LI EA A XIMA I F ELEC IC C E CALC LA I

G. BEYLKIN AND T. S. HAUT *

ABSTRACT. **We present a new method for electronic structure calculations based on novel algorithms for nonlinear approximations. We maintain a functional form for the spatial orbitals as a linear combination of products of decaying exponentials and spherical harmonics centered at the nuclear cusps. Although such representations bare some resemblance to the classical Slater-type orbitals, the complex-valued exponents in the representations are dynamically optimized via recently developed algorithms, yielding highly accurate solutions with guaranteed error bounds. These new algorithms make dynamic optimization an e□ective way to combine the e□ciency of Slater-type orbitals with the adaptivity of modern multiresolution methods.**

We develop numerical calculus suitable for electronic structure calculations.

to optimize the exponents in Slater-type approximations for diatomic molecules. Although such dynamic nonlinear optimization via traditional methods

can be expressed in terms of spherical harmonics (see Section 3.2) and, therefore, these two representations are functionally equivalent. In Section 3.2 we describe algorithms for converting between these two functional forms. We use one form or another depending on e□ciency considerations as is typical in pseudo-spectral methods where the interpolating representation is more convenient for multiplying functions, while the spectral representation is more convenient for computing convolutions.

We develop a numerical calculus based on these functional representations. For example, for any two functions $_1(\mathbf{r})$ and $_2(\mathbf{r})$ that are already in this functional form, we develop algorithms to represent the product $_1(\mathbf{r})\ _2(\mathbf{r})$ and the convolution $\Delta^{-1}\ _1(\mathbf{r})$ in the same functional form and with a small number of parameters. By casting the relevant electronic structure equations (e.g. the Hartree-Fock equations) in an integral form, we demonstrate that the functional forms in (1.1) and (1.2) can be used to solve for the ground states via iteration (using the framework developed in [18, 16, 17, 19]). The numerical calculus used within the iteration framework allows us to e□ciently build up highly e□cient representations for the solutions with guaranteed error bounds. It is also noteworthy that we compute the convolution $^2 + \Delta\ ^{-1}\ _1(\mathbf{r})$ in the spectral domain, which obviates the need to solve a large sparse matrix equation.

We consider the algorithms presented here to be of a preliminary nature, since

Algorithm Computing exponential representations

(1) Compute singular vectors $\mathbf{u}_m = \sigma_m \overline{\mathbf{u}}_m$ of $= [\ _{i+j}]_{i,j=0}^{N}$, σ = 0 $-$ 1, where M

(2) Form the $\times$ matrix $_3 = {}_1^{\dagger}\ _2$, where = ($\mathbf{u}_0$ $_{M-1}$), $_1$ = (0 : $-$1 1 :), and $_2$ = (1 : 1 :)

(3) Compute the

the odd moments (approximately) vanish to sufficiently high order. We note that this property naturally arises from application of Algorithm 1 and is not imposed as an additional constraint.

Remark 2.1. Although Algorithm 1 suffices for computing exponential approximations when the approximation error is no smaller than 10^{-7} and $f(x)$ is smooth and decays rapidly (which is sufficient for many electronic structure calculations), computing more accurate approximations using this algorithm may require quadruple precision.

For this reason, one of the possible alternative formulations described in Section 6 involves the so-called reduction algorithm [20]. Specifically, if the function $f(x)$ is already a linear combination of M decaying exponentials, then the reduction algorithm constructs another representation of the same form, but with a smaller number $\tilde{M}$ of exponents. The basic idea behind this approach is that it is straightforward to construct a sub-optimal representation of a given function as a sum of decaying exponentials (i.e., a representation that contains an excessive number of terms for a desired approximation error ε); the reduction algorithm may then be used to compute another exponential representation, but with a smaller number of terms. The reduction algorithm requires $\mathcal{O}(\tilde{M}^2)$ operations, and is therefore essentially linear in the number of sub-optimal exponentials. Moreover, in contrast to Algorithm 1, the reduction algorithm reliably yields (near) optimal representations with approximation error as small as 10^{-14}, and has high efficiency even when $f(x)$ decays slowly or has $1/r$-type singularities.

