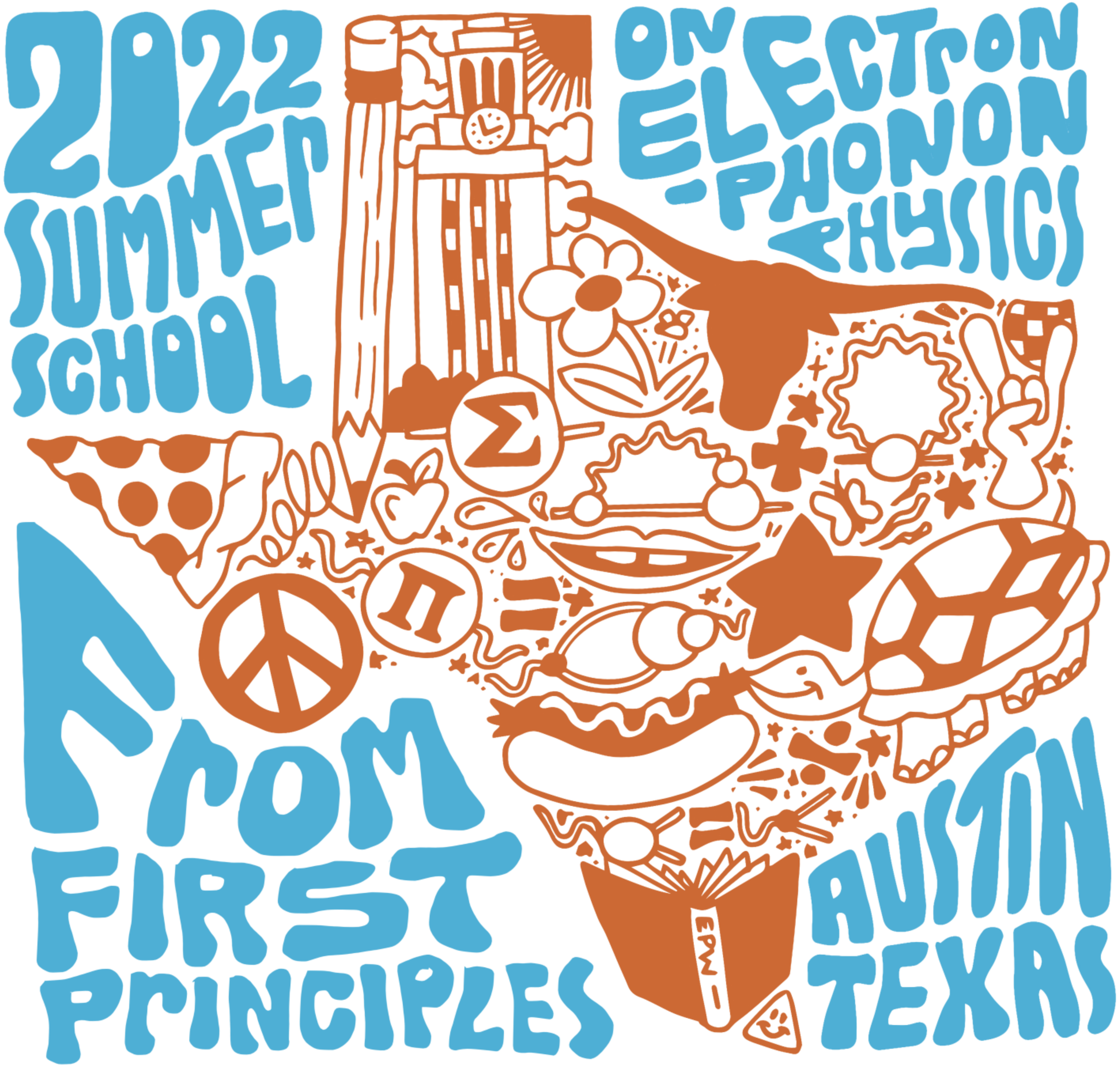
2022 Summer School on Electron-Phonon Physics from First Principles

Austin Texas



U.S. DEPARTMENT OF ENERGY

TACC
TEXAS ADVANCED COMPUTING CENTER

ODEN

Hands-on Tue.4

Hands-On Intro: Running EPW

Hyungjun Lee

Oden Institute for Computational Engineering and Sciences
The University of Texas at Austin

Contents

Computational flow of EPW

Computational flow

$$\boxed{g(\mathbf{k}_{\mathrm{f}}, \mathbf{q}_{\mathrm{f}})}$$

The most important
quantity
to study *e*-ph physics
(Mon. 1, Giustino)

Computational flow

$$\boxed{g(\mathbf{k}_\mathrm{f}, \mathbf{q}_\mathrm{f})}$$

The most important
quantity
to study *e*-ph physics
(Mon. 1, Giustino)

$$\textstyle\sum''_{n\mathbf{k}_\mathrm{f}}[g(\mathbf{k}_\mathrm{f}, \mathbf{q}_\mathrm{f})], \Pi''_{\nu\mathbf{q}_\mathrm{f}}[g(\mathbf{k}_\mathrm{f}, \mathbf{q}_\mathrm{f})], \ldots$$

Computational flow

$$\boxed{g(\mathbf{k}_{\mathrm{f}}, \mathbf{q}_{\mathrm{f}})}$$

The most important quantity
to study *e*-ph physics
(Mon. 1, Giustino)

$$\Sigma''_{n\mathbf{k}_{\mathrm{f}}}(\omega, T) = \pi \sum_{m\nu} \int_{\mathrm{BZ}} \frac{d\mathbf{q}_{\mathrm{f}}}{\Omega_{\mathrm{BZ}}} |g_{mn\nu}(\mathbf{k}_{\mathrm{f}}, \mathbf{q}_{\mathrm{f}})|^2$$
$$\{[n_{\mathbf{q}_{\mathrm{f}}\nu}(T) + f_{m\mathbf{k}_{\mathrm{f}}+\mathbf{q}_{\mathrm{f}}}(T)]\delta(\omega - (\varepsilon_{m\mathbf{k}_{\mathrm{f}}+\mathbf{q}_{\mathrm{f}}} - \varepsilon_{\mathrm{F}}) + \omega_{\mathbf{q}_{\mathrm{f}}\nu})$$
$$+ [n_{\mathbf{q}_{\mathrm{f}}\nu}(T) + 1 - f_{m\mathbf{k}_{\mathrm{f}}+\mathbf{q}_{\mathrm{f}}}(T)]\delta(\omega - (\varepsilon_{m\mathbf{k}_{\mathrm{f}}+\mathbf{q}_{\mathrm{f}}} - \varepsilon_{\mathrm{F}}) - \omega_{\mathbf{q}_{\mathrm{f}}\nu})\}$$

[Comput. Phys. Commun. **209**, 116 (2016), Rev. Mod. Phys. **89**, 1 (2017)]

Computational flow

$$\boxed{g(\mathbf{k}_{\mathrm{f}}, \mathbf{q}_{\mathrm{f}})}$$ ➤

The most important quantity
to study *e*-ph physics
(Mon. 1, Giustino)

Electron, phonon self-energy

(Tue. 4&5, Lee)

Transport

(Wed. 1,4&5, Poncé)

Superconductivity

(Wed. 2,6&7, Margine)

Indirect absorption

(Wed. 3, Thu. 4, Kioupakis, Zhang)

Polaron

(Thu. 1,4&5, Giustino,
Lian, and LaFuente-Bartolomé)

Computational flow

$$\boxed{g(\mathbf{k}_\mathrm{f}, \mathbf{q}_\mathrm{f})}$$ ➤

Fine sampling of
reciprocal points needed
(Mon. 1, Giustino)

Electron, phonon self-energy

(Tue. 4&5, Lee)

Transport

(Wed. 1,4&5, Poncé)

Superconductivity

(Wed. 2,6&7, Margine)

Indirect absorption

(Wed. 3, Thu. 4, Kioupakis, Zhang)

Polaron

(Thu. 1,4&5, Giustino,
Lian, and LaFuente-Bartolomé)

Computational flow

$$\boxed{g(\mathbf{k}_\mathrm{c}, \mathbf{q}_\mathrm{c})} \blacktriangleright \boxed{g(\mathbf{R}_e, \mathbf{R}_p)} \blacktriangleright \boxed{g(\mathbf{k}_\mathrm{f}, \mathbf{q}_\mathrm{f})} \blacktriangleright$$

Fine sampling of reciprocal points needed (Mon. 1, Giustino)
➤ Wannier interpolation

$$g_{mn\nu}(\mathbf{k}_\mathrm{f}, \mathbf{q}_\mathrm{f}) = \sqrt{\frac{\hbar}{2M_\kappa \omega_{\mathbf{q}_\mathrm{f}\nu}}} \sum_{pp'} e^{i(\mathbf{k}_\mathrm{f}\cdot\mathbf{R}_p + \mathbf{q}_\mathrm{f}\cdot\mathbf{R}_{p'})}$$
$$\times \left[U_{\mathbf{k}_\mathrm{f}+\mathbf{q}_\mathrm{f}}\, g(\mathbf{R}_p, \mathbf{R}_{p'}) \cdot e_{\kappa\alpha,\nu}(\mathbf{q}_\mathrm{f})\, U^\dagger_{\mathbf{k}_\mathrm{f}}\right]_{mn}$$

