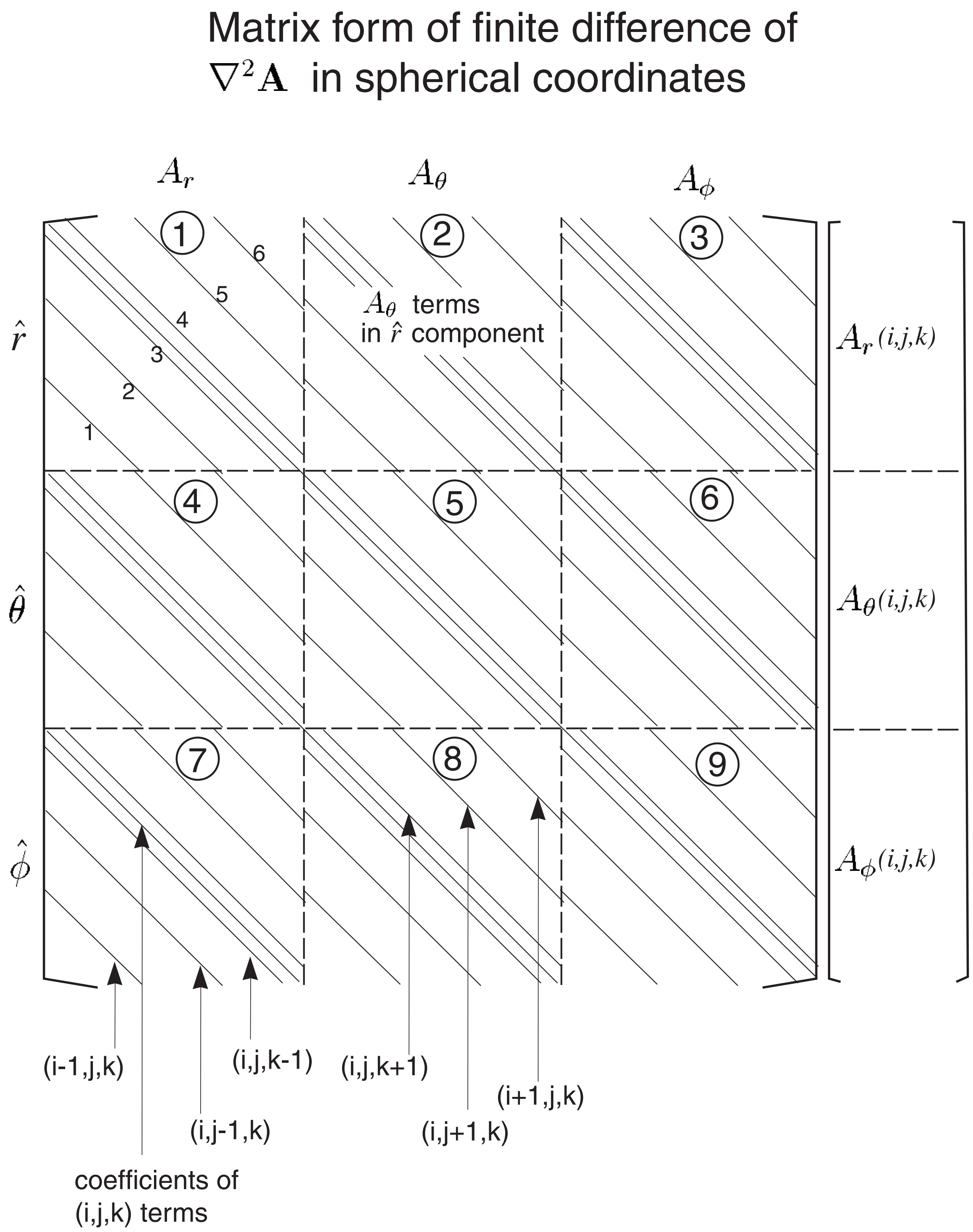

Figure 1: Finite difference of $\nabla^2\mathbf{A}$ in the spherical coordinates, expressed in a matrix form.

