Parallel algorithm for solving large-scale dynamic general equilibrium models

N. B. Melnikov[1], A. P. Gruzdev[1], M. G. Dalton[2], B. C. O'Neill[3]

Lomonosov Moscow State University[1]
National Oceanic and Atmospheric Administration, USA[2]
National Center for Atmospheric Research, USA[3]

We present a parallel algorithm for computing an equilibrium path in a large-scale economic growth model. We exploit the special block structure of the nonlinear systems of equations common in such models. Our algorithm is based on an iterative method of Gauss-Seidel type with prices of different time periods calculated simultaneously rather than recursively. We have implemented the parallel algorithm in OpenMP and MPI programming environments. The numerical results show that speedup improves almost linearly as number of nodes increases. Different methods for solving an individual block: Newton-type methods, Krylov subspace methods and trust-region methods, give similar results for the speedup.

1. Introduction

Dynamic general equilibrium (GE) models are used to quantify the effects on economy due to demographic, technological and climate change (see, e.g., [1, 2] and refs. therein). Dynamic GE is described by large-scale systems of nonlinear equations. One of the the most popular early methods for solving such nonlinear systems was the Fair-Taylor method [3], which is a variation of the Gauss-Seidel method. Recent contributions have concentrated on Newton-type methods that are now more widely used (see, e.g., [4]).

In this paper, we give the Fair-Taylor method a fresh look. We use the special block structure of the equations system common in the dynamic GE models to describe and implement a parallel algorithm. The idea of taking block structure into account was used previously for multi-country models (see, e.g., [5]). We implement parallel calculation of individual time blocks using OpenMP and MPI environments for systems with shared and distributed memory. For the solution of the time blocks, we compare the performance of different Newton-type methods: LU-factorization, Krylov methods and trust-region methods. The algorithm is demonstrated in the Population–Environmental–Technology (PET) model (see [1, 2] and refs. therein).

The paper is organized as follows. In Section 2 we present the block Gauss-Seidel method and briefly describe the Newton-type methods that we use for solving the equations system of individual blocks. In section 3, by example of the PET model, we derive the nonlinear system of equations that determines the GE. In section 4 we implement the sequential and two parallel versions (OpenMP and MPI) of the algorithm. Finally, in section 5 we present and discuss the numerical results.

2. Block Gauss-Seidel method

First, we briefly recall the Gauss-Seidel method for a linear equation system $Ax = b$, where A is a nondegenerate $n \times n$ matrix with nonzero diagonal elements (see, e.g., [6]). Assume that the kth iterate $x^k = (x_1^k, \ldots, x_n^k)^T$ and $i-1$ components $x_1^{k+1}, \ldots, x_{i-1}^{k+1}$ of the $(k+1)$th iterate x^{k+1} have been determined. Then to obtain x_i^{k+1} the ith equation of the system

$$\sum_{j=1}^{i-1} a_{ij} x_j^{k+1} + a_{ii} x_i + \sum_{j=i+1}^{n} a_{ij} x_j^k = b_i,$$

is solved with respect to x_i and the solution is taken as x_i^{k+1}. We write method in a more compact form by splitting the matrix as $A = D + L + U$, where D is diagonal and L and U are strictly lower and upper triangular parts. Then

$$x^{k+1} = Mx^k + c,$$

where $M = -(D + L)^{-1}U$ is the iteration matrix and $c = (D + L)^{-1}b$. The usual termination criterium is $||r_k||/||b|| < \varepsilon$, where $r_k = b - Ax_k$ is the residual. Since

$$r_k = (D + L)(x^{k+1} - x^k),$$

we can use the properly scaled error $\|x^{k+1} - x^k\|$ to terminate iterations.

The same idea is generalized to a nonlinear equations system $f(x) = 0$, where $f = (f_1, \ldots, f_n)$. To obtain x_i^{k+1} the ith equation

$$f_i(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_i, x_{i+1}^k, \ldots, x_n^k) = 0$$

is solved with respect to x_i and the solution is taken as x_i^{k+1}. An advantage of the method is that one does not need to calculate the Jacobian of the system but only values of the functions $f_i(x)$.

Now we assume that f_i and x_i are vectors:

$$f_i = (f_{i1}, \ldots, f_{ij}, \ldots, f_{il}), \qquad x_i = (x_{i1}, \ldots, x_{ij}, \ldots, x_{il}),$$

where the blocks f_{ij} and x_{ij} are also vectors having the same dimension. Let the jth block of functions f_{ij} depend only on the jth block of variables x_{ij}. Then to obtain x_i^{k+1} the equations

$$f_{ij}(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x, x_{i+1}^k \ldots, x_n^k) = 0, \qquad j = \overline{1, l}, \tag{1}$$

are solved independently with respect to x and the solution is taken as x_{ij}^{k+1}.

