Valuation Dynamics in Models with Financial Frictions

Model Solution

Lars Peter Hansen (University of Chicago)
Joseph Huang (University of Chicago)
Paymon Khorrami (University of Chicago)
Fabrice Tourre (Northwestern University)

Open Source Macroeconomics Bootcamp

July 11, 2018

Overview of Model Solution: Value Functions

Statement of the problem. Scaled value functions $\{\zeta_i\}_{i \in e,h}$ solve PDEs like

$$0 = K_i + A_i \zeta_i + B_i \cdot \partial_x \zeta_i + \text{trace}[C_i C_i' \partial_{xx'} \zeta_i], \quad x = (w, g, s, \varsigma),$$

where the coefficients are (nonlinearly):

$$\begin{aligned} K_i &= K_i(x, \zeta_e, \zeta_h, \partial_x \zeta_e, \partial_x \zeta_h) \\ A_i &= A_i(x, \zeta_e, \zeta_h, \partial_x \zeta_e, \partial_x \zeta_h) \\ B_i &= B_i(x, \zeta_e, \zeta_h, \partial_x \zeta_e, \partial_x \zeta_h) \\ C_i &= C_i(x, \zeta_e, \zeta_h, \partial_x \zeta_e, \partial_x \zeta_h) \end{aligned}$$

Strategy: Transform the problem into something that we are familiar with: solving linear PDEs using finite difference scheme.

Overview of Model Solution: Brief Introduction on Finite Difference Method

- Consider this simplified linear PDE:

$$
\begin{aligned}
& f(w) = -B(w)\frac{\partial^2 \zeta}{\partial w^2} \\
s.t. \quad & \zeta(w_{\min}) = a \\
& \zeta(w_{\max}) = b
\end{aligned}
$$

Overview of Model Solution: Brief Introduction on Finite Difference Method

- Consider this simplified linear PDE:

$$
\begin{aligned}
f(w) &= -B(w)\frac{\partial^2 \zeta}{\partial w^2} \\
s.t.\ \ \zeta(w_{\min}) &= a \\
\zeta(w_{\max}) &= b
\end{aligned}
$$

- Goal: find $\zeta(w)$ that satisfies the PDE above for all $w \in (0,1)$.

Overview of Model Solution: Brief Introduction on Finite Difference Method

- Consider this simplified linear PDE:

$$
\begin{aligned}
f(w) &= -B(w)\frac{\partial^2 \zeta}{\partial w^2} \\
s.t.\ \ \zeta(w_{\min}) &= a \\
\zeta(w_{\max}) &= b
\end{aligned}
$$

- Goal: find $\zeta(w)$ that satisfies the PDE above for all $w \in (0,1)$.
- Apply finite difference approximation to the PDE:

$$
f(w) = -B(w)\Big[\frac{\zeta(w+\Delta w) - 2\zeta(w) + \zeta(w-\Delta_w)}{(\Delta w)^2}\Big]
$$

Overview of Model Solution: Brief Introduction on Finite Difference Method

- Suppose we discretize the solution space, w, with 100 points. We can rewrite this problem in (sparse) matrix format.

$$\underbrace{\begin{pmatrix} 2B(w_1) & -B(w_1) & \cdots & 0 \\ -B(w_2) & 2B(w_2) & -B(w_2) & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & -B(w_{99}) & 2B(w_{99}) & -B(w_{99}) \\ 0 & 0 & -B(w_{100}) & 2B(w_{100}) \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} \zeta_1 \\ \zeta_2 \\ \vdots \\ \vdots \\ \zeta_{99} \\ \zeta_{100} \end{pmatrix}}_{\zeta} = \underbrace{(\Delta w)^2 \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_{99} \\ f_{100} \end{pmatrix} + \begin{pmatrix} c \\ 0 \\ \vdots \\ \vdots \\ 0 \\ d \end{pmatrix}}_{b}$$

Overview of Model Solution: Brief Introduction on Finite Difference Method

- Suppose we discretize the solution space, w, with 100 points. We can rewrite this problem in (sparse) matrix format.