2.2. Interpolation on the sphere. Let us briefly recall some results from [4] on quadratures for efficient integration and interpolation on the sphere with nodes invariant under the icosahedral group.

where $_j$, = 1 $_$ denote the positions of the nuclei. Here the radial component $_n^{(j)}$ ( ) satisfies

(3.3) $$ {}_n^{(j)}(\ ) = \sum_k {}_{k,n}^{(j)} \exp\ (-\ {}_{k,n}^{(j)} $$

and $_n^{\mathrm{c}}$ $_n^{(N_\mathrm{c},L_\mathrm{c})}$ denotes the interpolating function on the sphere associated with $_\mathrm{c}$ spherical quadrature nodes ω_n^{c} $\omega_n^{N_\mathrm{c}}$ and interpolation order $_\mathrm{c}$ (see Section 2.2 for details). For simplicity we assume that the number of quadrature nodes, $_\mathrm{c}$, does not depend on the singularity location $_j$ (in the examples in Section 5, $_\mathrm{c}$ = 12). In contrast to representations based on Slater-type orbitals or Gaussian-type orbitals, the number and the values of (complex-valued) exponents $_{k,n}^{(j)}$ in the radial representation (3.3) are not fixed in advance. Indeed, as described in Section 4, the exponents (and the coe□cients) are determined dynamically throughout the course of the computation to achieve a desired level of accuracy while using a near optimally small number of terms. We also note that the $_$ nuclei-centered terms that comprise the cusp part $_\mathrm{c}(\mathbf{r})$ may overlap with one another (and, in fact, do overlap in the examples in Section 5).

Similarly, we represent the smooth part $_\mathrm{s}(\mathbf{r})$ in the form

(3.4) $$ {}_\mathrm{s}(\mathbf{r}) = \sum_{n=1}^{N_\mathrm{s}} {}_n^{\mathrm{s}} \left(\frac{\mathbf{r}}{\mathbf{r}}\right) \sum_k {}_{k,n} \exp\ (-\ {}_{k,n}\ \mathbf{r}\ ) $$

where $_n^{\mathrm{s}}$ $_n^{(N_\mathrm{s},L_\mathrm{s})}$ denotes the interpolating function on the sphere associated with $_\mathrm{s}$ spherical quadrature nodes ω_n^{s} $\omega_n^{N_\mathrm{s}}$ and interpolation order $_\mathrm{s}$. The exponents and coe□cients are again determined dynamically f tff)rls de

In an analogous way, the smooth part $_{\mathrm{s}}(\mathbf{r})$ can also be expressed in terms of spherical harmonics,

$$(3.6)\qquad {}_{\mathrm{s}}(\mathbf{r}) = \sum_{l=0}^{L_{\mathrm{s}}} \sum_{m=-l}^{l} {}_{l}^{m}(\omega) \sum_{k} {}_{k,lm} \exp(-\gamma_{k,lm} \mathbf{r})$$

The representations (3.2) and (3.4) based on spherical interpolating functions are convenient for algorithms such as multiplication (see Sections 4.1 and 4.2), while the forms (3.5) and (3.6) are convenient for applying convolution operators (see Section 4.3).

3.2. Conv rt n b tw n nt rpo at n an sp r ca r pr s ntat ons We make use of two basic forms for representing $_{\mathrm{c}}(\mathbf{r})$ and $_{\mathrm{s}}(\mathbf{r})$ and now show how to convert between these two representations. It su□ces to consider

$$(3.7)\qquad (\mathbf{r}) = \sum_{l=0}^{L} \sum_{m=-l}^{l} {}_{l}^{m}(\omega)\ {}_{lm}(\ ) \qquad {}_{lm}(\ ) = \sum_{k} {}_{k,lm} \exp(-\gamma_{k,lm}\ )$$

and

$$(3.8)\qquad (\mathbf{r}) = \sum_{n=1}^{N} {}_{n}^{(N,L)}(\omega)\ {}_{n}(\ ) \qquad {}_{n}(\ ) = \sum_{k} {}_{k,n} \exp(-\gamma_{k,n}\ )$$

