The SCF-Anderson method for a Non-linear Eigenvalue Problem in Electronic Structure Computations

by

Peng Ni

A Project Report

Submitted to the Faculty

of

WORCESTER POLYTECHNIC INSTITUTE

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

in

Mathematical Sciences

by

August 2007

APPROVED:

Dr. Homer Walker, Project Advisor

1 Introduction

1.1 Background introduction

In the field of nanomaterials and devices, people often need to identify unknown structures.

Any material has a unique "ground state" — that is, the minimum-energy level where the material's electrons remain unless the material is perturbed by external sources. If it is possible to determine a material's ground state, then it is possible to identify the material itself.

A current approach is to find out the charge density associated with the ground state, and this envolves setting up nonlinear eigenvalue problems in the process.

Dr. Chao Yang from Lawrence Berkeley National Laboratory is working on this nonlinear eigenvalue problem. He is interested in finding out the properties of a particular existing algorithm, improving the performance of it, and developing alternative approaches if possible.

I got this project topic from Dr. Yang, and worked under the instruction from him and my advisor Dr. Walker. Thanks to both of them.

1.2 Problem introduction

In most situations in chemistry, it is legitimate to consider the nuclei as classical objects and as point-like particles with charges $(\lambda_1,\lambda_2,...,\lambda_p)$ at positions $(x_1,x_2,...,x_p)$, while treating the electrons as quantum particles. This is the so-called Born-Oppenheimer approximation. In view of this approximation, the determination of the ground state (that is, the state of minimum energy) structure of a molecular system consisting of p nuclei and n electrons amounts to solving a minimization problem. See details in [3].

The specific mathematical problem under this background is to minimize the

total energy function:

$$\begin{aligned} &\min E_{total}(X) \\ &\text{s.t. } X'X = I_p \end{aligned}$$

This problem is transformed into a nonlinear eigenvalue problem (the Kohn-Sham equation from [7]):

$$\begin{aligned} &H(X)X = X\Lambda_p \\ &X'X = I_p, \end{aligned} \tag{1}$$

where the columns of $X \in \mathbb{R}^{n\times p}$ $(p < n)$ are approximate electron wave functions, $H \in \mathbb{R}^{n\times n}$ is the discrete Hamiltonian and is dependent on X, $\Lambda_p \in \mathbb{R}^{p\times p}$ is a diagonal matrix with the p smallest eigenvalues of H on the diagonal, and $I_p \in \mathbb{R}^{p\times p}$ is identity matrix. $\rho = diag(XX')$ is called the charge density.

Current approaches to this problem include the Self-Consistent Field (SCF) method [7] and the SCF method accelerated by the Anderson Acceleration [1] (SCF-Anderson).

1.3 Algorithm introduction

In the following, some MATLAB notation is used. In particular, for $A \in \mathbb{R}^{p\times p}$, diag$(A)$ denotes the vector in $\mathbb{R}^p$ the components of which are the diagonal entries of A, and [ ] denotes the "empty" matrix.

1.3.1 The SCF method

Use $X \in \mathbb{R}^{n\times p}$ from previous iteration to calculate $H(X) \in \mathbb{R}^{n\times n}$, and view it as a constant matrix in the current iteration. Find the p smallest eigenvalues of $H(X)$, and use the corresponding eigenvectors to form the columns of a new X for next iteration.

Algorithm 1 SCF Method

Given $X \in \mathbb{R}^{n\times p}$, and $tol > 0$, evaluate $H(X) \in \mathbb{R}^{n\times n}$.
for $iter = 1$ to Max_iter **do**
  Find the p smallest eigenvalues $d \in \mathbb{R}^{p\times 1}$ and corresponding eigenvectors for $H(X)$ to form columns of $X_+ \in \mathbb{R}^{n\times p}$.
  Evaluate $\Delta\rho = \mathrm{diag}(X_+X_+') - \mathrm{diag}(XX')$.
  Evaluate $err = \|\Delta\rho\|$.
  Update $X \longleftarrow X_+$.
  if $err < tol$ **then**
    Break.
  end if
  Update $H(X) \longleftarrow H(X_+)$.
end for

In practice, the SCF method is less effective than expected: the convergence is slow, and sometimes it even diverges. So for the following parts of this report, we will focus on the SCF-Anderson method outlined in §1.3.2, in which SCF is augmented with a certain procedure to improve convergence.

1.3.2 The SCF-Anderson method

Actually, in the formula for $H(X)$, X always appears in the form of $\mathrm{diag}(XX')$, which is the electron density ρ. Basically, what we need to determine $H(X)$ is only ρ, so for simplicity, we will just use $H(\rho)$ instead from now on.

Based on the SCF method, instead of directly letting $\rho_+ = \mathrm{diag}(X_+X_+')$, choose ρ_+ to be a linear combination of the previous values of ρ to strengthen self-consistency.

Algorithm 2 SCF-Anderson Method

Given $\rho \in \mathbb{R}^{n\times 1}$, $tol > 0$, and Mdim.
Set $R = [\,]$, $D = [\,]$, and evaluate $H(\rho) \in \mathbb{R}^{n\times n}$.
for $iter = 1$ to Max_iter **do**
  Find the p smallest eigenvalues $d \in \mathbb{R}^{p\times 1}$ and corresponding eigenvectors from $H(\rho)$ to form columns of $X_+ \in \mathbb{R}^{n\times p}$.
  Evaluate X_+; let $\Delta\rho = \mathrm{diag}(X_+X_+') - \rho$
  Evaluate $err = ||\Delta\rho||$.
  if $err < tol$ **then**
    Update $\rho \longleftarrow \mathrm{diag}(X_+X_+')$; break.
  end if
  Let $k = \min\{iter, \mathrm{Mdim}\}$.
  if $k = \mathrm{Mdim}$ **then**
    Delete the first columns of R and D.
  end if
  Update $R \longleftarrow [R, \mathrm{diag}(X_+X_+')] \in \mathbb{R}^{n\times k}$.
  Update $D \longleftarrow [D, \Delta\rho] \in \mathbb{R}^{n\times k}$.
  Find $\alpha \in \mathbb{R}^{k\times 1}$ from

$$\begin{aligned} &\min_{\alpha} ||D\alpha|| \\ &\text{s.t.} \sum \alpha_i = 1 \end{aligned}$$

  Evaluate $\rho_+ = R\alpha$.
  Update $H(\rho) \longleftarrow H(\rho_+)$, $\rho \longleftarrow \rho_+$.
end for

1.4 Project overview

In this project, I tested the convergence of the existing algorithms, improved the approach to a certain subproblem in the SCF-Anderson method, and studied the effects of varying a parameter in the algorithm.

Section 2 gives a new view of the problem, and discusses several approaches to solving the subproblem; section 3 explains the convergence-related properties; section 4 draws a conclusion of the project and suggests the possible future work.

