`run.sh` that can be used to run this example :

```
$ cd localdir
$ ls
$ ./run.sh /install/dir
```

In case you omit the argument `/install/dir` the default `/usr/local` is assumed.

We recommend that you enter the commands manually to gain experience with the steps needed for a TranSIESTA calculation, and a short description of the required steps is following here:

In order to construct the electrode Hamiltonian, we need to run `transiesta` for the electrodes we wish to use. But first, we must copy or symbolically link the pseudopotentials to the directory we are running in:

```
$ cd elec
$ cp /install/dir/share/transiesta/potentials/vps/Al.vps .
```

Next, we run the code:

```
$ transiesta Alchain.fdf
$ ls
```

The script produces a series of files. In this context, only the file `AlElec.TSHS` is of interest, because it holds a description of the electrode hamiltonian.

We proceed to calculate the electronic structure of the C-H molecule in between two Al electrodes. But first, we must copy the pseuodpotentials to the run directory:

```
$ cd ..
$ cp /install/dir/share/transiesta/potentials/vps/Al.vps .
$ cp /install/dir/share/transiesta/potentials/vps/C.vps .
$ cp /install/dir/share/transiesta/potentials/vps/H.vps .
```

and then run the code:

```
$ transiesta AlCH+0.fdf > AlCH+0.out
$ ls
```

There are several output files. Those of interest include:

`Al.* H.* C.*` Hold the aluminium pseudopotential and basis sets for the aluminum, hydrogen and carbon atoms, respectively.

`AlCH.DM` Equilibrium density matrix

`AlCH.TSDE` Nonequilibrium density matrix and nonequilibrium energy density matrix

`AlCH.TSHS` Nonequilibrium Hamiltonian

`out.fdf` Output of which FDF parameters were used and what their values are.

Now to calculate the transmission coefficient in the energy window -2 eV to 2 eV around the Fermi level, we use the `$ tsscan` utility program:

```
$ tsscan AlCH+0.fdf
```

At the end of the run, the current, voltage and temperature are output to screen. Since there is no voltage applied, the current here is zero. The transmission coefficients and projected density of states are found the file `AlCH+0.TRANS` , which is a text file with three columns. The first column is the energy of the incident electrons, with $E = 0$ representing the average of the left and right electrochemical potentials. The second column contains the total transmission probability for electrons incident from deep within the left or right electrodes. The third column contains the density of states projected onto the orbitals of the contact region.

To visualize the left scattering state at 0.5 V, we use the utility program `tspsi`:

```
$ tspsi AlCH+0.fdf kl 0 0.5 psi
$ ls *.cube *.xyz
```

The files `psi.kl.0.*.cube` contain the atomic positions in cartesian coordinates and scattering state calculated on a real space grid in cube file format.

To perform a calculation with an applied bias of 1V and afterwards calculate the transmission coefficients type:

```
$ transiesta AlCH+10.fdf > AlCH+10.out
$ tsscan AlCH+10.fdf > AlCH+10.scan.out
```

The output from `tsscan` will also tell you the electric current flowing from left to right.

5 Electrode Setup

In order to calculate the electronic structure of a system under external bias, TranSIESTA attaches the system to semi-infinite electrodes which extend to the left and right of the contact region. Examples of electrodes would include surfaces, nanowires, nanotubes or even atomic chains. The electrode *must* be oriented along the z-axis and the unit cell along the z-direction must be large enough so that orbitals within the unit cell only interact with a single nearest neighbour cell (the size of the unit cell can thus be derived from the range of support for the orbital basis functions). The electrode description is also used in `tsscan` and `tspsi` .

The electrodes are generated by a separate `transiesta` run on a bulk system. The results are saved in a file with extension `.TSHS` which contains a description of the electrode unit cell, the position of the atoms within the unit cell, as well as the Hamiltonian and overlap matrices that describe the electronic structure of the lead. Using the commands below, one can generate a variety of electrodes based upon this initial run. Typical use of `transiesta` would involve reusing the same electrode for several `transiesta` runs. Future `transiesta` distributions will include a database of electrodes as well as a utility to align the contact region with the electrode. In TranSIESTA 0.9 , one must manually provide a valid description of the electrode and the atomic coordinates input to `transiesta` must conform with this description. At runtime, the `transiesta` coordinates are checked against the electrode coordinates and the program stops if there is a mismatch.

