Direct Inversion in the Iterative Subspace (DIIS)

Eric Berquist

University of Pittsburgh

October 7th, 2014

The Self-Consistent Field Procedure

- Calculate all one- and two-electron integrals.
- Generate a suitable start guess for the MO coefficients.
- Form the initial density matrix.
- Form the Fock matrix as the core (one-electron) integrals + the density matrix times the two-electron integrals.
- Diagonalize the Fock matrix. The eigenvectors contain the new MO coefficients.
- Form the new density matrix. If it is sufficiently close to the previous density matrix, we are done, otherwise go to step 4.

The Self-Consistent Field Procedure

$$\mathbf{S}\mathbf{L}_S = \mathbf{L}_S\Lambda_S \tag{1}$$

$$\mathbf{S}^{-1/2} \equiv \mathbf{L}_S\Lambda^{-1/2}\tilde{\mathbf{L}}_S \tag{2}$$

$$\mathbf{F}_0^{'} \equiv \tilde{\mathbf{S}}^{-1/2}\mathbf{H}^{\text{core}}\mathbf{S}^{-1/2} \tag{3}$$

$$\mathbf{F}_0^{'}\mathbf{C}_0^{'} = \mathbf{C}_0^{'}\epsilon_0 \tag{4}$$

$$\mathbf{C}_0 = \mathbf{S}^{-1/2}\mathbf{C}_0^{'} \tag{5}$$

$$(\mathbf{D}_0)_{\mu\nu} = \sum_{m}^{\text{occ}}(\mathbf{C}_0)_{\mu}^{m}(\mathbf{C}_0)_{\nu}^{m} \tag{6}$$

$$E_{\text{elec}}^{0} = \sum_{\mu\nu}^{\text{AO}} D_{\mu\nu}^{0}(H_{\mu\nu}^{\text{core}} + F_{\mu\nu}) = \text{tr}(\mathbf{D}^0(\mathbf{H}^{\text{core}} + \mathbf{F}^0)) \tag{7}$$

$$\mathbf{F}^{'} \equiv \tilde{\mathbf{S}}^{-1/2}\mathbf{F}\mathbf{S}^{-1/2} \tag{8}$$

Rinse and repeat from steps 4-8 until your chosen error metric is acceptable.

Techniques for SCF Convergence

- Damping
- Level Shifting
- **Extrapolation**
- Direct Minimzation

Some Working Equations

$$\mathbf{F}' = \sum_i c_i \mathbf{F}_i \tag{9}$$

$$\begin{aligned}\mathbf{e}' &= \sum_i c_i \mathbf{e}_i \\ &\approx \mathbf{0}\end{aligned} \tag{10}$$

where

$$\mathbf{e}_i \equiv \mathbf{F}_i \mathbf{P}_i \mathbf{S} - \mathbf{S} \mathbf{P}_i \mathbf{F}_i \tag{11}$$

$$= [\mathbf{F}_i, \mathbf{P}_i] \tag{12}$$

Derivation Part 1

Suppose we have a vector from the *i*th step of an iterative procedure that can be formed as the sum of the final quantity plus some error

$$\mathbf{p}_i = \mathbf{p}_f + \mathbf{e}_i \tag{13}$$

and that a good approximation to the final vector is a linear combination of the previous guesses

$$\mathbf{p} = \sum_i^m c_i \mathbf{p}_i \tag{14}$$

where m is a fixed integer (defaults: ORCA 5, GAMESS 10, Q-Chem 15). Every time there isn't a limit on the sum, assume it's m.

Derivation Part 2

Make a substitution for $\mathbf{p}_i$

$$\mathbf{p} = \sum_i c_i(\mathbf{p}_f + \mathbf{e}_i) \tag{15}$$

$$= \mathbf{p}_f \sum_i c_i + \sum_i c_i \mathbf{e}_i. \tag{16}$$

At convergence, the error must drop to zero, leaving us with

$$\mathbf{p} = \mathbf{p}_f \sum_i c_i \tag{17}$$

$$= \mathbf{p}_f. \tag{18}$$

So, we must minimize $\mathbf{e}'$ under the constraint $\sum_i c_i = 1$.