[Phys. Rev. B **76**, 165108 (2007), Rev. Mod. Phys. **89**, 1 (2017)]

Electron, phonon self-energy
(Tue. 4&5, Lee)

Transport
(Wed. 1,4&5, Poncé)

Superconductivity
(Wed. 2,6&7, Margine)

Indirect absorption
(Wed. 3, Thu. 4, Kioupakis, Zhang)

Polaron
(Thu. 1,4&5, Giustino, Lian, and LaFuente-Bartolomé)

Computational flow

Starting point
for EPW calculations

$$\boxed{g(\mathbf{k}_\mathrm{c}, \mathbf{q}_\mathrm{c})} \rightarrow \boxed{g(\mathbf{R}_e, \mathbf{R}_p)} \rightarrow \boxed{g(\mathbf{k}_\mathrm{f}, \mathbf{q}_\mathrm{f})}$$

$$g_{mn\nu}(\mathbf{k}_\mathrm{c}, \mathbf{q}_\mathrm{c}) = \langle u_{m\mathbf{k}_\mathrm{c}+\mathbf{q}_\mathrm{c}} | \Delta_{\mathbf{q}_\mathrm{c}\nu} v^{\mathrm{KS}} | u_{n\mathbf{k}_\mathrm{c}} \rangle_{\mathrm{uc}}$$

$$\Delta_{\mathbf{q}_\mathrm{c}\nu} v^{\mathrm{KS}} = \sum_{\kappa\alpha} \sqrt{\frac{\hbar}{2M_\kappa \omega_{\mathbf{q}_\mathrm{c}\nu}}} e_{\kappa\alpha,\nu}(\mathbf{q}_\mathrm{c}) \partial_{\kappa\alpha,\mathbf{q}_\mathrm{c}} v^{\mathrm{KS}}$$

, where $\partial_{\kappa\alpha,\mathbf{q}_\mathrm{c}} v^{\mathrm{KS}} = e^{-i\mathbf{q}_\mathrm{c}\cdot\mathbf{r}} \sum_p e^{i\mathbf{q}_\mathrm{c}\cdot\mathbf{R}_p} \frac{\partial V^{\mathbf{KS}}(\mathbf{r})}{\partial \tau_{\kappa\alpha p}}$

[Phys. Rev. B **76**, 165108 (2007), Rev. Mod. Phys. **89**, 1 (2017)]

Computational flow

$$\boxed{g(\mathbf{k}_\mathrm{c}, \mathbf{q}_\mathrm{c})} \rightarrow \boxed{g(\mathbf{R}_e, \mathbf{R}_p)} \rightarrow \boxed{g(\mathbf{k}_\mathrm{f}, \mathbf{q}_\mathrm{f})}$$

Wave functions from pw.x (NSCF)

$$g_{mn\nu}(\mathbf{k}_\mathrm{c}, \mathbf{q}_\mathrm{c}) = \langle u_{m\mathbf{k}_\mathrm{c}+\mathbf{q}_\mathrm{c}} | \Delta_{\mathbf{q}_\mathrm{c}\nu} v^{\mathrm{KS}} | u_{n\mathbf{k}_\mathrm{c}} \rangle_{\mathrm{uc}}$$

$$\Delta_{\mathbf{q}_\mathrm{c}\nu} v^{\mathrm{KS}} = \sum_{\kappa\alpha} \sqrt{\frac{\hbar}{2M_\kappa \omega_{\mathbf{q}_\mathrm{c}\nu}}} e_{\kappa\alpha,\nu}(\mathbf{q}_\mathrm{c}) \partial_{\kappa\alpha,\mathbf{q}_\mathrm{c}} v^{\mathrm{KS}}$$

, where $\partial_{\kappa\alpha,\mathbf{q}_\mathrm{c}} v^{\mathrm{KS}} = e^{-i\mathbf{q}_\mathrm{c}\cdot\mathbf{r}} \sum_p e^{i\mathbf{q}_\mathrm{c}\cdot\mathbf{R}_p} \frac{\partial V^{\mathbf{KS}}(\mathbf{r})}{\partial \tau_{\kappa\alpha p}}$

[Phys. Rev. B **76**, 165108 (2007), Rev. Mod. Phys. **89**, 1 (2017)]

Computational flow

$$\boxed{g(\mathbf{k}_\mathrm{c}, \mathbf{q}_\mathrm{c})} \rightarrow \boxed{g(\mathbf{R}_e, \mathbf{R}_p)} \rightarrow \boxed{g(\mathbf{k}_\mathrm{f}, \mathbf{q}_\mathrm{f})}$$

Wave functions from pw.x (NSCF)

Perturbing potentials
(In patterns basis)
from ph.x

$$g_{mn\nu}(\mathbf{k}_\mathrm{c}, \mathbf{q}_\mathrm{c}) = \langle u_{m\mathbf{k}_\mathrm{c}+\mathbf{q}_\mathrm{c}} | \Delta_{\mathbf{q}_\mathrm{c}\nu} v^{\mathrm{KS}} | u_{n\mathbf{k}_\mathrm{c}} \rangle_{\mathrm{uc}}$$

$$\Delta_{\mathbf{q}_\mathrm{c}\nu} v^{\mathrm{KS}} = \sum_{\kappa\alpha} \sqrt{\frac{\hbar}{2M_\kappa \omega_{\mathbf{q}_\mathrm{c}\nu}}} e_{\kappa\alpha,\nu}(\mathbf{q}_\mathrm{c}) \partial_{\kappa\alpha,\mathbf{q}_\mathrm{c}} v^{\mathrm{KS}}$$

, where $\partial_{\kappa\alpha,\mathbf{q}_\mathrm{c}} v^{\mathrm{KS}} = e^{-i\mathbf{q}_\mathrm{c}\cdot\mathbf{r}} \sum_p e^{i\mathbf{q}_\mathrm{c}\cdot\mathbf{R}_p} \frac{\partial V^{\mathbf{KS}}(\mathbf{r})}{\partial \tau_{\kappa\alpha p}}$

[Phys. Rev. B **76**, 165108 (2007), Rev. Mod. Phys. **89**, 1 (2017)]

Computational flow

$$\boxed{g(\mathbf{k}_\mathrm{c}, \mathbf{q}_\mathrm{c})} \rightarrow \boxed{g(\mathbf{R}_e, \mathbf{R}_p)} \rightarrow \boxed{g(\mathbf{k}_\mathrm{f}, \mathbf{q}_\mathrm{f})}$$

Wave functions from pw.x (NSCF)

$$g_{mn\nu}(\mathbf{k}_\mathrm{c}, \mathbf{q}_\mathrm{c}) = \langle u_{m\mathbf{k}_\mathrm{c}+\mathbf{q}_\mathrm{c}} | \Delta_{\mathbf{q}_\mathrm{c}\nu} v^{\mathrm{KS}} | u_{n\mathbf{k}_\mathrm{c}} \rangle_{\mathrm{uc}}$$

Perturbing potentials
(In patterns basis)
from ph.x

$$\Delta_{\mathbf{q}_\mathrm{c}\nu} v^{\mathrm{KS}} = \sum_{\kappa\alpha} \sqrt{\frac{\hbar}{2M_\kappa \omega_{\mathbf{q}_\mathrm{c}\nu}}} e_{\kappa\alpha,\nu}(\mathbf{q}_\mathrm{c}) \partial_{\kappa\alpha,\mathbf{q}_\mathrm{c}} v^{\mathrm{KS}}$$

Caculated from
dynamical matrices from ph.x

, where $\partial_{\kappa\alpha,\mathbf{q}_\mathrm{c}} v^{\mathrm{KS}} = e^{-i\mathbf{q}_\mathrm{c}\cdot\mathbf{r}} \sum_p e^{i\mathbf{q}_\mathrm{c}\cdot\mathbf{R}_p} \frac{\partial V^{\mathbf{KS}}(\mathbf{r})}{\partial \tau_{\kappa\alpha p}}$

[Phys. Rev. B **76**, 165108 (2007), Rev. Mod. Phys. **89**, 1 (2017)]

Computational flow

$\boxed{g(\mathbf{k}_\mathrm{c}, \mathbf{q}_\mathrm{c})}$ ➤ $\boxed{g(\mathbf{R}_e, \mathbf{R}_p)}$ ➤ $\boxed{g(\mathbf{k}_\mathrm{f}, \mathbf{q}_\mathrm{f})}$

Inputs to EPW

Wave functions from pw.x (NSCF)

Perturbing potentials
(In patterns basis)
from ph.x

Caculated from
dynamical matrices from ph.x

$$g_{mn\nu}(\mathbf{k}_\mathrm{c}, \mathbf{q}_\mathrm{c}) = \langle u_{m\mathbf{k}_\mathrm{c}+\mathbf{q}_\mathrm{c}} | \Delta_{\mathbf{q}_\mathrm{c}\nu} v^{\mathrm{KS}} | u_{n\mathbf{k}_\mathrm{c}} \rangle_{\mathrm{uc}}$$

$$\Delta_{\mathbf{q}_\mathrm{c}\nu} v^{\mathrm{KS}} = \sum_{\kappa\alpha} \sqrt{\frac{\hbar}{2M_\kappa \omega_{\mathbf{q}_\mathrm{c}\nu}}} e_{\kappa\alpha,\nu}(\mathbf{q}_\mathrm{c}) \partial_{\kappa\alpha,\mathbf{q}_\mathrm{c}} v^{\mathrm{KS}}$$

