

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
use Math::Trig ;
use strict;
```

```
#{my %coor,my $chnum}=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $mdir=$ARGV[1];
my $sch, my $chnum;
foreach my $r ($sch) {
```

Лекция 4. Введение в квантовую химию

Курс: Молекулярное моделирование в применении к биомолекулам

Головин А.В. ¹

¹МГУ им М.В. Ломоносова, Факультет Биоинженерии и Биоинформатики

Москва, 2013

```
my %qwa=find_quart( $coor{"O"} ); my $qnum=keys %qwa;
```

```
if ($qnum > 0){
#system("mkdir $ARGV[1]");
my $filename=$ARGV[0];
$filename=~ s/^\.//;
$filename=~ s/\.pdb//;
#$filename=$chnum.".".$qnum.".".$filename.".dat";
$filename="$mdir".$filename.".dat";
print "$filename\n";
open OUT,">$filename";
print OUT "#INFO chain $chnum qnum $qnum\n";
```

```
foreach my $m (sort { $a<=>$b } keys %coor){
my %qartets= %qwa ; #find quart( $coor{$m} );
my %q= find
```

```
# foreach my $q { keys %qartets } { print join " ",@{ $qartets{$q} }, "\n";
```

```
foreach my $q { keys %qartets } {
```

```
my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;
```

```
foreach my $res (@{ $qartets{$q} }){
```

```
# print "$q $coor{$m}{ $res } {"R"}->x,\n";
```

```
$nx=$nx+ $coor{$m}{ $res }{"N9"}->x;
```

```
$ny=$ny+ $coor{$m}{ $res }{"N9"}->y;
```

```
$nz=$nz+ $coor{$m}{ $res }{"N9"}->z;
```

```
$ox=$ox+ $coor{$m}{ $res }{"O6"}->x;
```

```
$oy=$oy+ $coor{$m}{ $res }{"O6"}->y;
```

```
$oz=$oz+ $coor{$m}{ $res }{"O6"}->z;
```

```
$r=$res;
```

```
}
```

Содержание

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );

#(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"0"}}){ my $ggg=substr($r,0,1); if ( $ggg ne $ch){ $chnum++; $ch=$ggg } };
```

```
my %qwa=find_quart( %coor{"0"} ); my $qnum=keys %qwa;
```

```
if ($qnum > 0){
#system("mkdir $ARGV[1]");
my $filename=$ARGV[0];
$filename=~ s/^\./\.\./;
$filename=~ s/\.pdb//;
#$filename=$chnum."_"$qnum."_"$filename.".dat";
$filename="$dir/".$filename.".dat";
print OUT "#INFO chain $chnum qnum $qnum\n";
```

```
foreach my $m (sort { $a<=>$b } keys %coor){
my %qartets= %qwa; #find_quart( %coor{$m} );
my %q= find_q( %coor{$m} );
```

```
# foreach my $q ( keys %qartets){ print join " ",@{ $qartets{$q} },"\n";
```

```
foreach my $q ( keys %qartets){
```

```
my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;
```

```
foreach my $res ( @{ $qartets{$q} }){
```

```
# print "$q $coor{$m} {$res} {"N"}->x,"\n";
$nx=$nx+ $coor{$m} {$res} {"N9"}->x;
$ny=$ny+ $coor{$m} {$res} {"N9"}->y;
$nz=$nz+ $coor{$m} {$res} {"N9"}->z;
```

```
$ox=$ox+ $coor{$m} {$res} {"O6"}->x;
$oy=$oy+ $coor{$m} {$res} {"O6"}->y;
$oz=$oz+ $coor{$m} {$res} {"O6"}->z;
```

Активные молекулы

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Волновая функция

```
my ($my %coor,$my $chain) = read_pdb($ARGV[0]);
my $chain = read_pdb($ARGV[0]);
```

Wikipedia* :

Волновая функция — комплекснозначная функция, используемая для описания чистого квантового состояния системы. Обычно функция имеет комплексные значения, а для одной частицы это функция пространства и времени. Изменение волновой функции сравнимо с поведением волны.

Физический смысл волновой функции заключается в том, что согласно копенгагенской интерпретации квантовой механики плотность вероятности нахождения частицы в данной точке пространства в данный момент времени считается равной квадрату абсолютного значения волновой функции этого состояния в координатном представлении.

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Уравнение Шредингера

```
my ($coor,$schnum)=read_pdb($ARGV[0]);
```

```
my %coor=read_pdb($ARGV[0]);
```

Итак обзовем оператором H (Гамильтониан):

```
foreach my $r ( sort keys %{$coor{"0"}} ){ my $sggg=substr($r,0,1); if ( $sggg ne $sch ){ $schnum++; $sch=$sggg } ;
```

```
my %qwa=find_quart( $coor{"0"} ); my $sqnum=keys %qwa;
```

$$H = \frac{-\hbar^2}{2m} \nabla^2 + V$$

```
if ($sqnum > 0){
```

```
  #system("mkdir $ARGV[1]");
```

```
  my $filename=$ARGV[0];
```

```
  $filename="-- s/^.*\//";
```

```
  $filename="-- s/\.pdb//";
```

```
  # $filename=$schnum." ".$sqnum." ".$filename." .dat";
```

```
  $filename="-- $dir". $filename." .dat";
```

```
  print OUT "\n";
```

```
  print OUT "chain $schnum qnum $sqnum \n";
```

```
  print OUT "INFO chain $schnum qnum $sqnum \n";
```

```
  foreach my $m ( sort { $a-<=>$b } keys %coor ){
```

```
    my %qartets = %qwa; #find_quart( $coor{ $m } );
```

```
    my %q = find_q( $coor{ $m } );
```

$$H\Psi = E\Psi$$

```
  #   foreach my $q { keys %qartets } { print join " ", @{$qartets{$q}}; "\n";
```