We make use of the fact that the interpolating function $_{n}^{(N,L)}(\omega)$ can be written in the form

$$(3.9)\qquad {}_{n}^{(N,L)}(\omega) = {}_{n}^{N} \sum_{l=0}^{L} \sum_{m=-l}^{l} {}_{l}^{m,*}\ \omega_n^N\ {}_{l}^{m}(\omega)$$

where ω_n^N and $_{n}^{N}$ denote the th node and weight associated with the -point quadrature rule (see Section 2.2).

In order to convert from (3.7) to (3.8), we have

$$\begin{aligned} {}_{lm}(\ ) &= \int_{\mathbb{S}^2} \sum_{n=1}^{N} {}_{n}^{(N,L)}(\omega)\ {}_{n}(\ )\ {}_{l}^{m,*}(\omega)\, d\Omega \\ &= \sum_{n=1}^{N} {}_{n}^{(N,L)}(\omega)\ {}_{l}^{m,*}(\omega)\, d\Omega\ {}_{n}(\ ) \\ &= \sum_{n=1}^{N} {}_{n}^{N}\ {}_{l}^{m,*}\ \omega_n^N\ {}_{n}(\ ) \end{aligned}$$

where the last equality uses formula (3.9). Thus, to convert from a spherical harmonic representation to an interpolating representation, it su□ces to use Algorithm 1 and the above formula for evaluating $_{lm}(\ )$.

Similarly, to convert from (3.8) to (3.7), we use

$${}_{n}(\ ) = \omega_n^N = \sum_{l=0}^{L} \sum_{m=-l}^{l} {}_{l}^{m}\ \omega_n^N\ {}_{lm}(\ )$$

Hence, it su□ces to use Algorithm 1 and the above formula for evaluating $_{n}(\ )$.

4.1. Mu t p cat on o r pr s ntat ons **Let us consider two functions** (r) = c(r) + s(r) **and** (r) = c(r) + s(r) **in the form (3.1) in Section 3.1. We present an algorithm to construct the same type of representation for the product,**

$$(\mathbf{r})\ \ (\mathbf{r}) = \tilde{}\ _{\mathrm{c}}(\mathbf{r}) + \tilde{}\ _{\mathrm{s}}(\mathbf{r}) + \mathcal{O}(\ )$$

where is the desired approximation error.

The basic idea behind the algorithm is that, in a neighborhood of the singularity at $_j$**, the product** (r) (r) **has the form**

$$(\ \omega + \ _j)\ \ (\ \omega + \ _j) = \sum_{l=0}^{L_{\mathrm{c}}} \sum_{m=-l}^{l} {}^{(j)}_{lm}(\ )\ _{lm}(\omega) + \ ^{L_{\mathrm{c}}+1}\ \tilde{}\ _{L_{\mathrm{c}}}(\ \omega)$$

where $^{(j)}_{lm}(\ )$ **is a polynomial and** $\tilde{}\ _{L_{\mathrm{c}}}(\ \omega)$ **is continuous. Therefore, the di erence**

$$(\ \omega + \ _j)\ \ \mathbf{0}\mathbf{0}($$

Here we can choose the cuto to be small, e.g. = min$_{k\neq}$

The advantage of maintaining this intermediate form is twofold. First, by incorporating the 1 -type singularities explicitly in the representation of $_{c}(r)$, g1.66.293(o)-5.r

where

$$\phi_j(\omega_n^c + \delta\omega_j) = \frac{1}{\sum_k \alpha_{k,n}^{(j)} \exp\left(-\beta_{k,n}^{(j)}\right) + \sum_n \alpha_{k,n}^{(j)} \exp\left(-\beta_{k,n}^{(j)}\right)}$$

Then extending $\phi_j(\mathbf{r})$ by interpolation,

$$\phi_j(\omega + \delta\omega_j) = \sum_{n=1}^{N_c} \chi_n^c(\omega) \sum_k \alpha_{k,n}^{(j)} \exp\left(-\beta_{k,n}^{(j)}\right) + \sum_n \alpha_{k,n}^{(j)} \exp\left(-\beta_{k,n}^{(j)}\right)$$

we arrive at the representation (4.6). The computation of the smooth part $\phi_s(\mathbf{r})$ proceeds in the same way as the computation of the smooth part of $\tilde{\rho}_s(\mathbf{r})$ of the multiplicand $\rho(\mathbf{r})$ $\phi(\mathbf{r})$ in Section 4.1.