$$
\underbrace{\begin{pmatrix}
2B(w_1) & -B(w_1) & \cdots & 0 \\
-B(w_2) & 2B(w_2) & -B(w_2) & \vdots \\
\vdots & \ddots & \ddots & \vdots \\
\vdots & -B(w_{99}) & 2B(w_{99}) & -B(w_{99}) \\
0 & 0 & -B(w_{100}) & 2B(w_{100})
\end{pmatrix}}_{L}
\underbrace{\begin{pmatrix}
\zeta_1 \\ \zeta_2 \\ \vdots \\ \vdots \\ \zeta_{99} \\ \zeta_{100}
\end{pmatrix}}_{\zeta}
\underbrace{= (\Delta w)^2 \begin{pmatrix}
f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_{99} \\ f_{100}
\end{pmatrix} + \begin{pmatrix}
c \\ 0 \\ \vdots \\ \vdots \\ 0 \\ d
\end{pmatrix}}_{b}
$$

- Solving for ζ is the same as solving linear system $L\zeta = b$.

Overview of Model Solution: Sparsity

- Notice that in matrix L, most of the elements are zero. We call it a **sparse** matrix.

Overview of Model Solution: Sparsity

- Notice that in matrix L, most of the elements are zero. We call it a **sparse** matrix.
- Significance of sparsity:
  - Storage: We only need to store the nonzero elements and their column and row indices.
  - Computation: Many factorization algorithms and iterative solvers have been developed for sparse matrices, where the time depends on the number of nonzeros.

Overview of Model Solution: Returning to nonlinear PDEs

Step 1: Augment the PDE with an artificial time-derivative $\partial_t \zeta_i$ (“false transient”).

Overview of Model Solution: Returning to nonlinear PDEs

Step 1: Augment the PDE with an artificial time-derivative $\partial_t \zeta_i$ ("false transient").

That is,

$$0 = K_i + A_i \zeta_i + B_i \cdot \partial_x \zeta_i + \text{trace}[C_i C_i' \partial_{xx'} \zeta_i]$$

Overview of Model Solution: Returning to nonlinear PDEs

Step 1: Augment the PDE with an artificial time-derivative $\partial_t \zeta_i$ ("false transient").

That is,

$$0 = K_i + A_i \zeta_i + B_i \cdot \partial_x \zeta_i + \mathrm{trace}[C_i C_i' \partial_{xx'} \zeta_i]$$

becomes

$$\partial_t \zeta_i = K_i + A_i \zeta_i + B_i \cdot \partial_x \zeta_i + \mathrm{trace}[C_i C_i' \partial_{xx'} \zeta_i],$$

Overview of Model Solution: Returning to nonlinear PDEs

Step 1: Augment the PDE with an artificial time-derivative $\partial_t \zeta_i$ ("false transient").

That is,

$$0 = K_i + A_i \zeta_i + B_i \cdot \partial_x \zeta_i + \text{trace}[C_i C_i' \partial_{xx'} \zeta_i]$$

becomes

$$\partial_t \zeta_i = K_i + A_i \zeta_i + B_i \cdot \partial_x \zeta_i + \text{trace}[C_i C_i' \partial_{xx'} \zeta_i],$$

Applying finite difference approximation to time yields:

$$\frac{\zeta_i^{t+\Delta t} - \zeta_i^t}{\Delta t} = K_i + A_i \zeta_i + B_i \cdot \partial_x \zeta_i + \text{trace}[C_i C_i' \partial_{xx'} \zeta_i]$$

Overview of Model Solution: Returning to nonlinear PDEs

Step 2. Linearize the PDEs.

Overview of Model Solution: Returning to nonlinear PDEs

Step 2. Linearize the PDEs.

First, write the coefficients K, A, B, C as functions of ζ_i^t and the derivatives of ζ_i^t. That is,

$$\frac{\zeta_i^{t+\Delta t} - \zeta_i^t}{\Delta t} = K_i^{(t)} + A_i^{(t)} \zeta_i + B_i^{(t)} \cdot \partial_x \zeta_i + \text{trace}[C_i^{(t)} C_i^{(t)\prime} \partial_{xx'} \zeta_i],$$

Overview of Model Solution: Returning to nonlinear PDEs

Step 2. Linearize the PDEs.