Для решения этого уравнения надо найти значения E и волновой

функции. Это уравнение относится к типу дифференциальных

уравнений с собственными значениями, где оператор действующий на

функцию возвращает произведение скалярной величины на функцию.

```
  $nx=$nx+ $coor{ $m } { $res } {"N9"}->x;
```

```
  $ny=$ny+ $coor{ $m } { $res } {"N9"}->y;
```

```
  $nz=$nz+ $coor{ $m } { $res } {"N9"}->z;
```

```
  $ox=$ox+ $coor{ $m } { $res } {"O6"}->x;
```

```
  $oy=$oy+ $coor{ $m } { $res } {"O6"}->y;
```

```
  $oz=$oz+ $coor{ $m } { $res } {"O6"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Операторы

```
#[my %coor,my $chnum]=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
```

Ожидаемое значение (можно рассматривать как среднее значение) какого либо свойства: энергии, положения, линейного момента, можно определить с помощью оператора.

Пример: гамильтониан это оператор для энергии можно сказать, что зная волновую функцию:

$$E = \frac{\int \dot{\Psi} H \Psi \partial r}{\int \dot{\Psi} \Psi \partial r}$$

Интегрировать надо по всем осям от $-\infty$ до $+\infty$.

Надо учитывать, что волновая может быть сложным числом и поэтому комплексная составляющая указывается явно.

```
# foreach my $q { keys %qartets } { print join " ", @qartets{$q} }, "\n";

my $nx,my $ny,my $nz;
my $ox,my $oy,my $oz;

print "$q $coor{$$m} {$res} {"$N"}->x, "\n";
$nx=$nx+ $coor{$$m} {$res} {"$N"}->x;
$ny=$ny+ $coor{$$m} {$res} {"$N"}->y;
$nz=$nz+ $coor{$$m} {$res} {"$N"}->z;

$ox=$ox+ $coor{$$m} {$res} {"$O6"}->x;
$oy=$oy+ $coor{$$m} {$res} {"$O6"}->y;
$oz=$oz+ $coor{$$m} {$res} {"$O6"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Приближение Борна-Оппергеймера

```

#(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $sch, my $chnum;
foreach my $r ( sort keys %{$coor{"O"}} ) { my $qqr=substr($r,0,1); if ( $qqr ne $sch ) { $chnum++; $sch=$qqr; }
}

```

Если мы считаем, что ядра двигаются сильно медленнее чем электроны, то мы можем считать

$$\Psi_{total} = \psi_{electronic} \psi_{nucleic}$$

$$E_{total} = E_{electronic} + E_{nucleic}$$

Google:

electron mass = $9.10938188 \times 10^{-31} \text{ kg}$

proton mass = $1.67262158 \times 10^{-27} \text{ kg}$

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Одно-электронный атом

Рассматривая одно-электронный атом можно и учитывая, что система имеет сферическую симметрию и её можно представить волновую функцию в сферических координатах:

```
my %qwa=find_quart( $coor{"0"} ); my $qnum=keys %qwa;
```

$$\Psi(r, \theta, \phi) = R(r)\Theta(\theta)\Phi(\phi)$$

```
if ($qnum > 0){
#system("mkdir $ARGV[1]");
my $filename=$ARGV[0];
```

Уравнение Шредингера разбивается на 3 уравнения с 1 или 2мя параметрами.

```
print "filename\n";
open OUT,">$filename";
print OUT "#INFO chain $qnum qnum $qnum\n";
```

$$\psi_{nlm}(r, \vartheta, \varphi) = \sqrt{\left(\frac{2}{na_0}\right)^3 \frac{(n-l-1)!}{2n(n+l)!}} e^{-\rho/2} \rho^l L_{n-l-1}^{2l+1}(\rho) Y_l^m(\vartheta, \varphi);$$

```
# foreach my $q { keys %qartets } { print join " ", @qartets{$q} }, "\n";
```

$L_{n-l-1}^{2l+1}(\rho)$ - Обобщённый полином Лагерра степени $n-l-1$; $\rho = \frac{2r}{na_0}$

$Y_l^m(\vartheta, \varphi)$ - Сферическая гармоника;

```
my $n=$qnum;
```

```
foreach my $res ( @ { $qartets{$q} } ) {
```

$$\Psi_{1,0,0}(1s) = \sqrt{\frac{Z^3}{\pi}} e^{-Zr}; \quad \Psi_{1,0,0}(2p_0) = \sqrt{\frac{Z^3}{2^5 \pi}} Z r e^{-Zr/2} \cos \theta$$

```
$ox=$ox+ $coor{$m}{$res}{"O6"}->x;
```

```
$oy=$oy+ $coor{$m}{$res}{"O6"}->y;
```

```
$oz=$oz+ $coor{$m}{$res}{"O6"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Многоэлектронный атом

```
#!/my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
```

Полное решение уравнение Шредингера для многоэлектронного атома затруднено, по ряду причин:

- N-body problem, суть вопроса, предсказать движение трёх и более тел на всём течении времени, если известны положение и скорости на текущий момент.
- Добавление четвёртого экспериментального квантового числа, спина, создаёт необходимость различать электроны.
- Квадрат волновой функции равен плотности. Трактование волновой функции как плотности электрона в данном месте, означает, что плотность может быть образована любым электроном. Этот факт сильно затрудняет расчёты.