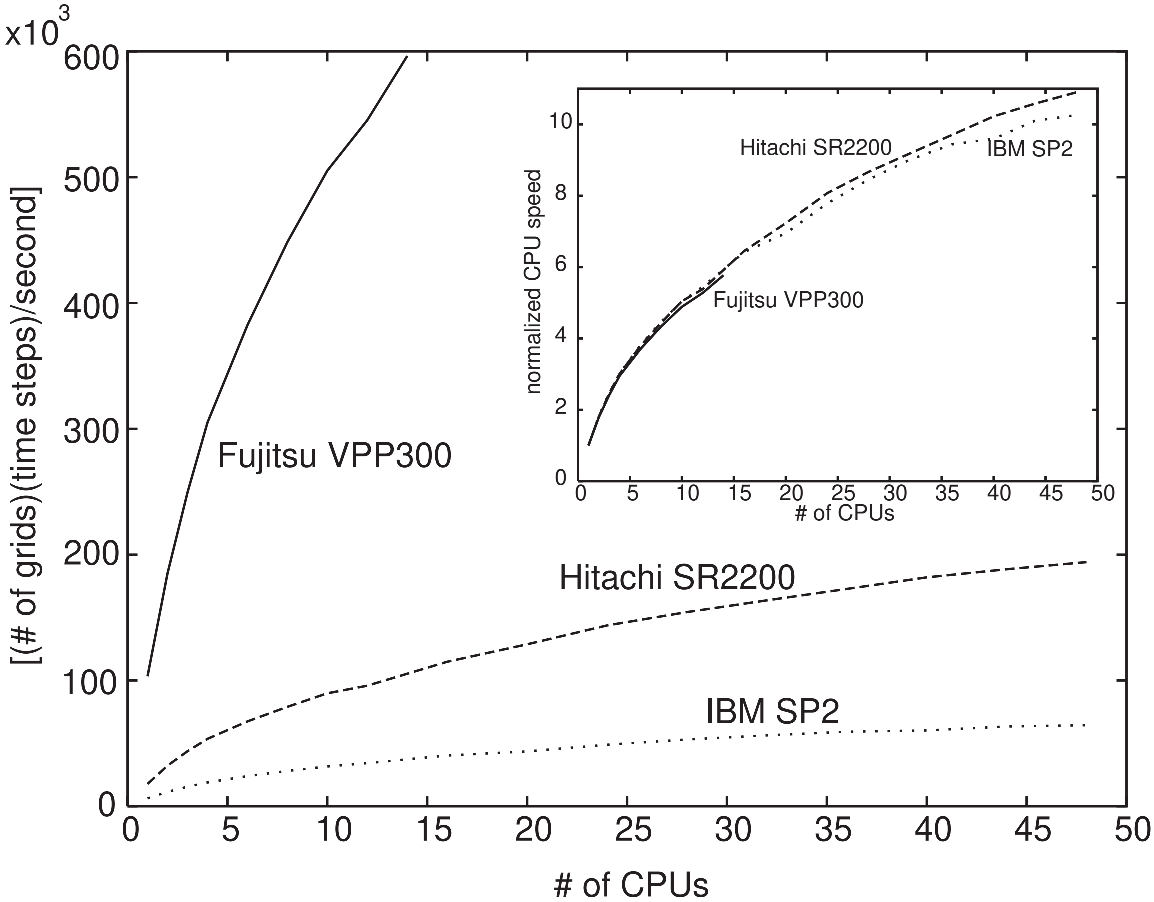

Figure 2: Parallel performance of our Cartesian code in Paper I, by fixing the number of grids in each CPUs ($128 \times 128 \times 20$). The inset shows the computational speed relative to a single CPU of each machine.