Derivation Part 3

Constrained minimization? Lagrange multipliers! To minimize the norm of the error

$$\langle \mathbf{e} \,|\, \mathbf{e} \rangle = \sum_{ij}^{m} c_i^* c_j \langle \mathbf{e}_i \,|\, \mathbf{e}_j \rangle \,, \tag{19}$$

define the Lagrangian

$$\mathcal{L} = \mathbf{c}^\dagger \mathbf{B} \mathbf{c} - \lambda \left(1 - \sum_{i}^{m} c_i \right) \tag{20}$$

where

$$B_{ij} = \langle \mathbf{e}_i \,|\, \mathbf{e}_j \rangle \,. \tag{21}$$

Derivation Part 4

The messy part is minimizing the Lagrangian

$$\frac{\partial \mathcal{L}}{\partial c_k} = 0 = \sum_j c_j B_{kj} + \sum_i c_i B_{ik} - \lambda \tag{22}$$

$$= 2 \sum_i c_i B_{ki} - \lambda, \tag{23}$$

where the 2 can be absorbed into λ and the coefficients are assumed to be real. We now need to solve $m+1$ linear equations

$$\begin{pmatrix} B_{11} & B_{12} & \cdots & B_{1m} & -1 \\ B_{21} & B_{22} & \cdots & B_{2m} & -1 \\ \vdots & \vdots & \ddots & \vdots & -1 \\ B_{m1} & B_{m2} & \cdots & B_{mm} & -1 \\ -1 & -1 & \cdots & -1 & 0 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -1 \end{pmatrix} \tag{24}$$

Some More Working Equations

All of this results in solving the linear system

$$\mathbf{B}\mathbf{c} = \mathbf{z}, \tag{25}$$

where $\mathbf{z}$ is the "zero vector". Finding the $\{c_m\}$ as

$$\mathbf{c} = \mathbf{B}^{-1}\mathbf{z} \tag{26}$$

is done with a math library, typically involving a call to the LAPACK routine `DGESV`.

A 1-by-1 Example

$$\begin{pmatrix} B_{11} & -1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} c_1 \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix} \tag{27}$$

$$\begin{pmatrix} c_1 \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ -1 & -B_{11} \end{pmatrix} \begin{pmatrix} 0 \\ -1 \end{pmatrix} \tag{28}$$

Surprise, c_1 is 1!

A 2-by-2 Example

$$\begin{pmatrix} B_{11} & B_{12} & -1 \\ B_{21} & B_{22} & -1 \\ -1 & -1 & 0 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \tag{29}$$

Results in the following set of equations:

$$c_1 B_{11} + c_2 B_{12} - \lambda = 0 \tag{30}$$

$$c_1 B_{21} + c_2 B_{22} - \lambda = 0 \tag{31}$$

$$-c_1 - c_2 = -1 \tag{32}$$

Outline of the algorithm

- Compute the Error Matrix in Each Iteration
- Build the B Matrix and Solve the Linear Equations
- Compute the New Fock Matrix

Computing the Error Matrix

```
/*!
 * @brief Build the DIIS error matrix.
 *
 * The formula for the error matrix at the ith iteration is:
 *   $e_i = F_i D_i S - S D_i F_i$
 */
arma::mat build_error_matrix(const arma::mat &F,
                             const arma::mat &D,
                             const arma::mat &S) {
  return (F*D*S) - (S*D*F);
}
```

Building the B Matrix

```
/*!
 * @brief Build the DIIS B matrix, or ``A'' in Ax = b.
 */
arma::mat build_B_matrix(const deque< arma::mat > &e) {
  int NErr = e.size();
  arma::mat B(NErr + 1, NErr + 1);
  B(NErr, NErr) = 0.0;
  for (int a = 0; a < NErr; a++) {
    B(a, NErr) = B(NErr, a) = -1.0;
    for (int b = 0; b < a + 1; b++)
      B(a, b) = B(b, a) = arma::dot(e[a].t(),  e[b]);
  }
  return B;
}
```

Computing the New Fock Matrix

```
/*!
 * @brief Build the extrapolated Fock matrix from the Fock vector.
 *
 * The formula for the extrapolated Fock matrix is:
 *   $F' = \sum_k^m c_k F_k$
 * where there are m elements in the Fock and error vectors.
 */
void build_extrap_fock(arma::mat &F_extrap,
                       const arma::vec &diis_coeffs,
                       const deque< arma::mat > &diis_fock_vec) {
  const int len = diis_coeffs.n_elem - 1;
  F_extrap.zeros();
  for (int i = 0; i < len; i++)
    F_extrap += (diis_coeffs(i) * diis_fock_vec[i]);
}
```

