PWscf ToolKit

(pwtk.ijs.si)

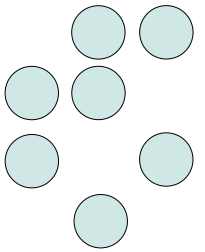
A **Tcl scripting interface** for **Quantum ESPRESSO**, aimed at flexible and productive framework.

Anton Kokalj

Jožef Stefan Institute, Ljubljana, Slovenia

based on presentation given at

2015 Quantum ESPRESSO developer's meeting

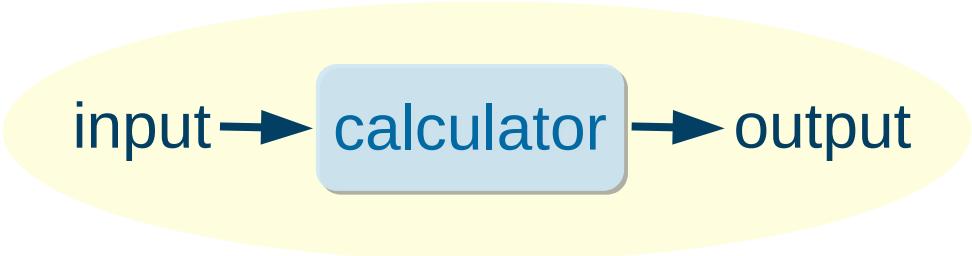


Why scripting?

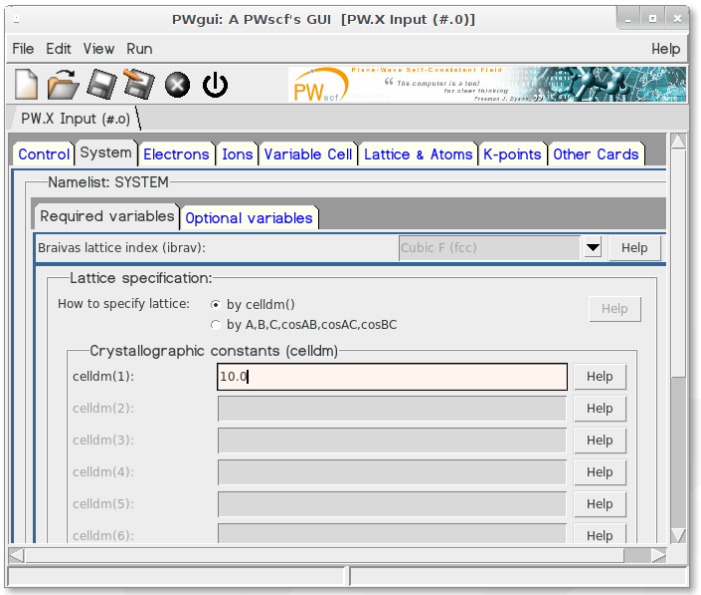
(a user's perspective)

Three paradigms

- oldie goldie I/O paradigm
  - not easy for novices
  - **rigid**



- GUI
  - *“press a button”*
  - easy for novices but **very rigid**



- Scripting
  - **flexible** & powerful
  - not easy for novices → starting barrier:
    - programming skills
    - yet another new “specific” syntax (API)

PWTK: How it got started ?

- I/O paradigm is too rigid (static)

consider performing many similar calculations
whose inputs differ only by a bit

↓

each calculation requires its own input file(s)

↓

many input files

↓

cumbersome and error prone

↓

easy way out: shell scripts

↓

soon get messy for more complicated cases

more systematic approach needed: **scripting interface**

PWTK: How it got started ?