, where $\partial_{\kappa\alpha,\mathbf{q}_\mathrm{c}} v^{\mathrm{KS}} = e^{-i\mathbf{q}_\mathrm{c}\cdot\mathbf{r}} \sum_p e^{i\mathbf{q}_\mathrm{c}\cdot\mathbf{R}_p} \frac{\partial V^{\mathbf{KS}}(\mathbf{r})}{\partial \tau_{\kappa\alpha p}}$

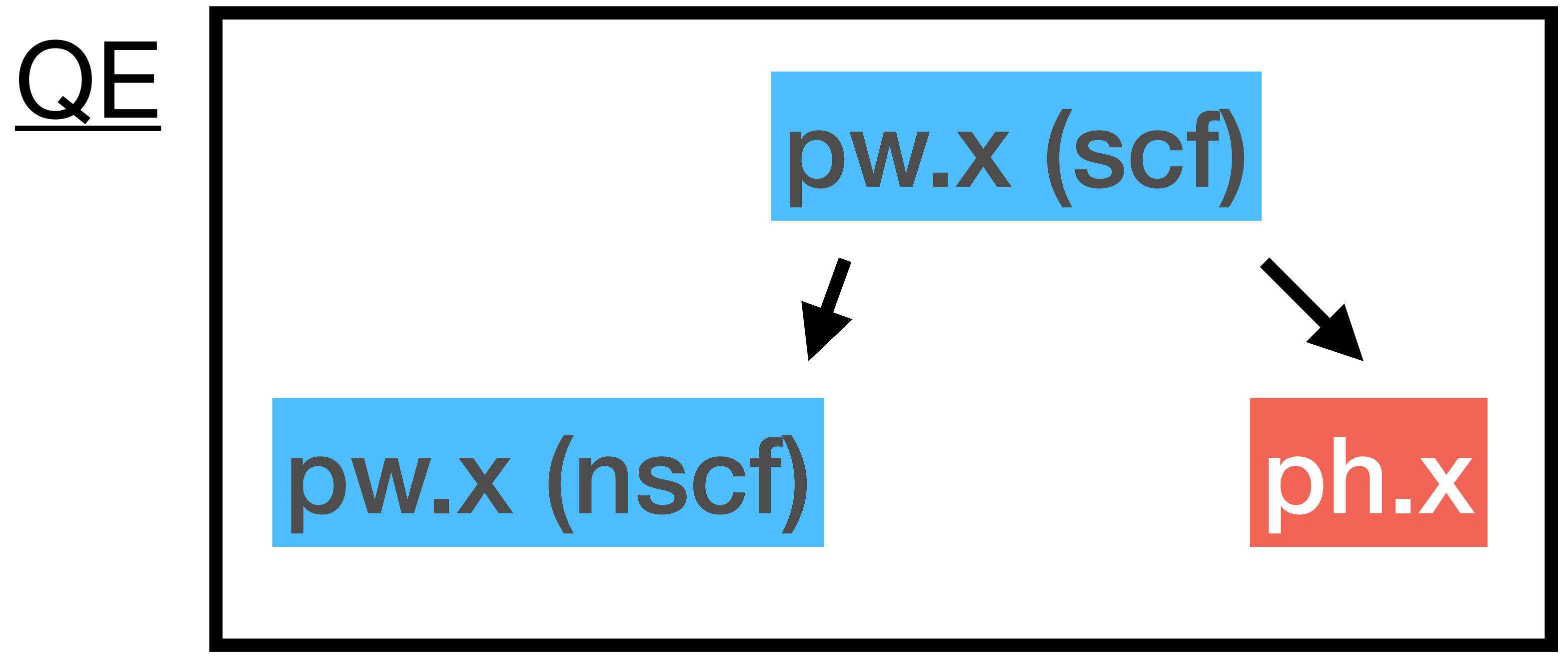
[Phys. Rev. B **76**, 165108 (2007), Rev. Mod. Phys. **89**, 1 (2017)]

Summary: Computational flow

QE

pw.x (scf)

Summary: Computational flow

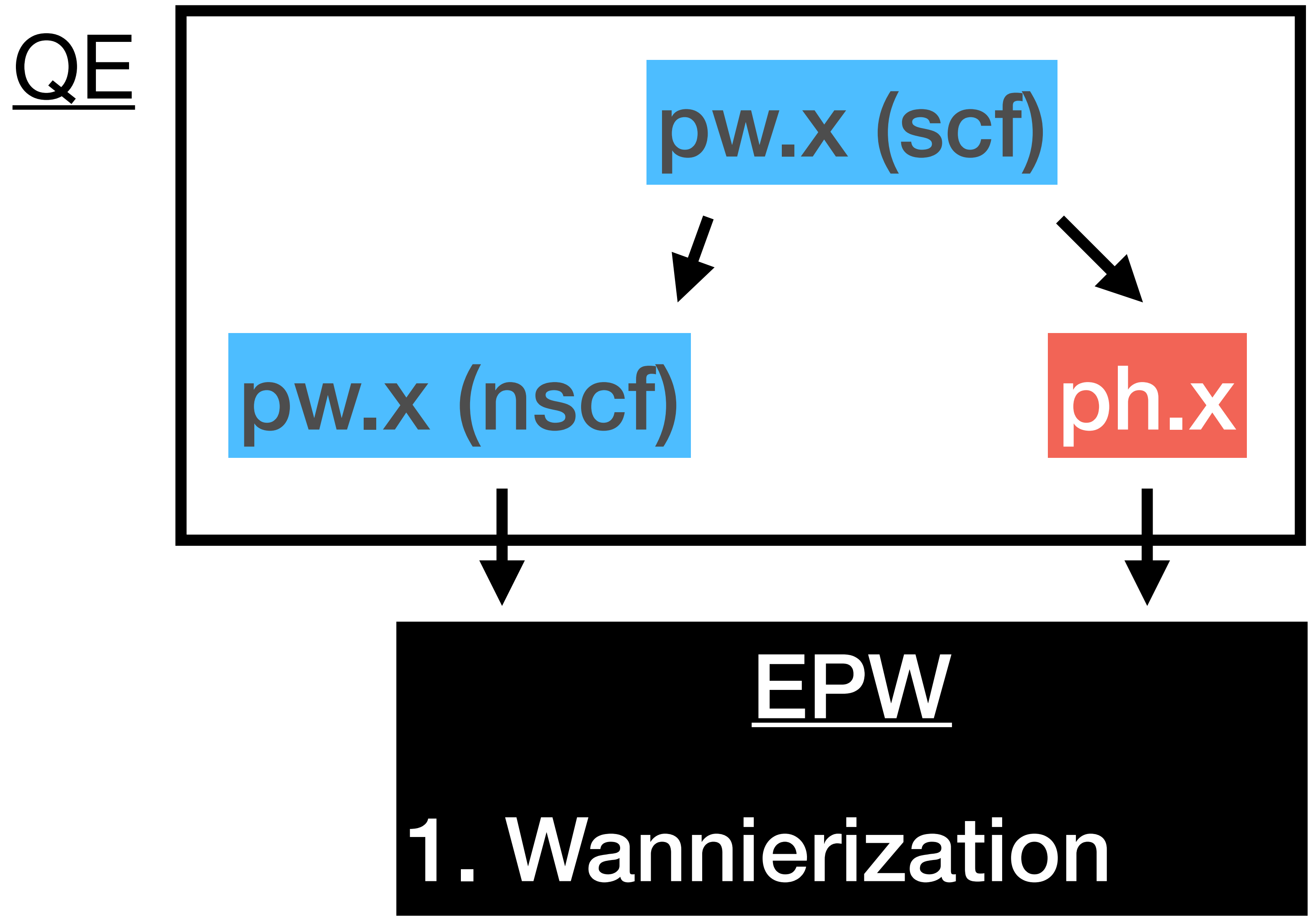


$$g_{mn\nu}(\mathbf{k}_{\rm c},\mathbf{q}_{\rm c}) = \langle u_{m\mathbf{k}_{\rm c}+\mathbf{q}_{\rm c}} | \Delta_{\mathbf{q}_{\rm c}\nu} v^{\rm KS} | u_{n\mathbf{k}_{\rm c}} \rangle_{\rm uc}$$