First, write the coefficients K, A, B, C as functions of ζ_i^t and the derivatives of ζ_i^t. That is,

$$\frac{\zeta_i^{t+\Delta t} - \zeta_i^t}{\Delta t} = K_i^{(t)} + A_i^{(t)}\zeta_i + B_i^{(t)} \cdot \partial_x \zeta_i + \mathrm{trace}[C_i^{(t)} C_i^{(t)\prime} \partial_{xx'} \zeta_i],$$

Second, substitute ζ_i with $\zeta_i^{t+\Delta t}$

$$\frac{\zeta_i^{t+\Delta t} - \zeta_i^t}{\Delta t} = K_i^{(t)} + A_i^{(t)}\zeta_i^{t+\Delta t} + B_i^{(t)} \cdot \partial_x \zeta_i^{t+\Delta t} + \mathrm{trace}[C_i^{(t)} C_i^{(t)\prime} \partial_{xx'} \zeta_i^{t+\Delta t}]$$

Overview of Model Solution: Returning to nonlinear PDEs

Step 2. Linearize the PDEs.

First, write the coefficients K, A, B, C as functions of ζ_i^t and the derivatives of ζ_i^t. That is,

$$\frac{\zeta_i^{t+\Delta t} - \zeta_i^t}{\Delta t} = K_i^{(t)} + A_i^{(t)}\zeta_i + B_i^{(t)} \cdot \partial_x \zeta_i + \text{trace}[C_i^{(t)} C_i^{(t)\prime} \partial_{xx'} \zeta_i],$$

Second, substitute ζ_i with $\zeta_i^{t+\Delta t}$

$$\frac{\zeta_i^{t+\Delta t} - \zeta_i^t}{\Delta t} = K_i^{(t)} + A_i^{(t)}\zeta_i^{t+\Delta t} + B_i^{(t)} \cdot \partial_x \zeta_i^{t+\Delta t} + \text{trace}[C_i^{(t)} C_i^{(t)\prime} \partial_{xx'} \zeta_i^{t+\Delta t}]$$

We now have a linear PDE.

Overview of Model Solution: Returning to nonlinear PDEs

Step 3: Iterate on time. At time $t = 0$, make a guess for $\zeta^{t=0}$. Then we know $\zeta^{t=0}$ and can solve for $\zeta^{t=1}$.

$$\frac{\zeta_i^{t=1} - \zeta_i^{t=0}}{\Delta t} = K_i^{(t=0)} + A_i^{(t=0)} \zeta_i^{t=1} + B_i^{(t=0)} \cdot \partial_x \zeta_i^{t=1}$$
$$+ \text{trace}[C_i^{(t=0)} C_i^{(t=0)\prime} \partial_{xx'} \zeta_i^{t=1}]$$

Overview of Model Solution: Returning to nonlinear PDEs

Step 3: Iterate on time. At time $t = 0$, make a guess for $\zeta^{t=0}$. Then we know $\zeta^{t=0}$ and can solve for $\zeta^{t=1}$.

$$\frac{\zeta_i^{t=1} - \zeta_i^{t=0}}{\Delta t} = K_i^{(t=0)} + A_i^{(t=0)}\zeta_i^{t=1} + B_i^{(t=0)} \cdot \partial_x \zeta_i^{t=1} + \text{trace}[C_i^{(t=0)} C_i^{(t=0)\prime} \partial_{xx'} \zeta_i^{t=1}]$$

After solving for $\zeta^{t=1}$, use $\zeta^{t=1}$ to solve for $\zeta^{t=2}$. Repeat the procedure until $\frac{\zeta^{t+\Delta t} - \zeta^t}{\Delta t} < \epsilon$.

Overview of Model Solution: Summary and Challenges

- Summary

Overview of Model Solution: Summary and Challenges

- Summary
  1. Make a guess at $t = 0$: $\zeta^{t=0}$.

Overview of Model Solution: Summary and Challenges

- Summary
  1. Make a guess at $t = 0$: $\zeta^{t=0}$.
  2. Use $\zeta^{t=0}$ to update the coefficients of the PDEs

Overview of Model Solution: Summary and Challenges

- Summary
  1. Make a guess at $t = 0$: $\zeta^{t=0}$.
  2. Use $\zeta^{t=0}$ to update the coefficients of the PDEs
  3. **Solve for $\zeta^{t=1}$ by solving the linear system:** $L\zeta^{t=1} = \zeta^{t=0}$ (major bottleneck)

Overview of Model Solution: Summary and Challenges

- Summary
  1. Make a guess at $t = 0$: $\zeta^{t=0}$.
  2. Use $\zeta^{t=0}$ to update the coefficients of the PDEs
  3. **Solve for $\zeta^{t=1}$ by solving the linear system:** $L\zeta^{t=1} = \zeta^{t=0}$ (major bottleneck)
  4. Repeat (2) and (3) until convergence