2 Implementation of the SCF and SCF-Anderson method

2.1 Fixed point iteration

As is mentioned in last section, what we are always updating is ρ, and we can view the SCF method iterates as follows:

$$\begin{aligned} \rho &\longrightarrow H(\rho) \\ H(\rho) &\longrightarrow \rho_+ \end{aligned}$$

Based on the SCF method, we define a function S such that:

$$\rho_+ = S(\rho) \tag{2}$$

Now the nonlinear eigenvalue problem becomes a fixed-point problem: $\rho = S(\rho)$. The SCF becomes fixed-point iteration:

Algorithm 3 SCF Method (with new notation)

Given $\rho \in \mathbb{R}^{n\times 1}$, and $tol > 0$, and evaluate $H(\rho) \in \mathbb{R}^{n\times n}$.
for $iter = 1$ to Max_iter **do**
  Let $\rho_+ = S(\rho)$, let $\Delta\rho = S(\rho) - \rho$, evaluate $err = ||\Delta\rho||$.
  Update $\rho \longleftarrow \rho_+$.
  if $err < tol$ **then**
    Break.
  end if
  Update $H(\rho) \longleftarrow H(\rho_+)$.
end for

The SCF-Anderson method is an acceleration to SCF:

Algorithm 4 SCF-Anderson Method (with new notation)

Given $\rho \in \mathbb{R}^{n\times 1}$, $tol > 0$, and Mdim.
Set $R = [\,]$, $D = [\,]$, and evaluate $H(\rho) \in \mathbb{R}^{n\times n}$.
for $iter = 1$ to Max_iter **do**
  Evaluate $S(\rho)$, let $\Delta\rho = S(\rho) - \rho$, and evaluate $err = ||\Delta\rho||$.
  if $err < tol$ **then**
    Update $\rho \longleftarrow S(\rho)$, break.
  end if
  Let $k = \min\{iter, \text{Mdim}\}$.
  if $k = \text{Mdim}$ **then**
    Delete the first columns of R and D.
  end if
  Update $R \longleftarrow [R, S(\rho)] \in \mathbb{R}^{n\times k}$, $D \longleftarrow [D, \Delta\rho] \in \mathbb{R}^{n\times k}$.
  Find $\alpha \in \mathbb{R}^{k\times 1}$ from $\begin{cases} \min_\alpha ||D\alpha|| \\ \text{s.t.} \sum \alpha_i = 1 \end{cases}$, evaluate $\rho_+ = R\alpha$.
  Update $H(\rho) \longleftarrow H(\rho_+)$, $\rho \longleftarrow \rho_+$.
end for

2.2 Linearly constrained least squares problem

In the Anderson subroutine, there is a linearly constrained least-square problem:

$$\begin{aligned} &\min_\alpha ||D\alpha|| \\ &\text{s.t.} \sum \alpha_i = 1 \end{aligned}$$

where $D \in \mathbb{R}^{n\times k}$, $\alpha \in \mathbb{R}^{k\times 1}$.

There are several approaches to this problem. Four are outlined below. The first two deal directly with the constrained least squares problem and are likely to involve ill-conditioned matrices. The second two reduce the constrained problem on $\mathbb{R}^k$ to an unconstrained problem on $\mathbb{R}^{k-1}$, thereby allowing solution approaches that involve much better conditioned matrices.

2.2.1 Lagrange multipliers

This method comes from [6], and treats the constrained least-squares problem using a Lagrange multipliers approach.

Set

$$\begin{aligned}\Phi(\alpha,\lambda) &= \frac{1}{2}|D\alpha|^2 - \lambda(\sum \alpha_i - 1)\\ &= \frac{1}{2}\alpha^T D^T D\alpha - \lambda(\sum \alpha_i - 1)\end{aligned}$$

In order to minimize $\Phi(\alpha,\lambda)$, the gradient should equal zero:

$$\nabla_\alpha \Phi(\alpha,\lambda) = D^T D\alpha + \lambda \begin{pmatrix}1\\ \vdots \\ 1\end{pmatrix} = 0 \tag{3}$$

$$\frac{\partial}{\partial\lambda}\Phi(\alpha,\lambda) = -(\sum \alpha_i - 1) = 0 \tag{4}$$

Solve this equation for minimizer α:

$$\begin{pmatrix}D^T D & -\vec{1}\\ -\vec{1}^T & 0\end{pmatrix}\begin{pmatrix}\alpha\\ \lambda\end{pmatrix} = \begin{pmatrix}\vec{0}\\ -1\end{pmatrix} \tag{5}$$

where $\vec{1} = (1,\ldots,1)^T$, $\vec{0} = (0,\ldots,0)^T$, $\vec{1},\vec{0} \in \mathbb{R}^{k\times 1}$.

This method involves the matrix $\begin{pmatrix}D^T D & -\vec{1}\\ -\vec{1}^T & 0\end{pmatrix}$. This may be ill-conditioned because of the term $D^T D$, which has condition number $\kappa(D^T D) = \kappa(D)^2$.

2.2.2 Matrix calculation

Set

$$f = \frac{1}{2}||D\alpha||^2 = \frac{1}{2}\alpha^T D^T D\alpha \tag{6}$$

The gradient of f should be orthogonal to the constraint hyperplane $\alpha^T\vec{1} = 1$ at the local minimizer point:

$$\nabla f = D^T D\alpha = \lambda\vec{1} \tag{7}$$

$$\Rightarrow \quad \alpha = (D^T D)^{-1}(\lambda\vec{1}) \tag{8}$$

$$1 = \vec{1}^T\alpha = \lambda\vec{1}^T(D^T D)^{-1}\vec{1} \tag{9}$$

$$\Rightarrow \quad \alpha = \frac{(D^T D)^{-1}\vec{1}}{\vec{1}^T(D^T D)^{-1}\vec{1}} \tag{10}$$

However, when solving linear systems with this method, we again may meet with high condition numbers, since $\kappa(D^T D) = \kappa(D)^2$.

2.2.3 Method of elimination

The general approach of the method of direct elimination is mentioned in the book "Numerical Methods for Least Squares Problems" [2]. The basic idea of the method is to use the constraint to express some of the variables in terms of others, and then to plug the expression into the function we want to minimize. Thus the constrained least-squares problem becomes an unconstrained problem in fewer variables.

The specific method comes from [5].

We want to solve α from:

$$\min_{\alpha} ||D\alpha||$$
$$\text{s.t.} \sum \alpha_i = 1,$$

and then let $\rho_m = \sum_{i=1}^{k} \alpha_i S(\rho_{m-k-1+i})$.