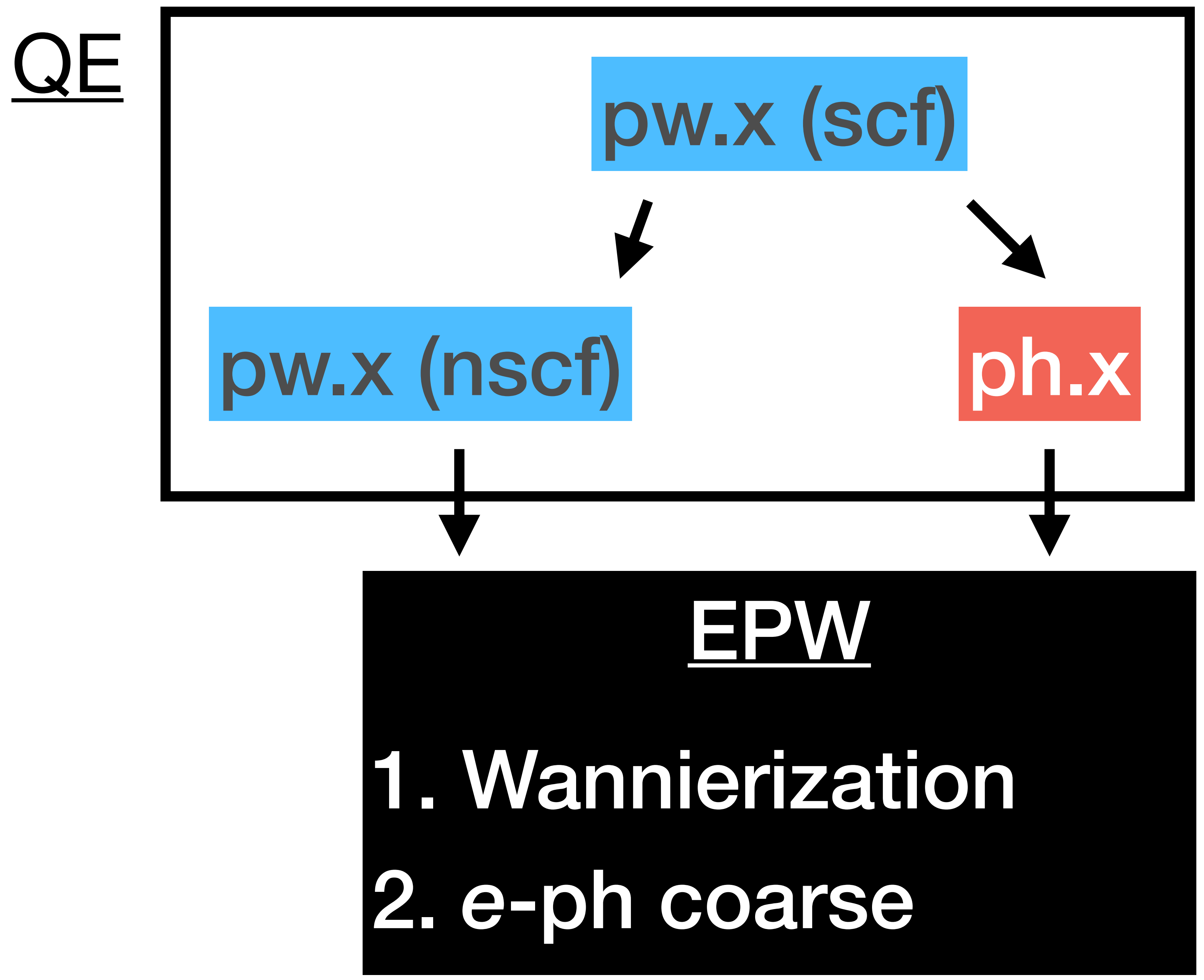
$$\Delta_{\mathbf{q}_{\rm c}\nu} v^{\rm KS} = \sum_{\kappa\alpha} \sqrt{\frac{\hbar}{2M_\kappa \omega_{\mathbf{q}_{\rm c}\nu}}}\, e_{\kappa\alpha,\nu}(\mathbf{q}_{\rm c})\, \partial_{\kappa\alpha,\mathbf{q}_{\rm c}} v^{\rm KS}$$

$$\partial_{\kappa\alpha,\mathbf{q}_{\rm c}} v^{\rm KS} = e^{-i\mathbf{q}_{\rm c}\cdot\mathbf{r}} \sum_p e^{i\mathbf{q}_{\rm c}\cdot\mathbf{R}_p} \frac{\partial V^{\rm KS}(\mathbf{r})}{\partial \tau_{\kappa\alpha p}}$$

Summary: Computational flow



Summary: Computational flow

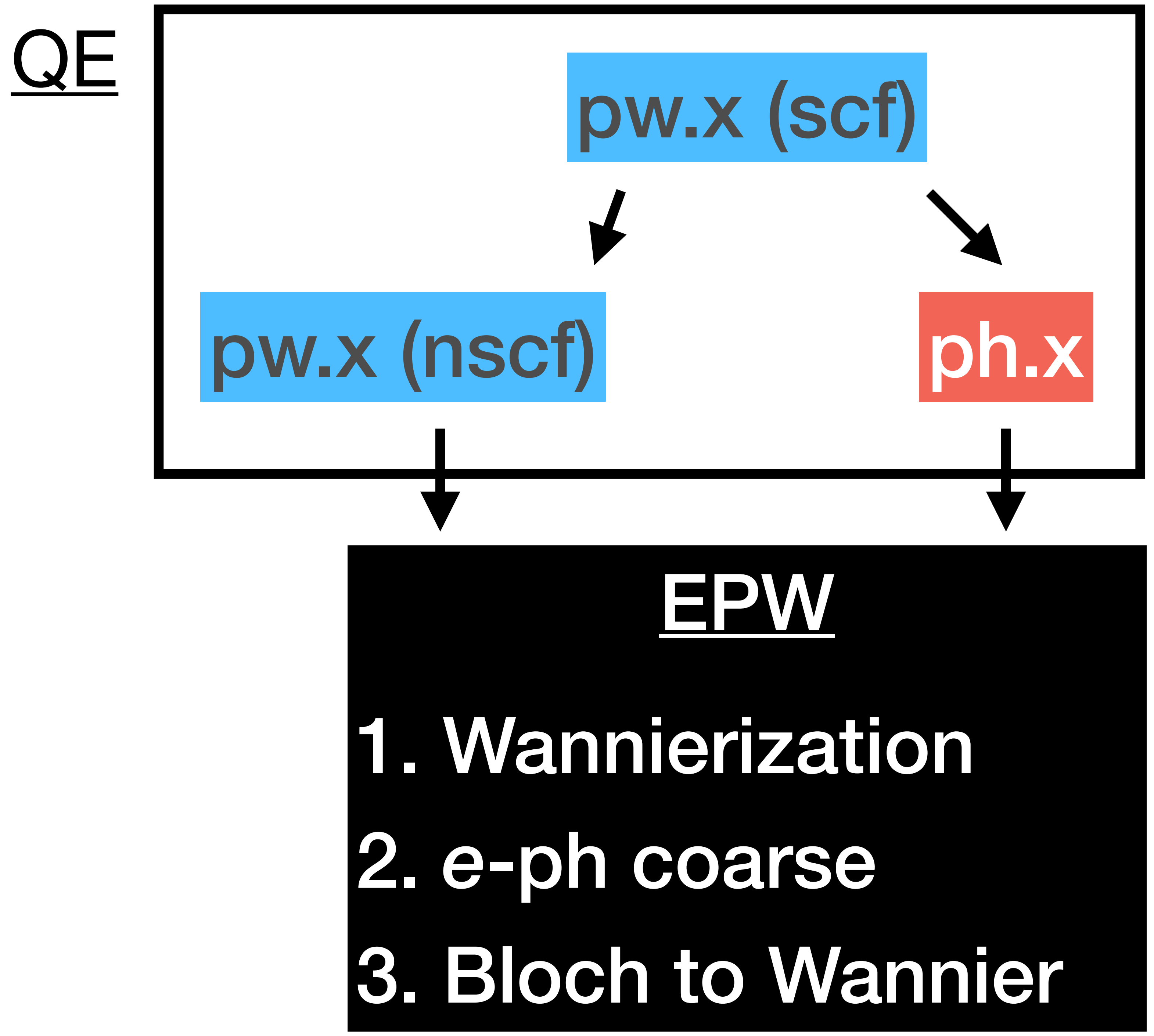


$$g_{mn\nu}(\mathbf{k}_\mathrm{c}, \mathbf{q}_\mathrm{c}) = \langle u_{m\mathbf{k}_\mathrm{c}+\mathbf{q}_\mathrm{c}} | \Delta_{\mathbf{q}_\mathrm{c}\nu} v^{\mathrm{KS}} | u_{n\mathbf{k}_\mathrm{c}} \rangle_{\mathrm{uc}}$$

$$\Delta_{\mathbf{q}_\mathrm{c}\nu} v^{\mathrm{KS}} = \sum_{\kappa\alpha} \sqrt{\frac{\hbar}{2M_\kappa \omega_{\mathbf{q}_\mathrm{c}\nu}}} e_{\kappa\alpha,\nu}(\mathbf{q}_\mathrm{c}) \partial_{\kappa\alpha,\mathbf{q}_\mathrm{c}} v^{\mathrm{KS}}$$

$$\partial_{\kappa\alpha,\mathbf{q}_\mathrm{c}} v^{\mathrm{KS}} = e^{-i\mathbf{q}_\mathrm{c}\cdot\mathbf{r}} \sum_p e^{i\mathbf{q}_\mathrm{c}\cdot\mathbf{R}_p} \frac{\partial V^{\mathrm{KS}}(\mathbf{r})}{\partial \tau_{\kappa\alpha p}}$$

