PHYSICAL REVIEW B **85**, 235103 (2012)

Massively parallel density functional calculations for thousands of atoms: KKRnano

A. Thiess,[1,2,*] R. Zeller,[1] M. Bolten,[3] P. H. Dederichs,[4] and S. Blügel[1,4]
[1]*Institute for Advanced Simulation, Forschungszentrum Jülich, D-52425 Jülich, Germany*
[2]*German Research School for Simulation Sciences, D-52425 Jülich, Germany*
[3]*Department of Mathematics and Science, University of Wuppertal, D-42097 Wuppertal, Germany*
[4]*Peter Grünberg Institut, Forschungszentrum Jülich and JARA, D-52425 Jülich, Germany*
(Received 25 January 2012; published 4 June 2012)

Applications of existing precise electronic-structure methods based on density functional theory are typically limited to the treatment of about 1000 inequivalent atoms, which leaves unresolved many open questions in material science, e.g., on complex defects, interfaces, dislocations, and nanostructures. KKRnano is a new massively parallel linear scaling all-electron density functional algorithm in the framework of the Korringa-Kohn-Rostoker (KKR) Green's-function method. We conceptualized, developed, and optimized KKRnano for large-scale applications of many thousands of atoms without compromising on the precision of a full-potential all-electron method, i.e., it is a method without any shape approximation of the charge density or potential. A key element of the new method is the iterative solution of the sparse linear Dyson equation, which we parallelized atom by atom, across energy points in the complex plane and for each spin degree of freedom using the message passing interface standard, followed by a lower-level OpenMP parallelization. This hybrid four-level parallelization allows for an efficient use of up to 100 000 processors on the latest generation of supercomputers. The iterative solution of the Dyson equation is significantly accelerated, employing preconditioning techniques making use of coarse-graining principles expressed in a block-circulant preconditioner. In this paper, we will describe the important elements of this new algorithm, focusing on the parallelization and preconditioning and showing scaling results for NiPd alloys up to 8192 atoms and 65 536 processors. At the end, we present an order-N algorithm for large-scale simulations of metallic systems, making use of the nearsighted principle of the KKR Green's-function approach by introducing a truncation of the electron scattering to a local cluster of atoms, the size of which is determined by the requested accuracy. By exploiting this algorithm, we show linear scaling calculations of more than 16 000 NiPd atoms.

DOI: 10.1103/PhysRevB.85.235103 PACS number(s): 71.15.Dx, 71.23.−k, 73.22.−f

I. INTRODUCTION

The microscopic understanding of multicomponent, patterned, or nanostructured solids with internal interfaces, structural and compositional disorder, point and extended defects is a central issue in material science, nanoelectronics, surface science, and chemistry. This understanding is of particular importance for the description of samples and ultimately of devices, the properties of which are determined by spatial extents on the nanoscale. In this case, the local properties depend on the shape, size, local chemical composition of small clusters, precipitates, formed filaments, or even single defects that interfere or even determine immediately the device functionality. Conversely, defects also provide a powerful toolbox to actively design system properties. From the theoretical point of view, the microscopic properties are determined by the electronic structure that is treated successfully *ab initio* by the density functional theory (DFT) using appropriate approximations to the unknown exchange and correlation functional. The treatment of nonideal structures such as interfaces, dislocations, or, e.g., amorphous systems or compositionally disordered systems, is, however, significantly more difficult since the symmetry is typically distinctively reduced as compared to the ideal periodic crystals. In addition, computational schemes have to account for the vastly enhanced chemical and structural complexity in the samples.

In order to face those challenges, two approaches are most commonly used for the *ab initio* description of nonideal systems: One approach is known as the coherent-potential approximation[1] (CPA) in which the scattering of the electrons in a random alloy is replaced by the scattering of an effective potential that has the same scattering properties as the alloy in average. Extensions have been developed such as the nonlocal CPA (Ref. 2) to include short-range compositional corrrelations. The incorporation of longer-ranged spatial correlation effects, finite-size effects, as well as more complex defects on a sufficiently accurate level is not straightforward. Aside from that, during the last decades, supercell techniques have been most commonly and successfully used in treating nonideal structures.[3,4] Here, *a priori* the geometrical and chemical freedom is not limited but, on the downside, the number of atoms in the supercell has to be often increased up to thousands or tens of thousands of atoms to both obtain statistically relevant results and minimize spurious interactions with periodic images. The *ab initio* description of such large-scaled systems presents a highly demanding computational task and is often not addressable on conventional computers.

The ongoing development of supercomputers, which nowadays combine the power of hundreds of thousands of processors, moves the computation of more than thousands of atoms per supercell into the bounds of possibility. The challenge is to invent efficient algorithms, i.e., algorithms which minimize communication between processors, and, which come up for the low memory resources typical for supercomputers. One approach which guarantees a high scalability is based on Kohn's nearsightedness principle, in other words, neglecting *a priori* long-range interactions. This principle is the fundamental assumption for the first-principles linear scaling methods such as CONQUEST,[5] ONESTEP,[6] SIESTA,[7] OPENMX,[8] LSMS,[9] or LGSF.[10] While those methods can be applied to a broad class of materials, the prerequisite of the presence of exclusively short-ranged electronic interactions can limit the applicability to, e.g., metallic systems where long-range interactions are crucial.

In this paper, we will present the newly developed first-principles Green's-function method KKRnano, which can be used to treat systems with both short- and long-range interactions and which is especially designed for the treatment of many thousands of atoms on massively parallel supercomputers with up to 100 000 processors. We will show, as proposed in Refs. 11–13, how several important techniques are combined to achieve an efficient quadratic or even linear scaling and thus a massively parallel algorithm within the all-electron full-potential Korringa-Kohn-Rostoker (KKR) method. Further, we will focus on the optimization of the computationally most demanding part, the iterative solution of the Dyson equation, by initial guess and preconditioning techniques without compromising on the accuracy.

II. KORRINGA-KOHN-ROSTOKER GREEN'S-FUNCTION METHOD

In a Green's-function method, the energy-resolved electron density can be obtained directly from the Green's function by

$$n(\mathbf{r},E) = -\frac{1}{\pi}\mathrm{Im}\, G(\mathbf{r},\mathbf{r},E). \tag{1}$$

The Green's function $G(\mathbf{r},\mathbf{r}',E)$ is the solution of the Schrödinger equation with a δ function as source term, which reads in atomic units as

$$\left[-\nabla_r^2 + V(\mathbf{r}) - E\right]G(\mathbf{r},\mathbf{r}';E) = -\delta(\mathbf{r}-\mathbf{r}'). \tag{2}$$

Equivalently, G can be given by an integral equation, which is computationally easier to solve than (2) and is from now on referred to as Dyson equation

$$G(\mathbf{r},\mathbf{r}';E) = G^r(\mathbf{r},\mathbf{r}';E) + \int d\mathbf{r}'' G^r(\mathbf{r},\mathbf{r}'';E)\Delta V(\mathbf{r}'')G(\mathbf{r}'',\mathbf{r}';E), \tag{3}$$

where $G^{\mathrm{r}}(\mathbf{r},\mathbf{r}';E)$ is the Green's function of an arbitrary reference system and $\Delta V(\mathbf{r}) = V(\mathbf{r}) - V^{\mathrm{r}}(\mathbf{r})$ represents the difference of the potential between the real and the reference systems. In the KKR method, the spatial integration in Eq. (3) is performed over space-filling Voronoi cells, and the scattering events are described by an angular momentum expansion in l and m around the site-centered coordinates $\mathbf{R}^n$. From now on, the index L will be used to abbreviate l and $|m| \leqslant l$ with $L_{\max} = (l_{\max}+1)^2$ possible combinations, where $l_{\max}$ denotes the maximal angular momentum contribution considered.

Within multiple-scattering theory, a clear separation between single-site and multiple-scattering quantities appears:

$$G(\mathbf{R}^n+\mathbf{r},\mathbf{R}^{n'}+\mathbf{r}';E) = \delta^{nn'}G_{\mathrm{s}}^n(\mathbf{r},\mathbf{r}';E) + \sum_{LL'} R_L^n(\mathbf{r};E)G_{LL'}^{nn'}(E)R_{L'}^{n'}(\mathbf{r}';E), \tag{4}$$

where G_{s}^n and R_L^n are the single-site Green's function and wave functions, which can be obtained locally on each site n with respect to a given local potential and thereby can be parallelized trivially in real space over sites. The expression of both single-site quantities and further details on the full-potential algorithm are given in Refs. 11 and 14. The remaining task is to determine the multiple-scattering Green's-function matrix elements $G_{LL'}^{nn'}(E)$, which obey the algebraic Dyson equation

$$G_{LL'}^{nn'}(E) = G_{LL'}^{\mathrm{r},nn'}(E) + \sum_{n'',L''L'''} G_{LL''}^{\mathrm{r},nn''}(E)\,\Delta t_{L''L'''}^{n''}(E)\,G_{L'''L'}^{n''n'}(E), \tag{5}$$

where $G_{LL'}^{\mathrm{r},nn'}(E)$ are the structure constants of the reference system and $\Delta t_{LL'}^n$ is the difference of the t matrices between the real and the reference systems:

$$\Delta t_{LL'}^n(E) = \int_n d\mathbf{r}\; j_l(r\sqrt{E})\; Y_L(\mathbf{r})V^n(\mathbf{r})R_{L'}^n(\mathbf{r};E) - \int_n d\mathbf{r}\; j_l(r\sqrt{E})\; Y_L(\mathbf{r})V^{\mathrm{r},n}(\mathbf{r})R_{L'}^{\mathrm{r},n}(\mathbf{r};E). \tag{6}$$

Here, $R_L^{\mathrm{r},n}$ and $V^{\mathrm{r},n}$ are the wave function and the potential of the reference system. Y_L denote spherical harmonics and j_l spherical Bessel functions. The integration in Eq. (6) is restricted for both integrals to the local site, which directly leads to an efficient parallelization of the computational work over sites. The computationally most demanding part remains the solution of the Dyson equation (5), on which this paper focuses.