To solve the equations system for a separate block (1), which we denote $F(x) = 0$ for now, it is reasonable to use a method with superlinear convergence. Here, we describe three methods that are further used for comparison: Newton's method, Krylov subspace method and trust-region method. In each of the three methods, at the current iterate x_k, the linearized system

$$F'(x_k)s_k = -F(x_k) \tag{2}$$

is solved with respect to the step s_k in order to obtain the next iterate $x_{k+1} = x_k + s_k$.

In Newton's method, a LU-factorization of the Jacobian $F'(x_k)$ is used (see, e.g., [6]). The inputs of the algorithm are the initial iterate x_0 and a termination tolerance tol.

```
   input : x_0, tol
   output: the solution x_k
1  k ← 0;
2  while ||F(x_k)|| > tol do
3  |  solve (2) with respect to s_k using the LU-factorization;
4  |  x_{k+1} ← x_k + s_k;
5  |  k ← k + 1;
6  end
```

Figure 1. Newton's method.

In the Newton Iterative Solver (NITSOL) [7, 8], instead of the direct method for solving the linearized system, an inexact Newton's method with backtracking is used. The termination criterium is

$$||F'(x_k)s_k + F(x_k)|| \leq \eta_k||F(x_k)||, \tag{3}$$

where $\eta_k \in [0, 1)$ is the tolerance. To obtain the step s_k, the generalized minimal residual (GMRES) Krylov subspace method is used (see, e.g., [9]). Once an initial s_k has been determined, it is tested and, if necessary, reduced in length through backtracking until an acceptable step is obtained. The parameter $t \in (0, 1)$ is used to judge sufficient reduction $\theta \in (\theta_{min}, \theta_{max})$. The Jacobian is calculated using the difference approximation, just as in Newton's method.

```
input : x_0, tol;
        η_max ∈ [0,1), t ∈ (0,1), 0 < θ_min < θ_max < 1
output: the solution x_k
k ← 0;
while ||F(x_k)|| > tol do
    solve (2) with respect to s_k using a Krylov subspace method (termination
    criterium (3));
    while ||F(x_k + s_k)|| > [1 − t(1 − η_k)]||F(x_k)|| do
        η_k ← 1 − θ(1 − η_k);
        s_k ← θ s_k, θ ∈ [θ_min, θ_max];
    end
    x_{k+1} ← x_k + s_k;
    k ← k + 1;
end
```

Figure 2. Inexact Newton with backtracking.

The third method solves the system of nonlinear equations using the trust-region method to obtain the current iterate (see, e.g., [10]). In the routine NEQBF, the following problem

$$||F(x_k) + B_k s_k||^2 \to \min_{s_k}, \tag{4a}$$

$$||s_k|| \le \delta_k. \tag{4b}$$

is solved to get the direction s_k, where δ_k is the size of the trust region and B_k is the approximate Jacobian evaluated at the current point x_k. Then, the function at the point $x_{k+1} = x_k + s_k$ is evaluated and used to decide whether the new point x_k should be accepted. Problem (4) is solved approximately by the double dogleg method. The Jacobian is approximated by Broyden's formula

$$B_{k+1} = B_k + \frac{([F(x_{k+1}) - F(x_k)] - B_k s_k) s_k^T}{s_k^T s_k}.$$

```
input : x_0, tol;
        δ_0, θ ∈ (0,1)
output: the solution x_k
k ← 0;
while ||F(x_k)|| < tol do
    δ_k ← δ_0;
    while f(x_{k+1}) ≥ f(x_k) do
        solve (4) with respect to s_k using the double dogleg method;
        δ_k ← θ δ_k;
        x_{k+1} ← x_k + s_k;
    end
    k ← k + 1;
end
```

Figure 3. Trust-region method.

3. Dynamic general equilibrium model

In this section, we show that a dynamic GE model leads to a block equations system. To be specific, we consider the one region PET model (see, e.g., [1, 2] for details). PET is a neoclassical growth models with three types of agents: consumers, producers and government. Consumers take the prices as given and solve the utility maximization problem over the infinite time-horizon under their budget constraints (rational expectations). Producers minimize costs given the levels of production and prices at each moment in time. The government plays a neutral role redistributing the capital through taxes and transfers. Prices are determined by the markets clearing conditions. The GE is defined as a solution to the nonlinear equations system that consists of the first-order optimality conditions for consumers and producers and supply-equals-demand conditions for markets.

3.1. Optimal economic growth

The consumer side of the model consists of N_c heterogeneous groups of households. Demand of the ith consumer group is determined by the intertemporal utility function

$$U_i(c_i) = \frac{1}{\psi} \sum_{t=0}^{\infty} \beta^t n_{it} u_i(c_{it}),$$

where $t = 0, 1, ...$ is time, c_{it} is the level of consumption (here and henceforth, small letters indicate the *per capita* values), $\beta \in (0, 1)$ is the discount rate, $\psi \in (-\infty, 1)$ is the time substitution rate and n_{it} is the size of population. The instantaneous utility function $u_i(c_{it})$ is given by the constant-elasticity of substitution (CES) function

$$u_i(c_{it}) = \left(\sum_{j=1}^{N_c} (\mu_{ijt} c_{ijt})^{\rho} \right)^{\frac{\psi}{\rho}},$$

