Simultaneous diagonalization and its C++ implementation

Yu Zhai
yuzhai@mail.huiligroup.org

Institute of Theoretical Chemistry, Jilin University, Changchun, China.

May 1, 2020

Introduction

> …If they (observables) do commute there exist so many simultaneous eigenstates that they form a complete set …
>
> P. A. M. Dirac, *The Principles of Quantum Mechanics*

Thus we do need an algorithm to obtain the simultaneous eigenstates of two (or more) matrices, which is described in References [1], [2] and [3].
Consider a set of matrices $\{\mathbf{A}\}$ of $N \times N$ complex matrices. In this work, however, I prefer limit us to real symmetric matrices, with which are easier to deal.
When the matrices in $\{\mathbf{A}\}$ are *normal commuting matrices*, their off-diagnoal terms can be set to zero by *one* unitary transform, thus simultaneously diagonalizing the set $\{\mathbf{A}\}$.

Mathematical description of the problem I

Define

$$\mathrm{off}(\mathbf{A}) \equiv \sum_{0 \le i \ne j < N} A_{ij}^2, \tag{1}$$

where A_{ij} denotes the row i column j entry of matrix $\mathbf{A}$.

```
double simultaneousDiagonalization::off_(const Eigen::MatrixXd & A){
  double res=0.0;
  for(int i=0; i!=A.rows(); ++i){
    for(int j=i; j!=A.rows(); ++j){
      if(i != j) res += pow(A(i,j),2);
    }
  }
  return res;
}
```

Mathematical description of the problem II

Simultaneous diagonalization may be obtained by minimizing the composite objective

$$\mathrm{offsum}(\{\mathbf{A}\}) \equiv \sum_{\mathbf{A}\in\{\mathbf{A}\}} \mathrm{off}(\mathbf{A}). \tag{2}$$

```
double simultaneousDiagonalization::offsum_()const{
  double res=0.0;
  for(auto && i : this -> matrices_){
    res += off_(i);
  }
  return res;
}
```

Extended Jacobi technique I

The extended Jacobi technique for simultaneous diagonalization constructs **U** as a product of plane rotations globally applied to all $\mathbf{A} \in \{\mathbf{A}\}$.
It is desired, for each choice of $i \neq j$, to find the c and s so that

$$O(c,s) \equiv \sum_{\mathbf{A}\in\{\mathbf{A}\}} \mathrm{off}(\mathbf{R}'(i,j,c,s)\mathbf{A}\mathbf{R}(i,j,c,s))$$

is minimized. $'$ in this document means adjoint (conjugate transpose).
Let us define yet another 2×2 real symmetric matrix $\mathbf{G}_{ij}$ for (i,j)

$$\mathbf{G}_{ij} \equiv \sum_{\mathbf{A}\in\{\mathbf{A}\}} \mathbf{h}(\mathbf{A})\mathbf{h}'(\mathbf{A}), \tag{3}$$

where

$$\mathbf{h}(\mathbf{A}) \equiv [A_{ii} - A_{jj}, A_{ij} + A_{ji}]'. \tag{4}$$

Extended Jacobi technique II

```
Eigen::Matrix2d simultaneousDiagonalization::G_(int i, int j) const {
  Eigen::Matrix2d res=Eigen::Matrix2d::Zero();
  for (auto && A: this -> matrices_){
    Eigen::Vector2d h;
    h << A(i,i)-A(j,j), A(i,j)+A(j,i);
#ifndef NDEBUG
    std::cout << "h   " << std::endl << h << std::endl;
#endif
    res += h*h.adjoint();
  }
  return res;
}
```

Extended Jacobi technique III

Denote $\mathbf{R}(i,j,c,s)$ the rotation matrix equal to the identity matrix but for the following entries:

$$\begin{pmatrix} R_{ii} & R_{ij} \\ R_{ji} & R_{jj} \end{pmatrix} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \text{ with } c,s \in \mathbb{R} \text{ and } c^2+s^2=1. \tag{5}$$

