1 Some Code

1.1 Outline of the algorithm

```
/*!
 * @brief Build the DIIS error matrix.
 *
 * The formula for the error matrix at the \textit{i}th 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);
}
```

```
/*!
 * @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;
}
```

```
/*!
 * @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]);
}
```

```
/*!
 * @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;
}
```

```
/*!
 * 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;
```

```
// 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);
  }
}
```