with constant electivity $\sigma = 1/(1-\rho)$. Here, the index $j = \overline{1, N_c}$ labels goods, $\rho \in (-\infty, 1) \setminus \{0\}$ is the substitution parameter among goods and μ_{ijt} is the preference coefficient. Capital dynamics is described as

$$(1 + \nu_{it})\, k_{i,t+1} = (1 - \delta)\, k_{it} + x_{it}, \qquad k_{i0} > 0, \tag{5}$$

where x_{it} is investment, $\delta \in (0, 1)$ is the capital depreciation coefficient, $1 + \nu_{it} = n_{i,t+1}/n_{it}$ is the population growth coefficient (ν_{it} is the growth rate). Budget constraint of the ith consumer group at time t is

$$\sum_{j=1}^{N_c} p_{jt} c_{ijt} + q_t x_{it} = (1 - \theta_{it})\, \omega_t l_{it} + (1 - \phi_{it})\, r_t k_{it} + g_{it}, \tag{6}$$

where p_{jt} is the price of jth consumer good, q_t and r_t are prices of investments and capital, ω_t is the wage rate, g_{it} is the government transfers, l_{it} is the labor supply, θ_{it} and ϕ_{it} are the tax rates on capital and labor incomes.

Variables k_{it}, c_{ijt} and x_{it} are determined by maximizing

$$U_i(c_i) \to \max_{c_{it}, k_{it}, x_{it}}, \tag{7}$$

under constraints (5) and (6). It is convenient to split the solution of problem (5)–(7) into two steps. The first one is to maximize the utility $u_i(c_{it})$ at time t given the expenditures m_{it}:

$$\left(\sum_{j=1}^{N_c} (\mu_{ij} c_{ij})^{\rho} \right)^{\frac{1}{\rho}} \to \max_{c_{ij}}, \qquad \sum_{j=1}^{N_c} p_j c_{ij} = m_i,$$

with respect to consumer demand of different types of goods c_{ij}, i.e. a static problem (we omit the index t where it does not lead to confusion). Solving the dual problem

$$\sum_{j=1}^{N_c} p_j c_{ij} \to \min_{c_{ij}}, \qquad \left(\sum_{j=1}^{N_c} (\mu_{ij} c_{ij})^{\rho}\right)^{\frac{1}{\rho}} = \overline{c}_i,$$

we obtain

$$\min\left(\sum_{j=1}^{N_c} p_j c_{ij}\right) = \bar{p}_i \overline{c}_i, \qquad \bar{p}_i = \left[\sum_{j=1}^{N_c} \left(\frac{p_j}{\mu_{ij}}\right)^{\frac{\rho}{\rho-1}}\right]^{\frac{\rho-1}{\rho}}.$$

The second step is to solve problem (5)–(7) for savings as a function of time (dynamics). Rewriting (5)–(7) in terms of $\bar{p}_{it}$ and $\overline{c}_{it}$:

$$\begin{aligned}
&\frac{1}{\psi}\sum_{t=0}^{\infty} \beta^t n_{it} {\overline{c}_{it}}^{\psi} \to \max_{\overline{c}_{it}, k_{it}},\\
&\bar{p}_{it}\overline{c}_{it} + q_t x_{it} = (1-\theta_{it})\,\omega_t l_{it} + (1-\phi_{it})\,r_t k_{it} + g_{it},\\
&(1+\nu_{it})\,k_{i,t+1} = (1-\delta)\,k_{it} + x_{it},
\end{aligned}$$

we obtain the first-order optimality condition (Euler equation)

$$\frac{q_t}{\bar{p}_{it}} {\overline{c}_{it}}^{\psi-1} = \beta \frac{q_{t+1}(1-\delta) + (1-\phi_{i,t+1}) r_{t+1}}{\bar{p}_{i,t+1}} {\overline{c}_{i,t+1}}^{\psi-1}. \tag{8}$$

The transversality conditions

$$\lim_{t\to\infty} \lambda_{it} k_{it} = 0, \tag{9}$$

where λ_{it} is the Lagrange multiplier, ensures the existence of the optimal trajectory (see, e.g., [11]).

3.2. Duality theory for production functions

The production side of the PET model consists of competitive firms divided into sectors. Each sector produces either a final good: N_C consumer goods and "investment good", or an intermediate good: N_E energy types and the rest, which is called materials ($N_X = N_C{+}1{+}N_E{+}1$ is the total number of production sectors).

