

# Program Reference

## Contents

<b>Overview of libcint usage</b>	<b>1</b>
Preparing args . . . . .	1
<b>Interface</b>	<b>1</b>
C routine . . . . .	1
Fortran routine . . . . .	3
Supported angular momentum . . . . .	4
Data ordering . . . . .	4
Tensor . . . . .	5
<b>Built-in function list</b>	<b>6</b>

## Overview of libcint usage

### Preparing args

...

## Interface

### C routine

```
dim = CINTgto_cart(bas_id, bas);
dim = CINTgto_spheric(bas_id, bas);
dim = CINTgto_spinor(bas_id, bas);
f1e(buf, shls, atm, natm, bas, nbas, env);
f2e(buf, shls, atm, natm, bas, nbas, env, opt);
f2e_optimizer(&opt, atm, natm, bas, nbas, env);
CINTdel_optimizer(&opt);
```

- buf: column-major double precision array.
  - for 1e integrals of shells (i,j), data are stored as [i1j1 i2j1 ... ]
  - for 2e integrals of shells (i,j|k,l), data are stored as  
[i1j1k1l1 i2j1k1l1 ... i1j2k1l1 ... i1j1k2l1 ... ]

- complex data are stored as two double elements, first is real, followed by imaginary, e.g. [Re Im Re Im ...]
  - shls: 0-based basis/shell indices.
    - int[2] for 1e integrals
    - int[4] for 2e integrals
  - atm: int[natm\*6], list of atoms. For ith atom, the 6 slots of atm[i] are
    - atm[i\*6+0] nuclear charge of atom i
    - atm[i\*6+1] env offset to save coordinates (env[atm[i\*6+1]], env[atm[i\*6+1]+1], env[atm[i\*6+1]+2]) are (x,y,z)
    - atm[i\*6+2] nuclear model of atom i, = 2 indicates gaussian nuclear model  $\rho(r) = Z(\frac{\zeta}{\pi})^{3/2} \exp(-\zeta r^2)$
    - atm[i\*6+3] env offset to save the nuclear charge distribution parameter  $\zeta$
    - atm[i\*6+4] unused
    - atm[i\*6+5] unused
  - natm: int, number of atoms, natm has no effect **except nuclear attraction** integrals
  - bas: int[nbas\*8], list of basis. For ith basis, the 8 slots of bas[i] are
    - bas[i\*8+0] 0-based index of corresponding atom
    - bas[i\*8+1] angular momentum
    - bas[i\*8+2] number of primitive GTO in basis i
    - bas[i\*8+3] number of contracted GTO in basis i
    - bas[i\*8+4] kappa for spinor GTO.
      - < 0 the basis  $\sim j = l + 1/2$ .
      - > 0 the basis  $\sim j = l - 1/2$ .
      - = 0 the basis includes both  $j = l + 1/2$  and  $j = l - 1/2$
    - bas[i\*8+5] env offset to save exponents of primitive GTOs. e.g. 10 exponents env[bas[i\*8+5]] ... env[bas[i\*8+5]+9]
    - bas[i\*8+6] env offset to save column-major contraction coefficients. e.g. 10 primitive -> 5 contraction needs a  $10 \times 5$  array
- ```

env[bas[i*8+6] ] | env[bas[i*8+6]+10] |      | env[bas[i*8+6]+40]
env[bas[i*8+6]+1] | env[bas[i*8+6]+11] |      | env[bas[i*8+6]+41]
.                | .                  | ... | .
.                | .                  |      | .
env[bas[i*8+6]+9] | env[bas[i*8+6]+19] |      | env[bas[i*8+6]+49]

```
- ‘bas[i\*8+7]’ unused
  - nbas: int, number of bases, nbas has no effect, can be set to 0
  - env: double[], save the value of coordinates, exponents, contraction coefficients
  - struct CINTOpt \*opt: so called “optimizer”, it needs to be initialized  
 CINTOpt \*opt = NULL; intname\_\_optimizer(&opt, atm, natm, bas, nbas, env);

every integral type has its own optimizer with the suffix `_optimizer` in its name, e.g. the optimizer for `cint2e_sph` is `cint2e_sph_optimizer`. “optimizer” is an optional argument for the integrals. It can roughly speed the integration by 10% without affecting the value of integrals. If no optimizer is wanted, set it to `NULL`.

optimizer needs to be released after using.