Now we change an expression:

$$\begin{aligned} D\alpha &= \sum_{i=1}^{k} \alpha_i d_i \\ &= \alpha_1 d_1 + \alpha_2 d_2 + \cdots + \alpha_k d_k \\ &= \bar{\alpha}_1(d_2 - d_1) + \bar{\alpha}_2(d_3 - d_2) + \cdots + \bar{\alpha}_{k-1}(d_k - d_{k-1}) + d_k \\ &= \bar{D}\bar{\alpha} + d_k \end{aligned}$$

where $\bar{D} = DW$,

$$W = \begin{pmatrix} -1 & & & \\ 1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1 \\ & & & 1 \end{pmatrix} \tag{11}$$

and

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{k-1} \\ \alpha_k - 1 \end{pmatrix} = \begin{pmatrix} -1 & & & \\ 1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1 \\ & & & 1 \end{pmatrix} \begin{pmatrix} \bar{\alpha}_1 \\ \bar{\alpha}_2 \\ \vdots \\ \bar{\alpha}_{k-1} \end{pmatrix} = W\bar{\alpha} \tag{12}$$

Then the minimization problem will become an unconstrained one:

$$\min_{\bar{\alpha}} ||\bar{D}\bar{\alpha} + d_k|| \tag{13}$$

One possibility is to solve the normal equation for $\bar{\alpha}$:

$$\bar{D}^T\bar{D}\bar{\alpha} = -\bar{D}^T d_k \tag{14}$$

Then use equation (12) to calculate α from $\bar{\alpha}$. This involves $\kappa(\bar{D}^T\bar{D}) = \kappa(\bar{D})^2$.

Fortunately, we can improve the condition number by doing QR decomposition to $\bar{D}$, i.e., $\bar{D} = QR$, where $Q \in \mathbb{R}^{n\times(k-1)}$ is orthogonal ($Q^TQ = I_k$) and $R \in \mathbb{R}^{k\times k}$ is upper-triangular. Then

$$\begin{aligned} & (QR)^T(QR)\bar{\alpha} = -(QR)^T d_k \\ \Rightarrow \quad & R^TQ^TQR\bar{\alpha} = -R^TQ^T d_k \\ \Rightarrow \quad & R^TR\bar{\alpha} = -R^TQ^T d_k \\ \Rightarrow \quad & R\bar{\alpha} = -Q^T d_k \end{aligned}$$

Thus one can obtain $\bar{\alpha}$ by solving a linear system with R, which has condition number:

$$\kappa(R) = \kappa(QR) = \kappa(\bar{D}) \leq \kappa(D)\kappa(W) \tag{15}$$

and $\kappa(W) \approx 2k/\pi$ for all but the smallest values of k.

2.2.4 Null-space method

The general approach of the null-space method is described in the book "Numerical methods for least squares problems" [2]. The basic idea of the method is to decompose the vector we want into a constant vector that satisfies the constraint and another vector in the null space of the constraint matrix. Thus the constrained problem becomes an unconstrained problem, which only involves solving for the vector in the null space.

We developed a new specific null-space approach with low condition number and some numerical advantages.

Denote $v = (0, \ldots, 0, 1)^T$, $v \in \mathbb{R}^{k\times 1}$. Set $\alpha = v + \beta$, so we have:

$$\begin{aligned} & 1 = \sum \alpha_i = \vec{1}^T\alpha = \vec{1}^T(v+\beta) = 1 + \vec{1}^T\beta \\ \Rightarrow \quad & \vec{1}^T\beta = 0 \end{aligned} \tag{16}$$

In order to satisfy (16), set $\beta = V\gamma$, where $V \in \mathbb{R}^{k\times(k-1)}$ is such that $\vec{1}^T V = 0$ (i.e. the columns of V are an orthonormal basis ofthe orthogonal complement of $\vec{1}$), and $\gamma \in \mathbb{R}^{(k-1)\times 1}$. Then the minimization problem becomes:

$$\min_{\alpha} ||D\alpha|| = \min_{\gamma} ||D(v + V\gamma)|| = \min_{\gamma} ||Dv + DV\gamma|| \tag{17}$$

V is so chosen that $V = (v_1, \ldots, v_{k-1})$, where

$$v_j = \begin{pmatrix} -1/\sqrt{j(j+1)} \\ \vdots \\ -1/\sqrt{j(j+1)} \\ \sqrt{j/(j+1)} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} j \text{ components} \\ \\ \\ \\ \end{matrix}, \qquad j = 1, \ldots, k-1. \tag{18}$$

The normal equation of this minimization system is:

$$(DV)^T(DV)\gamma = -(DV)^T Dv \tag{19}$$

Do QR decomposition to DV, such that $DV = QR$, and the normal equation becomes (note that $Dv = d_k$):

$$\begin{aligned} & (QR)^T(QR)\gamma = -(QR)^T Dv \\ \Rightarrow \quad & R^T Q^T QR\gamma = -R^T Q^T d_k \\ \Rightarrow \quad & R^T R\gamma = -R^T Q^T d_k \\ \Rightarrow \quad & R\gamma = -Q^T d_k \end{aligned}$$

In this way, we only have to solve one linear system, with condition number much smaller than the original approach:

$$\kappa(R) = \kappa(QR) = \kappa(DV) \leq \kappa(D)\kappa(V) = \kappa(D) \tag{20}$$

Since V is upper-Hessenberg, it will allow inexpensive ($O(nk)$ flops) updating of the QR decomposition of DV when columns are dropped from or added to D. Specifically, we store Q_0, R_0, where $Q_0 R_0 = D$, and update them at each iteration:

- When deleting the first column of D, we also delete the first column of R_0, such that we still have $Q_0 R_0 = D$. Now that R_0 becomes upper-Hessenberg, we use Givens rotations (see details in [4, §5.1.8]) to get $R_0 = G_0 R_0^{new}$ (here G_0 is orthogonal), and update $Q_0^{new} = Q_0 G_0$.
- When adding a last column to D, we apply the Gram-Schmidt process on the last column and update Q_0, R_0.
- When evaluating Q, R, we apply Givens rotations on $R_0 V$ since it is upper-Hessenberg. We get $R_0 V = GR$ (here G is orthonormal), and update $Q = Q_0 G$.

Also Dv is only the last column of D, which saves a lot of calculation.

2.2.5 Method comparison

Here is a table comparing the condition number of the linear systems each method meets with, and the number of operations (for convenience, denote $M = \text{Mdim}$):

Methods	Cond. No.	No. of Operations
Lagrange Multipliers	$\approx \kappa(D)^2$	$O(nM) + O(M^3)$
Matrix Calculation	$\kappa(D)^2$	$O(nM) + O(M^2)$
Method of Elimination	$\kappa(\bar{D}) \lesssim \frac{2M}{\pi}\kappa(D)$	$O(nM) + O(M^2)$
Null-Space Method	$\kappa(DV) \leq \kappa(D)$	$O(nM) + O(M^2)$

The analysis of condition number and number of operations on the four methods indicates that the null-space method and the method of elimination will have better performance, but the difference between these two seems unlikely to be significant.