```
foreach my $atom {
  my %q=();
  my %q= find_q( $coor{$m} );
  # for( keys %q ) {
  for( keys %q ) {
    my $ox=$x+ $coor{$m} {$sres} {"N9"}->x;
    $ny=$ny+ $coor{$m} {$sres} {"N9"}->y;
    $nz=$nz+ $coor{$m} {$sres} {"N9"}->z;
  }
  $ox=$ox+ $coor{$m} {$sres} {"O6"}->x;
  $oy=$oy+ $coor{$m} {$sres} {"O6"}->y;
  $oz=$oz+ $coor{$m} {$sres} {"O6"}->z;
}
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Метод самосогласованного поля, SCF

```
my ($coor,$my $chnum)=read_pdb($ARGV[0]);
```

```
my $file=read_pdb($ARGV[0]);
```

Межелектронное отталкивание вычисляется как влияние общего (среднего) поля на данный электрон, и это зависит только от положения данного электрона.

Это приближение позволяет повторить разделение переменных в сферических координатах.

$$H_i = \frac{-\hbar^2}{2m} \nabla^2 - \frac{Ze^2}{4\pi\epsilon_0 r_i} + \sum_{j \neq i}^N \left\langle \left(\frac{e^2}{4\pi\epsilon_0 r_{ij}} \right) \right\rangle_j$$

Эти уравнения называют одноэлектронными.

Суть решения состоит в итеративном изменении параметров в функциях, до тех пор пока изменение энергии не станет незначительным.

```
my $x=$my $ny; my $z;
```

```
$ny=$ny+ $coor{$m} {$res} {"N9"}->y;
```

```
$nz=$nz+ $coor{$m} {$res} {"N9"}->z;
```

```
$ox=$ox+ $coor{$m} {$res} {"O6"}->x;
```

```
$oy=$oy+ $coor{$m} {$res} {"O6"}->y;
```

```
$oz=$oz+ $coor{$m} {$res} {"O6"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Итак, энергия:

Для не возбуждённых состояний или closed shell:

```
#!/my %$coor,my $schnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $sch, my $schnum;
foreach my $r ( sort keys %{$coor{"0"}} ) { my $qnum=keys %$qwa;
my %qwa=find_quart( $coor{"0"} ); my $qnum=keys %qwa;
```

$$E = H^{core} + E^{rep} + E^{exch}; \quad \sum_{i=1}^{N/2} 2H_{ii}^{core}$$

```
if ($qnum > 0){
```

Существует четыре способа: как электроны с одной орбитали взаимодействуют с электронами с другой орбитали и есть всего два способа получить спаренные электроны и:

```
print OUT "#INFO chain $schnum qnum $qnum \n";
```

$$E = 2 \sum_{i=1}^{N/2} 2H_{ii}^{core} + \sum_{i=1}^{N/2} \sum_{j=i+1}^{N/2} (4J_{ij} - 2K_{ij}) + \sum_{i=1}^{N/2} J_{ii}$$

Или если примем $J_{ii} = K_{ii}$

$$E = 2 \sum_{i=1}^{N/2} 2H_{ii}^{core} + \sum_{i=1}^{N/2} \sum_{j=i+1}^{N/2} (J_{ij} - K_{ij})$$

```
$sox=$sox+ $coor{$m}{$sres}{"O6"}->x;
$soy=$soy+ $coor{$m}{$sres}{"O6"}->y;
$soz=$soz+ $coor{$m}{$sres}{"O6"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( // );
```

Метод: Хартри-Фока:

Аппроксимация многоэлектронной волновой функции детерминантом Слейтора и решение методов самосогласованного поля приводят к методу Хартри-Фока, в котором точный гамильтониан заменён оператором Фока.

```
my $dir=$ARGV[1];
my $filename=$ARGV[0];
$filename=~ s/~/./;
$filename=~ s/\./_/;
my $filename="$dir"/"$filename".dat;
print "filename=$filename\n";
open my $fh, "<",$filename or die "cannot open $filename";
print OUT "#INFO $dir $filename\n";
```

$$F_i = -\frac{1}{2}\nabla_i^2 - \frac{Z}{r_i} + \sum_j^N \left[\int \chi_j^*(x_j) \frac{1}{r_{ij}} \chi_j(x_j) \partial x_j - \int \chi_j^*(x_j) \frac{1}{r_{ij}} \chi_i(x_j) \partial x_j \right] = \epsilon_i \chi_i(x_i)$$

энергия электрона на орбитали χ_i

```
my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;
```

$$\epsilon_i = H_i + \sum_{j \neq i}^N [J_{ij} - K_{ij}]$$

```
foreach my $res (@{ $qartets{$sq} }) {
    print "$q $coor{$sm} {$res} {"$N"}->x,\n";
    $nx=$nx+ $coor{$sm} {$res} {"$N"}->x;
    $ny=$ny+ $coor{$sm} {$res} {"$N"}->y;
    $nz=$nz+ $coor{$sm} {$res} {"$N"}->z;
    $ox=$ox+ $coor{$sm} {$res} {"$O6"}->x;
    $oy=$oy+ $coor{$sm} {$res} {"$O6"}->y;
    $oz=$oz+ $coor{$sm} {$res} {"$O6"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Перейдём к молекулам:

```

#(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"O"}} ){ my $ggg=substr($r,0,1); if ( $ggg ne $ch ){ $chnum++; $ch=$ggg } ;

