

IMPLEMENTING THE TRUNCATED DENSITY CONVOLUTION

Kangbo Li & Jalen Harris

March 6, 2024

1 Theory Overview

This document describes the details of what we want to implement. The optimization problem can be written in abstract as

$$\min_U \Omega(U), \tag{1}$$

$$\Omega(U) = 2 \sum_{n,b} w_b (1 - |\hat{\rho}_n(b)|), \quad \hat{\rho}_n(b) = \frac{1}{N_k} \sum_k (U^{k\dagger} S^{k,b} U^{\text{add}(k,b)})_{n,n}, \tag{2}$$

$$\text{s.t } U^{k\dagger} U^k = I, \tag{3}$$

$$U \in \mathbb{C}^{N \times N \times N_k}, S \in \mathbb{C}^{N \times N \times N_b \times N_k}, w \in \mathbb{R}^{N_b}. \tag{4}$$

This is a constrained minimization problem, where the independent variable is a complex tensor. The gradient of the objective function is

$$(\nabla \Omega(U))_{ij}^k = -\frac{4}{N_k} \sum_{b,s} w_b \frac{\hat{\rho}_j^*(b)}{|\hat{\rho}_j(b)|} S_{i,s}^{k,b} U_{s,j}^{\text{add}(k,b)}. \tag{5}$$

The gist of the project is to make Ω and $\nabla \Omega$ fast. The add function is a world of hurt, so we will just tabulate the result of the reference code and not worry about it.

This constrained optimization problem is solved with a manifold conjugate gradient descent. The algorithm for a nonlinear conjugate gradient is roughly

```
function cg(f, grad_f)
    x = zeros
    p = grad_f(x)
    while not converged
        x = x - alpha * p
        p = grad_f(x) + beta * p
    end
end
```

The step length alpha is determined by a (quadratic) line search whereas the formula for beta has many variants. The one implemented in the reference code is the Fletcher Reeves.

It would be ideal if we could use an off-the-shelf optimizer, but the complication is that we are working on a manifold, which modifies the algorithm

```
function cg(f, grad_f)
    x = zeros
```

```

    p = project(grad_f(x))
    while not converged
        x = retract(x, -alpha * p)
        p = project(grad_f(x)) + beta * p
    end
end

```

The projection and the retraction implicitly used in Wannier90 are

$$\text{project}(X) = (X - X^\dagger)/2, \quad (6)$$

$$\text{retract}(U, \Delta U) = k \rightarrow U^k \exp(U^{k\dagger} \Delta U^k). \quad (7)$$

The purpose of the retraction is to enforce the constraint, so the exponential is unnecessarily expensive and a QR of $U + \Delta U$ is likely fine.

2 Implementation Plan

1. (x) Prepare the parameters of the problem.
2. Read the parameters from Fortran.
3. Serial implementation of the objective function, the gradient, and the retraction.
4. Implement a manifold optimization.
5. Speedup.

2.1 Prepare the data

The S and w are written in a Fortran binary file just as raw Fortran arrays whose dimensions match Eq. 4. The `add` function is tabulated and saved to a Fortran binary file as a $N_k \times N_b$ array, where the k^{th} row and b^{th} column is the new index `add(k, b)`.