5.1 Electrode Setup Options

TS.SaveHS (*logical*): Save the Hamiltonian in the file with extension `.TSHS`. Must be set to true when calculating the electrode Hamiltonian. The `.TSHS` file must also be generated

in `transiesta` calculations if `tsscan` or `tspsi` is to be used after the run. *Must* be true during an electrode calculation.

Default value: true

TS.HSFileLeft (*string*): Name of the `.TSHS` file output from the initial electrode run. *N.B.*: The program will stop if this file is not found.

Default value: ``_NONE_''

TS.HSFileRight (*string*): Name of `.TSHS` file describing right electrode. See **TS.HSFileLeft**.

Default value: ``_NONE_''

TS.ReplicateA1Left (*integer*): Number of replications of the left electrode unit cell along the A1 direction.

Default value: 1

TS.ReplicateA1Right (*integer*): Number of replications of the right electrode surface unit cell along the A1 direction.

Default value: 1

TS.ReplicateA2Left (*integer*): Number of replications of the left electrode surface unit cell along the A2 direction.

Default value: 1

TS.ReplicateA2Right (*integer*): Number of replications of the right electrode surface unit cell along the A2 direction.

Default value: 1

TS.NumUsedAtomsLeft (*integer*): The number of electrode atoms to include in the left lead (for example it could be 2 if only the Greens function of the first two layers of a fcc(111) surface is needed and in which case you need 3 atoms in the bulk unit cell to represent the A,B,C,A,.. stacking). Must be less than the number of atoms in the simple unit cell of the left electrode. If it is less than the number of atoms in the simple unit cell, the last `TS.NumUsedAtomsLeft` atoms are taken. If it is less than the number of atoms in the simple unit cell, the atoms in the left electrode *must* be ordered according to their coordinate along the z-direction, from smallest to largest.

Default value: Number of atoms in the simple unit cell of the Left electrode.

TS.NumUsedAtomsRight (*integer*): The number of electrode atoms to include in the right lead. Must be less than the number of atoms in the simple unit cell of the right electrode. If it is less than the number of atoms in the simple unit cell, the first `TS.NumUsedAtomsRight` atoms are taken. If it is less than the number of atoms in the simple unit cell, the atoms in the right electrode *must* be ordered according to their coordinate along the z-direction, from smallest to largest.

Default value: Number of atoms in the simple unit cell of the Right electrode.

TS.BufferAtomsLeft (*integer*): Number of atoms starting from the first atom to neglect in the TranSIESTA run (their density matrix will be fixed).

Default value: 0

TS.BufferAtomsRight (*integer*): Number of atoms starting from the last atom to neglect in the TranSIESTA run (their density matrix will be fixed).

Default value: 0

TS.SurfaceAtomsLeft (*integer*): Used to subdivide the contact region into a left surface, a molecular region, and a right surface (for post-processing purposes only). If nonzero, the first `TS.SurfaceAtomsLeft` atoms after the last atom in the left lead are counted as part of the surface.

TS.SurfaceAtomsRight (*integer*): If nonzero, the first `TS.SurfaceAtomsRight` atoms before the first atom in the right lead are counted as part of the surface.

5.2 Matching TranSIESTA Coordinates

Once an electrode description has been chosen, one must set up the atomic coordinates for use in TranSIESTA . It is important to note that the positions of the electrode atoms in the TranSIESTA unit-cell *must* be set up according to the following rules:

- The first `TS.BufferAtomsLeft` atoms correspond to buffer atoms that will not be used in the TranSIESTA calculation, but which are used in the SIESTA calculation.
- The next `TS.NumUsedAtomsLeft*TS.ReplicateA1Left*TS.ReplicateA2.Left` correspond to replications of the used atoms of the electrode. Starting with the first used atom in the primitive electrode, each atom in the unit cell is replicated in the **A1**-direction and then in the **A2**-direction. The vectors $(\mathbf{A1}, \mathbf{A2})$ correspond to vectors in the xy plane and **A3** *must* be in the z-direction. If for example $(\mathbf{A1}, \mathbf{A2}) = (\hat{x}, \hat{y})$ the x will change fastest, then comes y, and then comes z.
- The next atoms correspond to the contact region.
- The next `TS.NumUsedAtomsRight*TS.ReplicateA1Right*TS.ReplicateA2Right` to replications of the used atoms of the right electrode.
- The next `TS.BufferAtomsRight` correspond to atoms that are neglected in the `transiesta` part of the calculation and whose density is fixed by a SIESTA calculation.