```

Решать напрямую уравнения ХФ по отношению к молекулам, тяжело. Одной из успешных стратегий является введение базисных функций, т.е. волновая функция это комбинация одноэлектронных базисных функций и некоторых коэффициентов.

```

foreach my $m (sort { $a<=>$b } keys %coor){
  my %qartets = %qwa ; #find quart( $coor{$m} );
  my %q= find_q( $coor{$m} );

```

```

#   foreach my $q ( keys %qartets ){ print join
   foreach my $q ( keys %qartets){

```

```

    my $nx; my $ny; my $nz;
    my $ox; my $oy; my $oz;
    my $r;

```

```

    foreach my $res ( @{$qartets{$q}} ){
      print "$q $coor{$m} {$res} {"R"}->x,"n";
      $nx=$nx+ $coor{$m} {$res} {"N9"}->x;
      $ny=$ny+ $coor{$m} {$res} {"N9"}->y;
      $nz=$nz+ $coor{$m} {$res} {"N9"}->z;

```

```

    $ox=$ox+ $coor{$m} {$res} {"O6"}->x;
    $oy=$oy+ $coor{$m} {$res} {"O6"}->y;
    $oz=$oz+ $coor{$m} {$res} {"O6"}->z;

```

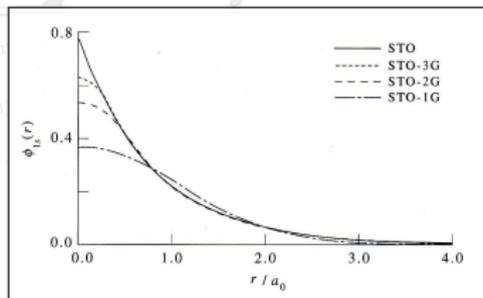
$$\psi_i = \sum_{\nu=1}^K c_{\nu i} \psi_{\nu}; \quad \frac{\partial E}{\partial c_{\nu i}} = 0$$

Подробнее:

При выборе базисных функций надо выполнить всего два условия:

- Они должны иметь физический смысл.
- Интегралы должны быть сходимыми.

Обычная практика – это использовать функции типа гауссиана, потому что их легко считать.



Каждая молекулярная орбиталь разлагается в набор базисных функций, центрированных около ядра и обычно называемых атомными орбиталями.

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Подход Рутхана-Хола

Если молекулярная орбиталь это сумма функций с коэффициентами, то и надо их искать.

Вспомним свойство операторов:

```
if ($qnum > 0){
#system("mkdir SARGV[1]");
my $filename=SARGV[0];
$filename-- s/^,*///;
$filename-- s/\.pdb//;
# $filename=$chnum.".".$qnum.".".$filename;
$filename="$dir"."$filename"."dat";
print "$filename\n";
open OUT,">$filename";
print OUT "#INFO: chain $chnum qnum $qnum\n";
```

$$\psi_i = \sum_{\nu=1}^K c_{\nu i} \chi_{\nu}; \quad F \sum_{\nu=1}^K c_{\nu i} \chi_{\nu} = e_i \sum_{\nu=1}^K c_{\nu i} \chi_{\nu}$$

FC=SCE, где S это интеграл перекрывания, $S_{ij} = \langle b_i | b_j \rangle = \int \chi_i \chi_j \partial r$

```
my %qartets = %qwa; #find quart ($m);
my %q = find_q($coor{$m});
```

```
# foreach my $q ( keys %qartets ){ print join " ", $qartets{$q} }
foreach my $q ( keys %qartets ){
my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;
foreach my $res ( @($qartets{$q}) ){
print "$q $coor{$m} $res";
$nx=$nx+ $coor{$m} {$res} {"N9"}->x;
$ny=$ny+ $coor{$m} {$res} {"N9"}->y;
$nz=$nz+ $coor{$m} {$res} {"N9"}->z;
$ox=$ox+ $coor{$m} {$res} {"O6"}->x;
$oy=$oy+ $coor{$m} {$res} {"O6"}->y;
$oz=$oz+ $coor{$m} {$res} {"O6"}->z;
```

$$C = \begin{vmatrix} c_{1,1} & c_{1,2} & \dots & c_{1,k} \\ c_{2,1} & c_{2,2} & \dots & c_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ c_{k,1} & c_{k,2} & \dots & c_{k,k} \end{vmatrix}$$

$$E = \begin{vmatrix} e_1 & 0 & \dots & 0 \\ 0 & e_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & e_i \end{vmatrix}$$

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Непосредственно процедура счёта

```

#(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort { $a-<=>$b } keys %coor ) {
```

Первичная
генерация
орбиталей

Координаты ядер

Матрица Фока

Диагонализация
матрицы Фока

Результат

Проверка SCF

```

my %qwa=find_orbitals($ch,$coor,$chnum,$dir);
if ($qnum > 0) {
#system("mkdir
my $filename=$dir.$chnum.$qnum;
$filename="-- s/";
$filename="-- s/\,pdb/";
#$filename=$chnum.".".$qnum.".".$filename.".dat";
$filename="$dir"."$filename.".dat";
print "$filename\n";
open OUT,">$filename";
print OUT "#INFO chain $chnum qnum $qnum\n";

foreach my $m ( sort { $a-<=>$b } keys %coor) {
my %qartets = %qwa ; #find_quart( $coor{ $m } );
my %q = find_q( $coor{ $m } );

# foreach my $q ( keys %qartets) { print join " ",@{ $qartets{ $q } },"\n";
foreach my $q ( keys %qartets) {
my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;
foreach my $res ( @{ $qartets{ $q } } ) {
print "$q $coor{ $m } { $res } { $r } -> x, \n";
$nx=$nx+ $coor{ $m } { $res } { "N9" }->x;
$ny=$ny+ $coor{ $m } { $res } { "N9" }->y;
$nz=$nz+ $coor{ $m } { $res } { "N9" }->z;

$ox=$ox+ $coor{ $m } { $res } { "O6" }->x;
$oy=$oy+ $coor{ $m } { $res } { "O6" }->y;
$oz=$oz+ $coor{ $m } { $res } { "O6" }->z;
```