It is important to note that the energy integration over the spectral density (1) can be considerably accelerated by taking advantage of the analytic properties of the Green's function in the complex plane and by using an electronic temperature T introduced by the Fermi-Dirac distribution $f_T(E)$:

$$n(\mathbf{r}) = -\frac{1}{\pi}\;\mathrm{Im}\int_{E_{\mathrm{B}}}^{\infty} dE\; f_T(E)\; G(\mathbf{r},\mathbf{r},E), \tag{7}$$

as shown by Wildberger *et al.*.[15] Here, E_B is chosen to lie between core states ($E < E_{\mathrm{B}}$), which are obtained locally in an atomiclike approach with wave functions satisfying atomic boundary conditions, and valence states ($E > E_{\mathrm{B}}$). The Fermi function $f_T(E) = (1+\exp(\frac{E-E_{\mathrm{F}}}{k_{\mathrm{B}}T}))^{-1}$ has poles at the Matsubara energies $E_j = E_{\mathrm{F}} + (2j+1)i\pi k_{\mathrm{B}}T$, whereas the Green's function is analytic away from the real axis and consequently rather smooth in the complex plane. This allows for a considerable reduction of integration points. Instead of using many hundreds of energy integration points along the real axis, by applying the artificial energy broadening to the complex contour integration, high accuracy can be reached considering only 20 to 40 energy points. Furthermore, the calculation of the Green's function at complex energy points on the integration contour and at the Matsubara poles is crucial since the iterative schemes to solve the Dyson equation work increasingly faster with larger distances from the real axis.[11]

III. TB-KKR GREEN'S-FUNCTION METHOD

Tight-binding (TB) schemes within *ab initio* electronic-structure methods have been very successfully realized in the TB linear muffin-tin orbital method[16] and, e.g., for interfaces and surfaces within the principal layer technique.[17] In this paper, we make use of the related tight-binding or screened KKR methods as developed by Zeller *et al.*.[18] Within this approach, an advantageous two-step procedure can be introduced instead of solving the Dyson equation (5) directly: First, the free-space Green's function g^0 is related to the one of a cluster of hard repulsive spheres, which are placed in the same geometry as the lattice sites of the actual system:

$$G^{\mathrm{r},nn'}_{LL'}(E) = g^{0,nn'}_{LL'}(E) + \sum_{n'',L''L'''} g^{0,nn''}_{LL''}(E)\, t^{\mathrm{r},n''}_{L''L'''}(E)\, G^{\mathrm{r},n''n'}_{L'''L'}(E). \quad (8)$$

Instead of the slow spatial decay of the free-space scattering matrix g^0 with respect to the distance between R^n and $R^{n'}$, the reference cluster scattering matrix G^{r} decays exponentially fast with increasing distance within the energy interval of the valence states.[18] This allows for the introduction of a radial cutoff without loss of accuracy, resulting in a finite number N_{cl} of interacting atoms within this cluster. The computation of (8) can be performed locally for each site and its individual cluster of atoms, and it is therefore ideally suited to be distributed to independent processes. The second step is the connection of this reference system to the actual system via

$$G^{nn'}_{LL'}(E,\mathbf{k}) = G^{\mathrm{r},nn'}_{LL'}(E,\mathbf{k}) + \sum_{n'',L''L'''} G^{\mathrm{r},nn''}_{LL''}(E,\mathbf{k})\, \Delta t^{n''}_{L''L'''}(E)\, G^{n''n'}_{L'''L'}(E,\mathbf{k}), \quad (9)$$

where we introduced here the $\mathbf{k}$ dependency induced by the periodic boundary conditions of the supercell. Consequently, in Eq. (9). the indices n and n' are now restricted to sites within the supercell of a total number of sites N. Since the introduction of empty cells is a frequently used procedure in KKR schemes, the number of sites of the supercell N is allowed to be greater than the number of atoms of the supercell N_{at}, i.e., $N \geqslant N_{\mathrm{at}}$. Accordingly, we are using N for the number of sites throughout this paper. Regarding (9), it is important to note that the matrix G^{r} is now sparse in the space of sites with only $\propto N_{\mathrm{cl}}N$ nonzero entries. This sparsity enables us to store this matrix for large systems on supercomputers and reduces the amount of floating-point operations, which are two crucial steps towards an efficient large-scale algorithm.

IV. ITERATIVE SOLUTION OF THE DYSON EQUATION

In order to identify and discuss the computational bottleneck arising from the solution of the Dyson equation (9), we reformulate the problem in the following. For this purpose, we drop the angular momentum and atomic-site indices as well as energy and $\mathbf{k}$-point dependencies from here on.

Starting from the Dyson equation (9),

$$G = G^{\mathrm{r}} + G^{\mathrm{r}}\Delta t\, G, \quad (10)$$

it is clear that G^{r} fulfills

$$(I - G^{\mathrm{r}}\Delta t)\, G = G^{\mathrm{r}}. \quad (11)$$

Instead of directly solving (11) for G, we avoid the matrix-matrix multiplication of G^{r} and the inverted matrix $(1 - G^{\mathrm{r}}\Delta t)^{-1}$ and exploit in the following that both Δt and $(\Delta t)^{-1}$ are block diagonal. With the identity

$$G^{\mathrm{r}} = -(I - G^{\mathrm{r}}\Delta t)\, \Delta t^{-1} + \Delta t^{-1}, \quad (12)$$

it follows after multiplication with $(1 - G^{\mathrm{r}}\Delta t)^{-1}$

$$G = -(\Delta t)^{-1} + (\Delta t)^{-1}[(\Delta t)^{-1} - G^{\mathrm{r}}]^{-1}(\Delta t)^{-1}. \quad (13)$$

Since the Δt matrices are block diagonal, all matrix-matrix multiplications in Eq. (13) are computationally inexpensive. The CPU time-consuming part of (13) to compute the inverse of

$$M = (\Delta t)^{-1} - G^{\mathrm{r}} \quad (14)$$

is often referred to as KKR matrix or scattering path operator, the direct inversion of which requires $O(N^3)$ floating-point operations. Although optimized sparse solvers can reduce the computational effort considerably, they require extensive communication if parallelized. The iterative inversion can be an efficient, alternative scheme for sparse and parallel calculations as it takes advantage of the sparsity and can be parallelized without any demands for communication during the inversion step. In order to conduct the iterative inversion, M and its inverse

$$X = M^{-1} = [(\Delta t)^{-1} - G^{\mathrm{r}}]^{-1} \quad (15)$$

can be set into relation as

$$\Delta t\ M\ X = \Delta t. \quad (16)$$

Inserting the actual expression (14) of M leads to

$$X = \Delta t + \Delta t\ G^{\mathrm{r}}X, \quad (17)$$

which can be solved using the scheme

$$X^{(\nu+1)} = \Delta t + \Delta t\ G^{\mathrm{r}}X^{(\nu)} \quad (18)$$

in an iterative cycle $X^{(0)} \to X^{(1)}$, $X^{(\nu)} \to X^{(\nu+1)}$, where ν is the iteration index and X is a quadratic matrix of size $N(l_{\max}+1)^2 \times N(l_{\max}+1)^2$. However, the convergency properties of such simple schemes are usually rather poor, which is motivating the choice of more sophisticated methods as, e.g., the generalized minimal residual[19] (GMRES) or the quasi-minimal residual (QMR) method.[20] Both algorithms are formulated to solve a set of linear equations

$$A\ X = B. \quad (19)$$

In this context, Eq. (18) can be reformulated as

$$(\Delta t\ G^{\mathrm{r}} - I)X = -\Delta t, \quad (20)$$

the solution X of which we obtain in this representation by an iterative transpose free QMR (TFQMR) solver.[21]

At this point, the computation time of the algorithm scales $\propto N^2 N_{\mathrm{cl}} N_{\mathrm{it}}$, where N_{it} stands for the number of TFQMR iterations required to converge X down to the predefined accuracy. An important property of the linear matrix equation (20) is that it decouples in $N(l_{\max}+1)^2$ problems, i.e., $N(l_{\max}+1)^2$ vectors each of the size $N(l_{\max}+1)^2$ can be iterated independently and together build the full solvent matrix X. In the following, X will stand for the solvent of one of those subproblems and is then accordingly referred to as vector. Beyond that, it is partly of conceptual advantage to group the $(l_{\max}+1)^2$ vectors X representing the same atom index into one matrix. For the discussion of scaling behavior, it is important to note that we find, in agreement with Refs. 11 and 22, that N_{it} depends only weakly on the system size for $N > 1000$ and it is assumed to be constant from now on. However, depending on the temperature T, for the energy point closest to the real axis up to $N_{\text{it}} \approx 1000$ iterations might be needed. For efficient computing, it is crucial to come up with schemes which reduce this factor considerably, such as finding a good starting vector as initial guess and/or preconditioning the matrix $A = (\Delta t\ G^{\text{r}} - I)$ in Eqs. (19) and (20), respectively. In the following, we will describe the implementation of both techniques and its performance acceleration focusing on one of the $N(l_{\text{max}} + 1)^2$ equivalent problems solving (20).