We also performed some preliminary experiments to support our analytical results with a modification of a code provided by C. Yang at Lawrence Berkeley National Laboratory. Our main interest was in observing the maximum condition numbers encountered by the methods with varying bounds on the maximum allowable value of vector storage Mdim. The following table shows typical results, which were obtained in the case of a water molecule. In the table, the first row indicates Mdim, and the second through fourth rows indicate the maximum condition numbers observed during the iterations.

Table 1: Maximum Observed Condition Numbers

Mixdim	5	10	15	20
Lagrange Multipliers	1.44e+011	4.70e+014	2.61e+016	2.61e+016
Matrix Calculation	9.72e+007	1.56e+016	6.21e+017	6.21e+017
Null-Space Method	1.19e+003	2.04e+007	1.55e+008	1.55e+008
Method of Elimination	1.01e+003	1.51e+007	1.38e+008	1.38e+008

From the table above, the maximum condition number comparison results are consistent with our previous analytical results. We also tracked the time expense of the four methods, which did not show significant difference. The main reason is that the Mdim's are much smaller than n's in our experiments.

3 Empirical convergence studies

3.1 The test problems

We have ten test problems:

bh3: The Bcl-2 homology domain 3.

butyne: Ethylacetylene, an extremely flammable and reactive alkyne with chemical formula C_4H_6. It occurs as a colorless gas with a garlic odor.

c2h6: Ethane, with chemical formula C_2H_6, the only two-carbon alkane, that is, an aliphatic hydrocarbon. It is a colorless, odorless gas at standard temperature and pressure.

ch4: Methane, a chemical compound with the molecular formula CH_4. It is the simplest alkane, and the principal component of natural gas.

co2: Carbon dioxide.

h2co: Formaldehyde, a colorless, volatile liquid, resembling acetic or ethyl aldehyde, and chemically intermediate between methyl alcohol and formic acid.

h2o: Water.

hnco: Isocyanic acid, a strong organic acid with a pKa value of about 3.5.

hooh: Hydrogen peroxide, used as a disinfectant or mild bleach.

ketene: Ketene, an organic compound, that is colorless, has a sharp odor and causes such things as eye, nose, throat, and lung irritation if humans are exposed to concentrated levels.

They are all derived from certain chemical materials. Numerical examples of them are done in Appendix A.

3.2 Details of the SCF-Anderson implementations

The real algorithm applied in the Matlab sample code package is slightly different from, but essentially the same as our theoretical algorithm discussed previously. The real approach is considered to be more efficient by the user community, while the theoretical approach is more straightforward. Let's compare these two directly:

In each iteration of the real approach, we:

- Build H_i with given ρ_{i-1};
- Calculate the residual R_i from H_i;
- Solve α by minimizing $||D\alpha||$ subject to $\sum \alpha_i = 1$, where $D = (R_{i-k+1}, \ldots, R_i)$;
- Use α and $H_{i-k+1}, \ldots, H_i$ to get $H_i^{Anderson}$;
- Apply SCF to get ρ_i from $H_i^{Anderson}$;
- Check the tolerance and go to the next iteration.

Here, the residual is a measure reflecting the size of the off-diagonal elements of $X'_{i-1} H_i X_{i-1}$, since this should be diagonal at the solution.

In each iteration of the theoretical approach, we:

- Calculate H_i from given ρ_{i-1};
- Get $S(\rho_{i-1})$ from H_i;
- Evaluate $\Delta\rho_i$, which is some kind of residual;
- Solve for α by minimizing $||D\alpha||$ subject to $\sum \alpha_i = 1$, where $D = (\Delta\rho_{i-k+1}, \ldots, \Delta\rho_{i-k+1})$;
- Update ρ_i from α and $S(\rho_{i-k}), \ldots, S(\rho_{i-1})$;

- Check the tolerance and go to the next iteration.

We can see that the real approach is mixing the Hamiltonian H, i.e., applying the Anderson Acceleration to H; while the theoretical approach is mixing the charge density ρ. Now let's look at the three critical steps in both approaches:

In the real approach:

- Mix H;
- Update ρ;
- Update H.

In the theoretical approach:

- Mix ρ;
- Update H;
- Update ρ.

Since H depends linearly on ρ, then mixing ρ and updating H would be equivalent to mixing H, and the two sets of algorithms would be identical.

Also, since $\Delta\rho$ is getting smaller in norm as the iteration converges, the D matrix may become more ill-conditioned if the columns from later iterationsare significantly smaller than those from earlier iterations. One possible way to overcome this disadvantage is the strategy of dropping: to delete the first several columns of D as necessary, when ill conditioning occurs.

3.3 Effects of varying Mdim

There is one important variable in the Anderson Acceleration method, Mdim, which is the number of $S(\rho)$'s that are stored. There are four points relating to this variable:

The larger Mdim is,

- the more information is contained in D,
- the more expensive the storage and computation are,
- the less useful the beginning columns of D are,
- the more ill-conditioned D is.

Because of the four conflicting points, there exists an optimum value of Mdim. We performed our experiments of varying Mdim with three different methods on the least squares subproblem:

1. Lagrange multipliers

2. Method of elimination

3. Null space method

We found in our test problems that 6 is a good choice, here is a table of performance data:

```
------------------------------------------------------------
prob =

bh3


Mdim  Iter#_1 Max_Cond_1 Iter#_2 Max_Cond_2 Iter#_3 Max_Cond_3
  1     20    2.618e+000   20    0.000e+000   20    1.000e+000
  2     16    8.252e+000   16    1.000e+000   16    3.090e+001
  3     14    3.960e+003   14    3.189e+001   14    3.494e+002
  4     13    7.477e+005   13    4.114e+002   13    7.119e+003
  5     13    1.651e+008   13    5.764e+003   13    5.631e+004
  6     12    4.347e+010   12    8.851e+004   12    2.001e+006
  7     12    5.768e+012   12    9.018e+005   12    2.178e+007
```

```
 8     12    1.468e+015   12    1.523e+007   12    3.690e+008
 9     12    4.550e+016   12    6.752e+007   12    1.739e+009
10     12    1.466e+019   12    1.116e+009   12    1.787e+010
11     12    9.221e+017   12    1.609e+010   12    3.334e+011
12     12    2.158e+018   12    3.262e+011   12    5.049e+012
-------------------------------------------------------------
```

Here, Iter# is the number of iterations before convergence; the upper limit is set to be 300. Max_Cond is the maximum condition number of the linear systems we solve.

We performed all the experiments without the strategy of dropping; complete data are in Appendix A.

3.4 Local and global convergence

In order to test the convergence properties of the SCF-Anderson method, experiments were performed in this way: We got an initial guess by perturbing the solution (X_*) with random matrices,

$$X_0 = X_* + \varepsilon \cdot \text{randn}(\text{size}(X_*)), \tag{21}$$

where $\text{randn}(n, k)$ is an $n \times k$ matrix with random entries normally distributed, and $\text{size}(X)$ with $X \in \mathbb{R}^{n\times k}$ equals (n, k). If one of the initial guesses made from ε led to divergence, we reduce ε by half until we got convergence for 1000 trials with the same ε, and the last ε, which we denote by ε_r, will be the approximate radius of convergence.

