

QMCkl

Anthony Scemama, Vijay Gopal Chilkur, William Jalby

1/03/2022

Lab. Chimie et Physique Quantiques, IRSAMC, UPS/CNRS, Toulouse (France)







- Application Programming Interface (API) for main algorithms of QMC
- Pedagogical (WP1) and high-performance (WP3) implementations
- API should have enough abstraction to let HPC implementations handle internal data structures, memory allocations, numerical precision, GPUs . . .
- API is expressed in C for maximum compatibility with codes
- Low level functions: linear algebra, small kernels (a.U.v)
- High-level functions: domain-specific
- In HPC implementation, everything is allowed as long as the HPC function returns the same value as the pedagogical one within the specified numerical precision.



- System functions in C (memory allocation, thread safety, *etc*)
- Computational kernels in Fortran for readability
- The API is C-compatible: QMCkI appears like a C library ⇒ can be used in all other languages
- A lot of documentation



Literate programming with *org-mode*:

- Comments are more important than code
- Can add graphics, LATEXformulas, tables, etc
- Documentation always synchronized with the code
- Some routines can be generated by embedded scripts
- Web site auto-generated when code is pushed
- Most of the first report was auto-generated from the documentation

Instead of writing comments documenting code, we write code illustrating documentation.



Literate programming with org-mode

File Edit Options Buffers Tools Table Org Text Help

Atomic Orbitals

#+SETUPFILE: ../docs/theme.setup
#+INCLUDE: ../tools/lib.org

The atomic basis set is defined as a list of shells. Each shells s is centered on a nucleus A, possesses a given angular nomentum l and a radial function R_s . The radial function is a linear combination of (sempliprimitive) functions that can be of type Slater (p=1) or Gaussian (p=2):

$$R_s(\mathbf{r}) = \mathcal{N}_s |\mathbf{r} - \mathbf{R}_A|^{n_s} \sum_{k=1}^{N_{\text{prim}}} a_{ks} \exp\left(-\gamma_{ks} |\mathbf{r} - \mathbf{R}_A|^p\right).$$

In the case of Gaussian functions, n. is always zero. The normalization factor \mathcal{N}_{c} snurses that all the functions of the shell are normalized to unity. As this normalization requires the ability to compute overlap integrals, it should be written in the file to ensure that the file is self-contained and does not require the client program to have the ability to compute such integrals.

Atomic orbitals (AOs) are defined as

 $\chi_i(\mathbf{r}) = P_{\eta(i)}(\mathbf{r}) R_{\theta(i)}(\mathbf{r})$

where $\theta(i)$ returns the shell on which the AO is expanded, and $\eta(i)$ denotes which angular function is chosen.

In this section we describe the kernels used to compute the values, gradients and Laplacian of the atomic basis functions.

		integer function gmckl_ao_gaussian_vgl_f(context, X, R, n,
Headers	:noe>	use qmckl
Context		implicit none
Polynomial part		<pre>integer*8 , intent(in) :: context real*8 , intent(in) :: X(3), R(3)</pre>
Radial part		<pre>integer*8 , intent(in) :: n</pre>
Gaussian basis functions		real*8 , intent(in) :: A(n)
		real*8 , intent(out) :: VGL(ldv,5)
~qmckl_ao_gaussian_vgl~ computes the values, gradients and		integer*8 , intent(in) :: ldv
Laplacians at a given point of ~n~ Gaussian functions centered at		
the same point:		integer*8 :: i,j
		real*8 :: Y(3), r2, t, u, v
$w = \exp(-a \cdot X - B ^2)$		
		U:9 mackl ap org 85% (1493 8) (N) Gitteoptext (Ora

I	-context- input clash istate -X(3)- input Array containing the coordinates of the points -R(3)- input Array containing the x,y,z coordinates of the center -n- input Array containing the x,y,z coordinates of the center -N- input Array containing the x,y,z coordinates of the center -N- input Array containing the x,y,z coordinates of the center -N- input Array containing the x,y,z coordinates of the center -N- input Array containing the x,y,z coordinates of the center -Vev input Array containing the x,y,z coordinates of the center
	Requirements :
	- context- is not 0 - $n > 0$ - $ldw > s s$ - $A(i) > 0$ for all -i- - $X + is allocated with at least 3 \times 8 bytes- A + is allocated with at least 3 \times 8 bytes- A + is allocated with at least n \times 8 bytes- A + is allocated with at least n \times 8 bytes$
	#+begin_src c :tangle (eval h_func)
	<pre>mckl_axit_code mckl_asit_code mckl_asit_context context,</pre>
P	#+end_src
i	#+begin.src f90 :tangle (eval f) nteger function quekl_ao_gaussian_vgl_f(context, X, R, n, A, VGL, ldv) result(info) use quekl
	<pre>implicit nome integer48, intent(in) :: context real48, intent(in) :: X(J), R(J) integer48, intent(in) :: n real48, intent(in) :: A(n) real48, intent(out) :: V(CL(Dv, 5)</pre>
	integer*8 . intent(in) :: ldv