In order to validate and show the increase in computational efficiency of the new concepts, which will be presented throughout this paper, KKRnano is applied to three different test systems with different physical properties. First, $\text{Ge}_1\text{Sb}_2\text{Te}_4$ is a phase-change material and crystallizes in the rocksalt lattice structure. The chemical composition is defined by two fcc sublattices, where one is fully occupied by Te and the other is disordered and is hosting 25% Ge, 50% Sb, and 25% vacancies. As a second system, we consider a semiconductor, pure Si in a diamond structure. In order to probe our method for purely metallic and spin-polarized samples, we append this set of sample systems with a quasirandom alloy of Ni and Pd crystallizing in the fcc structure. While the NiPd alloy is treated with a $l_{\text{max}} = 3$, the Si as well as $\text{Ge}_1\text{Sb}_2\text{Te}_4$ structures are described by an angular momentum cutoff of $l_{\text{max}} = 2$.

A. Initial guess

The default choice of the starting vector $X^{(0)}$ in the TFQMR package[21] is $0 + 0i$ for all entries of $X^{(0)}$. Obviously, this starting point is usually far from the required solution, which motivates us to introduce a (in general, arbitrary) starting vector $X^{(0)}$:

$$X = X^{(0)} + \tilde{X}, \tag{21}$$

where $\tilde{X}$ is the difference between the solution X and the starting vector $X^{(0)}$. Subtracting $X^{(0)}$ on both sides of (18) suggests the definition of

$$\Delta t' = \Delta t - X^{(0)} + \Delta t\ G^{\text{r}} X^{(0)}, \tag{22}$$

which leads to the effective iterative expression

$$\tilde{X}^{(\nu+1)} = \Delta t' + \Delta t\ G^{\text{r}} \tilde{X}^{(\nu)}, \tag{23}$$

with $\tilde{X}^{(\nu)} = X^{(\nu)} - X^{(0)}$. The matrix $\Delta t'$ directly determines the quality of the starting solution $X^{(0)}$. Comparing (22) with (18) shows that $\Delta t'$ expresses the perturbation remaining from the difference between the initial guess and the required solution. In fact, the calculation of the norm $\|\Delta t'\|$ directly corresponds to the check for quality by calculating the residual norm of the present iteration in the usual TFQMR procedure. For small $\|\Delta t'\|$, i.e., a reasonably good initial guess, the solution is expected to be computed in fewer iterations, as will be shown in the following.

Such a starting vector $X^{(0)}$ can be obtained by different physically motivated approaches:

(i) The solution of an ersatz geometry as a small locally defined cluster or a coherent potential or virtual-crystal approximation with an optionally smaller angular momentum cutoff.

(ii) The extrapolation of the solution at the previously calculated energy points $(E-1, E-2, \ldots)$ of the same self-consistency step (s), $X^{(0)}_{(s)}(E) = f[X_{(s)}(E-1), X_{(s)}(E-2), \ldots]$.

(iii) The result of the previous self-consistency step $(s-1)$, but same energy point E: $X^{(0)}_{(s)}(E) = X_{(s-1)}(E)$.

While the solution of an ersatz geometry can give an excellent initial guess for a class of materials as, e.g., the solution in a local cluster for semiconducting samples, the quality of the initial guess can be significantly worse in metallic systems. Moreover, the calculation of a precise initial guess can become computationally expensive and slows down the algorithm considerably. The extrapolation from previous energy points is computationally cheap. However, we find that the accuracy of such an extrapolation is rather limited. It is more advantageous to use the result of the previous DFT self-consistency cycle. This gives an initial guess, which exhibits usually already at the first self-consistency steps s, a quality of $(\|X_{(s)}\| - \|X_{(s-1)}\|)/\|X_{(s)}\| \approx 10^{-2}$. Keeping in mind the architectures of modern supercomputers, for this approach the memory limitations can be a bottleneck as for each atomic site the information on $X_{(s-1)}$ with size $N(l_{\text{max}} + 1)^2 \times (l_{\text{max}} + 1)^2$ has to be stored for each energy and $\mathbf{k}$ point. Therefore, in KKRnano contributions to represent $X^{(0)}$ that are larger than a specified cutoff are accordingly stored in a highly sparse representation. By applying the initial guess, the overall performed TFQMR iterations (Fig. 1) exhibit for all cases a significant reduction in the number of iterations by a factor γ

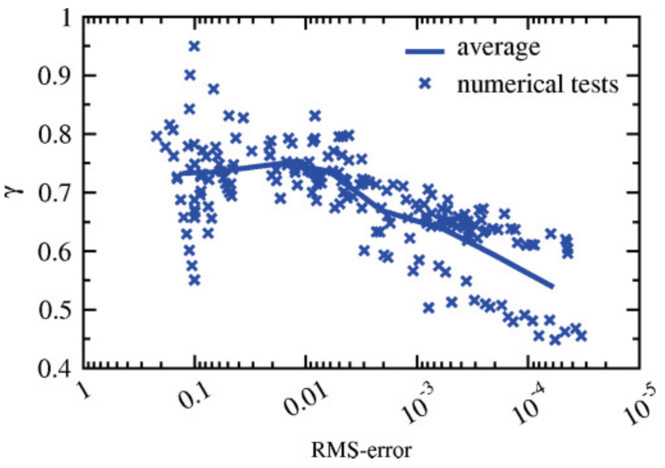

FIG. 1. (Color online) Reduction of the number of required iterations by application of the third initial guess strategy as described above for the system $\text{Ge}_1\text{Sb}_2\text{Te}_4$. The reduction ratio γ is defined as the ratio of the sum of all iterations needed for all sites, $\mathbf{k}$ and energy points, as well as lm components with and without application of the initial guess. Here, we considered system sizes from $N = 64$ to 512 and display γ as a function of the degree of convergency of the self-consistency steps, which is represented by the rms error, the variance of actual (s), and previous $(s-1)$ potential $\propto \| V_{(s)}(r) - V_{(s-1)}(r)\|$, from which $X^{(0)}_{(s)} = X_{(s-1)}$ is taken. The average reduction is plotted as a straight line.

of 0.45 to 0.95. As intuitively expected, the increase in quality of the initial guess for almost converged self-consistency iterations leads to a reliable incremental reduction of required TFQMR iterations.

B. Block-circulant preconditioning

In addition to the initial guess, the number of TFQMR iterations can be considerably reduced by application of preconditioning schemes. Here, the main challenge is finding a good approximation P to the matrix A in Eq. (19) that can be inverted easily. With the help of such an approximate matrix

$$P = P_1\ P_2, \tag{24}$$

a modified linear matrix equation

$$A'\ Y = B' \tag{25}$$

is solved, where $A' = P_1^{-1} A P_2^{-1}$, $B' = P_1^{-1}(B - AX^{(0)})$, and $Y = P_2(X - X^{(0)})$. $X^{(0)}$ denotes an optional initial guess to the solution X of $AX = B$. The residual vector $r'^{(\nu)}$ for the preconditioned system of TFQMR iteration ν then reads as $r'^{(\nu)} = B' - A'Y^{(\nu)}$. When a sufficiently small residual vector $r'^{(\nu)}$ is obtained, the solution of the original system can be calculated as

$$X^{(\nu)} = X^{(0)} + P_2^{-1} Y^{(\nu)}. \tag{26}$$

The residual of the original system translates as

$$r^{(\nu)} = P_1 r'^{(\nu)}. \tag{27}$$

For our purposes, we limit the preconditioning to right preconditioning by setting $P_1 = I$, which then leaves the required size of the minimal residual unchanged to the one of the original system. If the initial guess $X^{(0)}$ is zero, the additional computation consists of two steps:

(i) To calculate $A'Y = AP_2^{-1}Y$, the preconditioning matrix P_2^{-1} is applied to Y before every matrix multiplication with A. This step must be performed in every TFQMR iteration.

(ii) The solution of the original system X is obtained from Y by Eq. (26).

The remaining and most challenging task is to find an easily invertible matrix P_2 approximating A. One approach is to obtain P_2^{-1} by applying a sparse complete (LU) or incomplete (ILU) factorization of the matrix A, i.e. a decomposition as product of lower and upper triangular matrices, which would be functional also for cells with large relaxations or amorphous systems. However, since we aim to develop a highly parallelized code and ILU preconditioners are difficult to parallelize efficiently, we come up with a different scheme. A prerequisite for this alternative scheme is to restrict the approach to systems with structural relaxations of at most 5% of the lattice constant. Under this assumption, we can exploit the fact that in such lattices, G^{r}, and partly also Δt, are roughly periodic on a smaller length scale than the size of the actual supercell. This idea is the basis for preconditioning by a block-circulant matrix, which was recently introduced by Bolten *et al.*.[23] We will show that this scheme is optimally suited to obtain efficiently a high-quality preconditioning matrix in KKRnano.