Thus we have a theorem, whose proof is not given here: Under constraint $c^2+s^2=1$, $O(c,s)$ is minimized at

$$c=\sqrt{\frac{x+r}{2r}} \quad s=\frac{y}{\sqrt{2r(x+r)}} \quad r=\sqrt{x^2+y^2},$$

where $(x,y)'$ is any eigenvector associated to the largest eigenvalue of G. If the $(x,y)'$ are normalized, then $r=1$, formula above turns to

$$c=\sqrt{\frac{x+1}{2}} \quad s=\frac{y}{\sqrt{2(x+1)}}. \tag{6}$$

Extended Jacobi technique IV

```
Eigen::MatrixXd simultaneousDiagonalization::R_(int i, int j) const{
  Eigen::MatrixXd res = Eigen::MatrixXd::Identity(r_, c_);
  Eigen::Vector2d xy =
    Eigen::SelfAdjointEigenSolver<Eigen::Matrix2d>(G_(i,j))
    .eigenvectors().col(1);
  double x = xy[0];
  double y = xy[1];
  double c = sqrt((x+1.)/2.);
  double s = y/sqrt(2.*(x+1.));
  res(i,i) = res(j,j) = c;
  res(i,j) = -s;
  res(j,i) =  s;
  return res;
}
```

Extended Jacobi technique V

Then we have the Jacobi-like iteration scheme.

```
void simultaneousDiagonalization::compute_(double eps){
  for(int iter = 0; iter < 1000; ++iter){
    if(offsum_()>eps){
      for(int i=0; i<r_; ++i){
        for(int j=i+1; j<r_; ++j){
          Eigen::MatrixXd Rmat = R_(i,j);
          for(auto && A : matrices_){
            A= Rmat.adjoint() * A * Rmat;
          }
          eigenvectors_ = eigenvectors_ * Rmat;
        }
      }
... ...
}
```

C++ syntax related I

To make the program work, the class is defined as

```
class simultaneousDiagonalization{
  public:
    simultaneousDiagonalization(
        std::initializer_list<Eigen::MatrixXd> matrices,
        double eps = 1.e-8);
    inline Eigen::VectorXd eigenvalues(int i) const {
      return matrices_[i].diagonal();
    }
    inline Eigen::MatrixXd eigenvectors() const {
      return eigenvectors_; // all matrices share the same eigen vectors
    }
  private:
......
};
```

C++ syntax related II

And its constructor

```
simultaneousDiagonalization::simultaneousDiagonalization(
    std::initializer_list<Eigen::MatrixXd> matrices, double eps ):
  matrices_(matrices),
  r_ ( matrices_[0].rows() ), c_ ( matrices_[0].cols() )
{
  if(r_ != c_){
    throw std::runtime_error("Sizes inconsist.");
  }
  for(auto && i : matrices_){
    if(i.rows() != r_ || i.cols() != c_){
      throw std::runtime_error("Sizes inconsist.");
    }
  }
  eigenvectors_.setIdentity(r_,c_);
  compute_(eps); // solve the problem on construction...
}
```

Unit test

Three tests are designed while here I only show the last one.

- Both matrices **A** and **B** are generated from eigenvalues;
- They are both degenerated;
- Use the method presented CAN give reasonable solution.

Results:

```
eigenvectors
           1          0          0          0          0
           0 -0.0326343   0.579077   -0.77784   0.242014
           0   0.202113  -0.758439  -0.436193   0.440065
           0  -0.792972 -0.000725223  0.210606     0.5717
           0    0.57383   0.299066   0.400433   0.648793

A eigen values 0 0 1 1 2
B eigen values 0 1 1 2 2

CONVERGED.
sd.eigenvectors()

           1          0          0          0          0
           0  -0.242032   0.777834  0.0326344   0.579077
           0  -0.440075   0.436183  -0.202113  -0.758439
           0  -0.571695  -0.210619   0.792972 -0.000725226
           0  -0.648783  -0.400448   -0.57383   0.299066

sd.eigenvalues()

A eigen values          0           2           1 1.36642e-14           1
B eigen values 0 2 2 1 1
```

Thank you for your attention.

May 1, 2020

Reference

[1] Richard Dawes and Tucker Carrington. A multidimensional discrete variable representation basis obtained by simultaneous diagonalization. *The Journal of Chemical Physics*, 121(2):726–736, jul 2004.

[2] Angelika Bunse-Gerstner, Ralph Byers, and Volker Mehrmann. Numerical methods for simultaneous diagonalization. *SIAM Journal on Matrix Analysis and Applications*, 14(4):927–949, oct 1993.

[3] Jean-François Cardoso and Antoine Souloumiac. Jacobi angles for simultaneous diagonalization. *SIAM Journal on Matrix Analysis and Applications*, 17(1):161–164, jan 1996.