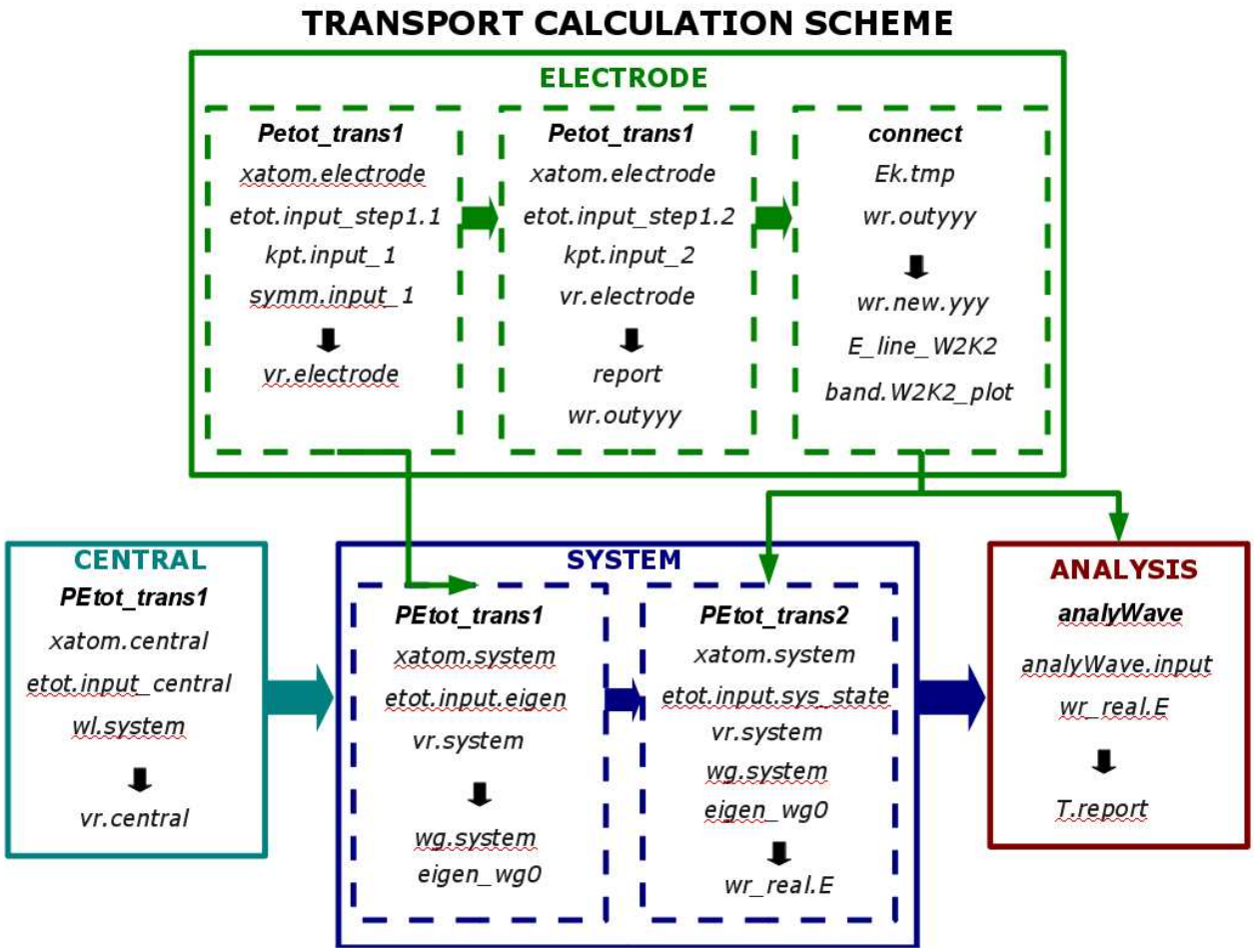
TRANSPORT CALCULATION USING PEtot_TRANS



As shown in the above schematics, there are four major parts and steps to do the transport calculation. The first is to calculate the electrode, then the central region, then use the electrode and the central region to construct the whole system. After the whole system is constructed, the system states (to be stored in wr_real.E) are calculated. Then, in the last step, the system states in wr_real.E are used to construct the scattering states and transmission curves in the analysis. The details of these four steps are described below:

1) ELECTRODE:

Purpose: to calculate selfconsistently the electrode charge density and potential, and the real k-point band structure and band states (to be used later for analysis and system state generation).
Current limitation: Left and right electrodes must be the same.
Test: in the example, go to ELECTRODE subdirectory**.**

1.1) Calculation of the self-consistent electrode potential

Code: PEtot_trans1
Input: kpt.input1.1, symm.input1.1 (generated from kpgen, use as many x,y,z k-points as necessary), etot.input_step1.1(copy to etot.input), xatom.electrode.
Output: vr.electrode (the selfconsistent potential of the electrode).