In all cubic or rectangular lattices, the supercell can be partitioned into M_{bl}^x, M_{bl}^y, and M_{bl}^z blocks in the x, y, and z directions in real space. Those in total $M_{\mathrm{bl}} = M_{\mathrm{bl}}^x M_{\mathrm{bl}}^y M_{\mathrm{bl}}^z$ blocks build a new coarse basis for the supercell, where each block contains $N_{\mathrm{bl}} = N/M_{\mathrm{bl}}$ atoms. [See Fig. 2(a) for a two-dimensional example of this spatial construction.] The matrix $A = (\Delta t G^r - I)$ can then be composed out of $M_{\mathrm{bl}} \times M_{\mathrm{bl}}$ submatrices and reads in full representation as

$$A_{LL'}^{nn'} = \begin{pmatrix} \left(a_{LL'}^{n_{\mathrm{bl}}n'_{\mathrm{bl}}}\right)_{11} & \left(a_{LL'}^{n_{\mathrm{bl}}n'_{\mathrm{bl}}}\right)_{12} & \cdots & \left(a_{LL'}^{n_{\mathrm{bl}}n'_{\mathrm{bl}}}\right)_{1N_{\mathrm{bl}}} \\ \left(a_{LL'}^{n_{\mathrm{bl}}n'_{\mathrm{bl}}}\right)_{21} & \left(a_{LL'}^{n_{\mathrm{bl}}n'_{\mathrm{bl}}}\right)_{22} & \cdots & \left(a_{LL'}^{n_{\mathrm{bl}}n'_{\mathrm{bl}}}\right)_{2N_{\mathrm{bl}}} \\ \vdots & \vdots & \ddots & \vdots \\ \left(a_{LL'}^{n_{\mathrm{bl}}n'_{\mathrm{bl}}}\right)_{N_{\mathrm{bl}}1} & \left(a_{LL'}^{n_{\mathrm{bl}}n'_{\mathrm{bl}}}\right)_{N_{\mathrm{bl}}2} & \cdots & \left(a_{LL'}^{n_{\mathrm{bl}}n'_{\mathrm{bl}}}\right)_{N_{\mathrm{bl}}N_{\mathrm{bl}}} \end{pmatrix},$$

where the submatrices $(a_{LL'}^{n_{\mathrm{bl}}n'_{\mathrm{bl}}})_{ij}$ are of dimension $N_{\mathrm{bl}}(l_{\max} + 1)^2 \times N_{\mathrm{bl}}(l_{\max} + 1)^2$ and n_{bl} is accordingly running from 1 to N_{bl}. In this representation, the diagonal submatrices ($i = j$) cover the intrablock interactions, while the interblock interactions are accounted for by the off-diagonal submatrices ($i \neq j$).

Although chemical or geometrical disorder on the lattice leads to a clear distinction between individual subblocks, we assume that average subblocks are in coarse approximation suitable to describe the entire lattice. By dropping the internal indices of the submatrices $a_{ij} = (a_{LL'}^{n_{\mathrm{bl}}n'_{\mathrm{bl}}})_{ij}$, all submatrices are replaced by a set of averaged submatrices $\overline{a}_i$, where the averaging process is carried out for each element separately. The mean submatrix carrying the intrablock interaction then reads as

$$\overline{a}_1 = \frac{1}{M_{\mathrm{bl}}} \sum_{j}^{M_{\mathrm{bl}}} a_{jj}. \tag{28}$$

However, for a generalization of (28) to interblock interactions, it is convenient to use not the row index i of submatrices directly, but instead a local relative index $i_l(j)$: From here on, each $i_l(j)$ marks a subblock of specific relative geometrical position to the central subblock $i = j$ of column j [for an illustration of this definition, see Fig. 2(c)]. This relative geometrical position of the block with respect to the diagonal block is given by $\mathbf{\Delta}_{i_l(j)} = (\Delta_{i_l(j)}^x, \Delta_{i_l(j)}^y, \Delta_{i_l(j)}^z)$. For example, in Fig. 2(c), the block neighboring the diagonal block in the x direction would be addressed by $\mathbf{\Delta}_2 = (1,0,0)$. By utilizing this notation, we can generalize (28) to

$$\overline{a}_{i_l(j)} = \frac{1}{M_{\mathrm{bl}}} \sum_{j}^{M_{\mathrm{bl}}} a_{i_l(j)j}, \tag{29}$$

which is for the intrablock interaction $i_l(j) = 1$ equivalent to Eq. (28). This averaging operation (29) is schematically visualized in Figs. 2(d) and 2(e). With this set of averaged blocks, we can proceed representing the full matrix $A_{LL'}^{nn'}$ by a block-circulant matrix. It is important to note that in practice we restrict the number of considered off-diagonal subblocks as indicated in Fig. 2(a) to $M'_{\mathrm{bl}} < M_{\mathrm{bl}}$, which is equivalent to introducing a number of zero matrices on the block-circulant matrix A. M'_{bl} is often and throughout this paper chosen such that nearest- and next-nearest-neighbor subblocks are included. This cutoff is justified by the fact that through the use of screened reference potentials, blocks being geometrically far

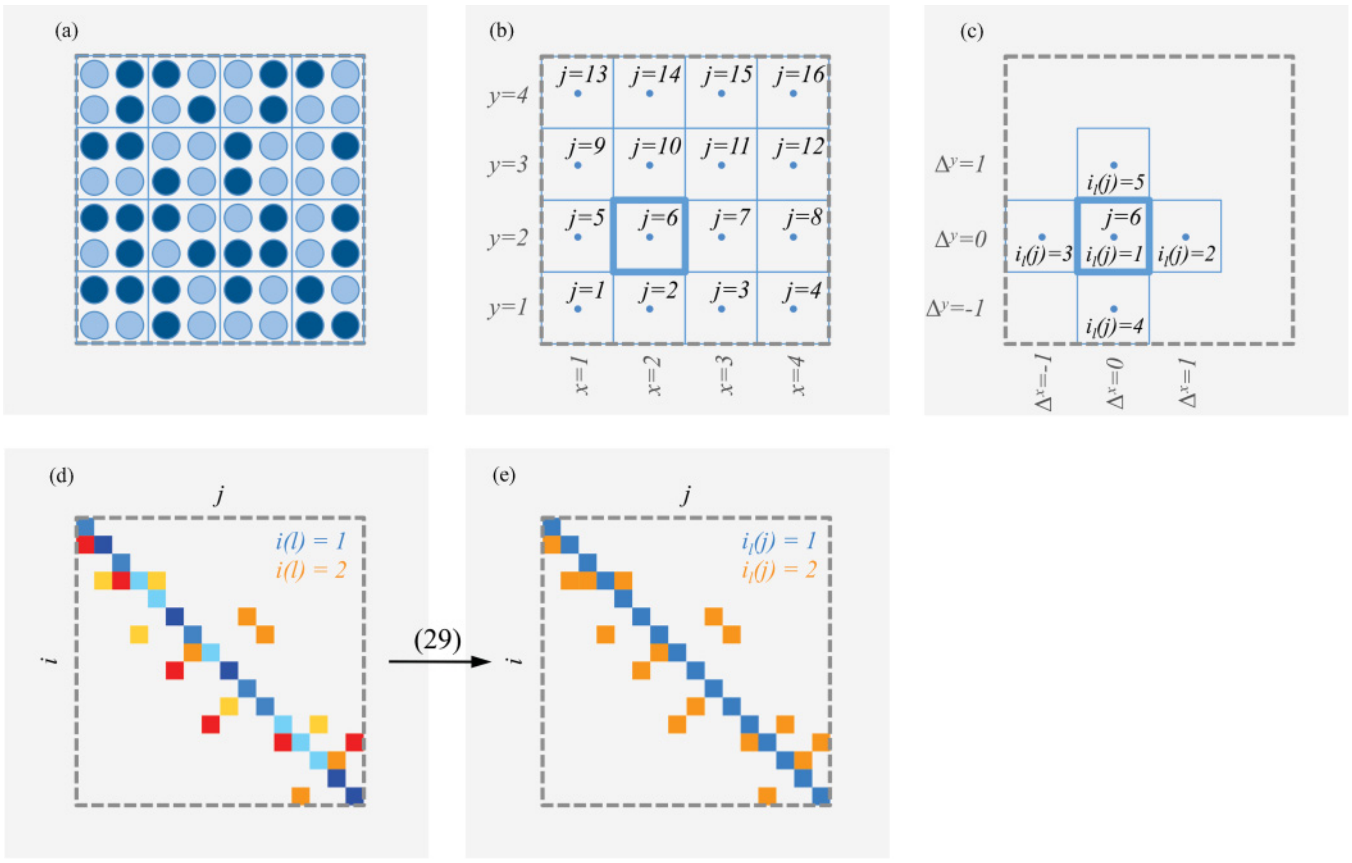

FIG. 2. (Color online) In all five panels, a two-dimensional supercell (boundary marked with gray dashed lines) is drawn and the five different conceptual steps needed for the block-circulant preconditioning scheme are illustrated. (a) Shows the supercell on the level of atoms, which is here exemplified by a two-dimensional disordered lattice of two arbitrary types of atoms (light and dark blue) with overall $N = 64$ atoms. In addition, the supercell is partitioned into $M_{\text{bl}} = 16$ subblocks, where each subblock contains $N_{\text{bl}} = 4$. The borders of the subblock are indicated by blue lines. Then, overall $M^x_{\text{bl}} = 4$ and $M^y_{\text{bl}} = 4$ subblocks in the x and y directions are required to represent the full supercell. (b) Illustrates how those subblocks are labeled over the entire supercell by the index j from $j = 1$ to $j = M_{\text{bl}} = M^x_{\text{bl}} M^y_{\text{bl}}$. In addition, it is shown how this index j is related to the position (x_j, y_j) in real space. In (c), the interblock interactions of subblock $j = 6$, which is highlighted by thick blue lines in (b) and (c), are depicted in the space of the row index $i_l(j)$. For the sake of simplicity, index $i_l(j)$ is running exclusively over nearest-neighboring subblocks. Further, the relative geometrical position of the interacting blocks $\Delta^x_{i_l(j)}$ and $\Delta^y_{i_l(j)}$ are specified for this simplified example. (d) Schematically shows the full interaction matrix of the supercell from all $M_{\text{bl}} = 16$ blocks amongst each other, highlighting two selected types of interaction $i_l(j) = 1$ (orange colors) and $i_l(j) = 2$ (blue colors), where the variations in color represent variations in the individual interactions. In direct contrast, (e) depicts the consequence of averaging the full interaction matrix to effective interactions by means of Eq. (29), which are accordingly represented by uniform colors. Note that (d) and (e) are schematical illustrations being not one-to-one related to (a)–(c).

from the centered diagonal block have small elements and can be neglected for the construction of the preconditioning matrix.