Production of the good X is determined by the the inputs of capital K, labor L, energy composite $\bar{E}$ and materials M according to a CES function

$$X = \gamma_X\left(\alpha_K (G_K K)^{\rho_X} + \alpha_L (G_L L)^{\rho_X} + \alpha_{\bar{E}} (G_{\bar{E}} \bar{E})^{\rho_X} + \alpha_M (G_M M)^{\rho_X}\right)^{\frac{1}{\rho_X}}, \tag{10}$$

where G_I is the productivity factor for $I = K, L, \bar{E}, M$ and γ_X normalizes α_I to sum to unity. The parameters α_I and G_I can be sector and time dependent. The producer of the good X minimizes the costs

$$P_K K + P_L L + P_{\bar{E}} \bar{E} + (1+\tau_M) P_M M \to \min_{K,L,\bar{E},M}$$

under the given level of production (10) (τ_M is the *ad-valorem* tax on the use of materials). The solution has the form

$$\min\left(P_K K + P_L L + P_{\bar{E}} \bar{E} + (1+\tau_M) P_M M\right) = P_X X,$$

where the price P_X is [1]

$$P_X = \frac{1}{\gamma_X}\left(\alpha_K^{\frac{1}{1-\rho_X}}\left(\frac{P_K}{G_K}\right)^{\frac{\rho_X}{\rho_X-1}} + \alpha_L^{\frac{1}{1-\rho_X}}\left(\frac{P_L}{G_L}\right)^{\frac{\rho_X}{\rho_X-1}} + \right.$$
$$\left. + \alpha_{\bar{E}}^{\frac{1}{1-\rho_X}}\left(\frac{P_{\bar{E}}}{G_{\bar{E}}}\right)^{\frac{\rho_X}{\rho_X-1}} + \alpha_M^{\frac{1}{1-\rho_X}}\left(\frac{(1+\tau_M)P_M}{G_M}\right)^{\frac{\rho_X}{\rho_X-1}}\right).$$

The input-output coefficients $A_I = I/X$ for $I = K, L, \bar{E}$ are given by

$$A_I = \left(\frac{1}{\alpha_I(\gamma_X G_I)^{\rho_X}}\frac{P_I}{P_X}\right)^{\frac{1}{\rho_X-1}},$$

and for $I = M$ by

$$A_M = \left(\frac{1}{\alpha_M(\gamma_X G_M)^{\rho_X}}\frac{(1+\tau_M)P_M}{P_X}\right)^{\frac{1}{\rho_X-1}}.$$

Similarly, production of the energy composite $\bar{E}$ is determined according to

$$\bar{E} = \gamma_E\left(\sum_i \alpha_{E_i}(G_{E_i}E_i)^{\rho_E}\right)^{\frac{1}{\rho_E}}, \tag{11}$$

where E_i, $i = \overline{1, N_E}$ are different energy types. Solving the cost-minimization problem

$$\sum_i (1+\tau_{E_i})P_{E_i}E_i \to \min_{E_i}$$

under the given the level of production (11), we derive the prices

$$P_{\bar{E}} = \frac{1}{\gamma_E}\left(\sum_i \alpha_{E_i}^{\frac{1}{1-\rho_E}}\left(\frac{(1+\tau_{E_i})P_{E_i}}{G_{E_i}}\right)^{\frac{\rho_E}{\rho_E-1}}\right)^{\frac{\rho_E-1}{\rho_E}}$$

and input-output shares $A_{E_i} = E_i/\bar{E}$ for the energy use:

$$A_{E_i} = \left(\frac{1}{\alpha_{E_i}(\gamma_E G_{E_i})^{\rho_E}}\frac{(1+\tau_{E_i})P_{E_i}}{P_{\bar{E}}}\right)^{\frac{1}{\rho_E-1}},$$

where τ_{E_i} is the *ad-valorem* tax on the energy use.

Government *revenues* include capital and labor income taxes on households and specific taxes on output plus *ad-valorem* taxes on energy use and materials for each industry,

$$GREV_t = \sum_{i=1}^{N_d} n_i(\phi_i r k_i + \theta_i \omega l_i) + \sum_{j=1}^{N_X}\left(\tau_{X_j}X_j + \sum_{s=1}^{N_E}\tau_{E_{S_j}}P_{E_{S_j}}E_{S_j} + \tau_{M_j}M_j\right) =$$
$$= \sum_{i=1}^{N_d}(\phi_i P_K K_i + \theta_i P_L L_i) + \sum_{j=1}^{N_X}\left(\tau_{X_j} + \sum_{s=1}^{N_E}\tau_{E_{s_j}}P_{E_{s_j}}A_{E_{s_j}}A_{\bar{E}_j} + \tau_{M_j}A_{M_j}\right)X_j.$$

Government *expenditures* are the sum of purchases and transfers, $GEXP_t = GP_t + GT_t$. Government *purchases* are assumed to be constant in the current prices: $GP_t/\bar{p}_t = (1+\xi)^t GP_0/\bar{p}_0$,

[1] By the envelope theorem, the price is identified with the marginal cost of production (see, e.g., [12]).

where $\bar{p}_t = (1/N_d)\sum_i^{N_d} \bar{p}_{it}$. It is assumed that GP_0 is given by a Cobb-Douglas production function, so that the input-output coefficients are

$$A_K^{GP} = \frac{\alpha_K}{P_K}, \quad A_L^{GP} = \frac{\alpha_L}{P_L}, \quad A_M^{GP} = \frac{\alpha_M}{P_M}, \quad A_{E_i}^{GP} = \frac{\alpha_{E_i}}{P_{E_i}}, \qquad i = \overline{1, N_E}.$$