Generated code

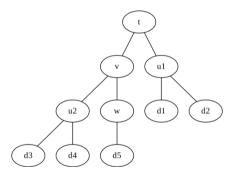
<pre>amekl_ao_f.f90 qmckl_numprec_fn_func.f90 qmckl_ao_fnc.h qmckl_ao_mprec_fnc.h qmckl_numprec_fnc.h qmckl_numprec_fnc.h qmckl_numprec_fnc.h qmckl_numprec_fnc.h qmckl_numprec_fnc.h qmckl_numprec_fnc.h qmckl_ao_r1vate_type.h qmckl_ao_r1vate_type.h qmckl_ao_r2vate_type.h qmckl_distance_f.f70 qmckl_distance_f.f70 qmckl_distance_f.f70 qmckl_distance_f.f70 test_qmckl_distance_f.f90 qmckl_distance_f.f70 test_qmckl_distance_f.f90 test_qmckl_distance_f.f90 test_qmckl_distance_f.f90 test_qmckl_distance_f.g qmckl_quarene_f.g test_qmckl_distance_f.g qmckl_quarene_f.g test_qmckl_distance_f.g qmckl_quarene_f.g test_qmckl_distance_f.g qmckl_quarene_f.g test_qmckl_distance_f.g qmckl_quarene_f.g test_qmckl_quarene_f.g qmckl_quarene_f.g test_qmckl_quarene_f.g test_qmckl_distance_f.g q qmckl_qmcd test_qmckl_qmcg q test_qmckl_qmcd test_qmckl_qmcd q qmckl_qmcd test_qmckl_qmcd q qmckl_qmcd test_qmckl_qmcd q qmckl_qmd q q qmckl_qmd q q qmckl_qmd q qmckl_qmd q q qmckl_qmd q q qmckl_qmd q q qmckl_qmd q q q qmckl_qmd q q qmckl_qmd q q q q qmckl_qmd q q q q qmckl_qmd q q q q q q qmckl_qmd q q q q q q q q q q q q q q q q q q</pre>	<pre>#define QHCKL_INVALD_CONTEXT ((qackL_exit_code) 183) #define QHCKL_INVALD_CATION_FAILED ((qackL_exit_code) 184) #define QHCKL_DEALLOCATION_FAILED ((qackL_exit_code) 184) #define QHCKL_INVALD_EXIT_CODE ((qackL_exit_code) 184) //* The context variable is a handle for the state of the library, */ /* and is stored in a data structure mich can't be seen outside of */ /* the library. To simulity compatibility with other languages, the */ /* pointer to the internal data structure is converted into a do-bit */ /* A value of -QHCKL_MULL_CONTEXT for the context is equivalent to a */ /* A value of -QHCKL_MULL_CONTEXT for the context is equivalent to a */ /* d#NAME: gackL_context ; #define QHCML_MULL_CONTEXT (qackL_context) 0 /* decide the error messages, -qackL_string_of_error- converts an */ /* error code into a string. */ /* winAME: MAX_STRING_LENGTH */ /* size /</pre>
}	const char* quckl_string_of_error(const quckl_exit_code error);
qmckl_context_struct* const ctx = (qmckl_context_struct* const) context;	void quckl_string_of_error_f(const quckl_exit_code error,
assert (ctx != NUL);	char result[128]);
int32_t mask = 1 << 4;	/* Updating errors in the context */
<pre>if (ctx->no_basis.uninitialized & mask) != 0) { return NULL } assert (ctx->ao_basis.shell_ang_mom != NULL); return ctr>ac_basis.shell_ang_mom;</pre>	/* The error is updated in the context using "gmckl_set_error". */ /* When the error is set in the context, it is mandatory to specify */ /* from which function the error is triggered, and a message */ /* explaining the error. The exit code can't be ~QMCKL_SUCCESS". */
)	/* # Header */
-/TREX/gnckl/src/gnckl_ao.c [unix] [C] [15%] (184/674,16)	#/TREX/gmckl/include/qmckl.h [unix] [CPP] [31%] (85/269,1)
1 2 3 Biblio OrgWork Web eMan us ◆+6°C ♀ 1.1K- 5658- Li	