The averaged block matrices are now used to set up a block-circulant matrix, which is utilized as preconditioning matrix P_2. For a fast computation of the inverse of P_2, an important property of circulant matrices is exploited: Given a Fourier transform defined as

$$\overline{\alpha}_j = \sum_i \overline{a}_i \, e^{-2\pi i \boldsymbol{\Delta}_{i_l(j)} \mathbf{k}_j}, \tag{30}$$

where

$$\mathbf{k}_j = (k_x, k_y, k_z) = \left(\frac{x_j - 1}{M^x_{\text{bl}}}, \frac{y_j - 1}{M^y_{\text{bl}}}, \frac{z_j - 1}{M^z_{\text{bl}}} \right). \tag{31}$$

An illustration of the definition of the spatial indices x_j, y_j, and z_j can be found in Fig. 2(b) and the blocks $\overline{\alpha}_j$ are of size $N_{\text{bl}}(l_{\max} + 1)^2 \times N_{\text{bl}}(l_{\max} + 1)^2$. This Fourier transform (30) of a circulant or block-circulant matrix P_2 creates a block-diagonal representation of P_2, $(P_2)_{\mathbf{k}}$, in reciprocal space.[23] The submatrices of $(P_2)_{\mathbf{k}}$, $\overline{\alpha}_j$, can now be block-wise inverted by means of LU decomposition. These are fast operations due to the small block sizes, e.g., for $N_{\text{bl}} = 10$ atoms and $l_{\max} = 3$ the blocks have a size of 160×160. The required matrix for preconditioning P_2^{-1} is then constructed out of

$$(P_2)^{-1}_{\mathbf{k}} = \begin{pmatrix} \overline{\alpha}_1^{-1} & 0 & \dots & 0 \\ 0 & \overline{\alpha}_2^{-1} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \overline{\alpha}_{M_{\text{bl}}}^{-1} \end{pmatrix}. \tag{32}$$

Multiplication of $(P_2)_{\mathbf{k}}^{-1}$ as operated in Eq. (26) as well as in every multiplication involving A' is then conducted in reciprocal space. Therefore, first a fast Fourier transform (FFT) is applied straightforwardly to Y, $Y \longmapsto Y_{\mathbf{k}}$, then the matrix multiplication $(P_2)_{\mathbf{k}}^{-1} Y_{\mathbf{k}}$ is done, and as a last step, the back transformation $(P_2)_{\mathbf{k}}^{-1} Y_{\mathbf{k}} \longmapsto P_2^{-1} Y$ gives the desired preconditioned vector $P_2^{-1} Y$.

After having introduced the preconditioning scheme, it is worthwhile commenting on the choice and construction of subblocks. Depending on the system size and complexity of the lattice, subblocks can be defined in several ways. To operate with small preprocessing times, which scale cubically with the number of atoms in the subblocks, in practice we choose block sizes not larger than 16 atoms. For cubic cells, N_{bl} is usually set to, e.g., $N_{\text{bl}} = 4$ for fcc or $N_{\text{bl}} = 8$ for rocksalt structures, while for structures incorporating additional interstitial sites, such as zinc blende or diamond, typically $N_{\text{bl}} = 16$ is selected.

The performance acceleration, or the reduction of the required number of TFQMR iterations, respectively, is shown in Fig. 3(a) for a sample system, i.e., a Si_{432} unit cell, at the

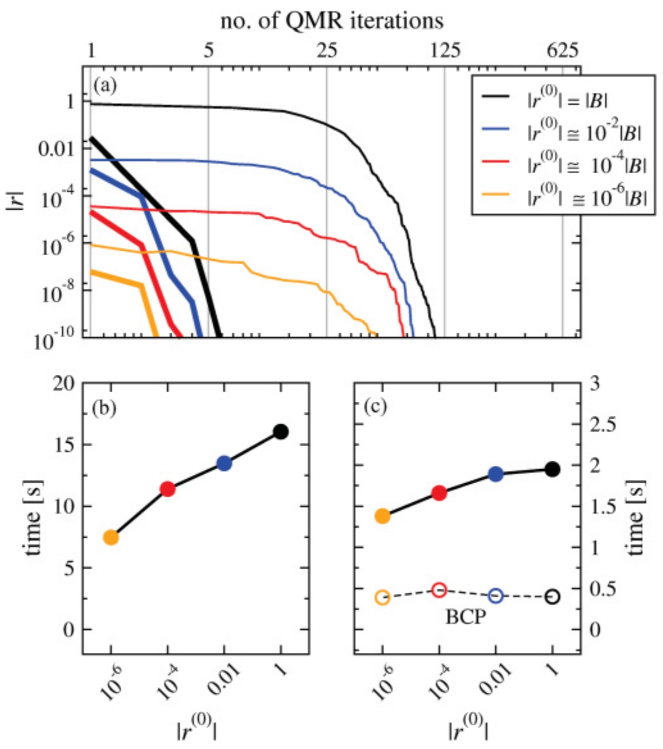

FIG. 3. (Color online) (a) Convergency of the norm of the residual vector $|r|$ as a function of the number of TFQMR iterations combining the initial guess approach and block-circulant preconditioning (BCP) for an arbitrary column of a matrix corresponding to Si_{432}. Four different qualities of starting vectors rated by the initial residual $|r^{(0)}| = AX^{(0)} - B$ are shown for comparison with (thick lines) and without (thin lines) applied preconditioning. Note that a log-log scale is used. In (b) and (c), the CPU time required to obtain convergency is shown for all qualities of initial guesses used in (a) in consistent color coding. Timings with and without applied preconditioning are shown in (c) and (b), respectively. Note that the scale is different in (b) and (c). For the sake of comparison, the CPU time required to set up the BCP is specified in (c). The CPU time to apply the initial guess is for all scenarios negligibly small and not shown.

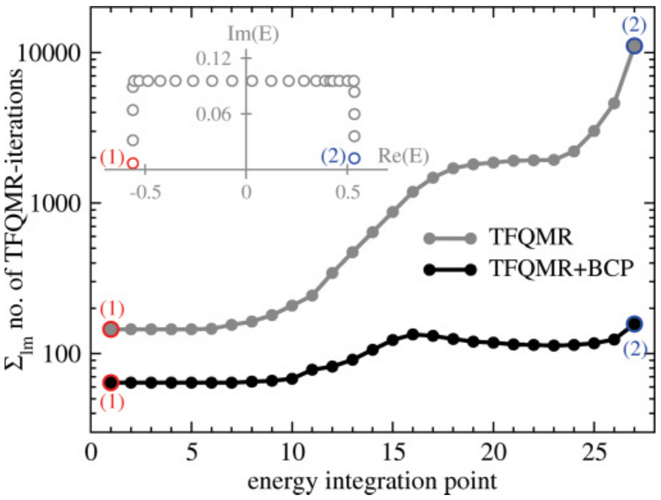

FIG. 4. (Color online) Sum of TFQMR and TFQMR+BCP iterations over all lm components and one arbitrary atom in the test system $\text{Ni}_5\text{Pd}_{251}$. Here, all 27 energy integration points are considered and no initial guess has been used. A random displacement from the ideal atomic sites on the order of 1% has been introduced on all sites. The inset shows the distribution of the energy points along the integration contour. The color-coded points illustrate the corresponding positions of energy points in the main graph.

energy point closest to the real axis. This point is of particular importance as the convergence at this energy point is most demanding. For all scenarios with different qualities for the initial guesses, convergency to sufficiently small norms $|r^{(\nu)}|$ of the residual vector $r^{(\nu)}$ is reached strikingly faster than without block-circulant preconditioning. By applying block-circulant preconditioning, at least a factor of 20 less TFQMR iterations have to be performed. To obtain a more general picture, Fig. 4 shows the required number of TFQMR iterations at different energy points. Here, the large variability over a range of 100 to 10 000 iterations is clearly visible for the unpreconditioned approach. Preconditioning leads to a significant reduction by a factor of 2 to 50 and to a much smaller spread of the number of iterations between 60 to 200 from the first to the last energy point.