Here is the experiment data:

Problem	bh3	butyne	c2h6	ch4	co2
ε_r	1.0000e-04	6.2500e-06	2.5000e-05	2.5000e-05	5.0000e-05
Problem	h2co	h2o	hnco	hooh	ketene
ε_r	5.0000e-05	1.0000e-04	5.0000e-05	1.0000e-04	1.0000e-04

In these experiments, the SCF-Anderson method was locally convergent but not globally convergent. Beyond the radius of convergence, as the perturbation gets bigger, convergence became less likely.

When the iterates converges, the convergence is linear, we use plots of iteration number versus log residual norm to illustrate this.

Here is an example of convergence under a "good" initial guess with $\varepsilon = \varepsilon_r = 1e-4$ for the bh3 problem:

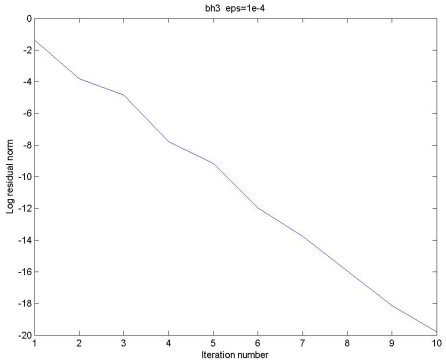


Figure 1: Convergence

Note that the convergence is approximately linear, see Apprendix A2 for additional examples.

Here is an example of divergence under a "bad" initial guess:

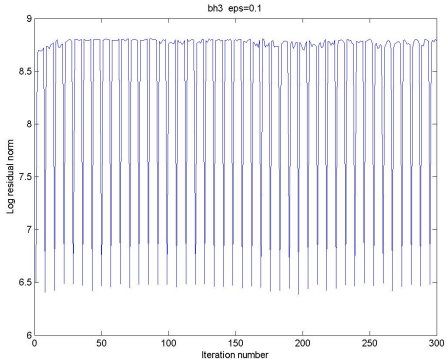


Figure 2: Divergence

4 Concluding summary

In this project, I:

- Learned the properties of SCF method and SCF-Anderson method, and found a new way to look at the problem.
- Studied several different approaches to the linearly constrained least-square subroutine.
- Studied the influence of the variable Mdim in the Anderson Acceleration method.

and I conclude that:

- For every Hamiltonian considered, there exists an optimal Mdim value.
- The Null-space method is better than the other three.
- The experiments suggest that the SCF-Anderson method could have local linear convergence.

For future work, I will:

- Find theoritical support for the Anderson Acceleration method, and improve the performance from it.
- Apply the Anderson Acceleration method as an acceleration algorithm to general fixed point problems.

Appendix

A1 Experiments with Mdim on sample problems

Here, Iter# is the number of iterations before convergence; the upper limit is set to be 300, Max_Cond is the maximum condition number of the linear systems we solve.

We tested three methods:

1. Lagrange multipliers
2. Method of elimination
3. Null space method

---

prob =

bh3

Mdim	Iter#_1	Max_Cond_1	Iter#_2	Max_Cond_2	Iter#_3	Max_Cond_3
1	20	2.618e+000	20	0.000e+000	20	1.000e+000
2	16	8.252e+000	16	1.000e+000	16	3.090e+001
3	14	3.960e+003	14	3.189e+001	14	3.494e+002
4	13	7.477e+005	13	4.114e+002	13	7.119e+003
5	13	1.651e+008	13	5.764e+003	13	5.631e+004
6	12	4.347e+010	12	8.851e+004	12	2.001e+006
7	12	5.768e+012	12	9.018e+005	12	2.178e+007
8	12	1.468e+015	12	1.523e+007	12	3.690e+008
9	12	4.550e+016	12	6.752e+007	12	1.739e+009
10	12	1.466e+019	12	1.116e+009	12	1.787e+010
11	12	9.221e+017	12	1.609e+010	12	3.334e+011
12	12	2.158e+018	12	3.262e+011	12	5.049e+012

prob =

butyne

Mdim	Iter#_1	Max_Cond_1	Iter#_2	Max_Cond_2	Iter#_3	Max_Cond_3
1	300	2.618e+000	300	0.000e+000	300	1.000e+000
2	47	1.727e+001	47	1.000e+000	47	2.434e+001
3	43	2.461e+002	43	1.230e+001	43	2.858e+001
4	33	2.772e+003	33	2.582e+001	33	7.643e+001
5	31	4.565e+004	31	9.483e+001	31	3.202e+002
6	30	4.245e+006	30	9.275e+002	30	4.459e+003
7	31	2.243e+007	31	1.887e+003	31	7.443e+003
8	30	4.003e+008	30	5.914e+003	30	5.300e+004
9	30	1.025e+011	30	1.222e+005	30	4.612e+005
10	30	1.878e+012	30	3.955e+005	30	2.502e+006
11	29	4.670e+013	29	1.706e+006	29	1.455e+007
12	29	5.076e+014	29	6.030e+006	29	5.499e+007
13	29	4.304e+015	29	2.022e+007	29	1.573e+008
14	29	2.325e+018	29	1.177e+008	29	8.703e+008
15	29	4.549e+017	29	2.453e+008	29	1.362e+009
16	29	9.578e+017	29	6.630e+008	29	4.389e+009
17	29	2.382e+018	29	2.590e+009	29	2.363e+010
18	29	1.448e+018	29	9.906e+009	29	5.870e+010
19	29	3.546e+018	29	4.091e+010	29	3.015e+011
20	29	3.891e+018	29	2.390e+011	29	1.797e+012