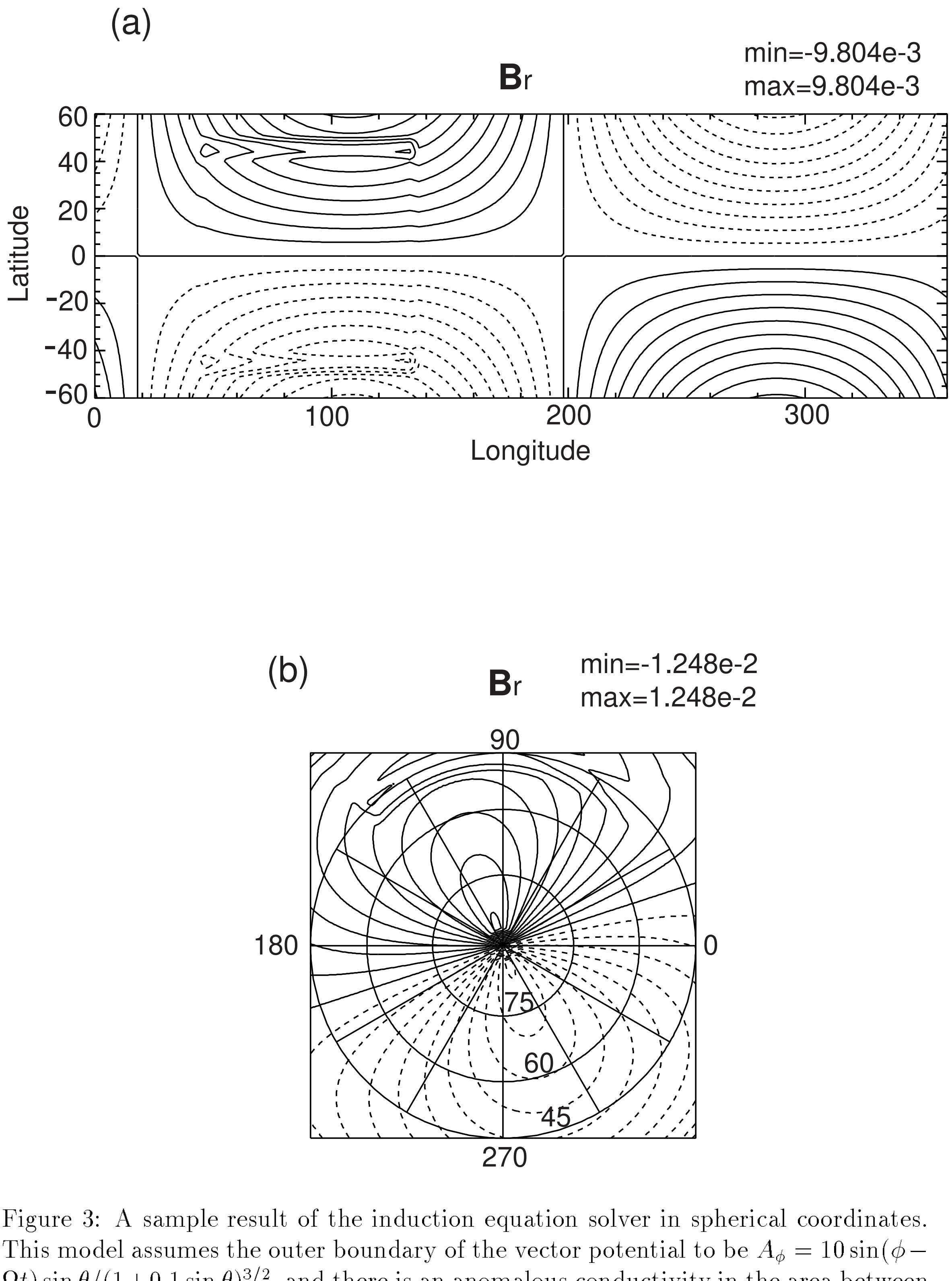

Figure 3: A sample result of the induction equation solver in spherical coordinates. This model assumes the outer boundary of the vector potential to be $A_\phi = 10\sin(\phi - \Omega t)\sin\theta/(1+0.1\sin\theta)^{3/2}$, and there is an anomalous conductivity in the area between 45° north and south and 45° to 135° in logitude. The Ω is assumed to be $2\pi/10000$s, and the result shown here is at t=500s. The figure shows the iso-contours of B_r. Solid curves correspond to positive values and dashed curves correspond to negative values. Panel (b) shows the polar view.

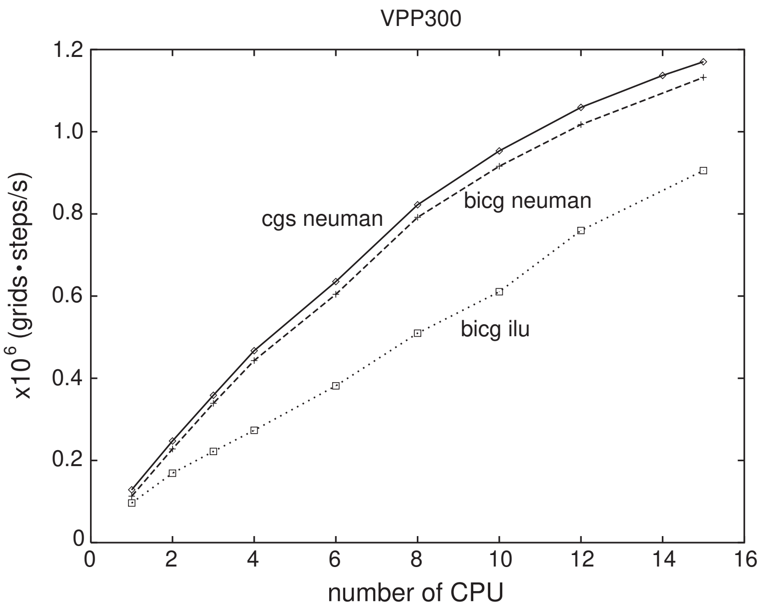

Figure 4: Performance of a Fujitsu VPP 300 computer to solve the induction equation in spherical coordinates. The ordinate corresponds to the computational speed. Three curves correspond to the test runs that adopt the CGS method with Neuman pre-conditioning, the BiCGstab method with Neuman pre-conditioning, and the BiCGstab method with ILU pre-conditioning, respectively.