This reduction of the number of iterations raises the question as to whether it can be translated into an overall speedup of the algorithm since additional computational work has to be performed: On the one hand, the setup of the calculation with initial guess and the creation of a preconditioning matrix P_2^{-1} is performed once prior to the start of the iterative procedure. On the other hand, P_2^{-1} is applied at each iterative step. While the setup of the initial guess requires only negligibly more computational overhead, Figs. 3(b) and 3(c) display the timings for both preconditioning steps. Apparent from Fig. 3(c), about 25% of the total time of the iterative solution with block-circulant preconditioning (BCP) is consumed to generate P_2^{-1}. However, comparing the timing with and without the BCP in Figs. 3(b) and 3(c) reveals that the extra amount of computational work is well invested. The strongly reduced number of iterations [even though each of them takes longer due to the multiplication with P_2^{-1} according to (25)] translates for this particular energy point and system into a significant speedup of the algorithm by a factor 6 to 9 as shown in Figs. 3(b) and 3(c).

Overall, and for many different metallic and semiconducting systems including small relaxations on the order of 5% of the lattice constant, we observe in total a speedup by block-circulant preconditioning by a factor of 3 to 10, depending predominantly on the temperature and distribution of energy points.

It is important to note that the presented introduction of the iterative solution of the Dyson equation does not limit the accuracy of the calculations: For example, with respect to the total energy, an error per atom on the order of one μeV is reached usually already by stopping the iterative steps for residuals on the order of $10^{-7}|b|$. Therefore, we find that calculations with KKRnano fully maintain the high accuracy as established with existing full-potential KKR methods.[14]

V. PARALLELIZATION AND SCALING

During the past decade, a rapid trend to parallelism evolved in high-performance computing. This development is still ongoing and manifests in the fact that nowadays the fastest supercomputers contain hundreds of thousands of processing units (cores). In order to guarantee the portability of KKRnano to existing and future platforms of that kind, we have developed and optimized our code on two modern computing architectures available at the Forschungszentrum Jülich, which are both among the 40 fastest computers in the world (Ref. 24). Those supercomputers are an IBM Blue Gene/P [JUGENE (Ref. 25)] with 294 912 cores, four cores per node, 2048 MB memory per node, and the JUROPA system[26] with 17 664 cores, eight cores per node, 24 GB memory per node. Based on these facts and the demand for portability to other platforms, we define three essential goals for the parallelization of KKRnano:

(i) parallelization up to at least 10 000 processors;

(ii) memory demand below the limit of 512 MB per message processing interface (MPI) process;

(iii) OpenMP parallelization.

While massive parallelization and low memory demands are obvious prerequisites to perform calculations on JUGENE, an additional level of OpenMP parallelization gives us the flexibility to operate in a shared memory approach and use considerable more memory, e.g., on JUGENE 2048 MB of memory or on JUROPA up to 24 GB of memory. To achieve these goals in KKRnano, four levels of parallelization are realized, which are schematically shown in Fig. 5. While the base frame of our method is the parallelization over atoms, which is always active, the other levels of parallelization can be used optionally. In the following, we will briefly describe the important steps of all levels of parallelization and independently illustrate their efficiency.

A. Atom parallelization

The entire program has been parallelized in real space over Voronoi cells. Those cells are constructed around the atomic sites and, if present in the lattice, around vacancies and/or interstitials. Although not all cells in general must

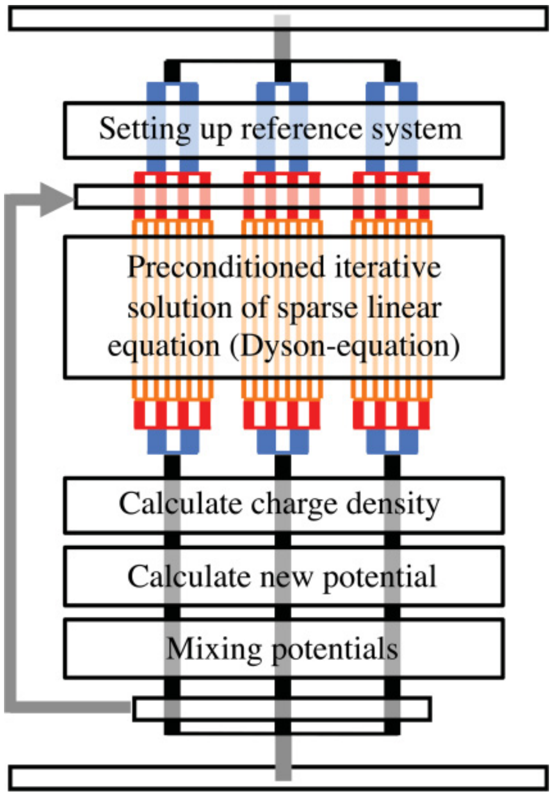
Setting up reference system
Preconditioned iterative solution of sparse linear equation (Dyson-equation)
Calculate charge density
Calculate new potential
Mixing potentials

FIG. 5. (Color online) Schematic workflow and nested parallelization of four levels of hierarchy as implemented in KKRnano. Starting with a serial process (gray), the computational work is distributed to the real-space parallelization (black). Here, three branches correspond to three atoms. These processes are further split, e.g., into two processes distributing operations over energy integration points (blue). Subsequently, those processes can be further split in spin-up and spin-down processes (red) in case of spin polarization. While up to this point all parallelization has been implemented in a distributed memory approach using MPI 2.0, the inner nested parallelization (orange) of the solution of the Dyson equation is based on OpenMP parallelization, which is in this example split into two threads.

contain atoms, we will for the sake of simplicity refer to this parallelization scheme as atom parallelization.

KKRnano is constructed such that the loop over atoms takes the highest hierarchy, which is the natural choice for the implemented iterative scheme. In order to be able to distribute the memory to the MPI processes, the number of processors N_p has to be equal or larger than the number of lattice sites in the supercell N. The computationally most demanding part, the iterative solution of the Dyson equation, can be split straightforwardly in terms of the atom parallelization as explained above. In nonideal structures, which explicitly include relaxations, each atomic site might have a different surrounding. For the screened KKR method, this requires the computation of the reference structure constants for all clusters. This work can be efficiently distributed using atom parallelization. The number of reference clusters is always smaller or equal to the number of sites and the computation of G^r can therefore be straightforwardly distributed to the atom parallelization and subsequently broadcasted by MPI communication.

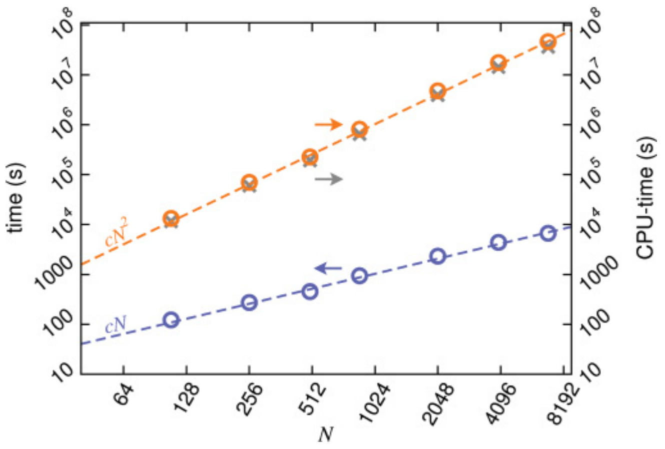

FIG. 6. (Color online) Scaling of KKRnano on a Blue Gene/P (Ref. 25) for a Ni_xPd_{1-x} system as a function of number of atoms per unit cell N from N =108 to 6912 with a Ni concentration of $x \approx 3\%$ using one **k** point and always one processor per atom. Block-circulant preconditioning has been applied with blocks of four atoms, and no initial guess has been used. Displayed are the execution time t_f (blue), the total CPU time $t_f N = t_f N_p$ (orange), as well as total CPU time without the time required for MPI communication (gray crosses). Straight lines are linear cN (blue) and quadratic cN^2 (orange) fits to the data points, where c is a constant, which is slightly different for both fits. All time measurements have been executed using the performance analysis tool SCALASCA (Ref. 27).

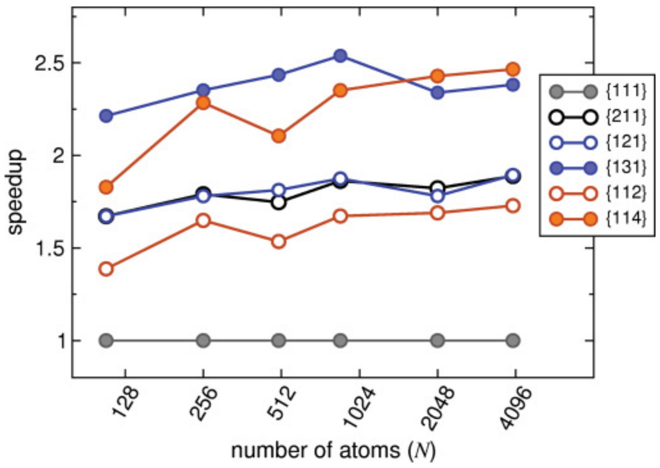

FIG. 7. (Color online) Speedup of KKRnano as a function of number of atoms N in the supercell. The number of processors is always equal to the number of atoms N defining the speedup of 1. All optional levels of parallelization are probed separately on a Blue Gene/P (Ref. 25) architecture for Ni_xPd_{1-x} alloys. The details are the same as described in the caption of Fig. 6. Labels $\{p_S, p_E, t_{OMP}\}$ with p_S, p_E, and t_{OMP} specify the number of processes/threads used per spin, energy, and OpenMP parallelization, respectively. For the energy parallelization, dynamic load balancing as explained in the text has been adopted.