ecutwfc.pwtk - GNU Emacs at cl

File Edit Options Buffers Tools Tcl Help

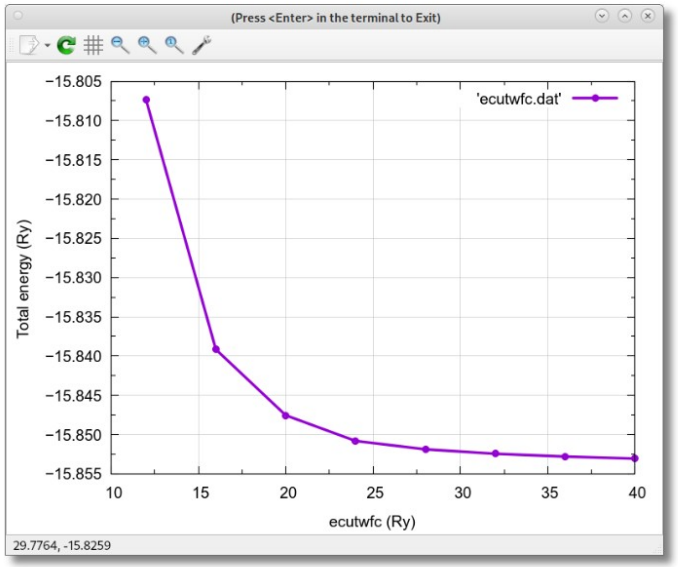
Shell script

```
#!/bin/sh

rm -f ecut.dat

for ecut in 12 16 20 24 28 32 36 40
do
    # create pw.x input file
    cat > scf.Si.e-$ecut.in << EOF
 &CONTROL
    prefix='silicon'
 /
 &SYSTEM
    ibrav =  2
    celldm(1) = 10.2
    nat =  2
    ntyp = 1
    ecutwfc = $ecut
 /
 &ELECTRONS
 /
ATOMIC_SPECIES
   Si  28.086  Si.pz-vbc.UPF
ATOMIC_POSITIONS
   Si 0.00 0.00 0.00
   Si 0.25 0.25 0.25
K_POINTS automatic
   4 4 4   1 1 1
EOF

    # run  pw.x
    mpirun -np 2 pw.x -in scf.Si.e-$ecut.in > scf.Si.e-$ecut.out

    # get "total energy" and write it to a file

    grep -e 'kinetic-energy cutoff' -e ! scf.Si.e-$ecut.out | \
        awk '/kinetic-energy/ {ecut=$(NF-1)}
             /!/              {print ecut, $(NF-1)}' >> ecut.dat

done
```

-:--- **ecutwfc.sh** All (40,0) (Shell-script[sh] AC)

PWTK script

```
# load the pw.x input from file
load_fromPWI scf.Si.in

scanpar ecut [seq 12 4 40] {
    # set ecutwfc & run pw.x
    SYSTEM "ecutwfc = $ecut"
    runPW scf.Si.e-$ecut.in

    # get "total energy" and write it to a file
    write ecut.dat [pwo_totene scf.Si.e-$ecut.out]
}

# plot the result

plot -xr 10:40 -yf %.3f -xl "ecutwfc (Ry)" -yl "Total energy (Ry)" ecut.dat
```



-:--- **ecutwfc.pwtk** All (17,1) (PWTK-mode)

Wrote /home/tone/prog/q-e/pwtk/slides/aux/ecutwfc.pwtk

About **PWTK**

- a mid-size software package
  (≈20,000 lines of code; ≈40,000 lines including documentation)
- written in Tcl
- free software (GNU General Public License)
- web page: **http://pwtk.ijs.si/**
  or **http://pwtk.quantum-espresso.org/**

PWTK: basic philosophy

- Keep the syntax close to original input syntax

pw.x input file

```
&CONTROL
   calculation = 'scf'
/
&SYSTEM
   ecutwfc = 25.0
   ecutrho = 200.0
   ...
/
ATOMIC_POSITIONS alat
   Si     0.00   0.00   0.00
   Si     0.25   0.25   0.25

K_POINTS automatic
   4 4 4    1 1 1
```