`CINTdel_optimizer(&opt);`

- if the return value equals 0, every element of the integral is 0
- short example

```
#include "cint.h"
...
CINTOpt *opt = NULL;
cint2e_sph_optimizer(&opt, atm, natm, bas, nbas, env);
for (i = 0; i < nbas; i++) {
    shls[0] = i;
    di = CINTcgto_spheric(i, bas);
    ...
    for (l = 0; l < nbas; l++) {
        shls[3] = l;
        dl = CINTcgto_spheric(l, bas);
        buf = malloc(sizeof(double) * di * dj * dk * dl);
        cint2e_cart(buf, shls, atm, natm, bas, nbas, env, opt);
        free(buf);
    }
}
CINTdel_optimizer(&opt);
```

## Fortran routine

```
dim = CINTgto_cart(bas_id, bas)
dim = CINTgto_spheric(bas_id, bas)
dim = CINTgto_spinor(bas_id, bas)
call f1e(buf, shls, atm, natm, bas, nbas, env)
call f2e(buf, shls, atm, natm, bas, nbas, env, opt)
call f2e_optimizer(opt, atm, natm, bas, nbas, env)
call CINTdel_optimizer(opt)
```

- atm and bas are 2D integer array
  - atm(1:6,i) is the (charge, offset\_coord, nuclear\_model, unused, unused, unused) of the ith atom
  - bas(1:8,i) is the (atom\_index, angular, num\_primitive\_GTO, num\_contract\_GTO, kappa, offset\_exponent, offset\_coeff, unused) of the ith basis

- parameters are the same to the C function. Note that those offsets atm(2,i) bas(6,i) bas(7,i) are 0-based.
- buf is 2D/4D double precision/double complex array
- opt: an integer(8) to hold the address of so called “optimizer”, it needs to be initialized by

integer(8) opt call f2e\_optimizer(opt, atm, natm, bas, nbas, env)

The optimizer can be banned by setting the “optimizer” to 0\_8

call f2e(buf, atm, natm, bas, nbas, env, 0\_8)

To release optimizer, execute

call CINTdel\_optimizer(opt);

- short example

```
...
integer,external CINTcgto_spheric
integer(8) opt
call cint2e_sph_optimizer(opt, atm, natm, bas, nbas, env)
do i = 1, nbas
  shls(1) = i - 1
  di = CINTcgto_spheric(i-1, bas)
  ...
  do l = 1, nbas
    shls(4) = l - 1
    dl = CINTcgto_spheric(l-1, bas)
    allocate(buf(di,dj,dk,dl))
    call cint2e_sph(buf, shls, atm, natm, bas, nbas, env, opt)
    deallocate(buf)
  end do
end do
call CINTdel_optimizer(opt)
```

## Supported angular momentum

$$l_{max} = 6$$

## Data ordering

- for Cartesian GTO, the output data in buf are sorted as

| s shell | p shell | d shell | ... |
|---------|---------|---------|-----|
| ...     | ...     | ...     |     |
| s       | p $x$   | d $xx$  |     |
| s       | p $y$   | d $xy$  |     |
| ...     | p $z$   | d $xz$  |     |
|         | p $x$   | d $yy$  |     |
|         | p $y$   | d $yz$  |     |
|         | p $z$   | d $zz$  |     |
|         | ...     | ...     |     |

- for real spheric GTO, the output data in buf are sorted as

| s shell | p shell | d shell       | f shell           | ... |
|---------|---------|---------------|-------------------|-----|
| ...     | ...     | ...           | ...               |     |
| s       | p $x$   | d $xy$        | f $y(3x^2 - y^2)$ |     |
| s       | p $y$   | d $yz$        | f $xyz$           |     |
| ...     | p $z$   | d $z^2$       | f $yz^2$          |     |
|         | p $x$   | d $xz$        | f $z^3$           |     |
|         | p $y$   | d $x^2 - y^2$ | f $xz^2$          |     |
|         | p $z$   | ...           | f $z(x^2 - y^2)$  |     |
|         | ...     |               | f $x(x^2 - 3y^2)$ |     |
|         |         |               | ...               |     |