Generated web site

Construction in the field of a serificity of the series o	Atomic Orbitals x +		Atomic Orbitals × +					
<section-header><section-header><text><text><text><equation-block><text><equation-block><text><text><text><text><text><text><text> http://> http://> \$<!--/--> <!--/ <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <// <//</td--><td>🗅 D C 🔲 🗈 tree-coe.github.io/gmckl/gmckl_ao.html 🔍 🦁 🛆 📨 🖪 B ∉</td><td>1 🕈 🕿 🎓 E</td><td>🕼 🗅 C 🛛 🗅 trex-coe.github.io/qmckl/qmckl_ao 0, 👽 🛆 🛛 🖬 B 🐇 🛦 🔶 🕿 🏟 🗉</td></text></text></text></text></text></text></text></equation-block></text></equation-block></text></text></text></section-header></section-header>	🗅 D C 🔲 🗈 tree-coe.github.io/gmckl/gmckl_ao.html 🔍 🦁 🛆 📨 🖪 B ∉	1 🕈 🕿 🎓 E	🕼 🗅 C 🛛 🗅 trex-coe.github.io/qmckl/qmckl_ao 0, 👽 🛆 🛛 🖬 B 🐇 🛦 🔶 🕿 🏟 🗉					
<text><text><text><text><text><equation-block> Atomic Orbitals under Comparison transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmission transmis</equation-block></text></text></text></text></text>	In Personal In OCaml In LCPQ In QP2 In Biblio In ToUpdate In TREX In ERC In AliDA In Micro	osoft 🖿 Padlets »	Ma Personal Ma OCaml Ma LCPQ Ma QP2 Ma Biblio Ma ToUpdate Ma TREX Ma ERC Ma AliDA Ma Microsoft »					
$ \begin{aligned} & \text{the term of the same of the left has a list of helds. Each held x is therefored on mucleus A, possesses a given angular momentum I and an dual function R, the model and the same of containing of employing interview functions that can be of the same of the same of the moleus (x - x), f = \alpha_1 \left(\frac{1}{2} X - R_1^2 \alpha_1 - \theta_1 v_1 \right)^{-1} \left(\frac{1}{2} \log \frac{1}{2$		í	$ abla_z v_i = -2a_i(X_z-R_z)v_i$					
The atomic basis set is defined as a list of shells. Each shell a is centered on a nucleus A_i possesses a given angular momentum I and a radial functions R_i . The nucleus A_i possesses a given angular momentum I and a radial functions R_i . The nucleus A_i possesses a given angular momentum I and a radial functions R_i . The nucleus A_i possesses a given angular momentum I and R_i for R_i . $ \begin{aligned} \hline R_i(r) = A_i \left(\sum_{n=1}^{\infty} q_n \exp(-\gamma_n) (r - R_i q^n) \right). \end{aligned} $ In the case of Gaussian functions n_i is always zero. The normalized to factor A_i momentum the functions of the shell are normalized to unity A takes is means that all be functions of the shell R_i constant M for R_i or R_i is a function R_i . The organized means the functions is the functions in the set of compared Gaussians More orbital to compare to explicit the shell R_i constant M_i for R_i or R_i is a function R_i . More orbital to compare to explicit the shell R_i or R_i is a function R_i . Subscience with a shell R_i is a function R_i . Ansatcal with a flast R_i is a function R_i is a function R_i . Subscience with a function R_i is a function R_i is a function R_i . Subscience with R_i is a function R_i . Subscience with R_i is a function R_i is a functi with R_i is a functi with R_i	Atomic Orbitals		$\Delta w = \alpha \left(4 Y - R ^2 \alpha - 6 \right) w$					
The atomic basis with defined as a late of the list. Each hand is a contracted on a mode we have angular momentum and a radial function R_{L} . The matrix R_{L} as a possesse at given angular momentum R_{L} and R_{L								
radial function is a linear combination of employmentively functions that can be of types later $(p = 1)$ or Gaussian $(p = 2)$: $ \frac{1}{\mu_{n}(p)} = \lambda_{n}^{2} \mathbf{r} - \mathbf{R}_{n} ^{2} \sum_{k=1}^{N_{m}} \alpha_{kk} \exp(-\gamma_{kk} \mathbf{r} - \mathbf{R}_{n} ^{2}). $ Rots that all the functions n_{k} is always zero. The normalized to unity, this is anomalized to unity, this is anomalized to unity. It is always the field is self-contained of the aussians $(N_{n}(-))$ is always zero. The normalized to unity is the differentiate of the dama set of		Table of Contents	context input Global state					
$ \begin{array}{l} \mbox{Figure functions} Figure functions$			X(3) input Array containing the coordinates of the points					
$\begin{aligned} & R_{i}(r) = \mathcal{N}_{i}[r - \mathbf{R}_{i}]^{n} \sum_{i=1}^{n} a_{is} \exp(-\gamma_{is}[r - \mathbf{R}_{i}]^{n}). \\ & \text{A (n)} \qquad \text{input } \text{Exponents of the Gaussians} \\ & \text{resources that all the functions, } \\ & \text{is a lawys zero. The normalization factor \mathcal{N}_{i} \\ & \text{resources that all the functions of the shall are normalized to unity. At this normalization requires the shall be interview of the functions of the shall are normalized to unity. At this normalization requires the shall be interview of the functions of the shall be soft-contained and does not require the cherring regress that wheth shall be soft-contained and does not require the cherring regress that wheth the soft-contained and does not require the cherring regress that wheth shall be soft-contained and does not require the cherring regress that wheth the soft-contained and m(r) denotes which angular function is chosen. \\ \text{The the section wheth the AO is expanded, and m(r) denotes which angular function is chosen. \\ \text{I.Polynomial part} \\ \text{I.1.Powers of } x = X_{i} \\ \text{The querk } _{ao} \text{ power function computes all the powers of the n input data up } \\ \text{The querk } _{ao} \text{ power function computes all the powers of the n input data up } \\ \text{The querk } _{ao} \text{ power function computes all the powers of the n input data up } \\ \text{The querk } _{ao} \text{ power function computes all the powers of the n input data up } \\ \text{The querk } _{ao} \text{ power function computes all the powers of the n input data up } \\ \text{The querk } _{ao} \text{ power function computes all the powers of the n input data up } \\ \text{The querk } _{ao} \text{ power function computes all the powers of the n input data up } \\ \text{The querk } _{ao} \text{ power function computes all the powers of the n input data up } \\ \text{The querk } _{ao} \text{ power function computes all the powers of the n input data up } \\ \text{The querk } _{ao} \text{ power function computes all the powers of the n input data up } \\ \text{The set of the set of the set of the normalized then input data up } \\ The set of the set of the data up and then t$			R(3) input Array containing the x,y,z coordinates of the center					
VectorVect	Naria		n input Number of computed Gaussians					