не
сошлось

сошлось

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Словами:

- Посчитать интегралы для заполнения матрицы F
- Посчитать матрицу перекрывания S
- Диагонализируем матрицу S
- Строим $S^{-1/2}$
- Угадываем или рассчитываем матрицу плотности P
- Строим матрицу F заполняя значениями интегралов и P
- Строим $F' = S^{-1/2} F S^{-1/2}$
- Решаем $|F' - E| = 0$ для поиска собственных значений E и C' диагонализации F'.
- Рассчитываем орбитальные коэффициенты $C = S^{-1/2} C'$
- Считаем новую матрицу плотности из матрицы C
- Если плотность изменилась не значительно заканчиваем или продолжаем заполнять матрицу F

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Итак ab initio:

```

#(my %coor,my $chnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $ch, my $chnum;
foreach my $r ( sort keys %{$coor{"0"}}){ my $ggg=substr($r,0,1); if ( $ggg ne $ch){ $chnum++; $ch=$ggg } ;

```

```
my %qwa=find_quart( %coor{"0"} ); my $qnum=keys %qwa;
```

```

if ($qnum > 0){
#system("mkdir $ARGV[1]");
my $filename=$ARGV[0];

```

Как мы видели выше, электронную структуру молекулы можно посчитать зная только основные физические константы, такие подходы и называются *ab initio*.

```

$filename=~ s/"/"/g;
my $qnum=$qnum / $filename;
open OUT,">$filename";

```

```

foreach my $m (sort { $a<=>$b } keys %coor){
my %qartets = %qwa; #find_quart( %coor{$m} );
my %q = find_q( %coor{$m} );

```

```
# foreach my $q ( keys %qartets){ print join " ",@{$qartets{$q}},"n";
```

```
foreach my $q ( keys %qartets){
```

```

my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;

```

```
foreach my $res ( @{$qartets{$q}}){
```

```
# print "$q %coor{$m} {$res} {"N"}->x,"n";
```

```
$nx=$nx+ %coor{$m} {$res} {"N9"}->x;
```

```
$ny=$ny+ %coor{$m} {$res} {"N9"}->y;
```

```
$nz=$nz+ %coor{$m} {$res} {"N9"}->z;
```

```
$ox=$ox+ %coor{$m} {$res} {"O6"}->x;
```

```
$oy=$oy+ %coor{$m} {$res} {"O6"}->y;
```

```
$oz=$oz+ %coor{$m} {$res} {"O6"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Базисные наборы

Часто орбитальные функции Слейтора аппроксимируют гауссиановскими функциями. В общем виде это:

```
my $scor = $scor * $Schnum;
my $q = $q + $Schnum;
foreach my $r ( sort keys %{$scor{"0"}} ) { my $ggg = substr($r,0,1); my $mm = $mm + $ggg; $Sch = $ggg; };
my %qwa = find_quart( $scor{"0"} ); my $qnum = keys %qwa;
```

$$x^a y^b z^c e^{-\alpha r^2}$$

```
if ($qnum > 0) {
#system("mkdir $ARGV[1]");
my $filename = $ARGV[0];
$filename = s/\^\.//;
$filename = s/\./_./;
#$filename = $Schnum . "_" . $qnum . "_" . $filename;
$filename = "dir" . $filename . ".dat";
print "$filename\n";
open OUT, ">$filename";
print OUT "#INFO chain $Schnum qnum $qnum\n";
```

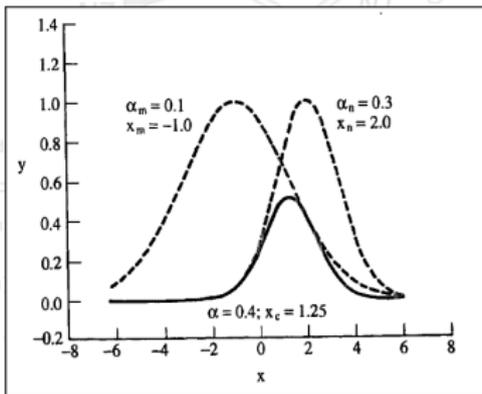
```
foreach my $m ( sort { $a <=> $b } keys %coor ) {
my %qartets = %qwa; #find_quart( $scor{$m} );
my %q = find_q( $coor{$m} );

# foreach my $q ( keys %qartets ) { print join
foreach my $q ( keys %qartets ) {
```

```
my $r;
foreach my $res ( @{$qartets{$q}} ) {
```

```
# print "$q $coor{$m} {$res} {"N9"}->x, "\n";
$nx = $nx + $coor{$m} {$res} {"N9"}->x;
$ny = $ny + $coor{$m} {$res} {"N9"}->y;
$nz = $nz + $coor{$m} {$res} {"N9"}->z;