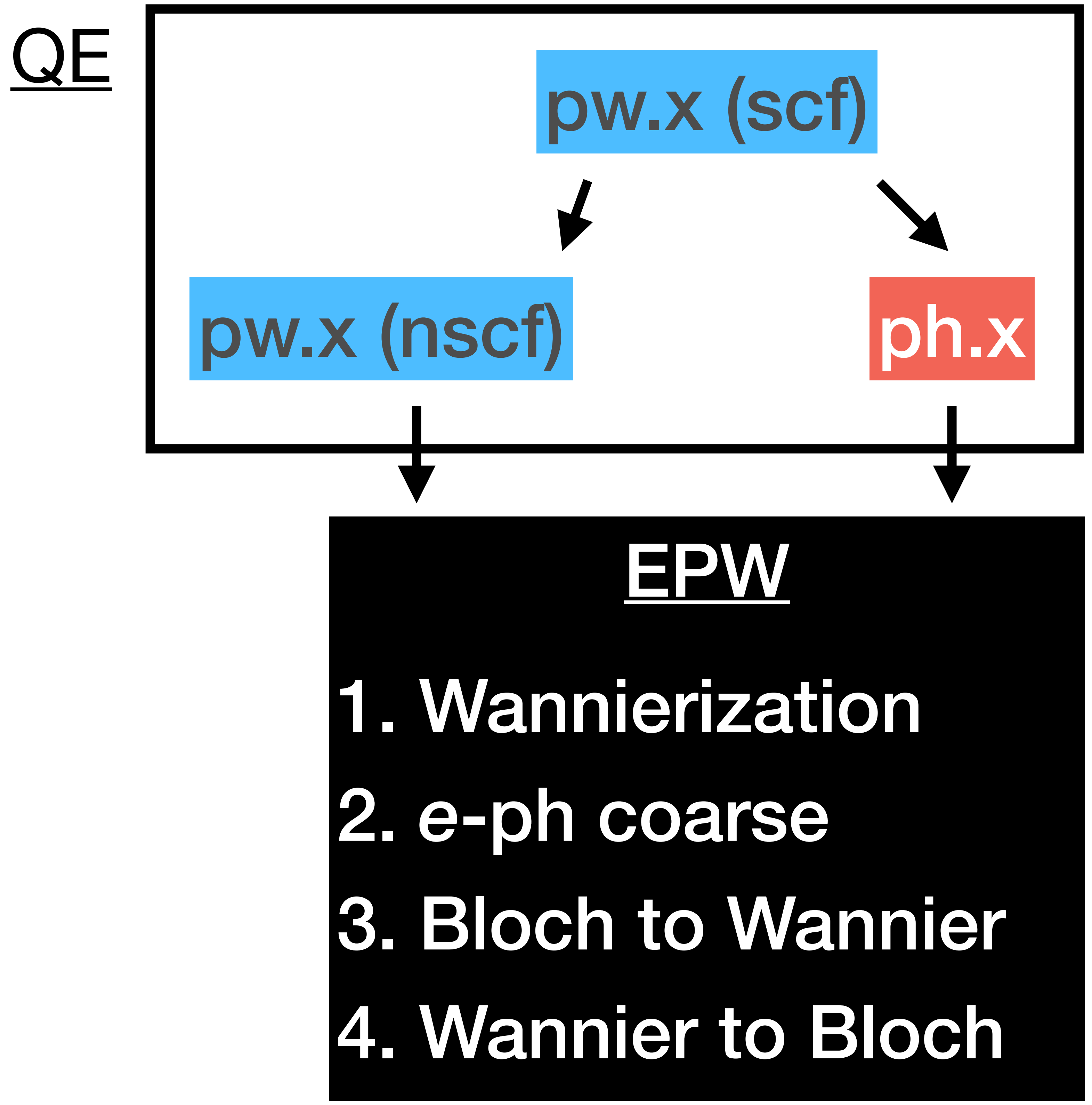
Summary: Computational flow



$$g_{mn\kappa\alpha}(\mathbf{R}_p, \mathbf{R}_{p'}) = \langle \mathrm{w}_{m0}(\mathbf{R}) | \frac{\partial V^{\mathrm{KS}}}{\partial \tau_{\kappa\alpha}}(\mathbf{r} - \mathbf{R}_{p'}) | \mathrm{w}_{m0}(\mathbf{R} - \mathbf{R}_p) \rangle_{\mathrm{sc}}$$

[Phys. Rev. B **76**, 165108 (2007), Rev. Mod. Phys. **89**, 1 (2017)]

Summary: Computational flow



$$
g_{mn\nu}(\mathbf{k}_{\mathrm{f}},\mathbf{q}_{\mathrm{f}}) = \sqrt{\frac{\hbar}{2M_{\kappa}\omega_{\mathbf{q}_{\mathrm{f}}\nu}}} \sum_{pp'} e^{i(\mathbf{k}_{\mathrm{f}}\cdot\mathbf{R}_p+\mathbf{q}_{\mathrm{f}}\cdot\mathbf{R}_{p'})}
\times [U_{\mathbf{k}_{\mathrm{f}}+\mathbf{q}_{\mathrm{f}}}\, g(\mathbf{R}_p,\mathbf{R}_{p'})\cdot e_{\kappa\alpha,\nu}(\mathbf{q}_{\mathrm{f}})\, U^{\dagger}_{\mathbf{k}_{\mathrm{f}}}]_{mn}
$$

[Phys. Rev. B **76**, 165108 (2007), Rev. Mod. Phys. **89**, 1 (2017)]

Summary: Computational flow

QE

pw.x (scf)

pw.x (nscf)

ph.x

EPW

1. Wannierization
2. e-ph coarse
3. Bloch to Wannier
4. Wannier to Bloch
5. Physical properties

Electron, phonon self-energy

(Tue. 4&5, Lee)

Transport

(Wed. 1,4&5, Poncé)

Superconductivity

(Wed. 2,6&7, Margine)

Indirect absorption

(Wed. 3, Thu. 4, Kioupakis, Zhang)

Polaron

(Thu. 1,4&5, Giustino,
Lian, and LaFuente-Bartolomé)

Basic inputs of EPW

EPW

Search docs

PROJECT
- Project News
- About
- Releases
- Documentation
  - Inputs
  - Theory
  - Tutorial
  - School2018
- Acknowledgement
- Papers using EPW
- Contact

SOURCE CODE
- Download and Install
- Ford automatic doc
- Code coverage
- Accuracy
- GitLab
- Test Farm
- Developers
- Benchmarks

DEVELOPERS
- Developers Meetings

Structure of the input data

title_line

&inputepw

...

/

nqs {cartesian}

xq(1) xq(2) xq(3) wq

Note: the k/q-points of the fine grids have to be provided in crystal coordinate only.

&inputepw

A a2f, amass, asr_typ, assume_metal

B band_plot, bands_skipped, bnd_cum, broyden_beta, broyden_ndim

C carrier, conv_thr_iaxis, conv_thr_racon, conv_thr_raxis, cumulant

D degaussq, degaussw, delta_approx, delta_qsmear, delta_smear, dvscf_dir

E efermi_read, eig_read, elecselfen, eliashberg, elph, ep_coupling, epbwrite, epbread, epexst, ephwrite, eps_acustic, epsiHEG, epwread, epwwrite, etf_mem

F fermi_diff, fermi_energy, fila2f, fildvscf, filkf, filqf, filukk, filukq, fsthick

G gap_edge

I imag_read, int_mob, iterative_bte, iverbosity

K kerread, kerwrite, kmaps

L lacon, laniso, lifc, limag, lindabs, liso, longrange, lpade, lphase, lpolar, lreal, lscreen, lunif

M max_memlt, meff, mob_maxiter, mp_mesh_k, mp_mesh_q, muc

N nbndsub, ncarrier, nc, nel, nest_fn, ngaussw, nk1, nk2, nk3, nkf1, nkf2, nqf3, nq1, nq2, nq3, nqf1, nqf2, nqf3, nqsmear, nqstep, n_r, nsiter, nsmear, nstemp, nswi, nswc, nswfc, nw, nw_specfun

O omegamax, omegamin, omegastep

P phonselfen, plselfen, prefix, prtgkk, pwc

R rand_nq, rand_nk, rand_q, rand_k, restart, restart_step

S scr_typ, scattering, scattering_serta, scissor, smear_rpa, specfun_el, specfun_ph, specfun_pl, system_2d, shortrange

T temps

V vme

W wannierize, wepexst, wmax, wmax_specfun, wmin, wmin_specfun, wscut, wsfc

/

nqs, xq(1) xq(2) xq(3)

https://docs.epw-code.org/doc/Inputs.html

Basic input templates

```
--
&inputepw
  prefix      = 'pb',
  outdir      = './'
  dvscf_dir   = '../phonon/save'

  elph        = .true.
  epwwrite    = .true.
  epwread     = .false.

  wannierize  = .true.
  nbndsub     =  4
  bands_skipped = 'exclude_bands = 1-5'
  num_iter    = 300
  dis_win_max = 21
  dis_froz_max= 13.5
  proj(1)     = 'Pb:sp3'
  wannier_plot= .true.

  nk1         = 6
  nk2         = 6
  nk3         = 6
  nq1         = 3
  nq2         = 3
  nq3         = 3
 /
```