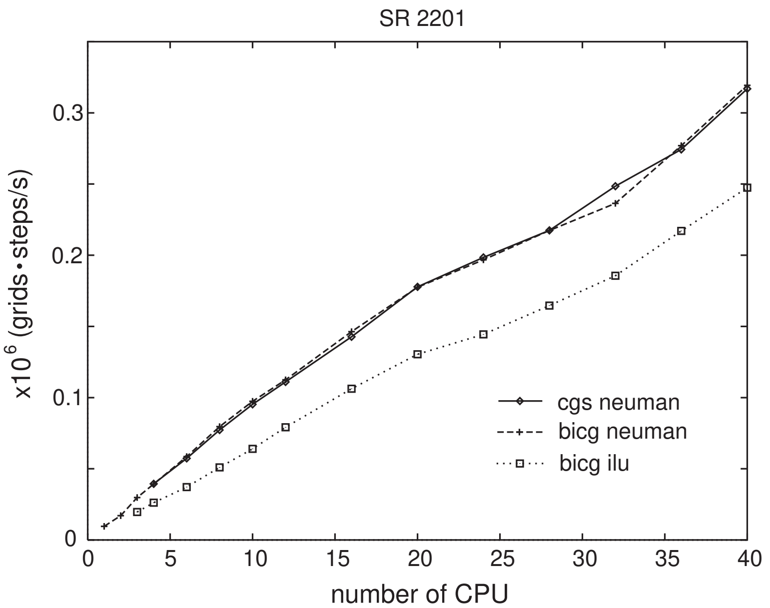

Figure 5: Performance of a Hitachi SR 2201 computer to solve the induction equation in spherical coordinates. The ordinate corresponds to the computational speed. Three curves correspond to the test runs that adopt the CGS method with Neuman pre-conditioning, the BiCGstab method with Neuman pre-conditioning, and the BiCGstab method with ILU pre-conditioning, respectively.

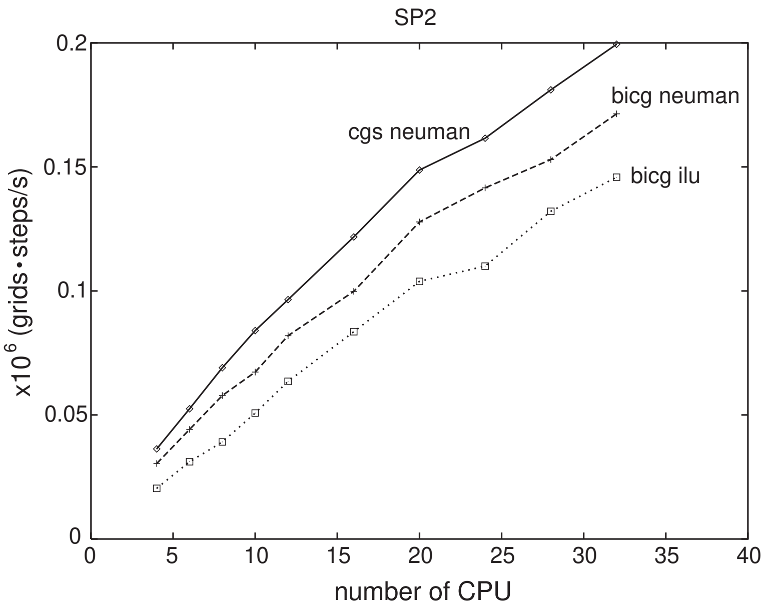

Figure 6: Performance of an IBM SP2 computer to solve the induction equation in spherical coordinates. The ordinate corresponds to the computational speed. Three curves correspond to the test runs that adopt the CGS method with Neuman pre-conditioning, the BiCGstab method with Neuman pre-conditioning, and the BiCGstab method with ILU pre-conditioning, respectively.