Overview of Model Solution: Summary and Challenges

- Summary
  1. Make a guess at $t = 0$: $\zeta^{t=0}$.
  2. Use $\zeta^{t=0}$ to update the coefficients of the PDEs
  3. **Solve for $\zeta^{t=1}$ by solving the linear system:** $L\zeta^{t=1} = \zeta^{t=0}$ (major bottleneck)
  4. Repeat (2) and (3) until convergence
- Challenges

Overview of Model Solution: Summary and Challenges

- Summary
  1. Make a guess at $t = 0$: $\zeta^{t=0}$.
  2. Use $\zeta^{t=0}$ to update the coefficients of the PDEs
  3. **Solve for $\zeta^{t=1}$ by solving the linear system:** $L\zeta^{t=1} = \zeta^{t=0}$ (major bottleneck)
  4. Repeat (2) and (3) until convergence

- Challenges
  1. Curse of dimensionality: The matrix size in step (3) increases exponentially in the number of state variables. Suppose we discretize 100 points in each state variable, then the matrix would be of the size $100^n \times 100^n$ (n being the number of state variables).

Overview of Model Solution: Summary and Challenges

- Summary
  1. Make a guess at $t = 0$: $\zeta^{t=0}$.
  2. Use $\zeta^{t=0}$ to update the coefficients of the PDEs
  3. **Solve for $\zeta^{t=1}$ by solving the linear system:** $L\zeta^{t=1} = \zeta^{t=0}$ (major bottleneck)
  4. Repeat (2) and (3) until convergence

- Challenges
  1. Curse of dimensionality: The matrix size in step (3) increases exponentially in the number of state variables. Suppose we discretize 100 points in each state variable, then the matrix would be of the size $100^n \times 100^n$ (n being the number of state variables).
  2. In each time step, the matrix gets updated (remember, the coefficients of the PDEs in each time step t depend on ζ^{t-1}). Therefore, we can't save the LU factors and re-apply them (more on this later).

Overview of Model Solution: Using Pardiso

Method 1: Use Pardiso ([1], [2], [3], [4]) (Parallel Sparse Direct Solver) for matrix decomposition (direct method)

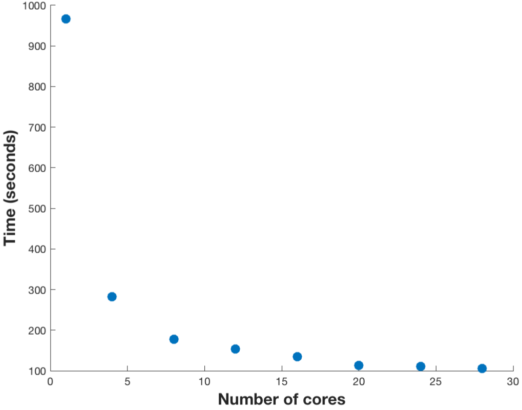
Performance: Without Parallel Processing

Suppose we initialize 100 grid points in each dimension and it takes 500 iterations (time steps) to converge:

Dimensions	Matrix Size	Time (seconds) per iteration	Total Time (Minutes)
1	$100^1 \times 100^1$	0.02	0.14
2	$100^2 \times 100^2$	0.12	1.03
3	$100^3 \times 100^3$	966.27	8052 (134 hours)

Overview of Model Solution: Using Pardiso

Using RCC's resources, we can employ parallel processors.
We show the time needed to solve two $100^3 \times 100^3$ linear systems with multiple cores per iteration:



Overview of Model Solution: Using Pardiso

However, we have a new matrix in each time step, but the procedure so far doesn't information from previous time steps. Let's try to change that.

Overview of Model Solution: Introduction to CG

Consider minimizing the quadratic form:

$$min\ f(\zeta) = \frac{1}{2}\zeta^T L\zeta - \zeta^T b$$

If L is symmetric, the first order condition of this minimization problem is

$$\partial_\zeta f(\zeta) = L\zeta - b = 0$$

Overview of Model Solution: Introduction to CG

Consider minimizing the quadratic form:

$$min\ f(\zeta) = \frac{1}{2}\zeta^T L\zeta - \zeta^T b$$

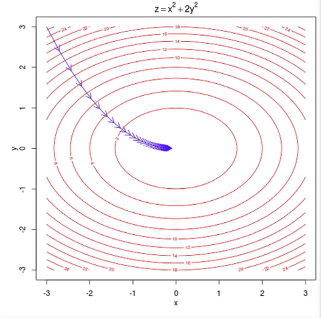
If L is symmetric, the first order condition of this minimization problem is

$$\partial_\zeta f(\zeta) = L\zeta - b = 0$$

- If L is positive definite, solving the linear system $L\zeta = b$ is equivalent to finding the minimum of the quadratic form.
- The matrix from our finite difference scheme is not symmetric or positive definite, but we can do the following transformation: $\widetilde{L} = L^T L$ and $\tilde{b} = L^T b$.