During a TranSIESTA run, several checks of the atomic coordinates will be carried out to validate that they are consistent with the description of the electrode. This is the most common problem when running TranSIESTA and thus much care should be taken when setting up a calculation.

6 Program Usage, Options and Output

6.1 `transiesta`

usage: `transiesta file.fdf`
The file `file.fdf` contains the input, while output is written to standard output.

6.1.1 General Options

SolutionMethod (*string*): Must be set to `transiesta` in order to perform a TranSIESTA calculation.

Default value: `diagon`

TS.Voltage (*physical*): The voltage applied along the z-direction of the unit cell. The Poisson equation is solved with the boundary condition that the bulk potentials are shifted with the applied voltage.

Default value: 0.0 eV

6.1.2 Complex Contour Integration Options

TS.ComplexContour.Emin (*physical*): The starting point of the complex energy contour. In a TranSIESTA run this value should be *below* the lowest energy in the energy spectrum – otherwise some charge will be missing in the integration.

Default value: -3 Ry.

TS.ComplexContour.NumCircle (*integer*): Number of points along the arc part of the contour (starting at **TS.ComplexContour.Emin** and ending at $E_F = 0$).

Default value: 24

TS.ComplexContour.NumLine (*integer*): Number of points on the line part of the contour.

Default value: 4

TS.ComplexContour.NumPoles (*integer*): Number of Fermi poles that the complex contour should include.

Default value: 6

6.1.3 Bias Contour Integration Options

TS.BiasContour.Eta (*real*): Small finite complex part of the real energy contour.

Default value: 10^{-5} Ry

TS.BiasContour.Method (*string*): This describes how the points on the real axis contour are chosen.

`Sommerfeld:` equally spaced points with Sommerfeld expansion for including the electron temperature

`GaussFermi:` Gaussian quadrature weighted with the Fermi distribution function

Default value: `GaussFermi`

TS.BiasContour.NumPoints (*integer*): Number of contour points on the close-to-real axis part of the contour in the voltage bias window.

Default value: 10

6.1.4 Output

`transiesta` generates several output files. The output files are named `Label.ext`, where `Label` is defined using the `SystemLabel` FDF command , and `.ext` depends on the type of the output. Below we list the `.ext` files which are specific to `transiesta` . For a description of the other output files, we refer the user to the SIESTA manual.

`.DM` : The SIESTA density matrix. SIESTA initially performs a calculation at zero bias assuming periodic boundary conditions, which is used as a starting point for the `transiesta` calculation. During a `transiesta` run, the `.DM` values are used for the density matrix in the regions where `.TSDM` is not valid.

`.TSDE` : The TranSIESTA density matrix and energy density matrix.

`.TSHS` : The Hamiltonian corresponding to `.TSDM`.

6.2 `tsscan`

usage: `tsscan file.fdf (n label)`

Utility program that is used after a selfconsistent calculation. The program scans an energy window and calculates energy dependent quantities defined in `file.fdf`. Typical use of this program is to obtain transmission spectra and the electron current. The program will also calculate the current if the energy window spans the energy range in between the electrochemical potentials.

6.2.1 Command Line Options

n (*integer*): Number of points used to sample the energy window. Overrides value set in `file.fdf`.

label (*integer*): Prefix for output files. Overrides the `SystemLabel` value set in `file.fdf`.

6.2.2 Analysis Options

TS.Scan.E1 (*real*): minimum energy for the scan

Default value: -2.0 eV

TS.Scan.E2 (*real*): maximum energy for the scan

Default value: 2.0 eV

TS.Scan.NumPoints (*real*): Number of points in the energy scan

Default value: 20

TS.Scan.GetEigenChannels (*logical*): Calculate the transmission eigenvalues and output to `.TRANS`.

Default value: false

TS.Scan.NumChannels (*integer*): restrict number of output eigenchannels (normally determined by number of incident bloch states at a given energy).

Default value: 4

6.2.3 Output

The incident energy, transmission, density of states and transmission eigenvalues are output to `Label.TRANS`. At the end of the scan, the current is written to standard output.

6.3 `tspsi`

usage: `tspsi file.fdf state n (e0 label)`

Utility program that performs an transmission analysis at a specific energy and calculates an eigenstate on a real space grid and saves it in Gaussian cube file format for visualization.