Similarly, government baseline *transfers* are neutral. There is additional lump-sum adjustment LSA_t to balance the government budget:

$$GT_t = \frac{\bar{p}_t}{\bar{p}_0}(1+\xi)^t \sum_{i=1}^{N_d} n_{it} g_{i0} + LSA_t.$$

3.3. Equilibrium conditions

Aggregate supply for capital and labor at each point in time are determined by summing over levels supplied by each consumer group:

$$K^{AS} = \sum_{i=1}^{N_d} n_i k_i, \qquad L^{AS} = \sum_{i=1}^{N_d} n_i l_i.$$

Aggregate demand for capital and labor are

$$K^{AD} = \sum_{j=1}^{N_X} A_K^j X_j + A_K^{GP} GP, \qquad L^{AD} = \sum_{j=1}^{N_X} A_L^j X_j + A_L^{GP} GP.$$

Similarly, for consumer goods and investment.

Total demand for intermediate goods $z = \left(X_1^{AD}, \ldots, X_{N_E+1}^{AD}\right)^T$ is the sum of demands to produce other intermediate goods, and those to produce final goods. Total demand for intermediate inputs by final goods producers and government is equal to

$$y = \left(\sum_{N_E+2}^{N_X} A_1^j X_j^{AD} + A_1^{GP} GP, \ldots, \sum_{N_E+2}^{N_X} A_{N_E+1}^j X_j^{AD} + A_{N_E+1}^{GP} GP \right)^T = (Y_1, \ldots, Y_{N_E+1})^T.$$

Since the production functions are homogeneous functions, equilibrium implies

$$X_i^{AD} = \sum_{j=1}^{N_E+1} A_i^j X_j^{AD} + \sum_{j=N_E+2}^{N_X} A_i^j X_j^{AD} + A_i^{GP} GP, \qquad i = \overline{1, N_E+1}.$$

or in the compact form $z = Az + y$, where $A = (A_i^j)$ is the $(N_E+1) \times (N_E+1)$ matrix of input-output coefficients. Hence outputs of the intermediate goods sectors z are determined by the final goods production y as

$$z = (I - A)^{-1} y,$$

where I is the $(N_E+1) \times (N_E+1)$ unity matrix.

A competitive equilibrium in each time moment is defined by markets clearing for factors of production, capital and labor, final goods and a balanced government budget:

$$K^{AD} = K^{AS}, \qquad L^{AD} = L^{AS}, \qquad X^{AD} = X^{AS}, \qquad GREV = GEXP.$$

Note that the consumer prices p_j and q are the gross (after-output-tax) prices P_X of consumer goods and investment good in industry. The rental rate of capital r and wage rate w are equal to the prices of capital P_K and labor P_L in industry.

3.4. Computation of equilibrium

The PET model, as well as other dynamic GE models used in application, evaluates changes during a transition period, and does not address possible effects on the long-run equilibrium. Therefore, the infinite time horizon is replaced by a sufficiently large finite time interval $[0, T]$. As $t \geq T$ the system is assumed to follow the balanced growth path, i.e. the quantities x_{it}, k_{it}, l_{it}, etc. grow exponentially with the growth rate ξ. Proceeding to the limit $t \to \infty$ in (8), we obtain

$$(1+\xi)^{1-\psi} = \beta\left(1-\delta+\frac{(1-\phi)r}{q}\right). \tag{12}$$

The boundary condition (12) at $t = T$ is used instead of transversality condition at infinity (9). This way, we get a two-point (discrete) boundary-value problem for each consumer group.

The nonlinear equations system describing the GE can be written as

$$f(K, P) = 0,$$

where K is the capital and P are prices (at all time moment $t = \overline{0, T}$). For brevity, we do not write consumption, investment and transfers since they can be obtained from K and P. The Gauss-Seidel method is applied to this system as follows. Assume that the sth iteration of capital K^s has been determined. Then solving the system

$$f(K^s, P) = 0$$

with respect to P, we obtain the $(s+1)$th iteration of prices, $P^{s+1} = P$; the previous iteration P^s is used as the initial point of the method. Different time blocks of this part of the algorithm, called the *inner loops*, can be calculated in parallel.

Once P^{s+1} has been determined, the next iteration of capital is obtained by solving the system

$$f(K, P^{s+1}) = 0$$

with respect to K and setting $K^{s+1} = K$. The part of the algorithm that calculates K^{s+1} from K^s is called the *outer loop* (see [1,3] for details of the capital iteration).

4. Algorithm implementation

First, we describe the sequential version (Algorithm 4). The input is the initial guess of the capital K^0 as a function of time and the outputs are the capital K and prices P as functions of time. Here, *tol* is the tolerance and *maxit* is the maximum number of iterations in the outer loop; T is the length of the time interval. Two types of arrays are used to store capital, consumption, prices, etc. as functions of time. The arrays of the first type, called *stor*, contain the results of the *it*-th iterate of the outer loop over all time moments t. The arrays of the second type, called *dyn*, contain the results of the *it*-th iterate of the outer loop for at the current t. Variable $diff$ is the error at the current iterate of the outer loop.