The weak scaling, i.e., increasing the system size and the number of processors used by the same factor, of the atom parallelization is shown in Fig. 6. Focusing on the pure atom parallelization, an O(N^2) scaling in CPU time can be observed. Alternative approaches to solve the Dyson equation are not preferable, as those direct methods run into the memory boundaries on supercomputers quickly and are not efficiently parallelizable. The quadratic and linear fits to the time measurement reveal that an O(N^2) scaling in CPU time and an O(N) scaling in execution time describe well the scaling performance of KKRnano (see Fig. 6). This efficient parallel performance has been rendered possible by designing the method such that MPI communication presents no bottleneck. In order to highlight this fact, we show in addition in Fig. 6 the required CPU time after subtracting all MPI communication related efforts: Independent on N, less than 10% of computational time is used up for MPI communication and synchronization of MPI processes.

B. Spin parallelization

In magnetic systems with collinear magnetic spin structures such as ferromagnets, antiferromagnets, ferrimagnets, or solids with spin-density waves, the parallelization over spin-up and -down electrons provides a natural second level of parallelization. The work performed for the two types of electrons is divided into two MPI processes, which include setups of the Δt matrices, solution of the Dyson equation, and computation of the electron density. The speedup shown in Fig. 7 (compare curve {211} with {111}) exhibits the high efficiency of this level of parallelization. For all examined system sizes, we find an acceleration by a factor of 1.7 to 1.8 in execution time compared to spin nonparallelized calculations. Here, as for all other optional levels of parallelization, the increased speedup for larger systems can be related to the fact that the ratio of parallelized parts to the nonparallelized overhead grows with N. Further, it should be pointed out that the spin parallelization as implemented in KKRnano at present can be applied straightforwardly only to collinear spin systems without spin-orbit coupling, which would couple the spin channels.

C. Energy parallelization

As a third level of parallelization, KKRnano has the option to distribute the energy integration points across processors. As introduced before, the energy integration is performed on a complex contour, thereby reducing the number of energy points to typically 30 to 40. Because the Dyson equation is solved iteratively, the number of iterations depends on the position of the energy point in the complex plane. Figure 4 shows a typical example for the workload at different energy points. Depending on the electronic temperature and the material, the computational time required to solve the Dyson equation for the last energy point closest to the real axis is in the range between 20% and 40% of the total time.

Balancing the workload, i.e., optimally distributing the energy points over the processors such that ideally all processes finish their individual task at the same time, is a highly nontrivial task that depends on the quality of preconditioning and the initial guess for the iterative solution. Therefore, we introduced a scheme which dynamically load balances the computation for the sth self-consistency step based on the performance of the $(s-1)$th step. The timing for each energy point is examined on the fly and used to reschedule the MPI processes under the condition to achieve the optimal load balance. With this approach, an estimate of work distribution, i.e., to each processor a group of energy points is allocated, has to be made only for the first self-consistency iteration. For all subsequent iterations, the dynamical load balancing reaches an efficiency between 90% and 100%. By exploiting the energy parallelization, we gain a speedup of 1.7 to 1.8 if two energy groups are used and 2.3 to 2.6 for three energy groups (compare curves {131}, {121}, and {111} in Fig. 7).

D. OpenMP parallelization

Up to this point, KKRnano has been exclusively parallelized by means of MPI parallelization. However, state-of-the-art supercomputers are nowadays predominantly built in hybrid architecture combining 2–32 CPUs in a shared memory environment on one node and up to tens of thousands of those nodes communicating as distributed memory units amongst each other. This shared memory approach provides an opportunity to circumvent the notorious memory resources on current supercomputers. To be able to exploit this important advantage, we introduce an OpenMP level of parallelization on top of the existing MPI parallelization, which is described in the following.

At this point, we have at least two options as to how to introduce an OpenMP parallelization for the dominant computational work required to solve the Dyson equation. The iterative approach solving (20) decouples all columns of the matrix X, which offers the possibility to introduce a parallel scheme over the $(l_{\max}+1)^2$ angular momentum expansion coefficients. The strict separation of lm components leads to the use of sparse matrix-vector operations, while in general the operation AX can be performed as sparse matrix-matrix multiplication, where X has then $N(l_{\max}+1)^2 \times (l_{\max}+1)^2$ elements. Our performance tests showed that beneficial cache access during the matrix-matrix operations leads to an acceleration of the algorithm by approximately a factor of 2. In order to incorporate this speedup in KKRnano, we implemented a flexible low-level OpenMP parallelization in various parts of the program, including the matrix-matrix operations in the iterative steps. The following performance analysis will be restricted to the inclusion of at most four OpenMP threads, which is the limit in exploiting the Blue Gene/P (Ref. 25) architecture. By distribution to two and four OpenMP threads, we observe an acceleration of 1.4 to 1.7 and 1.7 to 2.4, respectively (compare curves {114}, {112}, and {111} of Fig. 7). For the practical application, it is important to note that aside from this speedup, the OpenMP parallelization extends the memory bounds of KKRnano so that system sizes of more than 10 000 atoms per supercell are treatable.

The comparison of the three optional levels of parallelization on top of the natural one with respect to number of atoms leads to the conclusion that the spin or energy parallelization are the option of choice if the application is not memory bound. If the limited amount of memory restricts the use of these MPI levels of parallelization, the OpenMP parallelization can surmount this bottleneck and still leads to a speedup that is only 20–30% away from the ideal one.

The speedup of all four levels of parallelization and their combination is shown in Fig. 8. For 4096 atoms and up to eight processors per atom, we observe a speedup being

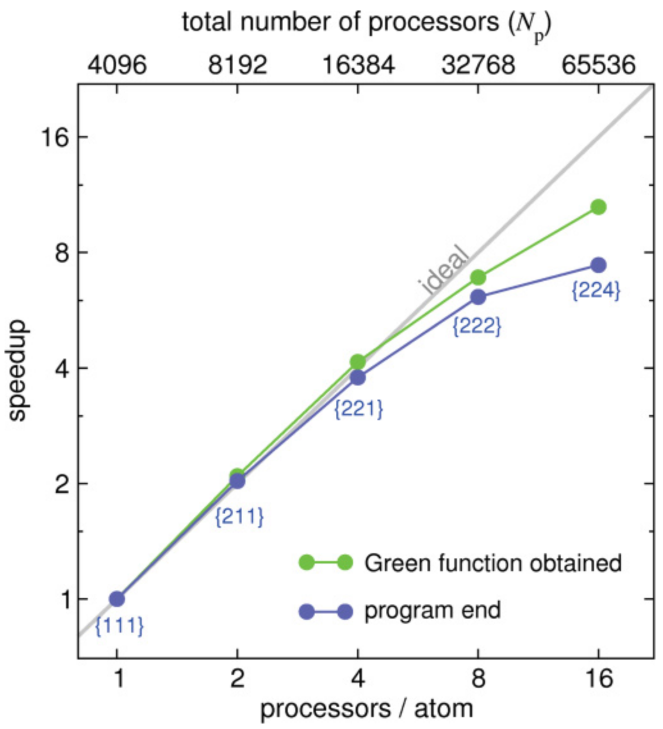

FIG. 8. (Color online) Speedup of KKRnano (blue) with respect to the number of processors combining subsequently all levels of parallelization on a Blue Gene/P (Ref. 25) architecture for a $\mathrm{Ni}_x\mathrm{Pd}_{1-x}$ alloy of 4096 atoms versus ideal speedup (gray line). The computational details are the same as described in the caption of Fig. 6. Labels $\{p_S, p_E, t_{OMP}\}$ with p_S, p_E, and t_{OMP} specify the number of processes/threads used per spin, energy, and OpenMP parallelization. For the energy parallelization, dynamic load balancing as explained in the text has been adopted.

larger than 60% of the ideal speedup. At a higher level of parallelization of up to 16 processes per atom, KKRnano still shows significant acceleration up to a speedup of eight, but the inefficiencies due to overhead and MPI communication are clearly visible. Hence, for the test supercell of 4096 atoms operating between 4096 and 32768 processors guarantees high efficiency. For larger cells of 10 000 atoms, the percentage of overhead is reduced and an even higher level of parallelization becomes efficiently usable. Therefore, KKRnano is ideally suited to run with up to 100 000 processors on present and future supercomputing architectures.[28]

VI. TRUNCATION OF INTERACTION

So far, we solved the Dyson equation and the electron density, respectively, without compromising on the accuracy. In the following, we explore the possibility of introducing in addition the nearsightedness of the density matrix as proposed by Kohn.[29] While the diagonal part of the density matrix $\rho(\mathbf{r},\mathbf{r}')$ is equivalent to the electron density, the full expression for ρ reads as

$$\rho(\mathbf{r},\mathbf{r}') = -\frac{1}{\pi}\,\mathrm{Im}\int_{-\infty}^{\infty} f_T(E)\, G(\mathbf{r},\mathbf{r}',E)\, dE. \qquad (33)$$

Here, analogously as in Eq. (7), the Fermi function $f_T(E)$ enters. Apparently, the Green's function gives the density matrix, which facilitates an application of the nearsighted principle within the KKR Green's-function approach. This route has been first exploited by Wang *et al.*.[9] Depending on material properties, e.g., the localization of electronic states, the multiple-scattering interaction can be restricted indeed to a local interaction zone described by a local cluster with a few hundred up to a few thousand atoms N_{tr}. Accordingly, interactions to sites being not part of the local cluster are truncated and neglected. The loss of accuracy, which is induced by this truncation of the interaction, can however be well controlled by the parameter N_{tr}.[11]