VectorVect	$R_s(\mathbf{r}) = \mathcal{N}_s \mathbf{r} - \mathbf{R}_A ^{n_s} \sum_{i=1}^{n_s} a_{ks} \exp(-\gamma_{ks} \mathbf{r} - \mathbf{R}_A ^p).$		A(n) input Exponents of the Gaussians					
instruction that the function of the shall are normalized to unity. At this normalization requires the shall be normalized not unity. At this normalization requires the shall be normalized not unity. At this normalization requires the shall be normalized not does not require the client program to have the shall by to compute shall the growt such that the shall nor which the AO is expanded, and $\eta(t)$ denotes which angular function is closent.Requirements $\chi(t) = P_{00}(t) R_{00}(t)$ $0 < 0 < 0 < 0 < 0 < 0 < 0 < 0 < 0 < 0 <$	x=1		VGL(ldv,5) output Value, gradients and Laplacian of the Gaussians					
normalization requires the ability to compute overlap integrals, it should be written in the file rooms with the file is set to ensure the the file set for the file file on set to integrals. Requirements: Atomic obtaits (Ao) are defined as \cdot context is not 0 $\chi(r) = P_{00}(r) R_{00}(r)$ \cdot defined as $\chi(r) = r_{00}(r) R_{00}(r)$ \cdot defined as $\chi(r) = r_{00}(r) R_{00}(r)$ \cdot defined as in this section we describe the karnels used to compute the values, gradients and Laplace for the atomic basis functions. \cdot is allocated with at least $n \times 5$ bytes I Polynomial part \cdot cont totals \cdot so bytes \cdot so bytes i.1.1 Powers of $x - X_i$ \cdot cont in the is, trans, the file powers of the n input data up \cdot cont total \cdot so, cont in tries, t tow, cont integrals, t tow, cont integr			ldv input Leading dimension of array VGL					
written in the file to ensure that the file is self-contained and does not require the distry groups to have the ability to compute with integrals. context is not 0 ld $v \to 5$ A(1) > 0 for all 1 A(1) > 0 for all 1			Reminaments :					
Atomic orbitals (AO3) are defined as • $n > 0$ $\chi_1(r) = P_{ij(1)}(r) R_{ij(1)}(r)$ • $l(y) > 5$ where $\theta(i)$ returns the shell on which the AO is expanded, and $\eta(i)$ denotes which angular function is toksen. • $n > 0$ In this section we describe the barnels used to compute the values, gradients and Laplacian of the atomic basis functions. • $n > 0$ 1 Polynomial part • $n > 0$ 1.1 Power's of $x - X_i$ • $n > 0$ The gatek1_ao gover function computes all the powers of the n input data up • $n > 0$								
Atomic orbitals (Ao) are defined as $\chi_i(r) = P_{ij(i)}(r) R_{iij(i)}(r) \chi_{ij(i)}(r) \chi_$	client program to have the ability to compute such integrals.							
$\chi_0(r) = R_{00}(r) R_{00}(r)$ • A(1) > 0 for all 1 where $\theta(t)$ returns the shall on which the A0 is expanded, and $\eta(t)$ denotes which angular function is closes. • X is allocated with at least 3 × 8 bytes In this action we describe the kernels used to compute the values, gradients and Laplacian of the atomic basis functions. • A is allocated with at least 3 × 8 bytes I. Polynomial part • Model is allocated with at least 1 × 8 bytes 1.1 Powers of $x - X_i$ • compute sall the powers of the n input data up	Atomic orbitals (AOs) are defined as							
where $\theta(i)$ returns the shell on which the AO is expanded, and $\eta(i)$ denotes which angular function is chosen. X is allocated with at least 3×8 bytes In this section we describe the kernels used to compute the values, gradients and Laplacian of the atomic basis functions. X is allocated with at least 3×8 bytes 1 Polynomial part VGL is allocated with at least $n \times 5 \times 8$ bytes 1.1 Power's of $x - X_i$ const double "x, const double "x, const double "x, const double "x, const double "x, const times,t "x, const ti	$\chi_i(\mathbf{r}) = P_{\eta(i)}(\mathbf{r}) R_{\theta(i)}(\mathbf{r})$							
angular function is chosen. • R is allocated with at least 3 × 8 bytes In this section we describe the kernels used to compute the values, gradients and Laplacian of the atomic basis functions. • R is allocated with at least n × 8 bytes I Polynomial part • We is allocated with at least n × 5 bytes 1.1 Powers of x - X _i • Const double **, const	where $\theta(i)$ returns the shell on which the AO is expanded, and $\eta(i)$ denotes which							
Laplacian of the atomic basis functions. • VGL is allocated with at least $n \times 5 \times 8$ bytes 1 Polynomial part qickl_ssile_code qickl_ssile_code qickl_ssile_code qickl_ssile_code qickl_ssile_code qickl_ssile_code qickl_ssile_code qickl_ssile_tode qicdit								
1 Polynomial part qckl_esit_code qcd qckl_esit_code qcd qckl_esit_code qcd qckl_esit_code qcd qckl_esit_code qcd	In this section we describe the kernels used to compute the values, gradients and		- A is allocated with at least $n imes 8$ bytes					
1 Polynomial part qmtl_so_gaustim_vgl(cont qutk)_context context, 1.1 Powers of x - X _i cont double **, 1.1 powers of x - X _i cont double **, The qmtkl_so_power function computes all the powers of the n input data up cont initial_t *a,	Laplacian of the atomic basis functions.		- VGL is allocated with at least $n imes 5 imes 8$ bytes					
1.Polynomial part qmtl_so_gaustian_vgl(cont qudk)_context context, cont dudke **, cont dudke **, cont dudke **, cont tinted_t **, cont tinted_t **, cont tinted_t **, cont tinted_t **, 1.Powers of $x - X_i$ cont tinted_t **, cont tinted_t **, cont tinted_t **, The gackLae_power function computes all the powers of the n input data up cont tinted_t **, cont tinted_t **,								
$(a) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{i=1}^{n} \sum_{j=1}^{n$	1 Polynomial part							
$\begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \begin{array}{c} \\ \end{array} \\ 1.1 \mbox{ Powers of } x-X_i \end{array} \end{array} \\ \begin{array}{c} \begin{array}{c} \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \begin{array}{c} \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \end{array} \\ \begin{array}{c} \end{array} \\ \end{array} $	21 olyhoma par							
const int64_t *A, The qmckl_ao_power function computes all the powers of the n input data up const double *VGL,								
The gmckl_ao_power function computes all the powers of the n input data up const double *VGL,	1.1 Powers of $x - X_i$							
	The omckl as power function computes all the powers of the n input data up							
1 1 2 3 Biblio OrgWork Web @M5ii us ♦ +6*C ≑ 1.24- 7478- LinksysRouter 81% 8 59*C ≜ /:216 ▮ 188% ⊞ 8 ¥ 82% 15 04/19 10:11 🖷			·					