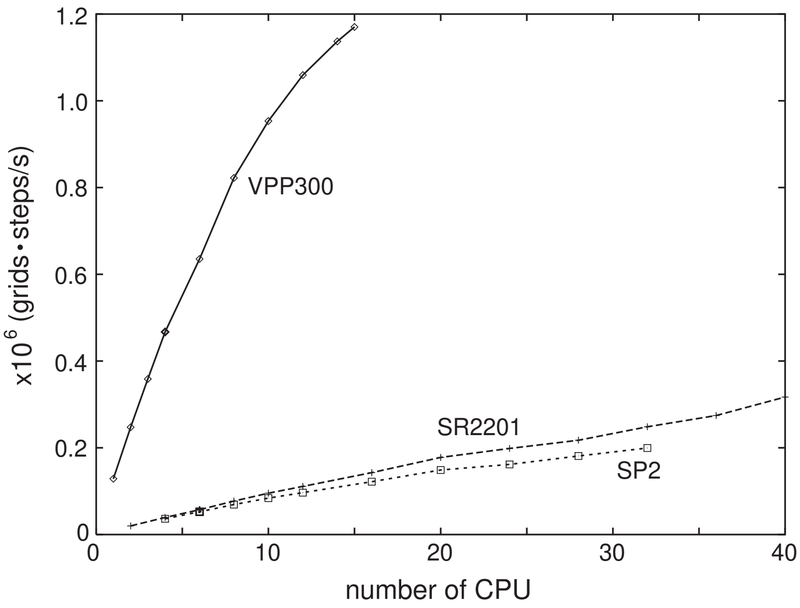

Figure 7: Relative performance of three computers (VPP300, SR2201, and SP2) when the same method (CGS with Neuman preconditioning) is used.

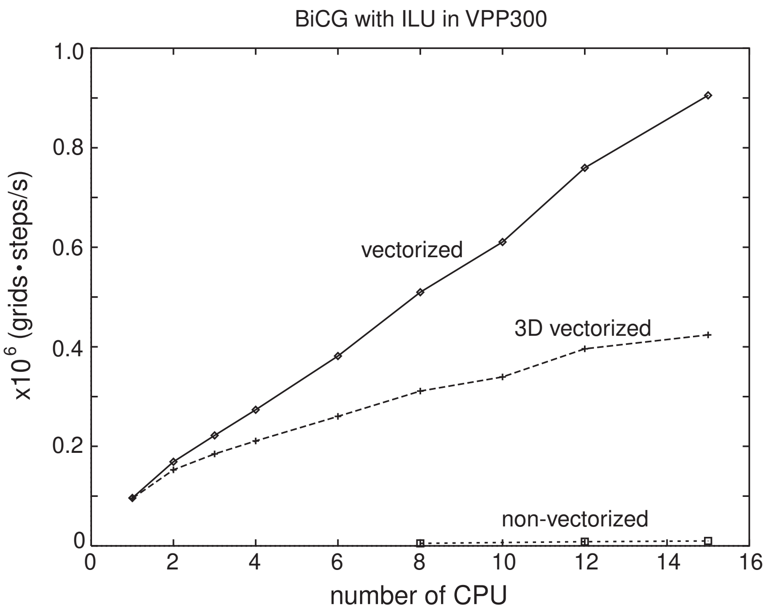

Figure 8: Comparison of the vectorization on a Fujitsu VPP 300 computer. Three test runs all adopt BiCGstab method with ILU preconditioning, but they are vectorized on I-J plane, vectorized on I-J-K plane, and not vectorized, respectively. Since the VPP 300 machine is a vector computer, the computational speed becomes extremely slow when the code is not vectorized.