$ox = $ox + $coor{$m} {$res} {"O6"}->x;
$oy = $oy + $coor{$m} {$res} {"O6"}->y;
$oz = $oz + $coor{$m} {$res} {"O6"}->z;
```



Важное свойство, сумму двух функций можно представить одним гауссианом, т.е.:

$$\phi_{\mu} = \sum_{i=1}^{L_{N7}} d_{i\mu} \phi_i(\alpha_{i\mu})$$

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Базисные наборы

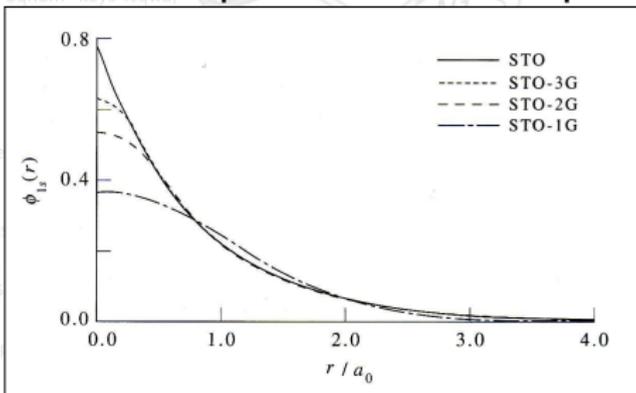
```
my ($coor,$schnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
```

Чем больше гауссианов описывают основные орбитали атомов тем ближе это описание к орбиталам Слейтора:

```
if ($qnum > 0){
#system("mkdir $ARGV[1]");
my $filename=$ARGV[0];
$filename="-- s/^.*\//";
$filename="-- s/\.pdb//";
#$filename=$schnum.".".$qnum.".".$file
$filename="$dir"."$filename"."dat";
print "$filename\n";
open OUT,">$filename";
print OUT "#INFO chain $schnum qnum\n";
```

```
foreach my $m (sort { $a<=>$b } keys
my %qartets = %qwa ; #find quartet
my %q = find_q( $coor{$m} );
```

```
# foreach my $q ( keys %qartets ){
foreach my $q ( keys %qartets ){
```



Естественно можно менять два параметра d и α , такие вычисления называют uncontracted. Но с точки зрения расчётов это не выгодно и часто используют contracted вычисления.

```
my $r;
foreach my $s ( @ { $qartets{$q} } ){
$nx=$nx+ $coor{$m} {$res} {"N9"}->x;
$ny=$ny+ $coor{$m} {$res} {"N9"}->y;
$nz=$nz+ $coor{$m} {$res} {"N9"}->z;
```

```
$ox=$ox+ $coor{$m} {$res} {"O6"}->x;
$oy=$oy+ $coor{$m} {$res} {"O6"}->y;
$oz=$oz+ $coor{$m} {$res} {"O6"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Basis set сокращения:

Рассмотрим самую популярную схему сокращения информации об использованных базисах на примере серии программ Gaussian.

```
my %qwa=find_quart( $coor{"0"} ); my $qnum=keys %qwa;
```

Minimal basis set: STO-nG, рассматриваются только орбитали атомов. Для элементов не содержащих d орбиталей STO-3G является абсолютным минимум. Плохо работает для несферических орбиталей и элементов в конце периода.

Double zeta basis: Это линейная комбинация contracted и diffuse функций, которая даёт результат более точный чем STO. Коэффициенты считаются в ходе итераций, что позволяет работать с анизотропией p_x, p_y, p_z .

Split valence double zetta: Подход простой, валентные оболочки описываем большим количеством функций, чем core электроны. 3-21G : 3 contracted гауссина описывают core орбитали, внешние орбитали: 2 contracted и 1 диффузный. 3-21G, 4-21G, 6-31G.

```
$ox=$ox+ $coor{$m} {$res} {"O6"}->x;
$oy=$oy+ $coor{$m} {$res} {"O6"}->y;
$oz=$oz+ $coor{$m} {$res} {"O6"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Basis set сокращения:

```
#!/(my %$coor,$my $schnum)=read_pdb($ARGV[0]);
my $coor=read_pdb($ARGV[0]);
my $mdir=$ARGV[1];
my $sch,$my $schnum;
```

Простое увеличение количества функций (triple, tetra) не обязательно приводит к улучшению модели. Это не поможет для систем с сильной анизотропией распределения заряда, например водородная связь.

В этих случаях вводят функции поляризации. Они соответствуют р орбиталям для водорода и d орбиталям первых двух рядов.

Пример:

```
{ $a<=>$b } keys %$coor{
my %$q= find_quart( $coor{ $m } );
my %$q= find_d( $coor{ $m } );
```

- 6-31G* поляризация для тяжёлый атомов
- 6-31G** поляризация для всех атомов, это важно для водородной связи!

```
foreach my $res ( @{ $qartets{ $q } } ){
my $ix,$my $iy,$my $iz;
my $sx,$my $sy,$my $sz;

print "$q $coor{ $m } { $res } { "N" }->x,"n";
$nx=$nx+ $coor{ $m } { $res } { "N9" }->x;
$ny=$ny+ $coor{ $m } { $res } { "N9" }->y;
$nz=$nz+ $coor{ $m } { $res } { "N9" }->z;

$ox=$ox+ $coor{ $m } { $res } { "O6" }->x;
$oy=$oy+ $coor{ $m } { $res } { "O6" }->y;
$oz=$oz+ $coor{ $m } { $res } { "O6" }->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Basis set сокращения:

```
#!/(my %coor,my $schnum)=read_pdb($ARGV[0]);
my %coor=read_pdb($ARGV[0]);
my $dir=$ARGV[1];
my $sch, my $schnum;
foreach my $r ( sort keys %{$coor{"0"}} ){ my $ggg=substr($r,0,1); if ( $ggg ne $sch ){ $schnum++; $sch=$ggg } ;
```

НО! этого не достаточно для расчёта атомов с не поделёнными парами электронов, там используют "высоко диффузные" функции:

- 6-311++G(3df,3pd): ++ диффузные функции на тяжёлых и водороде. 3df,3pd три набора d функций и 1 набор f функций для атомов первого ряда и три набора p функций и 1 набор d функций для водорода.