The quantities of interest are organized in a dependency graph:



• The user asks for v. w and u_2 were computed.

Dependency graph

- If the user asks for t, v is re-used and only u₁ is computed.
- The sub-trees are independent, so they can be computed in parallel



• Each quantity is expressed as a function of other quantities:

 $\chi_i(\mathbf{r}) = P_{\eta(i)}(\mathbf{r}) R_{\theta(i)}(\mathbf{r})$

Here, χ depends on P and \mathcal{R} .

- The context is a data structure containing the state of the library (equivalent to global variables).
- It contains all the computed quantities that may be re-used
- It is passed to all the functions
- The date is an integer which increments each time the electron coordinates change.
- The quantities are recomputed only if their date is older than the current date, like in a Makefile



```
qmckl_exit_code provide_chi(qmckl_context* ctx) {
 gmckl_exit_code rc;
  if (ctx->date > ctx->date chi) {
   rc = provide_p(ctx); // asserts that ctx->p is computed
    if (rc != QMCKL_SUCCESS) return rc;
   rc = provide_r(ctx); // asserts that ctx \rightarrow r is computed
    if (rc != QMCKL_SUCCESS) return rc;
    ctx->chi = compute_chi(ctx->p, ctx->r);
    ctx->chi_date = ctx->date:
  ን
 return QMCKL_SUCCESS;
ን
```



3

qmckl_exit_code qmckl_get_chi(qmckl_context ctx, double* chi, int64_t size_max) {
 // First, check context is valid, and that size_max is large enough. Then,

```
qmckl_exit_code rc = provide_chi(ctx);
if (rc != QMCKL_SUCCESS) return rc;
memcpy(chi, ctx->chi, ctx->chi_size);
return QMCKL_SUCCESS;
```

Note: the memcpy can be avoided using qmckl_get_chi_inplace, to be used with caution.



All QMCkl functions return an error code. A convenient way to handle errors is to write an error-checking function that displays the error in text format and exits the program.

```
subroutine qmckl_check_error(rc, message)
 use gmckl
 implicit none
 integer(qmckl_exit_code), intent(in) :: rc
 character(len=*) , intent(in) :: message
 character(len=128)
                                      :: str_buffer
 if (rc /= QMCKL_SUCCESS) then
    print *, message
     call qmckl_string_of_error(rc, str_buffer)
    print *. str_buffer
     call exit(rc)
 end if
end subroutine qmckl_check_error
```



- The user gives input parameters to the library to initialize the context
- The arrays can be given one by one, but the easy way is to read a TREXIO file:

```
use amckl
integer(qmckl_context) :: qmckl_ctx
integer(qmckl_exit_code) :: rc
double precision
                         :: qmckl_ao_vgl(ao_num,5,elec_num, walk_num)
qmckl_ctx = qmckl_context_create()
rc = qmckl_trexio_read(qmckl_ctx, trexio_filename, len(trim(trexio_filename)))
call gmckl_check_error(rc, 'Read TREXIO')
rc = gmckl_set_electron_walk_num(gmckl_ctx, walk_num)
call gmckl_check_error(rc, 'Set walk_num'))
rc = amckl set electron coord(amckl ctx, 'N', elec coord, size(elec coord)) ! Increments date
call gmckl_check_error(rc, 'Set elec_coord'))
rc = qmckl_get_ao_basis_ao_vgl(qmckl_ctx, qmckl_ao_vgl, size(qmckl_ao_vgl)) ! Commputes AOs
```

```
call qmckl_check_error(rc, 'Get ao_vgl'))
```