4.3. Convolution with the Helmholtz and Poisson kernels. We first discuss how to evaluate the convolution operator,

$$(-\Delta + \mu^2)^{-1} \rho(\mathbf{r}) = \frac{1}{4\pi} \int_{\mathbb{R}^3} \frac{e^{-\mu\|\mathbf{r}-\mathbf{y}\|}}{\|\mathbf{r}-\mathbf{y}\|} \rho(\mathbf{y}) d\mathbf{y} \tag{4.8}$$

to obtain the representation in the form described in Section 3.1. We assume that $\rho(\mathbf{r})$ is already given in this form and, for now, $\mu > 0$. As is typical in pseudo-spectral methods, it is more efficient to apply this operator in the Fourier domain—in this case, the space of spherical harmonics. Thus, we first convert $\rho(\mathbf{r})$ from an interpolating representation to a spherical harmonic representation, as described in Section 3.2.

By linearity and translation invariance of $(-\Delta + \mu^2)^{-1}$, it also suffices to consider $\rho(\mathbf{r})$ in the form $\rho(\mathbf{r}) = \rho_0(r) Y_l^m(\omega)$. Now, by expanding the kernel in (4.8) via spherical harmonics, it can be shown that

$$(-\Delta + \mu^2)^{-1} (\rho_0(r) Y_l^m(\omega)) = Y_l^m(\omega) \phi_l(r)$$

where $\phi_l(r)$ is given by

$$\phi_l(r) = 2\mu \, i_l(\mu r) \int_0^r i_l(\mu s) \rho_0(s) s^2 ds + \mu i_l(\mu r) \int_r^\infty k_l(\mu s) \rho_0(s) s^2 ds \tag{4.9}$$

and $i_l(r)$ and $k_l(r)$ are defined in terms of the modified Bessel functions

$$i_l(r) = \sqrt{\frac{\pi}{2r}} I_{l+1/2}(r) \qquad k_l(r) = \sqrt{\frac{2}{\pi r}} K_{l+1/2}(r)$$

Thus, computing a representation of $(-\Delta + \mu^2)^{-1} \rho(\mathbf{r})$ simply involves using Algorithm 1 to construct an exponential representation of $\phi_l(r)$. We also note that when $\rho(\mathbf{r})$ is of the intermediate form (4.6), the only difference is that we also have integrals of the form

$$\phi_l(r) = 2\mu \, i_l(\mu r) \int_0^r i_l(\mu s) \rho_0(s) s^2 ds + \mu i_l(\mu r) \int_r^\infty k_l(\mu s) \rho_0(s) s^2 ds$$

for which we also need to construct exponential representations.

Recall that Algorithm 1 requires sampling $\phi_l(r)$ on an equispaced grid $r_n = n\delta$ (δ),

The integral between $_n$ and $_{n+1}$ can be accurately evaluated using a small number of quadrature points since $_{\sim l}(\ )$ does not oscillate for $_{\sim}$ 0 (e.g., we use 5 quadrature nodes in our examples in Section 5). Once the samples $_{\sim 1}(\ _n)$ are computed, samples of the first integral in (4.9) may be readily obtained. We note that the values of $_{\sim l}(\ )$ at the quadrature points may be computed using an interpolation table constructed o□-line. Also, by assumption the function $_0(\ )$ is represented with a near optimally small number of exponents, and is thus also inexpensive to sample. The second integral defining $_l(\ )$ may be e□ciently sampled in a similar manner.