1.2) Nonselfconsistent calculation of electrode band structure

Code: PEtot_trans1
Input: vr.electrode (from 1.1), kpt.input101 (many, e.g., 100, k-points along x direction), etot.input_step1.2 (copy to etot.input, idens_out=12, wr.out in line 14).
Output: report (eigen energies), wr.outyyy (real space band eigen state, yyy=1,101, kpts).

1.3) Postprocessing the electrode band structure

Purpose: To identify connection bands, rotate the degenerated wave functions.
Code: connect.f (in SRC/UTILITIES/ELECTRODE).
Input: copy report to Ek.tmp (delete the headers), wr.outyyy.
Output: E_line_W2K2 (containing the band information), band.W2K2_plot (for plotting), wr.new.yyy (rotated and aligned real space electrode eigen states. One can remove the original wr.outyyy).

2) CENTRAL:

Purpose: to selfconsistently calculate the central region (the actual object for transport calculation), even for biased systems (non equilibrium cases).
Limitation: We don't do selfconsistent calculation using scattering states. Instead, an approximated way using a mask function is used for biased systems. In the future, a more sophisticated local Fermi potential approximation can be used. We think there is no advantage in using scattering states (there are many approximations anyway). Note, there could be different ways to do this central region, we just provide one practical way to do it here. The central region potential can also be done nonselfconsistently using an artificial potential drop profile (e.g., using add_bias.f from UTILITIES/CENTRAL).
Test: go to CENTRAL under EXAMPLE.
Code: PEtot_trans1, gen_mask.f (SRC/UTILITIES/CENTRAL).
Input: xatom.central, mask.central (from gen_mask.f, mask.input), etot.input (note: the line 25: 1, mask.system, dV_bias).
Output: vr.central.
Note: In xatom.central, in order to do dV_bias, we usually put vacuum at the left and right ends of the system. The left and right electrode atoms must be located at the same relative grid positions as in xatom.electrode (this can be achieved easily because the vacuums are added at the two ends, provided the central region can only be varied one grid point at a time). In etot.input, enough mx (usually much larger

than totNel/2) must be used in order for the eigen energies to reach the level of left and right Fermi energies.

3) SYSTEM

Purpose: set up the whole system from central potential and electrode potential, then calculate the system states to be used to construct the scattering states later.

3.1) Setup the full system atomic configuration and potential.

Purpose: take the vr.central, vr.electrode to generate vr.system.
Codes: construct_system.f (in SRC/UTILITIES/SYSTEM).
Input: construct.input (make sure, dV is the same as in etot.input for central).
Output: vr.system, xatom.system.

3.2) Calculation of the full system eigen states and eigen energies

Purpose: to calculate (nonselfconsistently) the full system eigenstates and eigen energies for later 3.3 system state calculations.
Code: PEtot_trans1
Input: etot.input.eigen (copy to etot.input), kpt.file (use 0.25, 0, 0), vr.system, xatom.system.
Ouput: wg.system, eigen_all.store.

3.3) Calculation of the system states

Purpose: calculate the system states (stored in wr_real.E), to be used later to generate the scattering states. Scanning through different energy points.
Code: PEtot_trans2
Input: etot.input.sys_state (copy to etot.input, style different from PEtot_trans1. etot.input, note line 23, dV must be the same as before); vr.system, xatom.system, wg.system, eigen_wg0 (copy report from 3.2 to eigen_wg0, remove the header, keep only the eigen energy part), E_line_W2K2, wr.new.yyy (link from the ELECTRODE directory using "ln –s ../ElECTRODE/wr.new.* ."), kpt.file (same as in 3.2).
Output: wr_real.E

4) ANALYSIS

Purpose: use the wr_real.E system states to generate the scattering states, and to calculate the transmission coefficients.
Code: analyWave.f (from SRC/ANALYSIS).
Input: analyWave.input (need specify the fitting position, dV must be the same as before), wr_real.E, wr.new.yyy, E_line_W2K2 (do it in the same dir as in 3).
Output: T.report (transmission coefficients, T-R should be close to 1), scattering states (scatt_st_3D.out).

Others: - curvefit.f (for interpolation of transmission curves and calculation of conductance for the corresponding bias voltage).

Other comments:

(1) Currently, one major limitation is that there is no k-point in the y and z directions. That can be added (the major change will be to the analyWave.f code when system_st, and system_st^* are both used).

(2) Both PEtot_trans1, and PEtot_trans2 in this project use cylindrical energy cut off along the n1 direction. Thus, it is not recommended to use PEtot_trans1 to relax the atom in the central region. To relax the atom, please use the conventially PEtot.

(3) The can be some difficulty to find the proper n1 (for the total system) in order to do FFT for a given system. This is solved by adding some extra points at the left side end (x=0). But one also needs to make sure there will be conduction at the boundary. Otherwise the whole thing will not work.