prob =

c2h6

Mdim	Iter#_1	Max_Cond_1	Iter#_2	Max_Cond_2	Iter#_3	Max_Cond_3
1	28	2.618e+000	28	0.000e+000	28	1.000e+000
2	22	8.244e+000	22	1.000e+000	22	5.291e+001
3	17	1.183e+004	17	5.440e+001	17	3.763e+002
4	17	4.776e+005	17	3.309e+002	17	1.414e+003

```
  5     16    1.926e+007   16    2.164e+003   16    1.228e+004
  6     15    8.454e+008   15    1.126e+004   15    1.086e+005
  7     15    8.942e+010   15    1.076e+005   15    1.778e+006
  8     15    2.251e+013   15    1.848e+006   15    1.916e+007
  9     15    2.052e+015   15    1.580e+007   15    1.433e+008
 10     15    5.051e+016   15    1.060e+008   15    1.096e+009
 11     15    7.850e+017   15    6.765e+008   15    7.900e+009
 12     15    2.492e+018   15    5.020e+009   15    6.188e+010
 13     15    5.898e+018   15    5.556e+010   15    8.937e+011
 14     15    5.898e+018   15    9.268e+011   15    1.408e+013
 15     15    5.898e+018   15    1.575e+012   15    1.678e+013

prob =

ch4


Mdim  Iter#_1 Max_Cond_1 Iter#_2 Max_Cond_2 Iter#_3 Max_Cond_3
  1     23    2.618e+000   23    0.000e+000   23    1.000e+000
  2     18    9.050e+000   18    1.000e+000   18    8.724e+001
  3     16    1.840e+003   16    2.241e+001   16    2.328e+002
  4     15    3.897e+006   14    8.584e+002   15    5.637e+003
  5     14    5.571e+008   14    1.023e+004   14    1.674e+005
  6     14    3.892e+011   14    2.469e+005   14    3.128e+006
  7     14    7.340e+013   14    3.589e+006   14    3.027e+007
  8     14    4.809e+015   14    2.681e+007   14    2.662e+008
  9     14    7.708e+016   14    1.163e+008   14    1.412e+009
 10     14    3.726e+017   14    8.723e+008   14    8.447e+009
 11     14    1.486e+018   14    8.631e+009   14    7.488e+010
 12     14    3.821e+019   14    4.517e+010   14    4.959e+011
 13     14    3.821e+019   14    9.116e+010   14    7.283e+011
 14     14    3.821e+019   14    7.140e+011   14    6.754e+012

prob =

co2
```

Mdim	Iter#_1	Max_Cond_1	Iter#_2	Max_Cond_2	Iter#_3	Max_Cond_3
1	61	2.618e+000	61	0.000e+000	61	1.000e+000
2	24	8.919e+000	24	1.000e+000	24	2.244e+001
3	21	1.687e+003	21	1.718e+001	21	9.381e+001
4	19	1.888e+005	19	2.185e+002	19	1.032e+003
5	18	1.003e+007	18	1.404e+003	18	8.396e+003
6	17	8.304e+008	17	1.131e+004	17	8.228e+004
7	17	7.700e+010	17	1.002e+005	17	7.596e+005
8	17	3.153e+012	17	7.267e+005	17	3.855e+006
9	17	1.235e+014	17	4.528e+006	17	2.902e+007
10	17	4.723e+015	17	2.443e+007	17	2.235e+008
11	16	8.054e+017	16	1.971e+008	16	1.075e+009
12	17	1.205e+020	17	1.240e+009	17	6.599e+009
13	17	1.762e+019	17	6.644e+009	17	6.010e+010
14	17	7.533e+018	17	6.380e+010	17	3.239e+011
15	16	7.085e+018	16	6.301e+010	16	4.778e+011
16	16	8.109e+018	16	1.580e+011	16	1.297e+012

prob =

h2co

Mdim	Iter#_1	Max_Cond_1	Iter#_2	Max_Cond_2	Iter#_3	Max_Cond_3
1	300	2.618e+000	300	0.000e+000	300	1.000e+000
2	30	2.697e+001	30	1.000e+000	30	9.621e+001
3	22	1.802e+003	22	2.218e+001	22	1.009e+002
4	21	3.285e+004	21	9.785e+001	21	6.636e+002
5	20	2.434e+006	20	6.303e+002	20	3.934e+003
6	18	4.514e+008	18	8.158e+003	18	5.055e+004
7	18	2.810e+010	18	6.343e+004	18	5.014e+005
8	18	2.249e+012	18	5.272e+005	18	3.151e+006
9	17	1.769e+014	17	4.936e+006	17	4.808e+007
10	17	9.441e+015	17	3.485e+007	17	3.519e+008
11	17	1.695e+017	17	1.383e+008	17	1.556e+009
12	17	1.503e+017	17	3.308e+008	17	2.030e+009
13	17	7.979e+018	17	1.158e+009	17	9.621e+009
14	17	4.299e+019	17	5.316e+009	17	3.904e+010

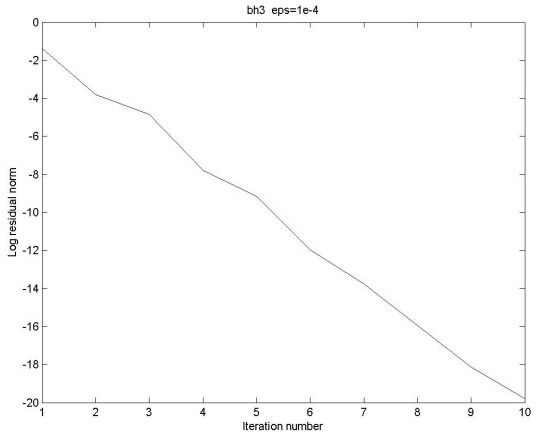
```
 15     17    5.932e+018   17    2.275e+010   17    2.740e+011
 16     17    1.663e+019   17    2.645e+011   17    2.268e+012
 17     17    1.663e+019   17    4.065e+011   17    2.270e+012

prob =

h2o


Mdim  Iter#_1 Max_Cond_1 Iter#_2 Max_Cond_2 Iter#_3 Max_Cond_3
  1     49    2.618e+000   49    0.000e+000   49    1.000e+000
  2     26    1.062e+001   26    1.000e+000   26    1.838e+001
  3     20    4.105e+003   20    3.750e+001   20    2.469e+002
  4     20    9.722e+004   20    1.386e+002   20    9.818e+002
  5     18    7.298e+006   18    1.015e+003   18    6.684e+003
  6     17    1.331e+009   17    1.546e+004   17    1.356e+005
  7     17    4.830e+011   17    2.796e+005   17    2.783e+006
  8     16    2.688e+012   16    5.123e+005   16    1.264e+007
  9     17    6.057e+014   17    8.096e+006   17    8.439e+007
 10     16    2.390e+016   16    6.027e+007   16    7.847e+008
 11     17    4.032e+017   17    9.211e+008   17    9.747e+009
 12     17    2.990e+018   17    7.369e+009   17    9.511e+010
 13     17    1.221e+018   17    2.551e+010   17    4.267e+011
 14     17    2.290e+019   17    1.878e+011   17    2.044e+012
 15     17    9.437e+019   17    8.985e+011   17    1.281e+013
 16     17    7.450e+020   17    1.465e+012   17    1.383e+013
 17     17    1.214e+018   17    2.817e+012   17    3.543e+013

prob =

hnco


Mdim  Iter#_1 Max_Cond_1 Iter#_2 Max_Cond_2 Iter#_3 Max_Cond_3
  1    300    2.618e+000  300    0.000e+000  300    1.000e+000
  2     41    1.539e+001   41    1.000e+000   41    2.667e+001
  3     35    7.285e+002   35    1.359e+001   35    3.918e+001
  4     31    9.958e+003   31    6.137e+001   31    1.973e+002
```