Basic input templates

```
--
&inputepw
  prefix      = 'pb',
  outdir      = './'
  dvscf_dir   = '../phonon/save'

  elph        = .true.
  epwwrite    = .true.
  epwread     = .false.

  wannierize  = .true.
  nbndsub     =  4
  bands_skipped = 'exclude_bands = 1-5'
  num_iter    = 300
  dis_win_max = 21
  dis_froz_max= 13.5
  proj(1)     = 'Pb:sp3'
  wannier_plot= .true.

  nk1         = 6
  nk2         = 6
  nk3         = 6
  nq1         = 3
  nq2         = 3
  nq3         = 3
 /
```

Input&Output

Control

Wannierization

k, ***q*** coarse grids

Basic input templates

```
--
&inputepw
  prefix       = 'pb',
  outdir       = './'
  dvscf_dir    = '../phonon/save'

  elph         = .true.
  epwwrite     = .true.
  epwread      = .false.

  wannierize  = .true.
  nbndsub      =  4
  bands_skipped = 'exclude_bands = 1-5'
  num_iter     = 300
  dis_win_max = 21
  dis_froz_max= 13.5
  proj(1)      = 'Pb:sp3'
  wannier_plot= .true.

  nk1          = 6
  nk2          = 6
  nk3          = 6
  nq1          = 3
  nq2          = 3
  nq3          = 3
 /
```

1. outdir/prefix.save dir. points to the directory where nscf results are stored.
2. prefix is a prefix of EPW outputs.
3. Most (not all) EPW outputs are stored in outdir.

Basic input templates

```
--
&inputepw
  prefix      = 'pb',
  outdir      = './'
  dvscf_dir   = '../phonon/save'

  elph        = .true.
  epwwrite    = .true.
  epwread     = .false.

  wannierize  = .true.
  nbndsub     =  4
  bands_skipped = 'exclude_bands = 1-5'
  num_iter    = 300
  dis_win_max = 21
  dis_froz_max= 13.5
  proj(1)     = 'Pb:sp3'
  wannier_plot= .true.

  nk1         = 6
  nk2         = 6
  nk3         = 6
  nq1         = 3
  nq2         = 3
  nq3         = 3
 /
```

1. dvscf_dir points to the directory where outputs from ph.x are stored; they can be simply collected by the script pp.py.

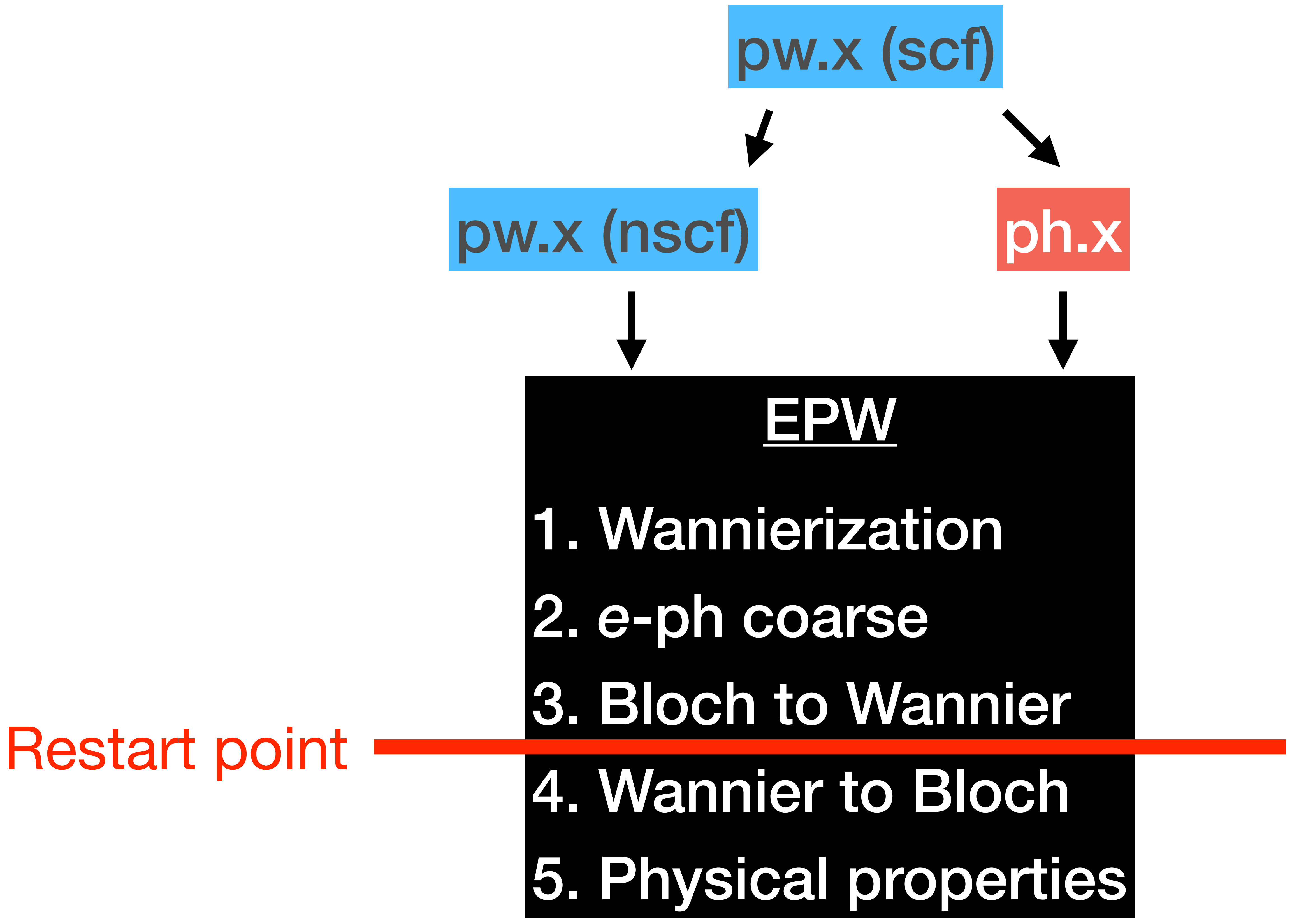
Basic input templates

```
--
&inputepw
  prefix      = 'pb',
  outdir      = './'
  dvscf_dir   = '../phonon/save'

  elph        = .true.
  epwwrite    = .true.
  epwread     = .false.

  wannierize  = .true.
  nbndsub     =  4
  bands_skipped = 'exclude_bands = 1-5'
  num_iter    = 300
  dis_win_max = 21
  dis_froz_max= 13.5
  proj(1)     = 'Pb:sp3'
  wannier_plot= .true.

  nk1         = 6
  nk2         = 6
  nk3         = 6
  nq1         = 3
  nq2         = 3
  nq3         = 3
 /
```

1. e-ph vertex calculation is done only when elph=.true. (in default, .false.)

Basic input templates

```
--
&inputepw
  prefix       = 'pb',
  outdir       = './'
  dvscf_dir    = '../phonon/save'

  elph         = .true.
  epwwrite     = .true.
  epwread      = .false.

  wannierize   = .true.
  nbndsub      =  4
  bands_skipped = 'exclude_bands = 1-5'
  num_iter     = 300
  dis_win_max = 21
  dis_froz_max= 13.5
  proj(1)      = 'Pb:sp3'
  wannier_plot= .true.

  nk1          = 6
  nk2          = 6
  nk3          = 6
  nq1          = 3
  nq2          = 3
  nq3          = 3
 /
```



1. For future restart from the Wannier basis, it is desirable to set epwwrite=.true. and epwread=.false. .
2. When restarting from the Wannier basis, set epwread=.true. and epwwrite=.false.