- for spinor GTO, the output data in buf correspond to

| ... | kappa=0,p shell | kappa=1,p shell | kappa=0,d shell | ... |
|-----|-----------------|-----------------|-----------------|-----|
|     | ...             | ...             | ...             |     |
|     | $p_{1/2}(-1/2)$ | $p_{1/2}(-1/2)$ | $d_{3/2}(-3/2)$ |     |
|     | $p_{1/2}(1/2)$  | $p_{1/2}(1/2)$  | $d_{3/2}(-1/2)$ |     |
|     | $p_{3/2}(-3/2)$ | $p_{1/2}(-1/2)$ | $d_{3/2}(1/2)$  |     |
|     | $p_{3/2}(-1/2)$ | $p_{1/2}(1/2)$  | $d_{3/2}(3/2)$  |     |
|     | $p_{3/2}(1/2)$  | $p_{1/2}(-1/2)$ | $d_{5/2}(-5/2)$ |     |
|     | $p_{3/2}(3/2)$  | $p_{1/2}(1/2)$  | $d_{5/2}(-3/2)$ |     |
|     | $p_{1/2}(-1/2)$ | ...             | $d_{5/2}(-1/2)$ |     |
|     | $p_{1/2}(1/2)$  |                 | $d_{3/2}(-3/2)$ |     |
|     | $p_{3/2}(-3/2)$ |                 | $d_{3/2}(-1/2)$ |     |
|     | $p_{3/2}(-1/2)$ |                 | ...             |     |
|     | ...             |                 |                 |     |

## Tensor

Integrals like Gradients have more than one components. The output array is ordered in Fortran-contiguous. The tensors are ordered as

- 3-component tensor
  - X buf(:,0)
  - Y buf(:,1)
  - Z buf(:,2)
- 9-component tensor

- XX buf(:,0)
- XY buf(:,1)
- XZ buf(:,2)
- YX buf(:,3)
- YY buf(:,4)
- YZ buf(:,5)
- ZX buf(:,6)
- ZY buf(:,7)
- ZZ buf(:,8)

## Built-in function list

- Cartesian GTO integrals
  - CINTcgto\_cart(int shell\_id, int bas[]): Number of cartesian functions of the given shell
  - cint1e\_ovlp\_cart  $\langle i|j \rangle$
  - cint1e\_nuc\_cart  $\langle i|V_{nuc}|j \rangle$
  - cint1e\_kin\_cart  $.5\langle i|\vec{p} \cdot \vec{p}|j \rangle$
  - cint1e\_ia01p\_cart  $\langle i|\frac{\vec{r}}{r^3} \times \vec{\nabla}|j \rangle$
  - cint1e\_irixp\_cart  $\langle i|(\vec{r} - \vec{R}_i) \times \vec{\nabla}|j \rangle$
  - cint1e\_ircxp\_cart  $\langle i|(\vec{r} - \vec{R}_o) \times \vec{\nabla}|j \rangle$
  - cint1e\_iking\_cart  $0.5i\langle \vec{p} \cdot \vec{p}|U_g|j \rangle$
  - cint1e\_iovlp\_cart  $i\langle i|U_g|j \rangle$
  - cint1e\_inucg\_cart  $i\langle i|V_{nuc}|U_g|j \rangle$
  - cint1e\_ipovlp\_cart  $\langle \vec{\nabla}_i|j \rangle$
  - cint1e\_ipkin\_cart  $0.5\langle \vec{\nabla}_i|\vec{p} \cdot \vec{p}|j \rangle$
  - cint1e\_ipnuc\_cart  $\langle \vec{\nabla}_i|V_{nuc}|j \rangle$