In the OpenMP version, steps of the loop over time are performed in the parallel region (Algorithm 5). All *dyn* and *stor* arrays are shared by all threads. The parameters arrays are transferred through the *copyin* clause. The time steps are distributed into blocks using *schedule(auto)*.

In the MPI version, we use only one communicator with number of nodes *nsize*. Each node has its own copy of the *dyn* and *stor* arrays. Moreover, each node works with its own part of the array, and the rest entries are set to zero. To synchronize the *dyn* and *stor* arrays, we use the collective communication routine MPI_AllReduce with the operation MPI_SUM that sums up different copies of an array and sends the result back to all nodes (lines 20 and 23 of Algorithm 6).

```
   input : K^0
   output: K, P
1  marker:  if diff > tol and it < numIt then
2  |  for t ← 0 to T do
3  |  |  Calculate prices P(t) (inner loop);
4  |  |  Update dyn arrays;
5  |  end
6  |  Update stor arrays;
7  |  it ← (it + 1);
8  |  diff ← update (K, P);
9  end
10 goto marker
```

Figure 4. Sequential version.

```
   input : K^0
   output: K, P
1  marker:  if diff > tol and it < numIt then
2  |  omp parallel default(private)
3  |  omp shared(dyn arrays, stor arrays)
4  |  omp copyin(parameters)
5  |  omp for
6  |  for t ← 0 to T do
7  |  |  Calculate prices P(t) (inner loop);
8  |  |  Update dyn arrays;
9  |  end
10 |  omp end parallel
11 |  Update stor arrays;
12 |  it ← (it + 1);
13 |  diff ← update (K, P);
14 end
15 goto marker
```

Figure 5. OpenMP version.

To calculate *diff* in the outer loop, we normalize the error using the variable *norm*. In the sequential version this variable is calculated in the first step of the loop over time by one of the functions. The variable *normCounter* is used to calculate the number of calls of this function. For better comparison of with the sequential version, we keep the same way of calculating *norm* in the MPI version. To this end, at each iteration of the outer loop, all nodes wait till the node with rank zero is completed. The zero node sends *norm* and *normCounter* using MPI_SEND to all other nodes and they resume their work (lines 5 – 7 of Algorithm 6). This approach does not affect the productivity much since most of the time is spent on collective operations (lines 20 and 23).

5. Results and discussion

The input data of the PET model consists of projections over time and base year data. Projections are used for the population, i.e. the number of consumers in different groups, and for productivity growth coefficients over the interval $[0, T]$. The base year data divides into production data from input-output tables, i.e. production shares, and consumer survey data, i.e. capital, expenditures, savings, etc. of different consumer groups.

The calculations were carried out over $T = 200$ years with the time step of one year. The tolerance is $\text{tol} = 10^{-5}$ for the outer loop and $\text{tol}' = 10^{-6}$ for the inner loops. The total number of production sectors is $N_X = 10$. Then the number of variables is of the order $T \times N_X \sim 10^3$.

```
   input : K^0
   output: K, P
 1 MPI_INIT();
 2 nsize ← MPI_COMM_SIZE();
 3 rank ← MPI_COMM_RANK();
 4 marker:  if diff > tol and it < numIt then
 5 |   if rank! = 0 then
 6 |   |   MPI_RECV for norm and normCounter from node rank;
 7 |   end
 8 |   for t ← (rank) to T by nsize do
 9 |   |   Calculate prices P(t) (inner loop);
10 |   |   Update dyn arrays;
11 |   end
12 |   Update stor arrays;
13 |   it ← (it + 1);
14 |   diff ← update (K, P);
15 |   if rank = 0 then
16 |   |   for rankId ← 1 to (nsize − 1) do
17 |   |   |   MPI_SEND norm and normCounter to node rankId;
18 |   |   end
19 |   end
20 |   MPI_AllReduce(MPI_SUM) for all dyn arrays;
21 end
22 goto marker
23 MPI_AllReduce(MPI_SUM) for all stor arrays;
24 MPI_FINALIZE();
```

Figure 6. MPI version.

With a more detailed multisector production and several regions the number of variables can be of order of hundred thousands.

For the OpenMP runs, we used Intel Visual Fortran Compiler XE 12.0. Calculations were carried out on Intel Core i5 (4 cores, 2.70 GHz). For comparison, solution of the equations system in the inner loop was obtained by Newton's method, NITSOL and NEQBF. For each method we made runs with optimization (-o3) and without it. The speedup graphs are presented at Fig. 7. As we see, parallelization with four cores gives a more than three times speedup.