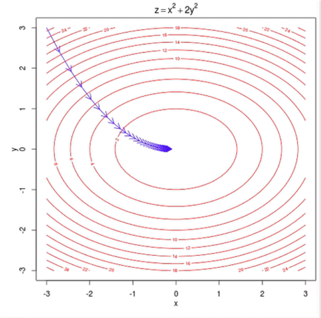
Overview of Model Solution: Introduction to CG

- We use the conjugate gradient (CG) method to find the minimum. As a starter, consider the steepest descent algorithm.



Overview of Model Solution: Introduction to CG

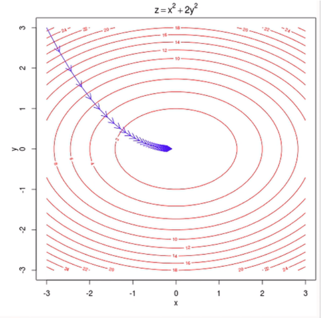
- We use the conjugate gradient (CG) method to find the minimum. As a starter, consider the steepest descent algorithm.



- CG chooses step sizes and directions in a more intelligent way, so that it does not have to step in the same direction repeatedly (details omitted). Hence, the number of steps required is bounded by the dimension of the matrix.

Overview of Model Solution: Introduction to CG

- We use the conjugate gradient (CG) method to find the minimum. As a starter, consider the steepest descent algorithm.



- CG chooses step sizes and directions in a more intelligent way, so that it does not have to step in the same direction repeatedly (details omitted). Hence, the number of steps required is bounded by the dimension of the matrix.
- **Note**: CG gives an approximate solution, instead of the exact solution.

Overview of Model Solution: New Algorithm

The new algorithm, therefore, becomes:

1. Make a guess at $t = 0$: $\zeta^{t=0}$.

Overview of Model Solution: New Algorithm

The new algorithm, therefore, becomes:

1. Make a guess at $t = 0$: $\zeta^{t=0}$.
2. Use $\zeta^{t=0}$ to update the coefficients of the PDEs

Overview of Model Solution: New Algorithm

The new algorithm, therefore, becomes:

1. Make a guess at $t = 0$: $\zeta^{t=0}$.
2. Use $\zeta^{t=0}$ to update the coefficients of the PDEs
3. Solve for $\zeta^{t=1}$ by ~~solving the linear system:~~ $\cancel{L\zeta^{t=1} = \zeta^{t=0}}$ **minimizing** $\frac{1}{2}(\zeta^{t=1})^T L\zeta^{t=1} - (\zeta^{t=1})^T \zeta^{t=0}$ **via CG**

Overview of Model Solution: New Algorithm

The new algorithm, therefore, becomes:

1. Make a guess at $t = 0$: $\zeta^{t=0}$.
2. Use $\zeta^{t=0}$ to update the coefficients of the PDEs
3. Solve for $\zeta^{t=1}$ by ~~solving the linear system:~~ $\cancel{L\zeta^{t=1} = \zeta^{t=0}}$ **minimizing** $\frac{1}{2}(\zeta^{t=1})^T L\zeta^{t=1} - (\zeta^{t=1})^T\zeta^{t=0}$ **via CG**
4. Repeat (2) and (3) until convergence.

Overview of Model Solution: Smart Guesses

On top of that, in each time step, we can inject information from previous time steps.

Overview of Model Solution: Smart Guesses

On top of that, in each time step, we can inject information from previous time steps.

- Recall that we assume $\{\zeta_t\} \to \zeta$ as $t \to \infty$. Further assume that $d(\zeta_{t-1}, \zeta_t)$ is **monotonically decreasing** almost everywhere in time.

Overview of Model Solution: Smart Guesses

On top of that, in each time step, we can inject information from previous time steps.

