# Using a user defined eigendecomposition function

Huo Chen

August 14, 2020

## 0.1 Initialize Hamiltonian with a custom eigendecomposition function

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

For example:

```
using QuantumAnnealingTools

# define a function to construct an eigendecomposition routine for the
# Hamiltonian. The routine should have signature: (H, t, lvl) -> (w, v).
# the argument is the cache of the Hamiltonian
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 precalculates the eigensystem of that Hamiltonian and build a function that returns those precalculated values. This is particular helpful for 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))

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

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])
```

### 0.1.2 Sparse Hamiltonian

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

For example, the default eigendecomposition algorithm for sparse Hamiltonian is to first convert it into dense matrices 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)

([-0.5730614258666127, -0.2941008362141902, -0.287913334436190l6, -0.285173
60703298933], Complex{Float64}[0.2449138662418381 + 0.0im -0.04569675946549
0115 + 0.0im 0.08758829485162403 + 0.0im -0.22716568903707304 + 0.0im; -0.2
4623433718823778 + 0.0im 0.08051061335559309 + 0.0im 0.26153955944388574 +
0.0im -0.11515272698366359 + 0.0im; ... ; -0.24623433718823762 + 0.0im -0.080
51061335559304 + 0.0im -0.2615395594438855 + 0.0im 0.11515272698366381 + 0.
0im; 0.24491386624183817 + 0.0im 0.04569675946549007 + 0.0im -0.08758829485
162418 + 0.0im 0.22716568903707304 + 0.0im])
```

If the size of the Hamiltonian becomes very large, we can use sparse algorithms provided by Arpack instead.

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

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
```

```julia
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
([-0.5737761928913452, -0.30615491148489093, -0.2969264601797298, -0.276479
67334551526], Complex{Float64}[0.2061651762191218 + 0.11129709243422096im 3
.112556437126249e-5 + 0.00029986598159737913im 0.05264915624361981 + 0.0497
9768463372329im 0.0022725328138583872 - 0.00010349689471567126im; -0.216281
80822683915 - 0.11675849842108478im -0.020570618478852037 - 0.1981788547978
54im 0.14652820794943514 + 0.13859225883215032im 0.2830380771434224 - 0.012
890270227088179im; ... ; -0.2162818082268392 - 0.11675849842108482im 0.020570
618478851978 + 0.19817885479785385im -0.14652820794943533 - 0.1385922588321
5034im -0.28303807714342216 + 0.012890270227088215im; 0.20616517621912173 +
 0.11129709243422094im -3.112556437119936e-5 - 0.00029986598159742711im -0.0
5264915624361982 - 0.04979768463372333im -0.0022725328138584328 + 0.0001034
9689471529653im])
```