Si_bulk.in

pwtk script

```
CONTROL {
   calculation = 'scf'
}
SYSTEM {
   ecutwfc = 25.0
   ecutrho = 8*25.0
   ...
}
ATOMIC_POSITIONS alat {
   Si     0.0   0.0   0.0
   Si     1/4   1/4   1/4
}
K_POINTS automatic {
   4 4 4    1 1 1
}

runPW pw.Si.in
```

Si_bulk.pwtk

Run from the terminal as:

```
pw.x -in Si_bulk.in > Si_bulk.out
```

Run from the terminal as:

```
pwtk Si_bulk.pwtk
```

PWTK: get started

- Simple script:

```
# load the input data

import Si_bulk.pwtk

# scan the lattice-parameter

foreach a {5.2 5.3 5.4 5.5 5.6 5.7} {

   SYSTEM "A = $a"

   # perform the pw.x calculation

   runPW Si_A$a
}
```

scan_a.pwtk

```
CONTROL {
   calculation = 'scf'
}
SYSTEM {
   ibrav = 2
     nat = 2
     ...
}
ATOMIC_POSITIONS alat {
   Si     0.0   0.0   0.0
   Si     1/4   1/4   1/4
}
K_POINTS automatic {
   4 4 4    1 1 1
}
```

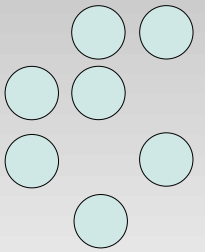
Si_bulk.pwtk

Run from the terminal as:

```
pwtk scan_a.pwtk
```

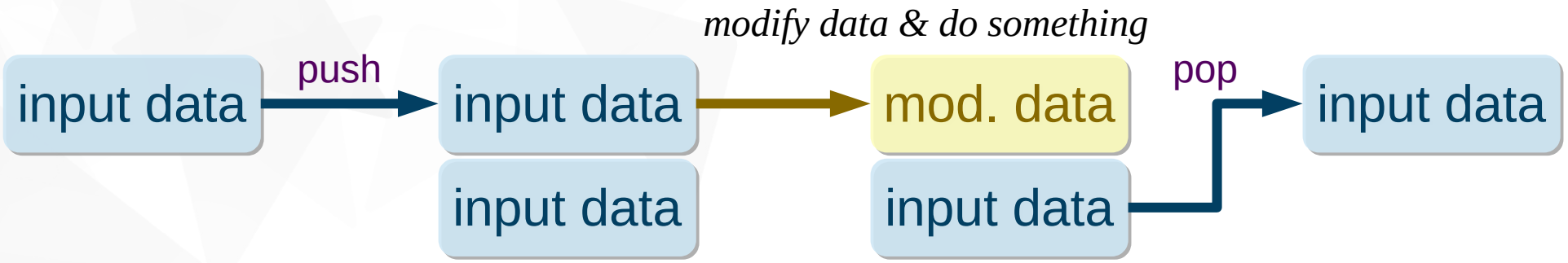
N.B.: otherwise PWTK can determine the lattice parameter automatically

PWTK's features



- (re)assignment of input data on the fly
- data retrieval and manipulation
- easy plotting of results
- workflows: e.g., charge density differences, flexible framework for PDOS calculations, automatic lattice parameter search...
- support for HPC job schedulers
- math parser
- stacking of input data
- asynchronous parallel execution of scripts
- hierarchical configuration
  - global (~/.pwtk/pwtk.tcl), project-based, local

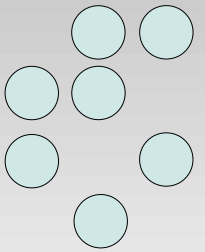
PWTK: input data stacking



... *define default input data* **(#0)**

```
# 1st calculation

input_push
    # here input data is the same as in (#0)

    ... modify input data for this calculation (#1)
    ... perform 1st calculation
input_pop

# input data is again the same as in (#0)
```