When using this approximation, only $N_{\mathrm{tr}}(l_{\max}+1)^2$ rather than $N(l_{\max}+1)^2$ nonzero entries of each column of the solvent of the linear matrix equation X have to be considered. As a direct consequence, the full matrix $A = \Delta t G^r - I$ does not have to be taken into account to solve for X: The fraction of the matrix A, which has to be considered, becomes independent of system size N and proportional to $N_{\mathrm{cl}}N_{\mathrm{tr}}$. In other words, the Dyson equation must be solved only in the individual local interaction zone for each of the atoms in the unit cell. With the reduced sizes of A and X, the required matrix-vector operations scale with N_{tr} instead of N. As those operations need about 90% of the computational work of one TFQMR cycle, the truncation leads in the optimal case to a reduction of computational time by N_{tr}/N. With this truncation, the algorithm used in KKRnano ideally scales $\propto N_{it}N_{\mathrm{cl}}N_{\mathrm{tr}}N$, i.e., linearly with number of atoms N. Hence, if losses due to MPI communication and overhead are neglected for the moment, the atom parallelization in KKRnano with its distribution of work to N_{p} processors ($N_{\mathrm{p}} \geqslant N$ and $N_{\mathrm{p}} \propto N$) leads to an execution time independent of N. Figure 9 reveals this conjecture in praxis. Except for minor losses due to load imbalance, MPI communication and overhead, the execution time indeed remains almost constant for $N > N_{\mathrm{tr}}$. In other words, KKRnano shows efficient O(N) scaling for $N > N_{\mathrm{tr}}$.

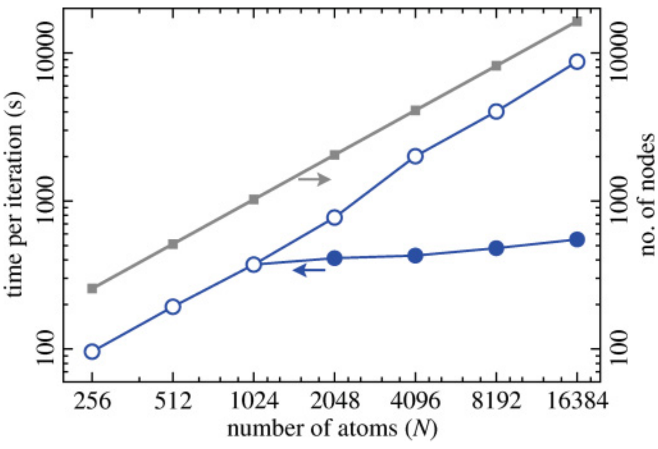

FIG. 9. (Color online) Scaling of KKRnano in double-logarithmic representation without (open blue circles) and with applied truncation (filled blue circles). The test system is $Ni_{1-x}Pd_x$ with $x = 5\%$ and one **k** point on JUGENE. Here, parallelization over atoms, spin, and two groups of energy points are used. The number of processing nodes (4 CPUs per node) is shown in dark gray.

With this advantageous scaling at hand, the question remains as to how the accuracy of the calculation is affected by truncation. For the metallic test system NiPd, the scaling of which is shown in Fig. 9, the applied cutoff of $N_{\mathrm{tr}} = 959$ atoms in the interaction zone leads to an energy error of less than 2 meV per atom. The loss of accuracy introduced by truncation depends in general strongly on the treated material and has be tested to gain ultimate control over the error. In any case, the accuracy can be increased if the electronic potential calculated self-consistently with a given interaction zone is refined in one additional self-consistency step with a larger zone.

An important advantage of our approach compared to other methods[9] is that linear scaling can be exploited optionally. This means, for systems where accuracy would demand interaction zones with tens of thousands of atoms, we can make use of the explicit periodicity of the supercell using one **k** point. Aside from the beneficial linear scaling, the high degree of sparsity arising from the truncation opens the possibility to extend the range of system sizes to more than 16 000 atoms.

VII. CONCLUSION

We have presented the successful development of a powerful massively parallel density functional full-potential Korringa-Kohn-Rostoker Green's-function method, KKR-nano, which is especially designed for large-scale applications of more than 10 000 atoms on hundreds of thousands of processors. As typical for all-electron methods without shape approximation, KKRnano is designed to give the density functional answer to the problem at hand and accuracy is not compromised by the algorithms introduced for large-scale applications on massively parallel computers. The advantageously quadratic or optionally linear scaling with system size and the high parallel scalability have been achieved by making use of screened reference systems combined with the iterative solution of the Dyson equation and optionally the truncation of long-range interactions. The excellent scaling behavior was demonstrated for alloys with more than 16 000 atoms and the test calculations involved more than 65 000 processors. In addition, we presented schemes to obtain high-quality initial guesses and preconditioning matrices for the iterative solution of the Dyson equation, which enable us to speed up KKRnano by up to one order of magnitude. The high parallel efficiency of KKRnano is thereby the key to exploit the full strength of today's and future massively parallel supercomputing architectures.

ACKNOWLEDGMENTS

We like to thank W. Lambrecht and P. Mavropoulos for fruitful discussions. Financial support of the DAAD and both computational resources as well as technical support of the Jülich Supercomputing Center are gratefully acknowledged. This work benefited from discussions within the SFB 917 Nanoswitches.

*a.thiess@fz-juelich.de

[1]H. Shiba, Prog. Theor. Phys. **46**, 77 (1971).
[2]D. A. Rowlands, Rep. Prog. Phys. **72**, 086501 (2009).
[3]R. M. Nieminen, Top. Appl. Phys. **104**, 29 (2007).
[4]A. E. Mattsson, P. A. Schultz, M. P. Desjarlais, T. R. Mattsson, and K. Leung, Modell. Simul. Mater. Sci. Eng. **13**, 1(R) (2005).
[5]M. J. Gillan, D. R. Bowler, A. S. Torralba, and T. Miyazaki, Comput. Phys. Commun. **177**, 14 (2007).
[6]C.-K. Skylaris, P. D. Haynes, A. A. Mostofi, and M. C. Payne, J. Chem. Phys. **122**, 084119 (2005).
[7]J. M. Soler, E. Artacho, J. D. Gale, A. García, J. Junquera, P. Ordejón, and D. Sánchez-Portal, J. Phys.: Condens. Matter **14**, 2745 (2002).
[8]T. Ozaki, Phys. Rev. B **82**, 075131 (2010).
[9]Y. Wang, G. M. Stocks, W. A. Shelton, D. M. C. Nicholson, Z. Szotek, and W. M. Temmerman, Phys. Rev. Lett. **75**, 2867 (1995).
[10]I. A. Abrikosov, A. M. N. Niklasson, S. I. Simak, B. Johansson, A. V. Ruban, and H. L. Skriver, Phys. Rev. Lett. **76**, 4203 (1996).
[11]R. Zeller, J. Phys.: Condens. Matter **20**, 294215 (2008).
[12]T. Ozaki, Phys. Rev. B **74**, 245101 (2006).
[13]A. V. Smirnov and D. D. Johnson, Phys. Rev. B **64**, 235129 (2001).
[14]N. Papanikolaou, R. Zeller, and P. H. Dederichs, J. Phys.: Condens. Matter **14**, 2799 (2002).
[15]K. Wildberger, P. Lang, R. Zeller, and P. H. Dederichs, Phys. Rev. B **52**, 11502 (1995).
[16]O. K. Andersen and O. Jepsen, Phys. Rev. Lett. **53**, 2571 (1984).
[17]B. Wenzien, J. Kudrnovsky, V. Drchal, and M. Sob, J. Phys.: Condens. Matter **1**, 9893 (1989).
[18]R. Zeller, P. H. Dederichs, B. Újfalussy, L. Szunyogh, and P. Weinberger, Phys. Rev. B **52**, 8807 (1995).
[19]Y. Saad and M. Schultz, SIAM J. Sci. Stat. Comput. **7**, 856 (1986).
[20]R. W. Freund and N. M. Nachtigal, Numer. Math. **60**, 315 (1991).
[21]R. W. Freund, SIAM J. Sci. Comput. **14**, 470 (1993).
[22]A. V. Smirnov and D. D. Johnson, Comput. Phys. Commun. **148**, 74 (2002).
[23]M. Bolten, A. Thiess, I. Yavneh, and R. Zeller, Linear Algebra Appl. **436**, 436 (2012).
[24]http://www.top500.org.
[25]http://www.fz-juelich.de/jsc/jugene.
[26]http://www.fz-juelich.de/jsc/juropa.
[27]M. Geimer, F. Wolf, B. J. N. Wylie, E. Ábrahám, D. Becker, and B. Mohr, Concurrency Comput: Pract. Experience **22**, 702 (2010).
[28]KKRnano has been successfully tested on the full Blue Gene/P JUGENE (Ref. 25) consisting of 294 912 cores (Ref. 30).
[29]W. Kohn, Phys. Rev. Lett. **76**, 3168 (1996).
[30]R. Zeller and A. Thiess, Jülich Supercomputing Centre, Forschungszentrum Jülich, Germany, Report No. FZJ-JSC-IB-2010-03 (unpublished).