Basic input templates

```
--
&inputepw
  prefix      = 'pb',
  outdir      = './'
  dvscf_dir   = '../phonon/save'

  elph        = .true.
  epwwrite    = .true.
  epwread     = .false.

  wannierize  = .true.
  nbndsub     =  4
  bands_skipped = 'exclude_bands = 1-5'
  num_iter    = 300
  dis_win_max = 21
  dis_froz_max= 13.5
  proj(1)     = 'Pb:sp3'
  wannier_plot= .true.

  nk1         = 6
  nk2         = 6
  nk3         = 6
  nq1         = 3
  nq2         = 3
  nq3         = 3
 /
```

1. Enables Wannierization through a library call to W90.

Basic input templates

```
--
&inputepw
  prefix       = 'pb',
  outdir       = './'
  dvscf_dir    = '../phonon/save'

  elph         = .true.
  epwwrite     = .true.
  epwread      = .false.

  wannierize   = .true.
  nbndsub      =  4
  bands_skipped = 'exclude_bands = 1-5'
  num_iter     = 300
  dis_win_max = 21
  dis_froz_max= 13.5
  proj(1)      = 'Pb:sp3'
  wannier_plot= .true.

  nk1          = 6
  nk2          = 6
  nk3          = 6
  nq1          = 3
  nq2          = 3
  nq3          = 3
 /
```

1. Specifies the number of Wannier functions

Basic input templates

```
--
&inputepw
  prefix      = 'pb',
  outdir      = './'
  dvscf_dir   = '../phonon/save'

  elph        = .true.
  epwwrite    = .true.
  epwread     = .false.

  wannierize  = .true.
  nbndsub     =  4
  bands_skipped = 'exclude_bands = 1-5'
  num_iter    = 300
  dis_win_max = 21
  dis_froz_max= 13.5
  proj(1)     = 'Pb:sp3'
  wannier_plot= .true.

  nk1         = 6
  nk2         = 6
  nk3         = 6
  nq1         = 3
  nq2         = 3
  nq3         = 3
 /
```

1. To reduce computing load, exclude unnecessary bands from the band manifold; ex) low-lying semi-core states

Basic input templates

```
--
&inputepw
  prefix       = 'pb',
  outdir       = './'
  dvscf_dir    = '../phonon/save'

  elph         = .true.
  epwwrite     = .true.
  epwread      = .false.

  wannierize  = .true.
  nbndsub      =  4
  bands_skipped = 'exclude_bands = 1-5'
  num_iter     = 300
  dis_win_max = 21
  dis_froz_max= 13.5
  proj(1)      = 'Pb:sp3'
  wannier_plot= .true.

  nk1          = 6
  nk2          = 6
  nk3          = 6
  nq1          = 3
  nq2          = 3
  nq3          = 3
 /
```

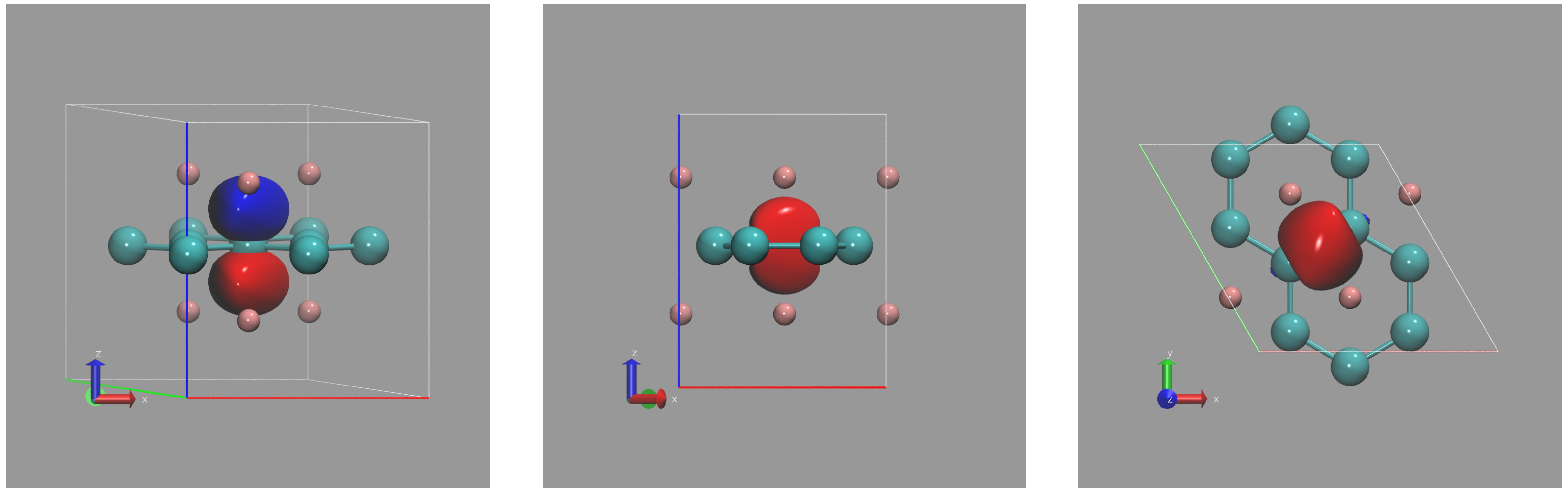
1. Specifies initial projections (or can use a few W90 techniques)

(Tue. 1&3, Marrazzo)

Basic input templates

```
--
&inputepw
  prefix       = 'pb',
  outdir       = './'
  dvscf_dir    = '../phonon/save'

  elph         = .true.
  epwwrite     = .true.
  epwread      = .false.

  wannierize  = .true.
  nbndsub      =  4
  bands_skipped = 'exclude_bands = 1-5'
  num_iter     = 300
  dis_win_max = 21
  dis_froz_max= 13.5
  proj(1)      = 'Pb:sp3'
  wannier_plot= .true.

  nk1          = 6
  nk2          = 6
  nk3          = 6
  nq1          = 3
  nq2          = 3
  nq3          = 3
 /
```

Example: MgB_2 [*P6/mmm* (No. 191)]



1. If necessary, we can directly generate cube files in EPW.

Basic input templates

```
--
&inputepw
  prefix      = 'pb',
  outdir      = './'
  dvscf_dir   = '../phonon/save'

  elph        = .true.
  epwwrite    = .true.
  epwread     = .false.

  wannierize  = .true.
  nbndsub     =  4
  bands_skipped = 'exclude_bands = 1-5'
  num_iter    = 300
  dis_win_max = 21
  dis_froz_max= 13.5
  proj(1)     = 'Pb:sp3'
  wannier_plot= .true.

  nk1         = 6
  nk2         = 6
  nk3         = 6
  nq1         = 3
  nq2         = 3
  nq3         = 3
 /
```

1. If additional W90 inputs are needed, use the keywords wdata .

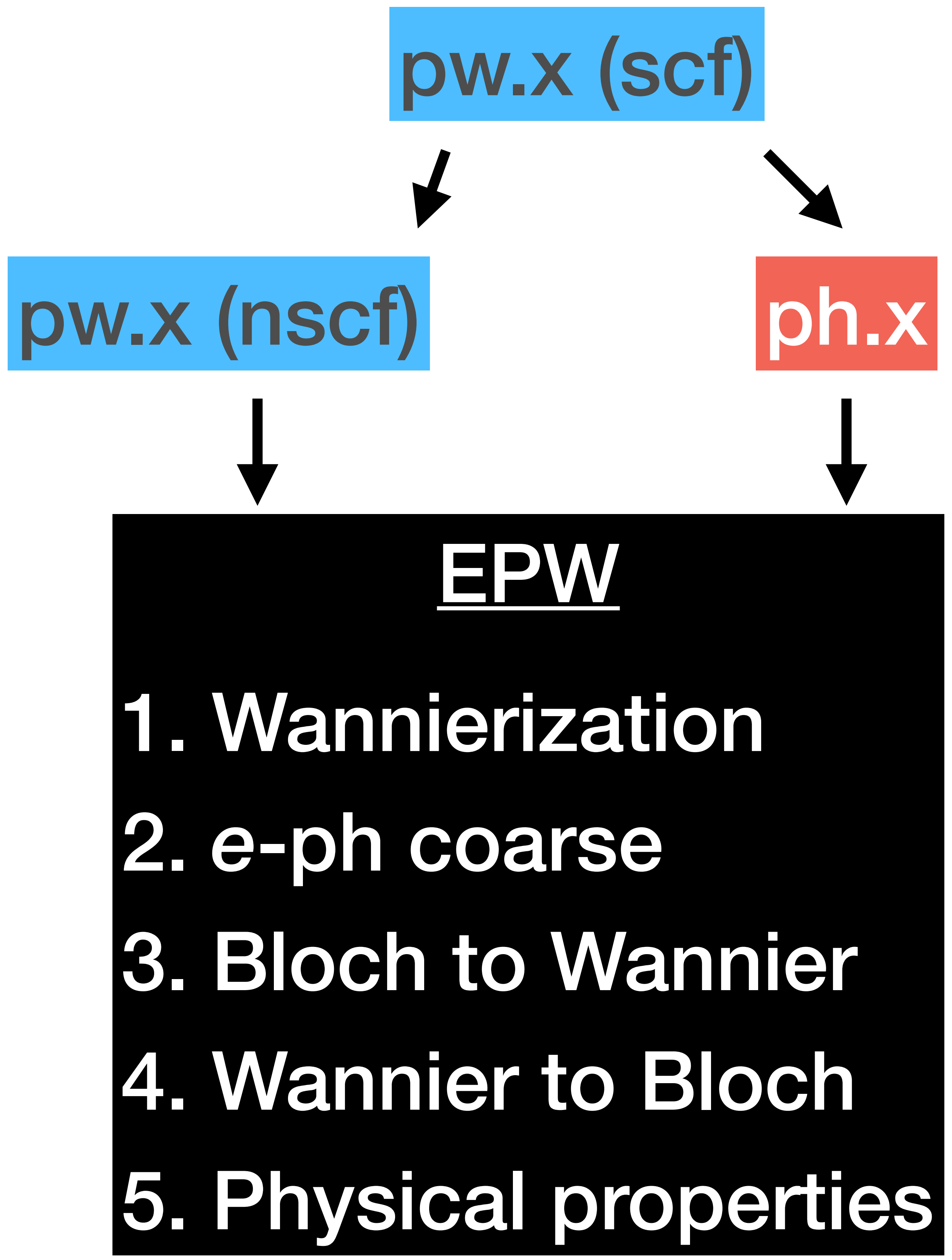
```
Ex)
  wdata(1) = 'bands_plot = .true.'
  wdata(2) = 'begin kpoint_path'
  wdata(3) = 'G 0.00 0.00 0.00 M 0.50 0.00 0.00'
  wdata(4) = 'G 0.00 0.00 0.00 K 0.33333333333 0.33333333333 0.00'
  wdata(5) = 'G 0.00 0.00 0.00 A 0.00 0.00 0.50'
  wdata(6) = 'end kpoint_path'
  wdata(7) = 'bands_plot_format = gnuplot'
```