- `cint1e_iprinv_cart`  $\langle \vec{\nabla} i | r^{-1} | j \rangle$
- `cint1e_rinv_cart`  $\langle i | r^{-1} | j \rangle$
- `cint2e_cart`  $(ij | kl)$
- `cint2e_ig1_cart`  $i \langle i U_g j | kl \rangle$
- `cint2e_ip1_cart`  $\langle \vec{\nabla} i j | kl \rangle$
- Spheric GTO integrals
  - `CINTcgto_spheric(int shell_id, int bas[])`: Number of  
spheric functions of the given shell
  - `cint1e_ovlp_sph`  $\langle i | j \rangle$
  - `cint1e_nuc_sph`  $\langle i | V_{nuc} | j \rangle$
  - `cint1e_kin_sph`  $0.5 \langle i | \vec{p} \cdot \vec{p} | j \rangle$
  - `cint1e_ia01p_sph`  $\langle i | \frac{\vec{r}}{r^3} \times \vec{\nabla} | j \rangle$
  - `cint1e_irixp_sph`  $\langle i | (\vec{r}_c - \vec{R}_i) \times \vec{\nabla} | j \rangle$
  - `cint1e_ircxp_sph`  $\langle i | (\vec{r}_c - \vec{R}_o) \times \vec{\nabla} | j \rangle$
  - `cint1e_iking_sph`  $0.5 i \langle \vec{p} \cdot \vec{p} | U_g j \rangle$
  - `cint1e_iovlp_sph`  $i \langle i | U_g j \rangle$
  - `cint1e_inucg_sph`  $i \langle i | V_{nuc} | U_g j \rangle$
  - `cint1e_ipovlp_sph`  $\langle \vec{\nabla} i | j \rangle$
  - `cint1e_ipkin_sph`  $0.5 \langle \vec{\nabla} i | \vec{p} \cdot \vec{p} | j \rangle$
  - `cint1e_ipnuc_sph`  $\langle \vec{\nabla} i | V_{nuc} | j \rangle$

- `cint1e_iprinv_sph`  $\langle \vec{\nabla} i | r^{-1} | j \rangle$
- `cint1e_rinv_sph`  $\langle i | r^{-1} | j \rangle$
- `cint2e_sph`  $(ij|kl)$
- `cint2e_ig1_sph`  $i(iU_g j|kl)$
- `cint2e_ip1_sph`  $(\vec{\nabla} ij|kl)$
- Spinor GTO integrals
  - `CINTcgto_spinor(int shell_id, int bas[])`: Number of spinor functions of the given shell
  - `cint1e_ovlp`  $\langle i | j \rangle$
  - `cint1e_nuc`  $\langle i | V_{nuc} | j \rangle$
  - `cint1e_nucg`  $\langle i | V_{nuc} | U_g j \rangle$
  - `cint1e_srsr`  $\langle \vec{\sigma} \cdot \vec{r} i | \vec{\sigma} \cdot \vec{r} j \rangle$
  - `cint1e_sr`  $\langle \vec{\sigma} \cdot \vec{r} i | j \rangle$
  - `cint1e_srsp`  $\langle \vec{\sigma} \cdot \vec{r} i | \vec{\sigma} \cdot \vec{p} j \rangle$
  - `cint1e_spsp`  $\langle \vec{\sigma} \cdot \vec{p} i | \vec{\sigma} \cdot \vec{p} j \rangle$
  - `cint1e_sp`  $\langle \vec{\sigma} \cdot \vec{p} i | j \rangle$
  - `cint1e_spspsp`  $\langle \vec{\sigma} \cdot \vec{p} i | \vec{\sigma} \cdot \vec{p} \vec{\sigma} \cdot \vec{p} j \rangle$
  - `cint1e_spnuc`  $\langle \vec{\sigma} \cdot \vec{p} i | V_{nuc} | j \rangle$
  - `cint1e_spnucsp`  $\langle \vec{\sigma} \cdot \vec{p} i | V_{nuc} | \vec{\sigma} \cdot \vec{p} j \rangle$
  - `cint1e_srnucsr`  $\langle \vec{\sigma} \cdot \vec{r} i | V_{nuc} | \vec{\sigma} \cdot \vec{r} j \rangle$