Calculations with the MPI version were carried out using Fortran compiler IBM XL Fortran Advanced Edition. Calculations were carried out on Blue Gene/P V 11.1. Solution of the equations system in the inner loop was obtained by Newton's method and NITSOL. For each method we made runs with optimization (-o3) and without it. The speedup graphs are presented at Fig. 8. Here, parallelization with 200 nodes and one rank per node gives a speedup of up to 45 times.

The numerical results imply that the parallel version of the algorithm gives a substantial improvement both with OpenMP and MPI. The efficiency of the parallel method is especially pronounced with the MPI version, where the number of ranks can be as large as the number of time steps. The increase of the speedup is almost linear and does not depend much on the iterative method used in the inner loop and compiler optimization.

The results allow us to analyze the scalability. We see that the parallelization method shows good strong scaling, i.e. the speedup increases with the number of processors for the fixed problem size. This justifies the use of parallelization in the Fair-Taylor method.

Work is underway to determine the reasons why some of the methods used in the inner loop perform better than others. Future work could address the effectiveness of the MPI version with several ranks per node and hybrid realizations, such as OpenMP+MPI, with several processes per node (see, e.g., [13, 14]).

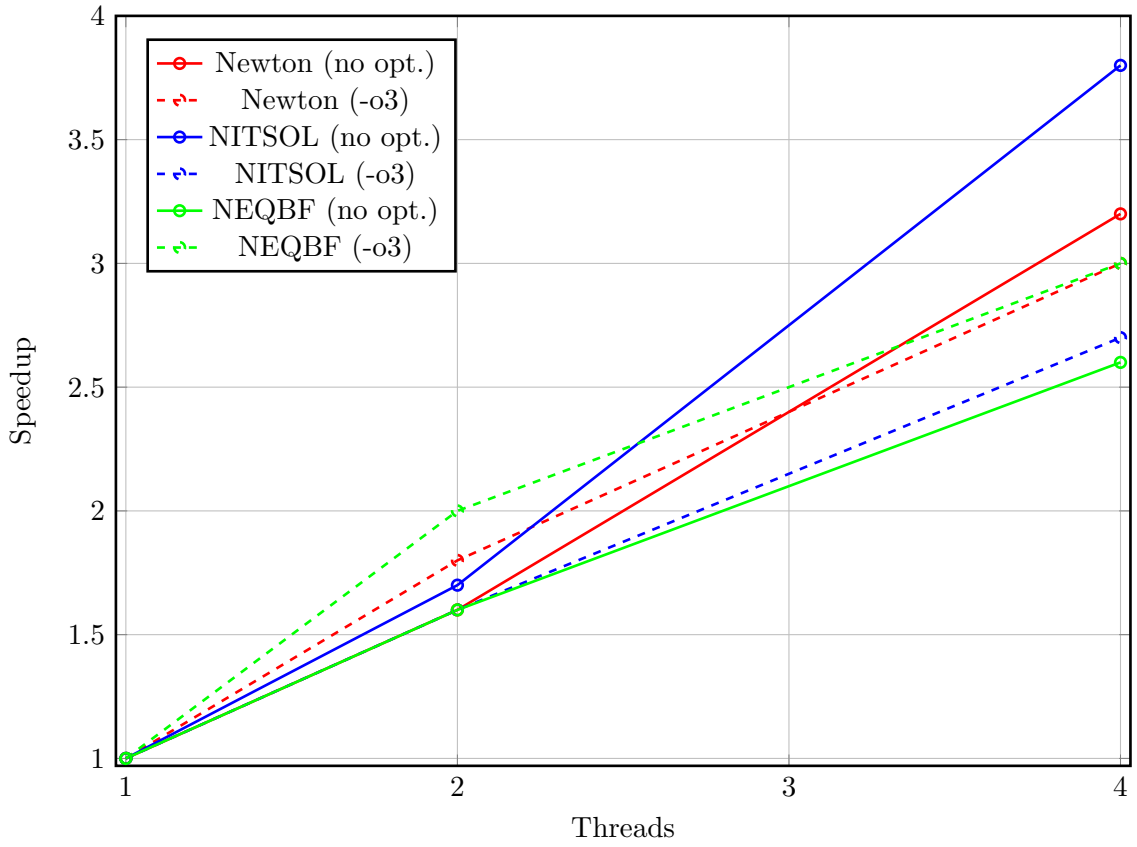

Figure 7. The speedup for the OpenMP parallel version of the algorithm.

