```
export tdvp!
using ITensors: position!

singlesite!(PH::ProjMPO) = (PH.nsite = 1)
twosite!(PH::ProjMPO) = (PH.nsite = 2)

struct TDVP2 end

"""
    tdvp!(psi,H::MPO,dt,tf; kwargs...)
Evolve the MPS `psi` up to time `tf` using the two-site time-dependent variational
principle as described in Ref. [1].

# Keyword arguments:
All keyword arguments controlling truncation which are accepted by ITensors.replaceBond!,
namely:
- `maxdim::Int`: If specified, keep only `maxdim` largest singular values after applying gate.
- `mindim::Int`: Minimal number of singular values to keep if truncation is performed according to
    value specified by `cutoff`.
- `cutoff::Float`: If specified, keep the minimal number of singular-values such that the discarded weight is
    smaller than `cutoff` (but bond dimension will be kept smaller than `maxdim`).
- `absoluteCutoff::Bool`: If `true` truncate all singular-values whose square is smaller than `cutoff`.

In addition the following keyword arguments are supported:
- `hermitian::Bool` (`true`) : whether the MPO `H` represents an Hermitian operator. This will be passed to the
    Krylov exponentiation routine (`KrylovKit.exponentiate`) which will in turn use a Lancosz algorithm in the
    case of an hermitian operator.
- `exp_tol::Float` (1e-14) : The error tolerance for `KrylovKit.exponentiate`.
    (note that default value was not optimized yet, so you might want to play around with it)
- `progress::Bool` (`true`) : If `true` a progress bar will be displayed

# References:
[1] Haegeman, J., Lubich, C., Oseledets, I., Vandereycken, B., & Verstraete, F. (2016).
Unifying time evolution and optimization with matrix product states.
Physical Review B, 94(16).
https://doi.org/10.1103/PhysRevB.94.165116
"""
function tdvp!(psi,H::MPO,dt,tf; kwargs...)
    nsteps = Int(tf/dt)
    cb = get(kwargs,:callback, NoTEvoCallback())
    hermitian = get(kwargs,:hermitian,true)
```

```
    exp_tol = get(kwargs,:exp_tol, 1e-14)
    krylovdim = get(kwargs,:krylovdim, 30 )
    maxiter = get(kwargs,:maxiter,100)
    normalize = get(kwargs,:normalize,true)

    pbar = get(kwargs,:progress, true) ? Progress(nsteps,
desc="Evolving state... ") : nothing
    τ = 1im*dt
    imag(τ) == 0 && (τ = real(τ))

    N = length(psi)
    orthogonalize!(psi,1)
    PH = ProjMPO(H)
    position!(PH,psi,1)
    for s in 1:nsteps
        stime = @elapsed begin
        for (b,ha) in sweepnext(N)
            #evolve with two-site Hamiltonian
            twosite!(PH)
            ITensors.position!(PH,psi,b)
            wf = psi[b]*psi[b+1]
            wf, info = exponentiate(PH, -τ/2, wf;
ishermitian=hermitian , tol=exp_tol, krylovdim=krylovdim)
            dir = ha==1 ? "left" : "right"
            info.converged==0 && throw("exponentiate did not
converge")
            spec = replacebond!(psi,b,wf;normalize=normalize, ortho
= dir, kwargs... )
            # normalize && ( psi[dir=="left" ? b+1 : b] /=
sqrt(sum(eigs(spec))) )

            apply!(cb,psi; t=s*dt,
                   bond=b,
                   sweepend= ha==2,
                   sweepdir= ha==1 ? "right" : "left",
                   spec=spec,
                   alg=TDVP2())

            # evolve with single-site Hamiltonian backward in time.
            # In the case of imaginary time-evolution this step
            # is not necessary (see Ref. [1])
            i = ha==1 ? b+1 : b
            if 1<i<N && !(dt isa Complex)
                singlesite!(PH)
                ITensors.position!(PH,psi,i)
                psi[i], info = exponentiate(PH,τ/2,psi[i];
ishermitian=hermitian, tol=exp_tol, krylovdim=krylovdim,
                                            maxiter=maxiter)
                info.converged==0 && throw("exponentiate did not
converge")
            elseif i==1 && dt isa Complex
                # TODO not sure if this is necessary anymore
                psi[i] /= sqrt(real(scalar(dag(psi[i])*psi[i])))
            end
```

```
        end
        end
        !isnothing(pbar) && ProgressMeter.next!(pbar,
showvalues=[("t", dt*s),

("dt step time", round(stime,digits=3)),

("Max bond-dim", maxlinkdim(psi))])
        checkdone!(cb) && break
    end
end
```