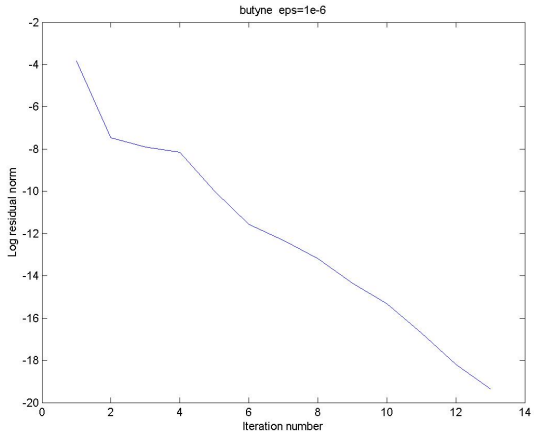
```
  5     26    8.583e+004   26    1.337e+002   26    4.182e+002
  6     26    1.382e+006   26    6.032e+002   26    2.696e+003
  7     24    6.935e+007   24    2.852e+003   24    9.660e+003
  8     24    5.586e+008   24    7.593e+003   24    3.391e+004
  9     23    1.191e+010   23    3.188e+004   23    1.514e+005
 10     23    2.182e+011   23    1.660e+005   23    7.035e+005
 11     23    5.896e+012   23    8.097e+005   23    2.441e+006
 12     23    5.814e+013   23    2.408e+006   23    7.643e+006
 13     22    3.954e+014   22    6.047e+006   22    2.304e+007
 14     22    4.611e+015   22    2.100e+007   22    9.425e+007
 15     22    6.326e+016   22    8.848e+007   22    3.444e+008
 16     22    7.275e+017   22    3.334e+008   22    1.511e+009
 17     22    2.729e+018   22    1.602e+009   22    4.844e+009
 18     22    8.779e+018   22    5.307e+009   22    1.835e+010
 19     22    4.732e+018   22    1.807e+010   22    6.928e+010
 20     22    3.499e+018   22    6.258e+010   22    2.759e+011

prob =

hooh


Mdim  Iter#_1 Max_Cond_1 Iter#_2 Max_Cond_2 Iter#_3 Max_Cond_3
  1     59    2.618e+000   59    0.000e+000   59    1.000e+000
  2     47    2.284e+001   47    1.000e+000   47    9.985e+000
  3     28    8.450e+002   28    1.486e+001   28    4.892e+001
  4     26    1.800e+004   26    7.597e+001   26    2.336e+002
  5     21    9.884e+005   21    4.470e+002   21    1.419e+003
  6     22    1.425e+007   22    1.382e+003   22    4.346e+003
  7     21    9.749e+007   21    3.480e+003   21    1.580e+004
  8     21    1.598e+009   21    1.255e+004   21    6.132e+004
  9     21    3.461e+010   21    6.247e+004   21    4.011e+005
 10     21    1.588e+012   21    4.155e+005   21    5.539e+006
 11     20    4.408e+013   20    1.982e+006   20    1.978e+007
 12     20    3.658e+015   20    1.792e+007   20    1.248e+008
 13     20    5.268e+016   20    7.694e+007   20    7.710e+008
 14     19    2.966e+017   19    9.018e+007   19    1.062e+009
 15     19    1.843e+018   19    3.227e+008   19    2.113e+009
```

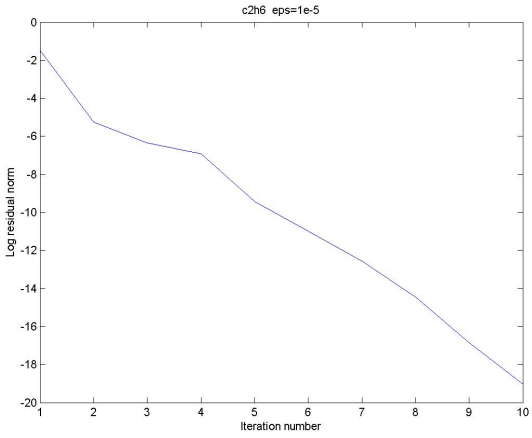
```
 16     20    1.990e+018   20    1.716e+009   20    1.793e+010
 17     20    2.339e+018   20    1.325e+010   20    1.179e+011
 18     20    1.315e+019   20    4.736e+010   20    3.457e+011
 19     20    1.315e+019   20    3.407e+011   20    3.054e+012
 20     20    1.315e+019   20    6.071e+011   20    3.798e+012

prob =

ketene


Mdim  Iter#_1 Max_Cond_1 Iter#_2 Max_Cond_2 Iter#_3 Max_Cond_3
  1    300    2.618e+000  300    0.000e+000  300    1.000e+000
  2     52    1.843e+005   52    1.000e+000   52    4.061e+002
  3     37    7.978e+004   37    3.079e+001   37    9.144e+002
  4     29    1.070e+006   29    3.119e+002   29    9.179e+002
  5     26    2.325e+006   26    5.756e+002   26    2.870e+003
  6     26    1.182e+008   26    5.099e+003   26    2.020e+004
  7     25    1.492e+008   25    5.146e+003   25    3.539e+004
  8     24    1.760e+010   24    5.356e+004   24    3.415e+005
  9     24    3.180e+012   24    7.822e+005   24    3.108e+006
 10     24    3.825e+013   24    2.196e+006   24    1.506e+007
 11     24    5.413e+014   24    6.580e+006   24    4.604e+007
 12     23    1.210e+016   23    3.826e+007   23    2.635e+008
 13     23    3.725e+017   23    1.782e+008   23    1.354e+009
 14     23    5.336e+017   23    4.882e+008   23    4.346e+009
 15     23    6.295e+018   23    1.558e+009   23    7.138e+009
 16     23    6.480e+018   23    5.510e+009   23    2.866e+010
 17     23    7.872e+018   23    2.552e+010   23    1.566e+011
 18     23    6.101e+018   23    9.034e+010   23    6.537e+011
 19     23    5.391e+018   23    1.340e+011   23    7.622e+011
 20     23    1.444e+020   23    7.638e+011   23    4.785e+012
-------------------------------------------------------------
```

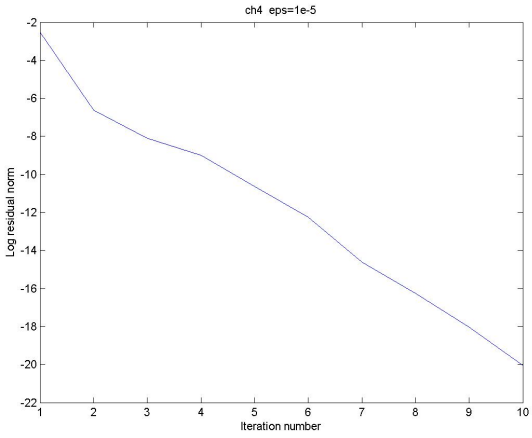
A2 Experiments with initial guess of sample problems