\$ ao_grid <trexio_file> <AO_id> <point_num>

Start by fetching the command-line arguments:

```
if (iargc() /= 3) then
   print *. 'Syntax: ao_grid <trexio_file> <AO_id> <point_num>'
   call exit(-1)
end if
call getarg(1, trexio_filename)
call getarg(2, str_buffer)
read(str_buffer, *) ao_id
call getarg(3, str_buffer)
read(str_buffer, *) point_num_x
if (point_num_x < 0 .or. point_num_x > 300) then
   print *, 'Error: 0 < point_num < 300'</pre>
   call exit(-1)
end if
```



Create the QMCkl context and initialize it with the wave function present in the TREXIO file:

```
qmckl_ctx = qmckl_context_create()
rc = qmckl_trexio_read(qmckl_ctx, trexio_filename, 1_8*len(trim(trexio_filename)))
call qmckl_check_error(rc, 'Read TREXIO')
```

We need to check that ao_id is in the range, so we get the total number of AOs from QMCkI:

```
rc = qmckl_get_ao_basis_ao_num(qmckl_ctx, ao_num)
call qmckl_check_error(rc, 'Getting ao_num')
if (ao_id < 0 .or. ao_id > ao_num) then
    print *, 'Error: 0 < ao_id < ', ao_num
    call exit(-1)
end if</pre>
```



Compute the limits of the box. For that, we first need to ask QMCkl the coordinates of nuclei.

```
rc = qmckl_get_nucleus_num(qmckl_ctx, nucl_num)
call qmckl_check_error(rc, 'Get nucleus num')
```

```
allocate( nucl_coord(3, nucl_num) )
rc = qmckl_get_nucleus_coord(qmckl_ctx, 'N', nucl_coord, 3_8*nucl_num)
call qmckl_check_error(rc, 'Get nucleus coord')
```

We now compute the coordinates of opposite points of the box:

```
rmin(1) = minval( nucl_coord(1,:) ) - 5.d0
rmin(2) = minval( nucl_coord(2,:) ) - 5.d0
rmin(3) = minval( nucl_coord(3,:) ) - 5.d0
```

```
rmax(1) = maxval( nucl_coord(1,:) ) + 5.d0
rmax(2) = maxval( nucl_coord(2,:) ) + 5.d0
rmax(3) = maxval( nucl_coord(3,:) ) + 5.d0
```

dr(1:3) = (rmax(1:3) - rmin(1:3)) / dble(point_num_x-1)



We produce the list of point coordinates where the AO will be evaluated:

```
point_num = point_num_x**3
allocate( points(point_num, 3) )
ipoint=0
z = rmin(3)
do k=1,point_num_x
   \mathbf{v} = \mathbf{rmin}(2)
   do j=1,point_num_x
       x = rmin(1)
       do i=1,point_num_x
           ipoint = ipoint+1
           points(ipoint,1) = x
           points(ipoint, 2) = y
           points(ipoint,3) = z
           \mathbf{x} = \mathbf{x} + \mathbf{dr}(1)
       end do
       \mathbf{v} = \mathbf{v} + \mathbf{dr}(2)
   end do
   z = z + dr(3)
end do
```



We give the points to QMCkl:

```
rc = qmckl_set_point(qmckl_ctx, 'T', points, point_num)
call qmckl_check_error(rc, 'Setting points')
```

We allocate the space required to retrieve the values, gradients and Laplacian of all AOs, and ask the data to QMCkI:

```
allocate( ao_vgl(ao_num, 5, point_num) )
rc = qmckl_get_ao_basis_ao_vgl(qmckl_ctx, ao_vgl, ao_num*5_8*point_num)
call qmckl_check_error(rc, 'Setting points')
```

We finally print the value of the AO:

```
do ipoint=1, point_num
    print '(3(F16.10,X),E20.10)', points(ipoint, 1:3), ao_vgl(ao_id,1,ipoint)
end do
```



Loop-based: Splits loops in *m* chunks and distributes chunks in different threads.

!\$OMP PARALLEL DO
do i=1,N
...
end do
!\$OMP END PARALLEL DO

Advantages

- Very low scheduling overhead
- Control of memory locality
- Easy to write and to think about
- Difficulties
 - Not optimal for inhomogeneous workloads
 - Limited to the scope of the loop



Task based Splits work to do into independent tasks, adds them to a queue and lets the threads perform these tasks

```
void provide_chi(qmckl_context* ctx) {
    if (ctx->date > ctx->date_chi) {
        #pragma omp task
        provide_p(ctx);
        #pragma omp task
        provide_r(ctx);
        #pragma omp taskwait
        ctx->chi = compute_chi(ctx->p, ctx->r);
        ctx->chi_date = ctx->date;
    }
}
```

- Advantages
 - Better load balancing
 - Tasks can be distributed on CPUs and GPUs
 - Dependencies between tasks can be expressed
- Difficulties
 - Scheduling tasks requires some CPU power ⇒ tasks need enough work to do

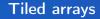




Usual storage: double precision :: A(8,8)

1	9	17	25	33	41	49	57
2	10	18	26	34	42	50	58
3	11	19	27	35	43	51	59
4	12	20	28	36	44	52	60
5	13	21	29	37	45	53	61
6	14	22	30	38	46	54	62
7	15	23	31	39	47	55	63
8	16	24	32	40	48	56	64





Tiled storage: double precision :: A(4,4,4,4)