|                     |                                                                                                           |
|---------------------|-----------------------------------------------------------------------------------------------------------|
| – cint1e_sa10sa01   | $0.5\langle\vec{\sigma}\times\vec{r}_ci \vec{\sigma}\times\frac{\vec{r}}{r^3} j\rangle$                   |
| – cint1e_ovlpg      | $\langle i U_gj\rangle$                                                                                   |
| – cint1e_sa10sp     | $0.5\langle\vec{r}_c\times\vec{\sigma}i \vec{\sigma}\cdot\vec{p}j\rangle$                                 |
| – cint1e_sa10nucsp  | $0.5\langle\vec{r}_c\times\vec{\sigma}i V_{nuc} \vec{\sigma}\cdot\vec{p}j\rangle$                         |
| – cint1e_sa01sp     | $\langle i \frac{\vec{r}}{r^3}\times\vec{\sigma} \vec{\sigma}\cdot\vec{p}j\rangle$                        |
| – cint1e_spgsp      | $\langle U_g\vec{\sigma}\cdot\vec{p}i \vec{\sigma}\cdot\vec{p}j\rangle$                                   |
| – cint1e_spgnucsp   | $\langle U_g\vec{\sigma}\cdot\vec{p}i V_{nuc} \vec{\sigma}\cdot\vec{p}j\rangle$                           |
| – cint1e_spgsa01    | $\langle U_g\vec{\sigma}\cdot\vec{p}i \frac{\vec{r}}{r^3}\times\vec{\sigma} j\rangle$                     |
| – cint1e_ipovlp     | $\langle\vec{\nabla}i j\rangle$                                                                           |
| – cint1e_ipkin      | $0.5\langle\vec{\nabla}i p\cdot pj\rangle$                                                                |
| – cint1e_ipnuc      | $\langle\vec{\nabla}i V_{nuc} j\rangle$                                                                   |
| – cint1e_iprinv     | $\langle\vec{\nabla}i r^{-1} j\rangle$                                                                    |
| – cint1e_ipspnucsp  | $\langle\vec{\nabla}\vec{\sigma}\cdot\vec{p}i V_{nuc} \vec{\sigma}\cdot\vec{p}j\rangle$                   |
| – cint1e_ipsprinvsp | $\langle\vec{\nabla}\vec{\sigma}\cdot\vec{p}i r^{-1} \vec{\sigma}\cdot\vec{p}j\rangle$                    |
| – cint2e            | $(ij kl)$                                                                                                 |
| – cint2e_spsp1      | $(\vec{\sigma}\cdot\vec{p}i\vec{\sigma}\cdot\vec{p}j kl)$                                                 |
| – cint2e_spsp1spsp2 | $(\vec{\sigma}\cdot\vec{p}i\vec{\sigma}\cdot\vec{p}j \vec{\sigma}\cdot\vec{p}k\vec{\sigma}\cdot\vec{p}l)$ |

- cint2e\_srsr1  $(\vec{\sigma} \cdot \vec{r}_i \vec{\sigma} \cdot \vec{r}_j | kl)$   
 - cint2e\_srsr1srsr2  $(\vec{\sigma} \cdot \vec{r}_i \vec{\sigma} \cdot \vec{r}_j | \vec{\sigma} \cdot \vec{r}_k \vec{\sigma} \cdot \vec{r}_l)$   
 - cint2e\_sa10sp1  $0.5(\vec{r}_c \times \vec{\sigma}_i \vec{\sigma} \cdot \vec{p}_j | kl)$   
 - cint2e\_sa10sp1spsp2  $0.5(\vec{r}_c \times \vec{\sigma}_i \vec{\sigma} \cdot \vec{p}_j | \vec{\sigma} \cdot \vec{p}_k \vec{\sigma} \cdot \vec{p}_l)$   
 - cint2e\_g1  $(iU_g j | kl)$   
 - cint2e\_spgsp1  $(\vec{\sigma} \cdot \vec{p}_i U_g \vec{\sigma} \cdot \vec{p}_j | kl)$   
 - cint2e\_g1spsp2  $(iU_g j | \vec{\sigma} \cdot \vec{p}_k \vec{\sigma} \cdot \vec{p}_l)$   
 - cint2e\_spgsp1spsp2  $(\vec{\sigma} \cdot \vec{p}_i U_g \vec{\sigma} \cdot \vec{p}_j | \vec{\sigma} \cdot \vec{p}_k \vec{\sigma} \cdot \vec{p}_l)$   
 - cint2e\_ip1  $(\vec{\nabla}_{ij} | kl)$   
 - cint2e\_ipspsp1  $(\vec{\nabla} \vec{\sigma} \cdot \vec{p}_i \vec{\sigma} \cdot \vec{p}_j | kl)$   
 - cint2e\_ip1spsp2  $(\vec{\nabla}_{ij} | \vec{\sigma} \cdot \vec{p}_k \vec{\sigma} \cdot \vec{p}_l)$   
 - cint2e\_ipspsp1spsp2  $(\vec{\nabla} \vec{\sigma} \cdot \vec{p}_i \vec{\sigma} \cdot \vec{p}_j | \vec{\sigma} \cdot \vec{p}_k \vec{\sigma} \cdot \vec{p}_l)$   
 - cint2e\_ssp1ssp2  $(i \vec{\sigma} \vec{\sigma} \cdot \vec{p}_j | k \vec{\sigma} \vec{\sigma} \cdot \vec{p}_l)$