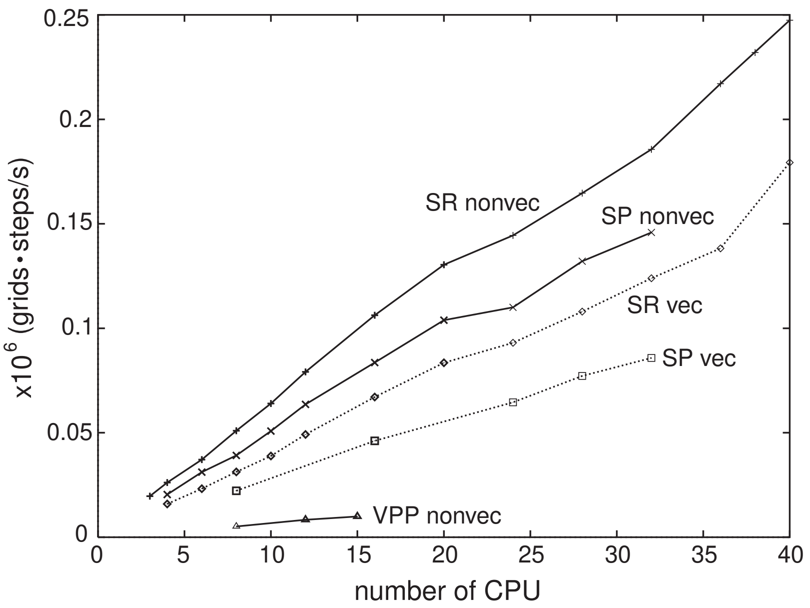

Figure 9: Comparison of the vectorization on scalar computers (Hitachi SR2201 and IBM SP2). Solid lines correspond to the non-vectorized code, and dotted lines correspond to the vectorized code. The performance of the non-vectorized code on a VPP300 computer is also shown here for comparison.

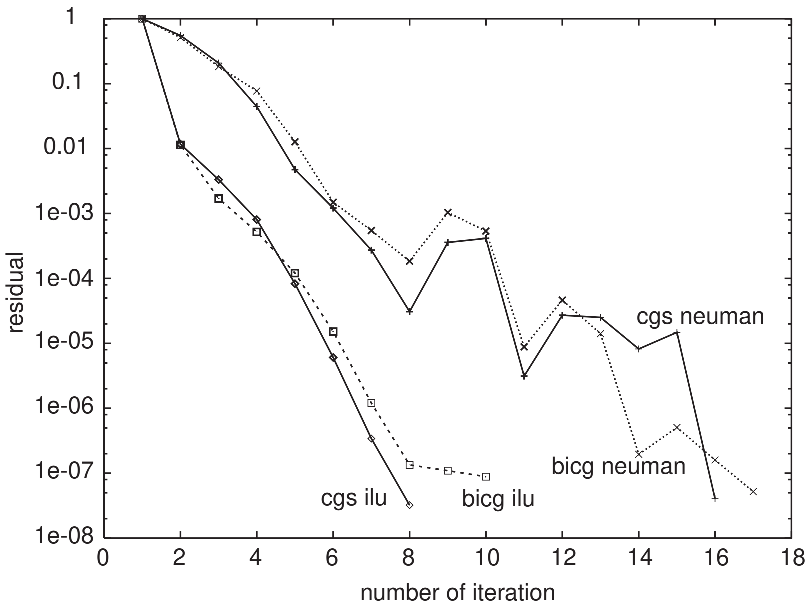

Figure 10: Iteration history for the BiCGstab (dashed and dotted lines) and CGS (solid lines) methods, adopting either the ILU or Neuman preconditioner.

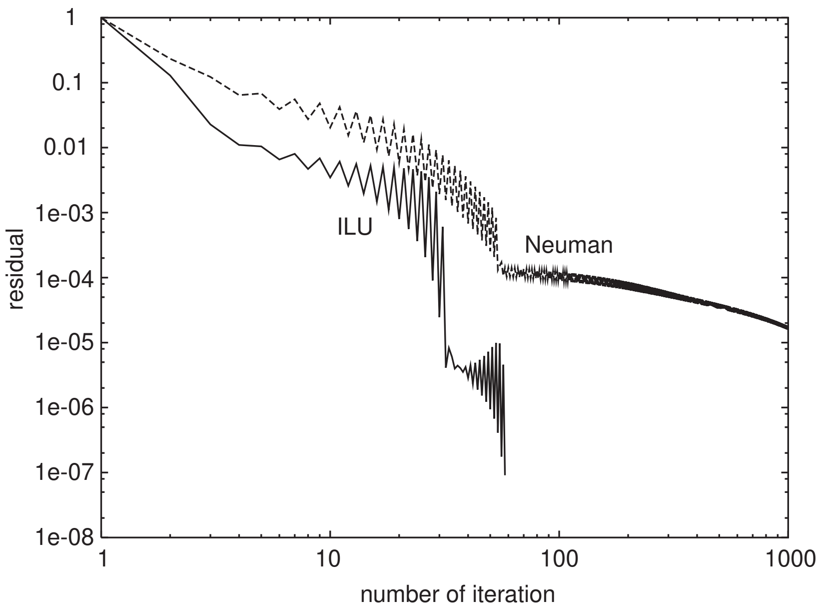

Figure 11: Iteration history for the ILU and Neuman preconditioners when a different boundary condition [Eq.(55)] is adopted.