1	5	9	13	33	37	41	45
2	6	10	14	34	38	42	46
3	7	11	15	35	39	43	47
4	8	12	16	36	40	44	48
17	21	25	29	49	53	57	61
18	22	26	30	50	54	58	62
19	13	27	31	51	55	59	63
20	24	28	32	52	56	60	64



Advantages

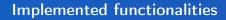
- Much Better locality in memory: sizes calibrated to cache sizes
- Performance of matrix multiplication is constant
- Fast transposition
- Blocks can be directly sent to GPUs: no problem with leading dimension
- Computation time of tasks is easy to estimate

Disadvantages

- Code more difficult to read/write: more nested loops
- Mapping between math and code is not simple
- Random access is very inconvenient
- Not naturally adapted to external libraries and codes



- In QMC, we manipulate small matrices
- Compilers with -03 usually generate efficient code for large data sets
- Optimized BLAS libraries work with 2D arrays, which is not necessarily the most efficient pattern
- Matrices in QMCkl will be internally tiled: needs efficient linear algebra for small (sub-)matrices
- We implement low-level functions using x86 or ARM assembly:
 - Performance is independent of the compiler
 - Implementation is specific to tiled arrays
- One generic kernel produces multiple assembly versions with a code generator





Pedagogical only

- MOs
- Potential (ee, eN, NN)
- Inverse Slater matrix

Optimized high-level

- AOs
- Jastrow
- Sherman Morrison
- Adjugate : $B = adj(A) = det(A) A^{-1}$

Optimized low-level functions for tildes matrices

- Matrix multiplication: heavily used in Jastrow
- Rank-k update: necessary for Sherman-Morrison
- Matrix-vector multiplication:

•
$$u^{\dagger}.A.v$$
: necessary for $\Psi = \sum_{ij} C_{ij} D_i^{\dagger} D_j^{\downarrow}$



$$J_{\text{een}}(\mathbf{r}, \mathsf{R}) = \sum_{\alpha=1}^{N_{\text{nucl}}} \sum_{i=1}^{N_{\text{elec}}} \sum_{j=1}^{i-1} \sum_{p=2}^{N_{\text{nord}}} \sum_{k=0}^{p-1} \sum_{l=0}^{p-k-2\delta_{k,0}} c_{lkp\alpha} (r_{ij})^k \left[(R_{i\alpha})^l + (R_{j\alpha})^l \right] (R_{i\alpha} R_{j\alpha})^{(p-k-l)/2}$$

can be rewritten as

$$J_{\text{een}}(\mathbf{r}, \mathsf{R}) = \sum_{p=2}^{N_{\text{nord}}} \sum_{k=0}^{p-1} \sum_{l=0}^{p-k-2\delta_{k,0}} \sum_{\alpha=1}^{N_{\text{nucl}}} c_{lkp\alpha} \sum_{i=1}^{N_{\text{elec}}} \bar{\mathsf{R}}_{i,\alpha,(p-k-l)/2} \bar{\mathsf{P}}_{i,\alpha,k,(p-k+l)/2} (\downarrow \text{ complexity})$$

with

$$ar{\mathtt{P}}_{i,lpha,k,l} = \sum_{j=1}^{N_{\mathsf{elec}}} ar{\mathtt{r}}_{i,j,k} \ ar{\mathtt{R}}_{j,lpha,l}.$$
 (GEMM)

Jastrow factor



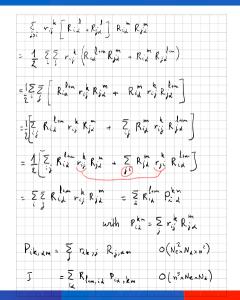
$$\nabla_{im} J_{een}(\mathbf{r}, \mathbf{R}) = \sum_{p=2}^{N_{nord}} \sum_{k=0}^{p-1} \sum_{l=0}^{p-k-2\delta_{k,0}} \sum_{\alpha=1}^{N_{nucl}} c_{lkp\alpha} \sum_{i=1}^{N_{elec}} \bar{\mathbf{G}}_{i,m,\alpha,(p-k-l)/2} \bar{\mathbf{P}}_{i,\alpha,k,(p-k+l)/2} + \\ \bar{\mathbf{G}}_{i,m,\alpha,(p-k+l)/2} \bar{\mathbf{P}}_{i,\alpha,k,(p-k-l)/2} + \bar{\mathbf{R}}_{i,\alpha,(p-k-l)/2} \bar{\mathbf{Q}}_{i,m,\alpha,k,(p-k+l)/2} + \\ \bar{\mathbf{R}}_{i,\alpha,(p-k+l)/2} \bar{\mathbf{Q}}_{i,m,\alpha,k,(p-k-l)/2} + \delta_{m,4} (\\ \bar{\mathbf{G}}_{i,1,\alpha,(p-k+l)/2} \bar{\mathbf{Q}}_{i,1,\alpha,k,(p-k-l)/2} + \bar{\mathbf{G}}_{i,2,\alpha,(p-k+l)/2} \bar{\mathbf{Q}}_{i,2,\alpha,k,(p-k-l)/2} + \\ \bar{\mathbf{G}}_{i,3,\alpha,(p-k+l)/2} \bar{\mathbf{Q}}_{i,3,\alpha,k,(p-k-l)/2} + \bar{\mathbf{G}}_{i,1,\alpha,(p-k-l)/2} \bar{\mathbf{Q}}_{i,1,\alpha,k,(p-k+l)/2} + \\ \bar{\mathbf{G}}_{i,2,\alpha,(p-k-l)/2} \bar{\mathbf{Q}}_{i,2,\alpha,k,(p-k+l)/2} + \bar{\mathbf{G}}_{i,3,\alpha,(p-k-l)/2} \bar{\mathbf{Q}}_{i,3,\alpha,k,(p-k+l)/2})$$