6.3.1 Command Line Options

`state` (*integer*): This determines which type of eigenstate is output, can have the following values (*n.b.* these are not case sensitive):

- `kl`: A scattering state incident from the left electrode.
- `kr`: A scattering state incident from the right electrode.
- `nl`: A transmission eigenstate incident from the left electrode.
- `nr`: A transmission eigenstate incident from the right electrode.
- `m`: A molecular eigenstate of the MPSH.

`n` (which starts from 0) indexes the eigenstates and has different meaning depending on which type of state is output:

- `kl` or `kr`: `n` indexes the different Bloch states incident from the electrodes, which at present are unsorted
- `nl` or `nr`: `n` indexes the different transmission eigenvalues which are sorted in terms of their eigenvalue so that `n = 0` is the most transmissive eigenstate.
- `m`: `n` indexes the different MPSH eigenvalues which are sorted in terms of their eigenvalue so that `n = 0` is the MPSH eigenstate lowest in energy (if `TS.MolecularStates.Restrict` is set, then this is the lowest in energy within the window defined by `TS.MolecularStates.DE`).

  If a scattering state or transmission eigenstate is requested, `tspsi` calculates these quantities for electrons incident at energy `e0`, with a default value of 0 eV (equal to the average of the left and right electrochemical potentials).

6.3.2 Molecular Eigenstate Options

TS.MolecularStates.DE (*physical*): Used to restrict the eigenstates of the Molecular Projected Selfconsistent Hamiltonian (MPSH), such that only the states with an energy of `TS.MoleculeStates.DE` of the average chemical potential are reported and indexed.

Default value: 3.0 eV

TS.MolecularStates.Restrict (*logical*): When calculating eigenstates of the Molecular Projected Selfconsistent Hamilotnian (MPSH), only report the values of those within an energy window ± **TS.MolecularStates.DE** relative to the average chemical potential.

Default value: true

6.3.3 Cube Description Options

In order to calculate the wavefunction on a real space grid, the user must specify the dimensions of the grid using the following FDF options:

`GenCube.X0` (*physical*): Minimum x-coordinate of box.

Default value: Minimum x-value of TranSIESTA unit cell described in `file.fdf`

`GenCube.X1` (*physical*): Maximum x-coordinate of box.

Default value: Maximum x-value of TranSIESTA unit cell described in `file.fdf`

`GenCube.Y0` (*physical*): Minimum y-coordinate of box.

Default value: Minimum y-value of TranSIESTA unit cell described in `file.fdf`

`GenCube.Y1` (*physical*): Maximum y-coordinate of box.

Default value: Maximum y-value of TranSIESTA unit cell described in `file.fdf`

`GenCube.Z0` (*physical*): Minimum z-coordinate of box.

Default value: Minimum z-value of TranSIESTA unit cell described in `file.fdf`

`GenCube.Z1` (*physical*): Maximum z-coordinate of box.

Default value: Maximum z-value of TranSIESTA unit cell described in `file.fdf`

6.3.4 Output

`label.*.cube` The wavefunction in Gaussian cube format. `label` defaults to `SystemLabel` but one can direct the output to different cube files, which is useful for looking at resonances at different incident energies.

`label.MPSHEIG` File containing the MPSH eigenvalues.

7 Projected Changes and Additions

The next major release of TranSIESTA is scheduled for May 2003. In this version, the following additional features are projected:

1. Calculation of electronic structure using k-point sampling in the xy-plane.
2. Calculation of spin polarized electronic structure.
3. Parallelization of the integration of charge density.
4. A database of pseudopotentials and basis sets will be included in the distribution.
5. A database of electrode files will be included in the distribution.
6. A tool to automate the set up of the electrodes and the aligning of electrode and molecule will be included in the distribution.
7. A tool to automatically calculate current-voltage characteristics will be included.
8. Support for different output formats of `tspsi` wavefunctions.
9. Use of platform independent binary formats for storage of output.
10. Will be built upon a newer version of SIESTA with additional functionality.

8 References

1. Mads Brandbyge, Jose-Luis Mozos, Pablo Ordejon, Jeremy Taylor and Kurt Stokbro, *Density-functional method for nonequilibrium electron transport*, Phys. Rev. **B** 65, 165401 (2002).