Computing a representation of Δ^{-1} (i.e., in the case = 0) proceeds in a similar manner. In particular, we have

$$\Delta^{-1}(\ _l^m(\omega)\ _0(\ )) = \ _l^m(\omega)\ _l(\ )$$

where

$$_l(\ ) = -\frac{1}{2\ +1}\ ^{-l-1}\int_0^r\ _0(\ )\ ^{2+l}_{\sim}\ +\ ^l\int_r^\infty\ _0(\ )\ ^{-l+1}_{\sim} \tag{4.10}$$

One technical di□erence is that $_l(\ )$ decays like $_l\ ^{-l-1}$, where $_l$ depends on the exponents and coe□cients of $_0(\ )$; thus, for small$_{\sim}$, the direct application of Algorithm 1 would require a prohibitively large number of samples for a representation on the entire half line 0. However, in the quantum chemistry applications considered here, we only need to represent products of the form Δ^{-1}, where the function (r) (and, thus, the overall product) decays exponentially fast. Therefore, it su□ces to construct an exponential approximation to $_l(\ )$ only within the numerical support of (r), which is the approach taken in the examples in Section 5.

Remark 4.1. We can alternatively use the methods in [8] and [20] to construct an e□cient exponential representation of $_l(\ )$ on the entire half line 0. In fact, the slowly decaying asymptotic part $_l\ ^{-l-1}$ can be represented with a small number of decaying exponentials via the discretization of an appropriate quadrature formula [8]. Once this exponential representation for $_l\ ^{-l-1}$ is available, an exponential representation for the rapidly decaying function $_l(\ )-\ _l\ ^{-l-1}$ can be constructed using Algorithm 1.

5. Exa ple of solving Hartree-Fock eq ations for iato ic olec les

We now use the representations and algorithms in Sections 3.1 and 4 to solve the Hartree-Fock equations for several diatomic molecules.

Exa p As our first example of using the representations and algorithms discussed above, we solve the Hartree-Fock equation,

$$-\frac{1}{2}\Delta\ +\ \ -4\ \ \Delta^{-1}\ \ ^2\ \ = \tag{5.1}$$

with the potential

$$(\mathrm{r}) = \frac{^1}{\mathrm{r}-\ _1}\ +\ \frac{^2}{\mathrm{r}-\ _2}$$

As in [19, 31, 32], our basic approach involves recasting (5.1) as an integral equation which we solve via iteration. However, in contrast to [19, 31, 32], we represent (r) as described in Section 3.1 and apply algorithms described in Section 4. Since the

spatial orbital $(\mathbf{r})$ has cusp-like singularities at the nuclei $\mathbf{r} = {}_1$ and $\mathbf{r} = {}_2$, the Hartree-Fock equation provides a useful accuracy test for our approach.

Exa p **For our second example, we consider the Hartree-Fock equation for Lithium Hydride, LiH. We have**

(5.2) $$\mathcal{F}_j(\mathbf{r}) = \quad_j \quad_j(\mathbf{r}) \qquad = 1\ 2$$

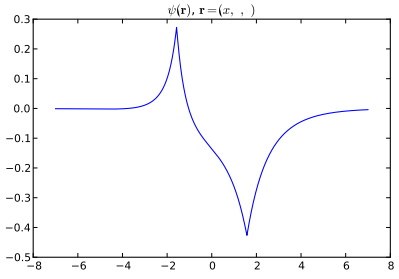
where $\mathcal{F} = -\frac{1}{2}\Delta +$ **+ 2** $-$ **,**

$$\quad_j = \quad_j \quad -4\ \Delta^{-1} \quad \quad_1{}^2 + \quad \quad_2{}^2$$

$$\quad_j = \quad_1 \quad -4\ \Delta^{-1}(\quad^*_1\ \quad_j) + \quad_2 \quad -4\ \Delta^{-1}(\quad^*_2\ \quad_j)$$

and

$$(\mathbf{r}) = \frac{\quad_1}{\mathbf{r} - \quad_1} + \frac{\quad_2}{\mathbf{r} - \quad_2}$$



(a)