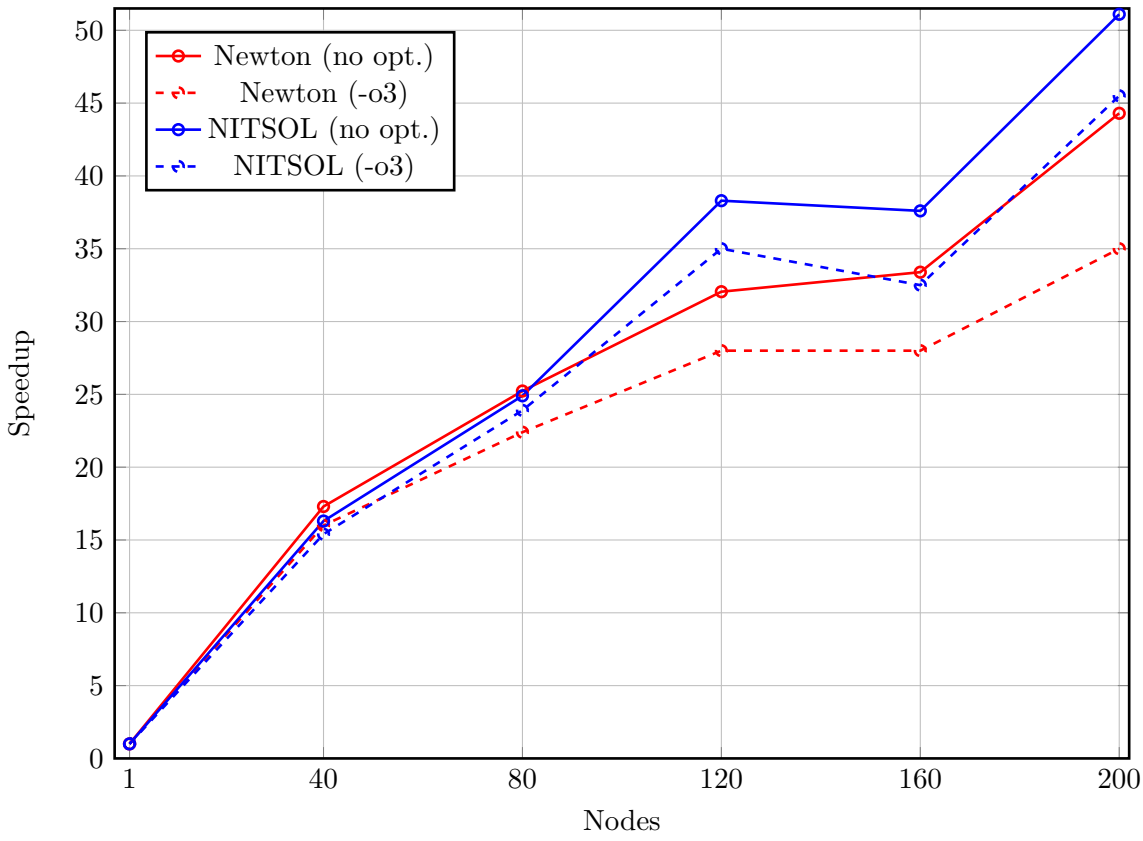

Figure 8. The speedup for the MPI parallel version of the algorithm.

References

1. Dalton M., O'Neill B., Prskawetz A., Jiang L., Pitkin J. Population aging and future carbon emissions in the United States // Energy economics. 2008. Vol. 30, N. 2. P. 642–675.

2. Melnikov N.B., O'Neill B.C., Dalton M.G. Accounting for the household heterogeneity in dynamic general equilibrium models // Energy economics. 2012. Vol. 34, N. 5. P. 1475–1483.

3. Fair R., Taylor J. Solution and maximum likelihood estimation of dynamic nonlinear rational expectations models // Econometrica. 1983. Vol. 51, N. 4. P. 1169–1185.

4. Brayton F. Two practical algorithms for solving rational expectations models // Finance and Economics Discussion Series. 2011–44, U.S. Federal Reserve Board.

5. Faust J., Tryon R. A distributed block approach to solving near-block-diagonal systems with an application to a large macroeconometric model // Computational Economics. 1995. Vol. 8, N. 4. P. 303–316.

6. Ortega J.M., Rheinboldt W.C. Iterative Solution of Nonlinear Equations in Several Variables. SIAM, 2000. 572 p.

7. Eisenstat S., Walker H. Globally convergent inexact Newton methods // SIAM Journal on Optimization. 1994. Vol. 4, N. 2. P. 393—422.

8. Pernice M., Walker H. Nitsol: A Newton Iterative solver for Nonlinear systems // SIAM Journal on Scientific Computing. 1998. Vol. 19, N. 1. P. 302–318.

9. Kelley C.T. Iterative Methods for Linear and Nonlinear Equations. SIAM, 1995. 166 p.

10. Conn A.R., Gould N.I.M., Toint P.L. Trust-region methods. SIAM, 2000. 959 p.

11. Stokey N.L., Lucas R.E., Prescott E.C. Recursive Methods in Economic Dynamics. Harvard University Press, 1989. 608 p.

12. Varian H.R. Microeconomic Analysis. Norton, 1992. 563 p.

13. Hasert M., Klimach H., Roller S. CAF versus MPI - Applicability of Coarray Fortran to a Flow Solver // Springer. 2011. Vol. 6960, P. 228-236.

14. Hoefler T., Dinan J., Buntinas D., Balaji P., Barrett B., Brightwell R., Gropp W., Kale V., Thakur R. MPI+MPI: A New Hybrid Approach to Parallel Programming with MPI Plus Shared Memory // Computing. 2013. Vol. 95, P. 1121-1136.