Algorithm: Form RHS

```
/*!
 * @brief Build the DIIS "zero" vector, or "b" in Ax = b.
 */
arma::vec build_diis_zero_vec(const int len) {
  arma::vec diis_zero_vec(len, arma::fill::zeros);
  diis_zero_vec(len - 1) = -1.0;
  return diis_zero_vec;
}
```

Algorithm: Data Structures

```
/*!
 * Prepare structures necessary for DIIS extrapolation.
 */
int NErr;
deque< arma::mat > diis_error_vec;
deque< arma::mat > diis_fock_vec;
int max_diis_length = 6;
arma::mat diis_error_mat;
arma::vec diis_zero_vec;
arma::mat B;
arma::vec diis_coeff_vec;
```

Algorithm: Main Loop

```cpp
// Start collecting elements for DIIS once we're past the first iteration.
if (iter > 0) {
  diis_error_mat = build_error_matrix(F, D, S);
  NErr = diis_error_vec.size();
  if (NErr >= max_diis_length) {
    diis_error_vec.pop_back();
    diis_fock_vec.pop_back();
  }
  diis_error_vec.push_front(diis_error_mat);
  diis_fock_vec.push_front(F);
  NErr = diis_error_vec.size();
  // Perform DIIS extrapolation only if we have 2 or more points.
  if (NErr >= 2) {
    diis_zero_vec = build_diis_zero_vec(NErr + 1);
    B = build_B_matrix(diis_error_vec);
    diis_coeff_vec = arma::solve(B, diis_zero_vec);
    build_extrap_fock(F, diis_coeff_vec, diis_fock_vec);
  }
}
```

Results: DIIS Off

Water, RHF/STO-3G (7 basis functions)

```
   0    -117.839710375889    -117.839710375889
   1     -70.284216222929    -117.839710375889
   2     -76.045949191625      47.555494152959       1.826673084479
   3     -74.714598899044      -5.761732968696       0.479570364860
   4     -74.984773695191       1.331350292581       0.086831688906
   5     -74.935766125703      -0.270174796147       0.031026136359
   6     -74.943904016889       0.049007569488       0.010799283179
   7     -74.942119723104      -0.008137891186       0.005254826831
   8     -74.942250190433       0.001784293785       0.002438579642
   9     -74.942136438892      -0.000130467329       0.001177279531
  10     -74.942111868815       0.000113751541       0.000564543180
...
  23     -74.942079930560       0.000000005264       0.000000043256
  24     -74.942079929335       0.000000002540       0.000000020871
  25     -74.942079928743       0.000000001226       0.000000010070
  26     -74.942079928458       0.000000000591       0.000000004859
  27     -74.942079928320       0.000000000285       0.000000002345
  28     -74.942079928254       0.000000000138       0.000000001131
  29     -74.942079928222       0.000000000066       0.000000000546
  30     -74.942079928206       0.000000000032       0.000000000263
  31     -74.942079928199       0.000000000016       0.000000000127
  32     -74.942079928195       0.000000000007       0.000000000061
  33     -74.942079928193       0.000000000004       0.000000000030
  34     -74.942079928193       0.000000000002       0.000000000014
  35     -74.942079928192       0.000000000001       0.000000000007
  36     -74.942079928192       0.000000000000       0.000000000003
  37     -74.942079928192       0.000000000000       0.000000000002
  38     -74.942079928192       0.000000000000       0.000000000001
  39     -74.942079928192       0.000000000000       0.000000000000
  40     -74.942079928192       0.000000000000       0.000000000000
```

Results: DIIS On

Water, RHF/STO-3G (7 basis functions), 6 error vectors

```
  0    -117.839710375889    -117.839710375889
  1     -70.284216222929    -117.839710375889
  2     -74.576672718926      47.555494152959       1.826673084479
  3     -75.105709804074      -4.292456495997       0.403889812696
  4     -74.954655933663      -0.529037085148       0.088003715430
  5     -74.938944396782       0.151053870411       0.020519848928
  6     -74.942105934721       0.015711536881       0.012108640030
  7     -74.942079965219      -0.003161537938       0.000460862177
  8     -74.942079928865       0.000025969502       0.000001081074
  9     -74.942079928192       0.000000036354       0.000000052668
 10     -74.942079928192       0.000000000673       0.000000000336
```

