Using a user-defined eigendecomposition function

Huo Chen

November 19, 2020

0.1 Initialize Hamiltonian with a custom eigendecomposition function

When defining a Hamiltonian object, a user-defined eigendecomposition routine can be supplied using a keyword argument `EIGS`. All the eigendecomposition calls inside the solver will call this function instead of the default one.

For example:

```
using OpenQuantumTools

# Define a function to construct an eigendecomposition routine for the
# Hamiltonian. The routine should have the signature: (H, t, lvl) -> (w, v).
# The argument of this function is the cache used by the Hamiltonian object.
function build_user_eigen(u_cache)
    EIGS = function(H, t, lvl)
        println("I am the user defined eigendecomposition routine.")
        w, v = eigen(Hermitian(H(t)))
        w[1:lvl], v[:, 1:lvl]
    end
end

H = DenseHamiltonian([(s)->1-s, (s)->s], [σx, σz], EIGS=build_user_eigen)

eigen_decomp(H, 0.5, lvl=2)
```

```
I am the user defined eigendecomposition routine.
([-0.7071067811865476, 0.7071067811865476], Complex{Float64}[-0.38268343236
508984 + 0.0im 0.9238795325112866 - 0.0im; 0.9238795325112868 - 0.0im 0.382
6834323650897 - 0.0im])
```

0.1.1 Constant Hamiltonian

There are two applications for this functionality. First, if the Hamiltonian is constant, one can precalculate that Hamiltonian's eigensystem and build a function that returns those precalculated values. This is particularly helpful for the adiabatic master equation solver.

```
function build_user_eigen(u_cache)
    # note that to keep the unit consistent, the unit of any value inside the routine
should be 1/h
    w, v = eigen(Hermitian(2*π*(σx+σz)))
    EIGS = function(H, t, lvl)
```

```
        w[1:lvl], v[:, 1:lvl]
    end
end

H = DenseHamiltonian([(s)->1.0], [σx+σz], EIGS=build_user_eigen)

print(eigen_decomp(H, 0.5, lvl=2))
print(eigen_decomp(H, 0.0, lvl=2))
```

```
([-1.4142135623730951, 1.4142135623730951], Complex{Float64}[0.382683432365
0897 + 0.0im -0.9238795325112867 + 0.0im; -0.9238795325112867 + 0.0im -0.38
26834323650897 + 0.0im])([-1.4142135623730951, 1.4142135623730951], Complex
{Float64}[0.3826834323650897 + 0.0im -0.9238795325112867 + 0.0im; -0.923879
5325112867 + 0.0im -0.3826834323650897 + 0.0im])
```

0.1.2 Sparse Hamiltonian

Another application is to supply special eigendecomposition algorithms for sparse matrices to take advantage of the sparsity.

For example, the default eigendecomposition algorithm for sparse Hamiltonian is to convert it into dense matrices first and then perform the decomposition.

```
Hd = standard_driver(4, sp=true);
Hp = two_local_term(rand(3), [[1,2],[2,3],[3,4]], 4, sp=true)
H = SparseHamiltonian([(s)->1-s, (s)->s], [Hd, Hp], unit=:ħ)

# the default eigen_decomposition using dense matrices algorithm
w, v = eigen_decomp(H, 0.1, lvl=4)
```

```
([-0.5732797484390657, -0.29691811582796696, -0.2955240607555121, -0.277711
5644819612], Complex{Float64}[0.2411143083590813 + 0.0im -0.000622371548841
455 + 0.0im -0.016752598819094133 + 0.0im -0.020609160452341346 + 0.0im; -0
.24931988852617384 + 0.0im -0.24481005482912538 + 0.0im -0.2278486238668494
3 + 0.0im 0.2727929026491988 + 0.0im; ...@*( ; -0.24931988852617398 + 0.0im
0.24481005482912535 + 0.0im 0.2278486238668492 + 0.0im -0.2727929026491988 + 0.0im;
0.2411143083590814 + 0.0im 0.0006223715488413366 + 0.0im 0.016752598819094303 + 0.0im
0.020609160452341423 + 0.0im])
```

If the Hamiltonian size becomes large, we can use sparse algorithms provided by Arpack instead. Let's first install `Arpack.jl` by running:

```
using Pkg
Pkg.add("Arpack")
using Arpack
```

Next, we can use `Arpack` function to replace the default eigendecomposition routine:

```
function build_user_eigen(u_cache)
    function (H, t, lvl)
        hmat = H(t)
        println("Using sparse matrix algorithm")
        # define all the Arpack routine parameters here
        eigs(hmat, nev = lvl, which=:SR, tol=0.0, maxiter=300)
    end
end

Hd = standard_driver(4, sp=true);
```

```
Hp = two_local_term(rand(3), [[1,2],[2,3],[3,4]], 4, sp=true)
H = SparseHamiltonian([(s)->1-s, (s)->s], [Hd, Hp], unit=:ħ, EIGS =build_user_eigen)

eigen_decomp(H, 0.1, lvl=4)

Using sparse matrix algorithm
([-0.5731989530240889, -0.2982288128055402, -0.28824707134237876, -0.284972
07667448826], Complex{Float64}[-0.23674004525642994 - 0.0497523004389308im
-0.009872337900299016 + 0.01606291433022861im -0.1313092339724105 - 0.14855
65834366283im 0.0014666989873390773 - 0.039708309298891395im; 0.24118386102
559383 + 0.050686194225270494im -0.07164962049182325 + 0.116578436372045711
m -0.07011771152789059 - 0.07932760970312207im -0.01313587089176387 + 0.355
63072503688614im; ...@*( ; 0.24118386102559386 + 0.05068619422527028im 0.0716496204918233
- 0.11657843637204574im 0.07011771152789051 + 0.07932760970312223im 0.013135870891763959 -
0.3556307250368861im; -0.2367400452564299 - 0.04975230043893063im 0.009872337900299183 -
0.01606291433022862im 0.13130923397241043 + 0.14855658343662845im -0.001466698987339105 +
0.039708309298891326im])
```