- Recall that we assume $\{\zeta_t\} \to \zeta$ as $t \to \infty$. Further assume that $d(\zeta_{t-1}, \zeta_t)$ is **monotonically decreasing** almost everywhere in time.
- In each time step t, when implementing CG, instead of starting with an arbitrary point, **start with $\zeta^{(t-1)}$ (i.e. solution from the previous time step) as your starting point**.

Overview of Model Solution: Smart Guesses

On top of that, in each time step, we can inject information from previous time steps.

- Recall that we assume $\{\zeta_t\} \to \zeta$ as $t \to \infty$. Further assume that $d(\zeta_{t-1}, \zeta_t)$ is **monotonically decreasing** almost everywhere in time.
- In each time step t, when implementing CG, instead of starting with an arbitrary point, **start with $\zeta^{(t-1)}$ (i.e. solution from the previous time step) as your starting point**.
- Since $d(\zeta_{t-1}, \zeta_t)$ decreases in t, the starting point becomes less and less arbitrary (hence the name "smart guess").

Overview of Model Solution: New Algorithm (2)

The **modified** new algorithm, therefore, becomes:

1. Make a guess at $t = 0$: $\zeta^{t=0}$.

Overview of Model Solution: New Algorithm (2)

The **modified** new algorithm, therefore, becomes:

1. Make a guess at $t = 0$: $\zeta^{t=0}$.
2. Use $\zeta^{t=0}$ to update the coefficients of the PDEs

Overview of Model Solution: New Algorithm (2)

The **modified** new algorithm, therefore, becomes:

1. Make a guess at $t = 0$: $\zeta^{t=0}$.
2. Use $\zeta^{t=0}$ to update the coefficients of the PDEs
3. Solve for $\zeta^{t=1}$ by ~~solving the linear system:~~ $\cancel{L\zeta^{t=1} = \zeta^{t=0}}$ **minimizing** $\frac{1}{2}(\zeta^{t=1})^T L\zeta^{t=1} - (\zeta^{t=1})^T \zeta^{t=0}$ **via CG**

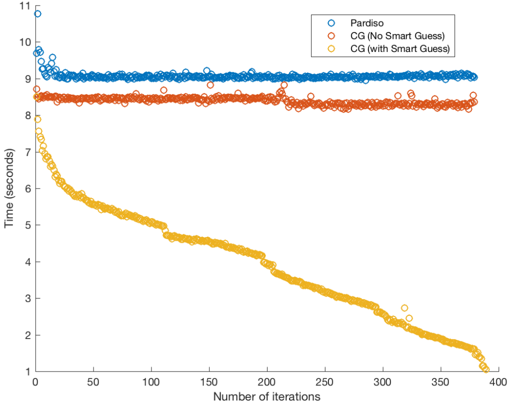
Overview of Model Solution: New Algorithm (2)

The **modified** new algorithm, therefore, becomes:

1. Make a guess at $t = 0$: $\zeta^{t=0}$.
2. Use $\zeta^{t=0}$ to update the coefficients of the PDEs
3. Solve for $\zeta^{t=1}$ by ~~solving the linear system:~~ $\cancel{L\zeta^{t=1} = \zeta^{t=0}}$ **minimizing** $\frac{1}{2}(\zeta^{t=1})^T L\zeta^{t=1} - (\zeta^{t=1})^T \zeta^{t=0}$ **via CG**
4. Repeat (2) and (3) until convergence. **If** $t > 1$**, start CG with point** ζ^{t-1}.

Overview of Model Solution: Performance

We show the performance of Pardiso (LU decomposition), CG without smart guesses, and CG with smart guesses. This test is run on matrix size $250,000 \times 250,000$.



References

[1] A. Fuchs D. Kourounis and O. Schenk. Towards the next generation of multiperiod optimal power flow solvers. *Transactions on Power Systems*, PP(99):1–10, 2018.

[2] D. Kourounis F. Verbosio, A. D. Coninck and O. Schenk. Enhancing the scalability of selected inversion factorization algorithms in genomic predictions. *Journal of Computational Science*, 22(Supplement C):99–108, 2017.

[3] D. Kourounis F. Verbosio O. Schenk S. Maenhout A. De Coninck, B. De Baets and J. Fostier. Needles: Toward large-scale genomic prediction with marker-by-environment interaction. *Genetics*, 203(1):543–555, 2016.

[4] Simon Urs Scheidegger. *Gravitational waves from 3D MHD core-collapse supernova simulations with neutrino transport*. PhD thesis, University of Basel, 2011.