DIIS Error Matrix

After iteration 0:

```
                1          2          3          4          5          6          7
   1   -0.0000000 -0.0135594 -0.0000000 -0.0102014  0.0000000 -0.0646745 -0.0646745
   2    0.0135594  0.0000000 -0.0000000 -0.0072338  0.0000000  0.1917308  0.1917308
   3    0.0000000  0.0000000  0.0000000  0.0000000 -0.0000000  0.2279110 -0.2279110
   4    0.0102014  0.0072338 -0.0000000  0.0000000  0.0000000  0.1787515  0.1787515
   5   -0.0000000 -0.0000000  0.0000000 -0.0000000  0.0000000  0.0000000 -0.0000000
   6    0.0646745 -0.1917308 -0.2279110 -0.1787515 -0.0000000  0.0000000  0.0000000
   7    0.0646745 -0.1917308  0.2279110 -0.1787515  0.0000000 -0.0000000  0.0000000
```

After iteration 2:

```
                1          2          3          4          5          6          7
   1    0.0000000 -0.0001491 -0.0000000  0.0005689 -0.0000000 -0.0004437 -0.0004437
   2    0.0001491  0.0000000 -0.0000000 -0.0156002 -0.0000000  0.0133253  0.0133253
   3    0.0000000  0.0000000  0.0000000  0.0000000 -0.0000000  0.0010536 -0.0010536
   4   -0.0005689  0.0156002 -0.0000000 -0.0000000 -0.0000000 -0.0044211 -0.0044211
   5    0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
   6    0.0004437 -0.0133253 -0.0010536  0.0044211 -0.0000000  0.0000000  0.0000000
   7    0.0004437 -0.0133253  0.0010536  0.0044211 -0.0000000  0.0000000  0.0000000
```

DIIS B matrix and coefficients

After two iterations:

```
              1          2          3
1  -0.0850416  0.1611509 -1.0000000  0.7287
2   0.1611509 -0.5000366 -1.0000000  0.2713
3  -1.0000000 -1.0000000  0.0000000 -0.0182
```

Just before convergence:

```
              1          2          3          4          5          6          7
1  -0.0000000  0.0000000  0.0000000  0.0000000 -0.0000000  0.0000000 -1.0000000   1.9028e+00
2   0.0000000 -0.0000000 -0.0000000 -0.0000000  0.0000000 -0.0000000 -1.0000000  -9.0514e-01
3   0.0000000 -0.0000000 -0.0000000 -0.0000000  0.0000000  0.0000000 -1.0000000   2.0913e-03
4   0.0000000 -0.0000000 -0.0000000 -0.0000000  0.0000000 -0.0000000 -1.0000000   2.2118e-04
5  -0.0000000  0.0000000  0.0000000  0.0000000 -0.0000001 -0.0000005 -1.0000000   3.1371e-07
6   0.0000000 -0.0000000  0.0000000 -0.0000000 -0.0000005 -0.0000142 -1.0000000   8.1938e-09
7  -1.0000000 -1.0000000 -1.0000000 -1.0000000 -1.0000000 -1.0000000  0.0000000  -1.2659e-27
```

GDIIS

In the case of a nearly quadratic energy function,

$$\mathbf{e}_i = -\mathbf{H}^{-1}\mathbf{g}_i \tag{33}$$

$$\mathbf{x}'_{m+1} = \sum_i c_i \mathbf{x}_i \tag{34}$$

$$\mathbf{g}'_{m+1} = \sum_i c_i \mathbf{g}_i \tag{35}$$

$$\mathbf{x}_{m+1} = \mathbf{x}'_{m+1} - \mathbf{H}^{-1}\mathbf{g}'_{m+1} \tag{36}$$

References

- http://sirius.chem.vt.edu/wiki/doku.php?id=crawdad:programming:project8 (T. Daniel Crawford)
- Pulay, J Comp Chem, 3 (1982) 556.
- http://vergil.chemistry.gatech.edu/notes/diis/diis.html (C. David Sherrill)
- Introduction to Computational Chemistry (2nd Ed.), Frank Jensen
- http://en.cppreference.com/w/cpp/container/deque
- Q-Chem 4.2 manual