В последнее время считается, что: 6-31G*=6-31G(d)

```
foreach my $res ( @{ $partets{$q} } ){
# print "$q $coor{$m} {$res} {"$R"}->x,"n";
$nx=$nx+ $coor{$m} {$res} {"$R"}->x;
$ny=$ny+ $coor{$m} {$res} {"$R"}->y;
$nz=$nz+ $coor{$m} {$res} {"$R"}->z;

$ox=$ox+ $coor{$m} {$res} {"$O6"}->x;
$oy=$oy+ $coor{$m} {$res} {"$O6"}->y;
$oz=$oz+ $coor{$m} {$res} {"$O6"}->z;
```

Наименьшие базисные наборы, обеспечивающие описание свойств молекул

Молекулярная геометрия	HF/6-31G	Исключение – расчет двугранных углов и геометрии пирамидальных структур, где необходимо использовать поляризационные функции
Словые постоянные	HF/6-31G	Учет поляризационных функций слабо влияет на результат в жестких молекулах
Вращательные и инверсионные барьеры	HF/6-31G**	Исключение – молекулы с осью вращения, пронизывающей два гетероатома (например, C-N): в этом случае требуется базис DZ + P
Химическая связь. Энергии реакций	HF/6-31G** MP2/6-31G**	Необходим учет электронной корреляции
Взаимодействие ионов и диполей. Водородные связи	HF/6-31++G**	Для расчетов молекулярных анионов и их взаимодействий необходимо дополнительно включать диффузные функции
Внутри- и межмолекулярные взаимодействия	MP2/6-311+G**	Необходимы как поляризационные, так и диффузные функции, а также учет энергии корреляции электронов

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Семи-эмпирические методы:

```
#{my %$coor,$schnum}=read_pdb($ARGV[0]);
my %$coor=read_pdb($ARGV[0]);
my $mdir=$ARGV[1];
my $sch,$schnum;
foreach my $r ( sort keys %{$coor{"0"}} ){ my $ggg=substr($r,0,1); if ( $ggg ne $sch ){ $schnum++; $sch=$ggg } ;
```

```
my %$qwa=find_quart( $coor{"0"} ); my $sqnum=keys %$qwa;
```

Задача: число двух электронных интегралов пропорционально M^4 , где M это количество или размерность атомного базиса.

Сложно считать большие электронные системы.

- **Основная идея:** это уменьшить стоимость счёта за счёт двух электронных интегралов.

- **Цель:** описание либо больших систем, либо качественная оценка.

```
# foreach my $s ( keys %$qartets ){ print join " ",@{$qartets{$s}} ,"\n";
foreach my $m ( keys %$qartets ){
my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;
foreach my $res ( @{$qartets{$s}} ){
# print "$s $coor{$m}{$res} {"$R"}->x,"\n";
$nx=$nx+ $coor{$m}{$res} {"$R"}->x;
$ny=$ny+ $coor{$m}{$res} {"$R"}->y;
$nz=$nz+ $coor{$m}{$res} {"$R"}->z;
$ox=$ox+ $coor{$m}{$res} {"O6"}->x;
$oy=$oy+ $coor{$m}{$res} {"O6"}->y;
$oz=$oz+ $coor{$m}{$res} {"O6"}->z;
```



```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Основные приближения

```
my ($my $coor, my $chnum)=read_pdb($ARGV[0]);
my $my $coor=read_pdb($ARGV[0]);
my $mdir=$ARGV[1];
my $ch_n=$ARGV[2];
foreach my $m ($mdir) {
```

- Рассматриваются только валентные электроны
- В молекулярных орбиталях учитываются АО с n , соответствующим высшим заселённым орбиталям
- Для двухэлектронных интегралов вводят приближение нулевого дифференциального перекрытия.

```
my %qwa;
```

```
if ($qnum) {
#system("cat $mdir/$chnum/$qnum/$qnum.out");
my $filename=$ARGV[0];
$filename=$mdir.$chnum.$qnum.$qnum.out;
$filename=$mdir.$chnum.$qnum.$qnum.out;
print "INFO: chain $chnum qnum $qnum\n";
}
foreach my $m (sort { $a-<=>$b } keys %coor) {
my %qarts= %qwa; #find_quart($coor{$m});
my %q= find_q($coor{$m});
```

$$\chi_{\mu}(r)\chi_{\nu}(r)dr = 0, \mu \neq \nu$$

- Двухэлектронные интегралы зависят только от природы атомов, на которых центрированы орбитали χ_{μ} и χ_{ν} , и не зависят от конкретного вида орбиталей. Для обозначения среднего значения интегралов используют γ_{AB}

```
#
foreach my $s ($mdir) {
foreach my $q (keys %qarts) {
my $sox, my $soy, my $soz;
foreach my $res ($mdir) {
print "sq $coor{$m} ($res) {"$q"}->x, \n";
$nx=$nx+ $coor{$m} {$res} {"$q"}->x;
$ny=$ny+ $coor{$m} {$res} {"$q"}->y;
$nz=$nz+ $coor{$m} {$res} {"$q"}->z;
$sox=$sox+ $coor{$m} {$res} {"$q"}->x;
$soy=$soy+ $coor{$m} {$res} {"$q"}->y;
$soz=$soz+ $coor{$m} {$res} {"$q"}->z;
```

