with economic models. It builds on the method presented in Achdou et al. (2016), but extends it to handle non-linearities.

Consider a PDE of the form

$$\partial_t V = f(x, V, \partial_x V, \partial_{xx} V) \tag{1}$$

As in Achdou et al. (2016), I first discretize the state x on a grid and I approximate derivatives $\partial_x V_t$ and $\partial_{xx} V_t$ using finite difference approximations. First order derivatives are upwinded, which helps with convergence and ensures that boundary counditions are satisfied.

This discretization allows one to rewrite the PDE as an ODE in a vector V:

$$\partial_t V = F(V)$$

where F is a non linear function of the vector V.

I then use a stiff method to solve for this ODE. More precisely, given V_t, I solve for

$$\frac{V_{t+1} - V_t}{\Delta} = F(V_{t+1}), \tag{2}$$

which is a non linear equation. Two solutions:

1. Solve for the non-linear equation using the Ralphson-Newson algorithm
2. Decompose the ODE into a linear and a non linear part, and only solve for the linear part implicitly:

$$\begin{aligned} \frac{V_{t+1} - V_t}{\Delta} &= \partial_V F(V_t) V_{t+1} + F(V_t) - \partial_V F(V_t) V_t \\ \Rightarrow V_{t+1} &= (1 - \Delta \partial_V F(V_t))^{-1} (V_t + \Delta (F(V_t) - \partial_V F(V_t) V_t)) \end{aligned}$$

*I thank Valentin Haddad, Ben Moll, and Dejanir Silva for useful discussions.

In the particular case in which F is linear, this becomes

$$V_{t+1} = (1 - \Delta \partial_V F(V_t))^{-1} V_t$$

which is the implicit method discussed in Achdou et al. (2016).

In both cases, the (sparse) Jabobian $\partial_V F$ is automatically computed using the Julia packages ForwardDiff and SparseDiffTools.

Stationary Solution In most cases, one is only interested in the stationary solution of the PDE (1), i.e.,

$$0 = f(x, V, \partial_x V, \partial_{xx} V) \tag{3}$$

In this case, I use the same method, but I adapt the time step Δ over time. More precisely, if the time iteratio (2) is successful (i.e., the Newton-Raphson method converges), Δ is increased; otherwise, it is decreased. This method ensures converges since, the Newton-Raphson method always converges for Δ small enough. Yet, it does not sacrifice speed: as $\Delta \to \infty$, the method becomes equivalent to a non-linear solver for the PDE, which ensures quadratic convergence around the solution.

The idea of adapting Δ comes from the Pseudo-Transient Continuation method used in the fluid dynamics literature. Formal conditions for the convergence of the algorithm are given in Kelley and Keyes (1998).

Applications Empirically, I find the method to be fast and robust — the examples folder shows that the algorithm solves a wide range of asset pricing models.

References

Achdou, Yves, Jiequn Han, Jean-Michel Lasry, Pierre-Louis Lions, and Benjamin Moll, "Heterogeneous Agent Models in Continuous Time," 2016. Working Paper.

Kelley, Carl Timothy and David E Keyes, "Convergence analysis of pseudo-transient continuation," *SIAM Journal on Numerical Analysis*, 1998, *35* (2), 508–523.