PWTK: input data stacking



```
... define default input data (#0)

# 1st calculation

input_push
   # here input data is the same as in (#0)

   ... modify input data for this calculation (#1)
   ... perform 1st calculation
input_pop

# input data is again the same as in (#0)

# 2nd calculation
input_push
   ... modify input data for this calculation (#2)
   ... perform 2nd calculation
   input_push
      ... modify input data further (#3)
      ... perform 3rd calculation
   input_pop
   # here input data is the same as in (#2)
   ... perform 4th calculation
input_pop

# input data is again the same as in (#0)
```

PWTK: configuration

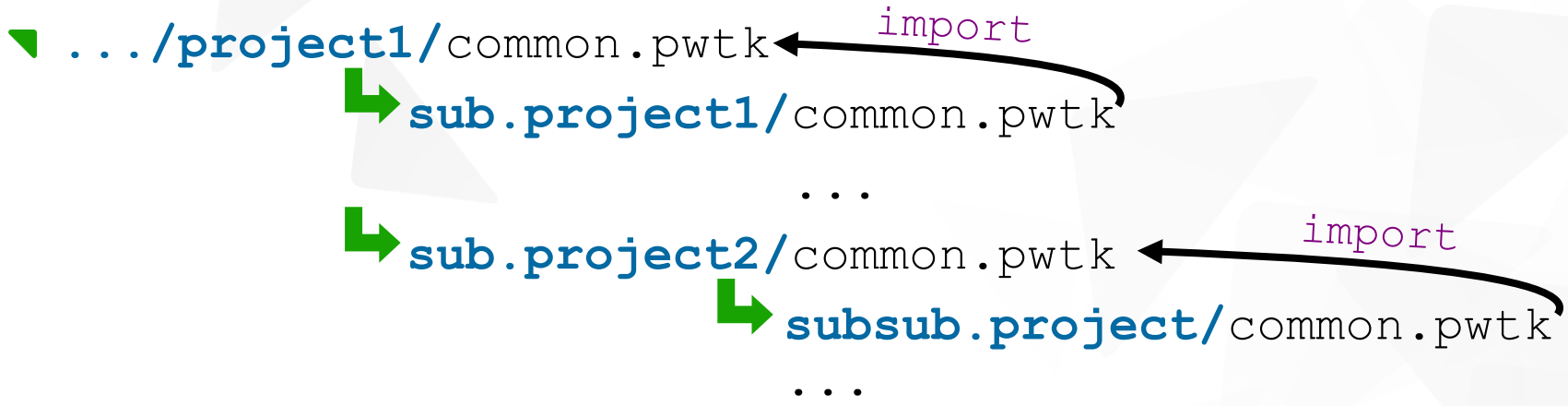
- Main configuration file: **~/.pwtk/pwtk.tcl**
  - executables and how to run them
  - special directories
  - ...

```
# HOW TO RUN QUANTUM ESPRESSO EXECUTABLES ...
# i.e.: prefix bin_dir/program postfix < in > out

prefix  mpirun -np 16
postfix -npool 2
bin_dir /opt/espresso-7.3.1


# DEFAULT DIRECTORIES ...

# top (root) directory where the scratch files are written
# i.e.: outdir == outdir_prefix/outdir_postfix

outdir_prefix /temp/$env(USER)/espresso

# location of pseudo-potentials
pseudo_dir    $env(HOME)/espresso/pseudo
```

~/.pwtk/pwtk.tcl

PWTK: hierarchical configuration

- Main configuration file: **`~/.pwtk/pwtk.tcl`**
  - executables and how to run them
  - special directories
  - ...
- Project-based configurations (via **import**), e.g.:



PWTK: data retrieval and manipulation functions

- retrieve energies, forces, pressure... from **`pw.x`** output files
- load cell parameters and atomic positions from:
  - **`pw.x`** input & output files
  - XSF files, **`neb.x`** CRD and PATH files...
- load input data from existing QE input files
- several simple structure manipulation functions
- ...

