	Shankar's (lindbladInit)	Mine (lindbladInit_for-DMD-4.4)
Output data files (required by later real-time dynamics)	A single file ldbd.dat	Multi files in the folder ./ldbd_data, each storing one kind of quantity (additionally ImSigma_ePh and eigenvectors for electron-impurity and electron-electron scatterings)
Output electron-phonon quantities	$G^{-} = g^{-}\sqrt{\delta}$ stored in sparse form	E-ph-related matrices for both Lindblad and conventional equation. For Lindblad, $P^{I}_{1234} = \sum_{\pm\lambda} G^{\pm}_{13} G^{\pm,*}_{24} n^{\pm}$ $P^{II}_{1234} = \sum_{\pm\lambda} G^{\pm}_{31} G^{\pm,*}_{42} n^{\mp}$ For conventional, $P^{I}_{1234} = \sum_{\pm\lambda} g^{\pm}_{13} \mathcal{G}^{\pm,*}_{24} n^{\pm}$ $P^{II}_{1234} = \sum_{\pm\lambda} g^{\pm}_{31} \mathcal{G}^{\pm,*}_{42} n^{\mp}$ $\mathcal{G} = g\delta$ of a k pair (ik,jk) with ik <= jk ($P^{I,kk\prime} = P^{II,k'k,*}$ and $P^{II,kk\prime} = P^{I,k'k,*}$) in dense form. Due to the difference of output e-ph quantities, energy conservation for k pair selection is not the same
Information in std::out		Spin-relaxation time calculated using P? matrices and G matrices with different kind of smearings (constant or ImSigma_ePh)
Energy ranges	Two – one for probe ; one for pump and e-ph for both conduction and valance bands	Three – for probe ; for pump ; for e-ph, if ePhOnlyElec (ePhOnlyHole) = 1, only e-ph of conduction (valence) bands will be written down; Parameter nkBT is added to control energy range, default 7, 7 is enough for real-time but may not be for rate formula
Energy conservation range	nEphDelta = 5	nEphDelta is an input parameter. This is important for (i) electron-electron scattering (ii) spin lifetime computations inside the code, since different smearings (e.g., max of ImSigme_ePh can be much larger than ePhDelta) will be used

VBM & CBM determination		In case VBM is not zero (metal, Fermi smearing or finite electric field), parameter band_skipped, starting band index of wannier relative to DFT, can be used to determine VBM and CBM correctly
k-point selection		In 3D cases, there can be too many k points. To speed up eLoop, we need temporally turn off spin and velocity matrices computations. This is achieved by introducing eEneOnly in FeynWann (see if (eEneOnly) return; in setState in FeynWann.cpp)
Parallel for e-ph	K points	K pairs
E-ph matrix elements computation	Use ePhLoop with mask (whether satisfying energy conservation)	Firstly, run eCalc at all selected k points in parallel (and use bcastState_inEphLoop to collect them) and store electronic states in std::vector<FeynWann::StateE> state_elec; Secondly, use ePhCalc for each k pair with stored FeynWann::StateE state_elec[ik] and state_elec[jk]. In some cases, compute1is slow, e.g., GaN, it is very helpful to run eCalc only once and reuse them in e-ph computations. Otherwise, compute1 must be run for each k pair instead of for each k point My code is however not suitable to the cases where most k points are selected (e.g., iron) or there are more than 10,000 k points so that state_elec will take too huge memory