with

$$\bar{\mathsf{G}}_{i,m,\alpha,l} = \frac{\partial \left(R_{i\alpha}\right)^{l}}{\partial r_{i}}, \qquad \bar{\mathsf{g}}_{i,m,j,k} = \frac{\partial \left(r_{ij}\right)^{k}}{\partial r_{i}}, \qquad \text{and} \ \bar{\mathsf{Q}}_{i,m,\alpha,k,l} = \sum_{j=1}^{N_{\mathsf{elec}}} \bar{\mathsf{g}}_{i,m,j,k} \ \bar{\mathsf{R}}_{j,\alpha,l}$$

Jastrow factor









$$R_{s}(\mathbf{r}) = \mathcal{N}_{s}|\mathbf{r} - \mathsf{R}_{A}|^{n_{s}} \sum_{k=1}^{N_{\text{prim}}} a_{ks} f_{ks} \exp\left(-\gamma_{ks}|\mathbf{r} - \mathsf{R}_{A}|^{p}\right).$$
$$\chi_{i}(\mathbf{r}) = \mathcal{M}_{i} P_{\eta(i)}(\mathbf{r}) R_{\theta(i)}(\mathbf{r})$$

$R_{\theta(i)}(r)$: Radial part

- For each nucleus, beyond a given e-N distance all exponentials are zero
- We call the exp function only if the argument is small enough: $\gamma_{ks} |\mathbf{r} \mathbf{R}_A|^2 \leq -\log(10^{-12})$
- The same radial part is reused for multiple AOs (p_x, p_y, p_z)





$$R_{s}(\mathbf{r}) = \mathcal{N}_{s}|\mathbf{r} - \mathsf{R}_{A}|^{n_{s}} \sum_{k=1}^{N_{\text{prim}}} a_{ks} f_{ks} \exp\left(-\gamma_{ks}|\mathbf{r} - \mathsf{R}_{A}|^{p}\right).$$
$$\chi_{i}(\mathbf{r}) = \mathcal{M}_{i} P_{\eta(i)}(\mathbf{r}) R_{\theta(i)}(\mathbf{r})$$

$P_{\eta(i)}(r)$: Polynomial part

- For each nucleus, we know the max angular momentum $I_{\rm max}$
- We compute all the powers of x, y, z up to I_{max} by successive multiplications, and their 1st and 2nd derivatives
- These values are re-used for all AOs





$$R_{s}(\mathbf{r}) = \mathcal{N}_{s}|\mathbf{r} - \mathsf{R}_{A}|^{n_{s}} \sum_{k=1}^{N_{prim}} a_{ks} f_{ks} \exp\left(-\gamma_{ks}|\mathbf{r} - \mathsf{R}_{A}|^{p}\right).$$
$$\chi_{i}(\mathbf{r}) = \mathcal{M}_{i} P_{\eta(i)}(\mathbf{r}) R_{\theta(i)}(\mathbf{r})$$

Combining *R* and *P*

- Very low arithmetic intensity
- P and R are computed together for each atom (P) and each shell (R) to reduce memory operations





$$R_s(\mathbf{r}) = \mathcal{N}_s |\mathbf{r} - \mathsf{R}_A|^{n_s} \sum_{k=1}^{N_{\mathsf{prim}}} a_{ks} f_{ks} \exp\left(-\gamma_{ks} |\mathbf{r} - \mathsf{R}_A|^p\right).$$

Work in progress

- Sort arrays of exponents γ_{ks} in increasing order per atom. Identify duplicates that can occur between shells. We know that after indice k_{max} all exponents are zero, so we can vectorize the computation of all exponentials centered on A.
- Express the coefficients in the *generally contracted* format (as in MOLCAS):

$$\gamma = \begin{bmatrix} 0.158\\ 0.502\\ 1.792\\ 7.903\\ 52.56 \end{bmatrix} e_k = e^{\gamma_k |r - R_A|^2} \mathbf{A} = \begin{bmatrix} 0.0 & 0.0 & 1.0\\ 0.0 & 1.0 & 0.0\\ 0.852933 & 0.0 & 0.0\\ 0.189684 & 0.0 & 0.0\\ 0.025374 & 0.0 & 0.0 \end{bmatrix} \mathbf{R} = \mathbf{A}^{\dagger} e$$



Introduce low-level functions

Next steps

- Introduce tiled arrays
- AOs into sparse format
- AOs in spherical coordinates
- $\blacksquare \text{ Sparse AO} \rightarrow \text{MO transformation}$
- Python interface for prototyping
- Multi-determinant wave functions



Pseudopotentials

- Should we pre-compute $\langle Y_{lm} | \phi(r) \rangle$ of the determinantal component and store it in TREXIO?
- I have no experience in programming pseudos with quadratures: I don't know how to do it efficiently

Periodic systems

I have no experience with periodic systems. What changes between isolated system and periodic?

Possible strategy

- Write a latex file with all formulas and detailed explanations
- It is the most difficult and time-consuming part of writing code in QMCkl. Writing code is faster than writing documentation.