PWTK: data retrieval

- **Example #1:**
  retrieve the data from an old calculation (from the I/O files), modify some input data, and make a new calculation

```
# load data from pw.x input file
load_fromPWI relax.in

# load optimized coordinates from pw.x output file
ATOMIC_POSITIONS_fromPWO relax.out

# change some input data
SYSTEM {
    ecutwfc = 35, ecutrho = 35*8
}
K_POINTS automatic {
    6 6 6    1 1 1
}

# run pw.x with new parameters
runPW relax.e35_k6
```

PWTK: structure manipulation

- **Example #2:**
  scan different lateral positions of a molecule @ surface

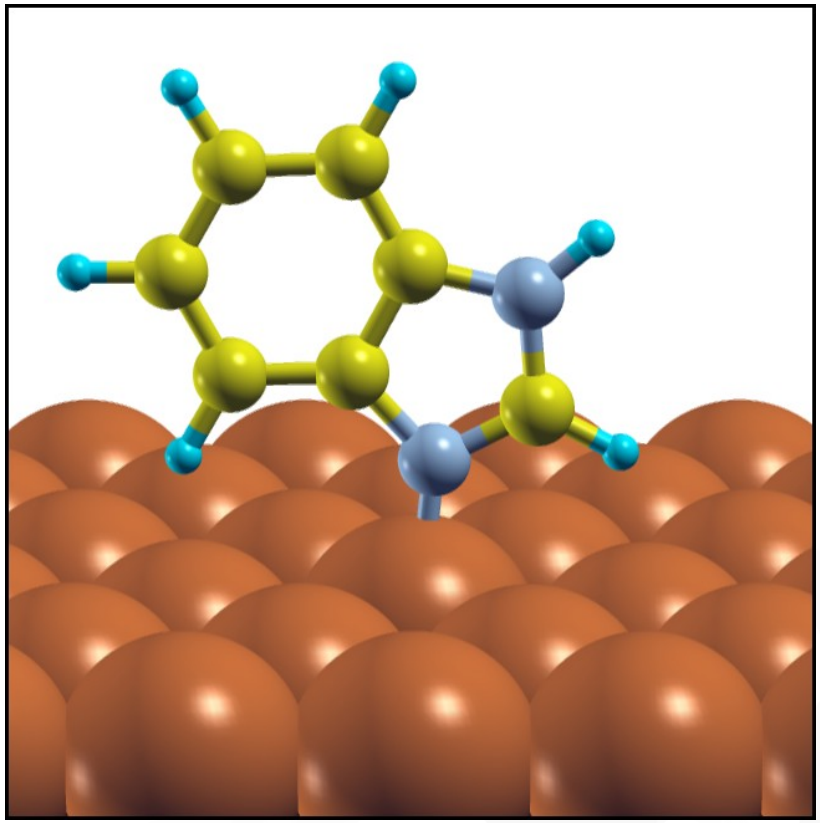
```
# load common input data with slab-only coordinates
import slab.pwtk

# scan lateral (x,y)-positions = ($i/4, $j/4)
for {set i 0} {$i < 4} {incr i} {
   for {set j 0} {$j < 4} {incr j} {

      input_pushpop {
         # add a molecule
         insertAtoms begin "
            H    0.1699+$i/4. -0.1262+$j/4.  1.2035
            H   -0.1699+$i/4. -0.1262+$j/4.  1.2035
            C    0.0928+$i/4.  0.0000+$j/4.  1.1907
            C   -0.0928+$i/4.  0.0000+$j/4.  1.1907
         "

         # perform $i,$j calculation
         runPW relax.xy-$i.$j
      }
   }
}
```

PWTK: workflow example (#1)

- **Workflow example #1:**
  charge density difference between the mol–surf

$$\Delta\rho(\mathbf{r}) = \rho_{\text{mol/surf}}(\mathbf{r}) - \rho_{\text{mol}}(\mathbf{r}) - \rho_{\text{surf}}(\mathbf{r})$$