Basic input templates

```
--
&inputepw
  prefix       = 'pb',
  outdir       = './'
  dvscf_dir    = '../phonon/save'

  elph         = .true.
  epwwrite     = .true.
  epwread      = .false.

  wannierize   = .true.
  nbndsub      =  4
  bands_skipped = 'exclude_bands = 1-5'
  num_iter     = 300
  dis_win_max = 21
  dis_froz_max= 13.5
  proj(1)      = 'Pb:sp3'
  wannier_plot= .true.

  nk1          = 6
  nk2          = 6
  nk3          = 6
  nq1          = 3
  nq2          = 3
  nq3          = 3
 /
```

1. Set coarse **k** grids (same as those in nscf) and **q** grids (same as those in ph)
2. **k** and **q** grids should be commensurate and nkX>=nqX (X=1,2,3)

Basic input templates




```
--
&inputepw
  prefix      = 'pb',
  outdir      = './'
  dvscf_dir   = '../phonon/save'

  elph        = .true.
  epwwrite    = .true.
  epwread     = .false.

  wannierize  = .true.
  nbndsub     =  4
  bands_skipped = 'exclude_bands = 1-5'
  num_iter    = 300
  dis_win_max = 21
  dis_froz_max= 13.5
  proj(1)     = 'Pb:sp3'
  wannier_plot= .true.

  nk1         = 6
  nk2         = 6
  nk3         = 6
  nq1         = 3
  nq2         = 3
  nq3         = 3
 /
```

1. Set coarse **k** grids (same as those in nscf) and **q** grids (same as those in ph)
2. **k** and **q** grids should be commensurate and nkX>=nqX (X=1,2,3)

Additional inputs depending on calculations

Added to the basic template

```
band_plot    = .true.

filkf        = 'path2.dat'
filqf        = ‘path2.dat'
```

If you want to plot electronic bands and phonon dispersions, set **band_plot=.true.** .
Additionally, provide the **k** (**q**)-point lists .

Additional inputs depending on calculations

Added to the basic template

```
prtgkk       = .true.

filqf        = 'path1.dat'
nkf1         = 1
nkf2         = 1
nkf3         = 1
```

If you want to plot *e*-ph vertex along **k** (or **q**)-point path, set **prtgkk=.true.** .
Additionally, provide the **k** (or **q**)-point lists .

Additional inputs depending on calculations

$$\gamma_{\mathbf{q}\nu} = \Pi''_{\mathbf{q}\nu} = 2\pi\omega_{\mathbf{q}\nu} \sum_{nm} \int_{\mathrm{BZ}} \frac{d\mathbf{k}}{\Omega_{\mathrm{BZ}}} |g_{mn,\nu}(\mathbf{k},\mathbf{q})|^2 \delta(\varepsilon_{n\mathbf{k}} - \varepsilon_{\mathrm{F}})\delta(\varepsilon_{m\mathbf{k}+\mathbf{q}} - \varepsilon_{\mathrm{F}}),$$

Added to the basic template

```
phonselfen  = .true.
delta_approx= .true.

fsthick      = 1 ! eV
temps        = 0.075 ! K
degaussw     = 0.2 ! eV

filqf        = 'path2.dat'
nkf1          = 20
nkf2          = 20
nkf3          = 20
```

If you want to calculate phonon self-energy within the delta approximation,

1. set **phonselfen=.true.** and **delta_approx= .true.**
2. **degaussw** for the Gaussian width of the Gaussian function for the approx. of the Delta function
3. Fine **k**- and **q**-point lists
4. Fermi window, **fsthick** : consider only states within Fermi energy +- fsthick
5. temps (temperature) is not used within Delta approximation

Additional inputs depending on calculations

$$\Sigma''_{n\mathbf{k}}(\omega, T) = \pi \sum_{m\nu} \int_{\mathrm{BZ}} \frac{d\mathbf{q}}{\Omega_{\mathrm{BZ}}} |g_{mn,\nu}(\mathbf{k},\mathbf{q})|^2 \{[n_{\mathbf{q}\nu}(T) + f_{m\mathbf{k}+\mathbf{q}}(T)]\delta(\omega - (\varepsilon_{m\mathbf{k}+\mathbf{q}} - \varepsilon_{\mathrm{F}}) + \omega_{\mathbf{q}\nu})$$
$$+ [n_{\mathbf{q}\nu}(T) + 1 - f_{m\mathbf{k}+\mathbf{q}}(T)]\delta(\omega - (\varepsilon_{m\mathbf{k}+\mathbf{q}} - \varepsilon_{\mathrm{F}}) - \omega_{\mathbf{q}\nu})\},$$

Added to the basic template

```
elecselfen  = .true.
efermi_read = .true.
fermi_energy= 9.6

fsthick     = 7.0
temps       = 20
degaussw    = 0.05
```

If you want to calculate electron self-energy,

1. set **elecselfen=.true.** .
2. **degaussw** for the Gaussian width of the Gaussian function for the approx. of the Delta function
3. Fine **k**- and **q**-point lists
4. Fermi window, **fsthick** : consider only states within Fermi energy +- fsthick
5. **temps** (temperature)
6. **efermi_read** and **fermi_energy** for Fermi energy (in default, Fermi energy is calculated)

Supp.1) Restart input flags

- **epbread** and **epbwrite**: enable restart from the *e*-ph vertex in the Bloch basis on coarse grids
- **epwread** and **epwwrite**: enable restart from the e-ph vertex in the Wannier basis
- + several restart flags [will be covered during other Hands-on sessions]

Supp.2) specification of k (q) points

- Coarse grids
  - (Γ-centered) Regular grids: **nkX** , **nqX** (X=1,2,3)
- Fine grids
  - (Γ-centered) Regular grids: **nkfX** , **nqfX** (X=1,2,3)
  - **k** (**q**)-point lists: **filkf** (**filqf**) for bands and e-ph calculations
  - Random grids.

Supp.2) specification of k (q) points

- Coarse grids
    - (Γ-centered) Regular grids: **nkX** , **nqX** (X=1,2,3)
- Fine grids
    - (Γ-centered) Regular grids: **nkfX** , **nqfX** (X=1,2,3)
    - **k** (**q**)-point lists: **filkf** (**filqf**) for bands and e-ph calculations
    - Random grids.

Supp.2) specification of k (q) points

- Coarse grids
  - (Γ-centered) Regular grids: **nkX** , **nqX** (X=1,2,3)
- Fine grids
  - (Γ-centered) Regular grids: **nkfX** , **nqfX** (X=1,2,3)
  - **k** (**q**)-point lists: **filkf** (**filqf**) for bands and e-ph calculations
  - Random grids.

Example of file for **k** (**q**)-points lists

\# of points

cartesian or crystal

weights
(usually, not meaningful)

```
41 cartesian
   1.000000000   0.000000000   0.000000000   1.0
   0.950000000   0.000000000   0.000000000   1.0
..
```

Supp.3) lpolar for polar materials

- lpolar = .true. : Enable the correct Wannier interpolation in case of polar materials (Mon. 1, Giustino)

Summary of tutorials

Summary of Exercises and Problem

Goal

Practice for how to check the quality of Wannier interpolation of physical quantities for a future production run

Summary of Exercises and Problem

Goal

Practice for how to check the quality of Wannier interpolation of physical quantities for a future production run

To be specific, check the quality of interpolation for the electron and phonon band structures and the electron-phonon matrix elements

Summary of Exercises and Problem

- Exercise 1: Pb
  - Electron and phonon band structures & phonon linewidth
  - Compare the electron and phonon band structures from EPW and DFPT

Summary of Exercises and Problem

- Exercise 2: SiC (polar material, **lpolar=.true.**)
  - *e*-ph vertex, electron and phonon band structures & electron linewidth
  - Compare *e*-ph vertex, electron and phonon band structures from EPW and DFPT

Summary of Exercises and Problem

- Problem: SiC (polar material, **lpolar=.true.**)
  - *e*-ph vertex, but different wave vector for initial states
  - Compared with that from DFPT

Additional notes

- The number of processes should be the same as that of *k* pools (-nk): relaxed in EPW v6
- Coarse k grids should cover the FBZ with the range [0, 1) in crystal units: relaxed in EPW v6
- (Not essential) Use of reduce_io=.true. to reduce I/O to the strict minimum in phonon calculations
- (Not essential) Connect Frontera with X11 forwarding if you want to use gnuplot

More Info

- https://docs.epw-code.org/doc/Inputs.html
- https://forum.epw-code.org

This slide (Tue.4.Lee.pdf) is at /work2/06868/giustino/EP-SCHOOL/ .