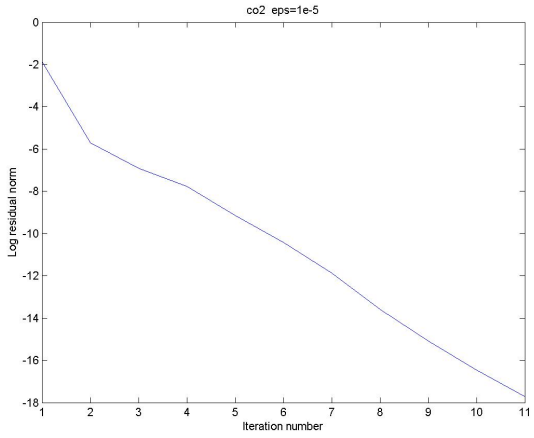
Plots of log residual norm of experiments on sample problems are as following:

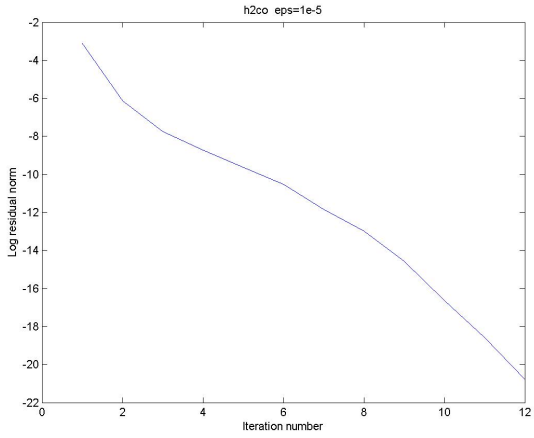


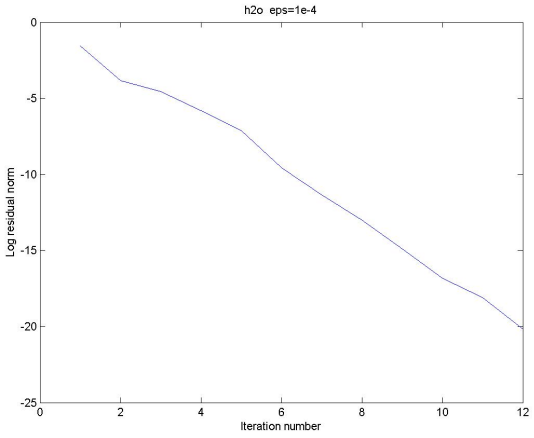


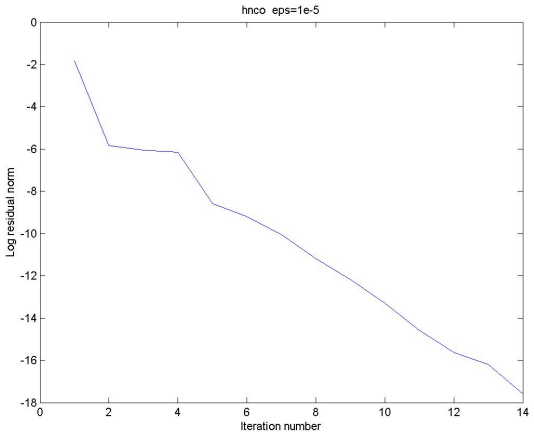

c2h6 eps=1e-5
Log residual norm
Iteration number
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
-20
1
2
3
4
5
6
7
8
9
10

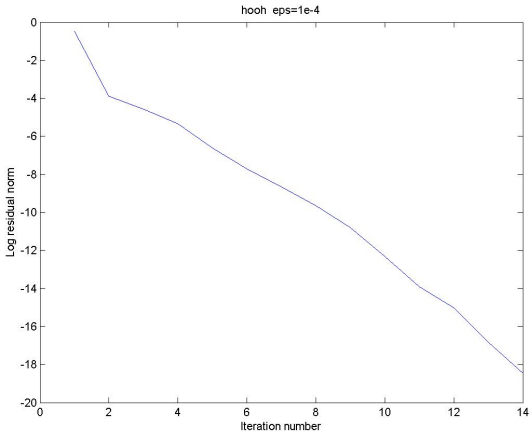

ch4 eps=1e-5
Log residual norm
Iteration number
-2
-4
-6
-8
-10
-12
-14
-16
-18
-20
-22
1
2
3
4
5
6
7
8
9
10

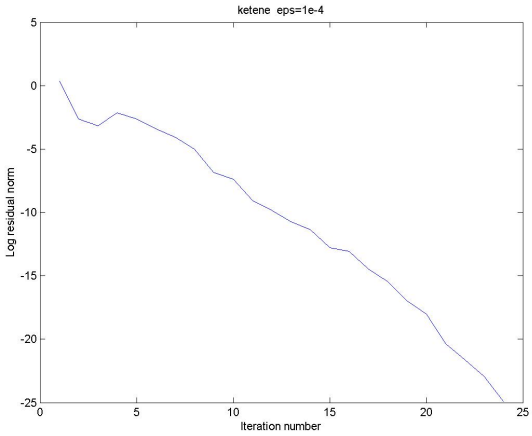

co2 eps=1e-5
Log residual norm
Iteration number
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
1
2
3
4
5
6
7
8
9
10
11


h2co eps=1e-5
Log residual norm
Iteration number
-2
-4
-6
-8
-10
-12
-14
-16
-18
-20
-22
0
2
4
6
8
10
12


h2o eps=1e-4
Log residual norm
Iteration number
0
-5
-10
-15
-20
-25
0
2
4
6
8
10
12


hnco eps=1e-5
Log residual norm
Iteration number
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
0
2
4
6
8
10
12
14


hooh eps=1e-4
Log residual norm
Iteration number
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
-20
0
2
4
6
8
10
12
14


ketene eps=1e-4
Log residual norm
Iteration number
5
0
-5
-10
-15
-20
-25
0
5
10
15
20
25

References

[1] D. G. Anderson. Iterative procedures for nonlinear integral equations. *J. Assoc. Comput. Mach.*, 12:547–560, 1965.

[2] Åke Björck. *Numerical methods for least squares problems*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996.

[3] Claude Le Bris. Computational chemistry from the perspective of numerical analysis. *Acta Numerica*, 14:363–444, 2005.

[4] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[5] Georg Kresse and Jürgen Furthmüller. Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. *Computational Materials Science*, 6:15–50, 1996.

[6] Peter Pulay. Convergence acceleration of iterative sequences. the case of scf iteration. *Chem. Phys. Letters*, 73(2):393–398, 1980.

[7] Chao Yang, Juan C. Meza, and Lin-Wang Wang. A trust region direct constrained minimization algorithm for the Kohn-Sham equation. *SIAM J. Sci. Comput.*, 29(5):1854–1875 (electronic), 2007.