I/O paradigm:
QE requires 6 (or 7) input files & calculations

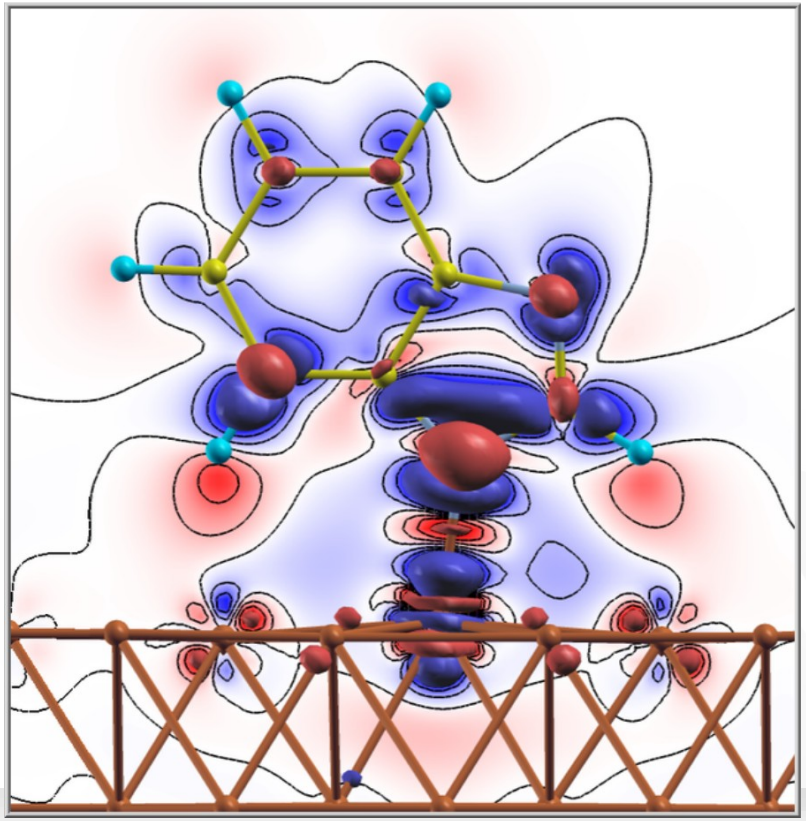


PWTK: workflow example (#1)

- **Workflow example #1:**
  charge density difference between the mol–surf

$$\Delta\rho(\mathbf{r}) = \rho_{\text{mol/surf}}(\mathbf{r}) - \rho_{\text{mol}}(\mathbf{r}) - \rho_{\text{surf}}(\mathbf{r})$$

```
# load the mol-surf input data
import mol-slab.pwtk

# instruct the pwtk how to calculate \Delta\rho
DIFDEN {
    name(1)    = 'mol-surf'
    segment(1) = 'all'
    weight(1)  = 1.0

    name(2)    = 'mol'
    segment(2) = 1-15
    weight(2)  = -1.0

    name(3)    = 'surf'
    segment(3) = 16-
    weight(3)  = -1.0
}

# do the actual Δρ(r) calculation
difden_run difden.mol-surf
```



PWTK: workflow example (#2)

- **Workflow example #2:**
  calculate and plot projected density of states

```
# load the FeO-bulk input data
import FeO.pwtk

# set the projwfc.x input data
PROJWFC {
    ngauss  = 0
    degauss = 0.01
    DeltaE  = 0.05
}

# calculate the PDOS
pdos_run -scf -k {6 6 6} FeO

# sum LDOSes for Fe1, Fe2, O atoms
sumldos -f FeO -s Fe1 Fe1.ldos
sumldos -f FeO -s Fe2 Fe2.ldos
sumldos -f FeO -s O   O.ldos

# plot LDOSes
ldos_plot -xr -10:5 -e nscf.FeO.out \
          {Fe1.ldos Fe2.ldos O.ldos} FeO.ldos
```