```
#!/usr/bin/perl
```

Приближение нулевого дифференциального перекрывания:

```
use Math::VectorReal qw( :all );
my %$coor; my $schnum; read_pdb($ARGV[0]);
my $coor=read_pdb($ARGV[0]);
my $kdir=$ARGV[1];
my $sch, my $schnum;
foreach my $r ( sort keys %{$coor{"0"}} ){ my $sggg=substr($r,0,1); if ( $sggg ne $sch ){ $schnum++; $sch=$sggg } };
my %$qwa=find_quart( $coor{"0"} ); my $sqnum=keys %$qwa;
```

Основная идея: уменьшение количества интегралов, за счёт обнуления перекрывания некоторых орбиталей.

Если два ядра далеко друг от друга то вероятно центрированные к ядрам функции не перекрываются.

Основной результат с точки зрения уравнений это матрица обмена, $S=1$ Тогда уравнение Рутан-Хола будет выглядеть как: $FC=CE$

Это приближение называют **ZDO**.

```
my $sr;
foreach my $res ( @{$ $qartets{$sq} } ){
# print "$q $coor{$m} {$res} {"$R"}->x,"n";
$nx=$nx+ $coor{$m} {$res} {"$R"}->x;
$ny=$ny+ $coor{$m} {$res} {"$R"}->y;
$nz=$nz+ $coor{$m} {$res} {"$R"}->z;

$ox=$ox+ $coor{$m} {$res} {"$O6"}->x;
$oy=$oy+ $coor{$m} {$res} {"$O6"}->y;
$oz=$oz+ $coor{$m} {$res} {"$O6"}->z;
```

Приближение нулевого дифференциального перекрытия:

Это приближение слишком сильное для атомов объединённых в молекулу.

Существуют несколько модификаций:

- CNDO: идея применяется ко всем парам функций, не нулевыми оставались только кулоновские интегралы
- INDO, MINDO/x: учитывалось перекрытие всех функций² центрированных на одном ядре.
- NDDO, MNDO: пренебрегается двухатомное перекрытие.

AM1, PM3: Улучшенные и современные варианты параметризации MNDO.

Программы: MOPAC, OPENMOPAC, AMPAC

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Источники параметризации

```

#(my %$coor,my $schnum)=read_pdb($ARGV[0]);
my %$coor=read_pdb($ARGV[0]);
my $mdir=$ARGV[1];
my $sch, my $schnum;
foreach my $r ( sort keys %{$coor{"0"}}){ my $ggg=substr($r,0,1); if ( $ggg ne $sch){ $schnum++; $sch=$ggg } ;

```

```
my %$qwa=find_quart( %$coor{"0"} ); my $qnum=keys %$qwa;
```

Метод

источник

CNDO/2

электронная плотность, электроны спарены

CNDO/S

спектры, электроны спарены

INDO

электронная плотность, электроны не спарены

INDO/S

спектры, электроны не спарены

ZNDO и ZNDO/S

тоже самое только для переходных элементов

MINDO/3

теплоты образования

```
# foreach my $q ( keys %$qartets){ print join " ",@{$qartets{$q}} , "\n";
```

```
foreach my $q ( keys %$qartets){
```

```

my $nx; my $ny; my $nz;
my $ox; my $oy; my $oz;
my $r;
```

```
foreach my $res ( @{$qartets{$q}}){
```

```
# print "$q $coor{$m}{$res}{\"R\"}->x,\"n\";
```

```
$nx=$nx+ $coor{$m}{$res}{\"N9\"}->x;
```

```
$ny=$ny+ $coor{$m}{$res}{\"N9\"}->y;
```

```
$nz=$nz+ $coor{$m}{$res}{\"N9\"}->z;
```

```
$ox=$ox+ $coor{$m}{$res}{\"O6\"}->x;
```

```
$oy=$oy+ $coor{$m}{$res}{\"O6\"}->y;
```

```
$oz=$oz+ $coor{$m}{$res}{\"O6\"}->z;
```

```
#!/usr/bin/perl
use Math::VectorReal qw( :all );
```

Сравнение методов

```
my ($coor,$snum)=read_pdb($ARGV[0]);
```

```
my %coor=read_pdb($ARGV[0]);
```

```
my %dir=$ARGV[1];
```

```
my $schnum=$ARGV[2];
```

```
my %qcart=sort keys %qcart;
```

```
my %qwa=find_quartets($schnum,%qcart);
```

Метод

Параметризуемое св-во

Хорошо воспроизводимые св-ва

Плохо воспроизводимые св-ва

CNDO/2

Разности энергий между занятыми МО

Дипольные моменты, длины связей, валентные углы, силовые константы

Теплоты образования, потенциал ионизации, сродство к электрону, спектры, ре-акции

CNDO/S,
INDO/S,
ZINDO

Электронный спектр

Спектр

Теплоты образования, геометрия молекул, реакции

INDO

Спиновые плотности

Спиновые плотности, константы сверхтонкого взаимодействия, геометрия молекул

Теплоты образования, потенциалы ионизации, срод-ство к электрону, электрон-ные спектры

MINDO/3

Потенциал атоматомного взаимодействия

Теплоты образования, потенциалы ионизации, длины связей

Электронные спектры, водородная связь

MNDO

Теплоты образования

Теплоты образования, геометрия молекул

Электронные спектры, водородная связь

AM1

Теплоты образования

Теплоты образования, геометрия молекул

Электронные спектры

PM3

Теплоты образования, параметры межмолекулярного взаимодействия

Теплоты образования, геометрия молекул, водородная связь, межмолекулярные взаимодействия

Электронные спектры