



TxPhysics

User's Guide

Version 2.0.0

Tech-X Corporation
5621 Arapahoe Avenue, Suite A
Boulder, CO 80303
<http://www.txcorp.com>
info@txcorp.com



Contents

1	Introduction	2
2	Building and Installation	2
3	What is built?	3
3.1	Language Interoperability	3
4	txstopping – Ion interactions with materials	4
4.1	Electronic Stopping Power	4
4.1.1	Bound Electronic Stopping Power	4
4.1.2	Free Electronic Stopping Power	4
4.2	Nuclear Stopping Power	5
4.3	Target specifications	5
5	txigenelec – Ion-solid interactions	6
5.1	Ion-induced secondary electron emission	6
5.2	Neutral desorption	7
6	txsputter – Physical sputtering	7
7	txegenelec — Electron-induced secondary electron emission	8
8	txionpack — Impact and field ionization routines	8
9	txradiation – Routines for computing radiated power	8
10	txrand — Routines for generating pseudo-random numbers	9
11	Troubleshooting	9
11.1	Python on AIX	9

1 Introduction

TxPhysics is a collection of numerical libraries for computing ion stopping in solids, secondary electron emission, physical sputtering, impact ionization, impurity radiation, and field ionization. TxPhysics includes bindings for C and Python, providing interfaces to physics simulation codes in a language-independent manner. In addition, TxPhysics is designed to be built on a variety of platforms by using autotools to detect configuration information and generate Makefiles.

The sub-libraries of TxPhysics include:

- `txstopping`: Algorithms for calculating ion stopping powers, including bound electronic, free electron, and nuclear stopping
- `txigenelec`: Algorithms for calculating ion-induced secondary electron emission
- `txsputter`: Algorithms for calculating ion-induced physical sputtering rates
- `txgenelec`: Algorithms for calculating electron-induced secondary electron emission
- `txradiation`: Algorithms for calculating impurity radiation rates
- `txionpack`: Algorithms for calculating field and impact ionization rates
- `txrand`: Custom random number generators

2 Building and Installation

To build and install TxPhysics, perform the following steps. These instructions will build TxPhysics out-of-place (the preferred method) and will generate python bindings.

For users:

```
% tar -xvzf txphysics-2.0.0.tar.gz
% cd txphysics
% mkdir build
% cd build
% ../configure --prefix=/install/path --enable-python
% make
% make install
% make install-docs
```

Note: For PowerPC-based Macs using gcc 3.x, you will need to configure with extra flags:

```
../configure --prefix=/install/path --enable-python FLIBS=/path/to/libg2c.a
```

For developers:

```
% svn co https://facets.txcorp.com/code/txphysics/trunk txphysics
% cd txphysics
% config/cleanconf.sh
% mkdir build
% cd build
% ../configure --prefix=/install/path --enable-python
% make
% make install
% make install-docs
% make distcheck
```

3 What is built?

In order to use the TxPhysics libraries you should build and install them using the directions above. A number of directories will be created containing the following files:

- **\$(prefix)/lib:** Contains the static and shared object libraries libtxstopping, libtxgenelec, libtxigenelec, libtxionpack, libtxionpack, libtxsputter, and libtxrand. See below for more detailed descriptions of the routines in these libraries. If python has been enabled, also contains the directory /lib/pythonX.X/site-packages/txphysics, which contains the python modules.
- **\$(prefix)/include:** Contains the header files for all of the TxPhysics routines.
- **\$(prefix)/docs:** Contains the TxPhysics license agreement and User's guide (this document). Also contains the API documentation if `make install-docs` has been invoked.
- **\$(prefix)/tests:** Contains regression tests for the TxPhysics routines. To run the regression tests either invoke `make check` in the build directory, or `check_txphysics.sh` in the tests directory. The individual test programs provide examples on the calling syntax for the TxPhysics routines.

3.1 Language Interoperability

TxPhysics is written in C, and builds static and shared object libraries which may be linked directly to your C or C++ simulation code. TxPhysics may also be configured to generate Python bindings so that the routines may be run in a python script or interactive environment.

To use the Python bindings, one needs to import the txphysics module into your Python program. To do this, set your PYTHONPATH environment variable to point to the txphysics python directory, eg,

```
% export PYTHONPATH="/usr/local/txphysics/lib/python2.5/site-packages:$PYTHONPATH"
```

You will also need to set your LD_LIBRARY_PATH to point to the location of the shared object libraries,

```
% export LD_LIBRARY_PATH="/usr/local/txphysics/lib:$LD_LIBRARY_PATH"
```

Then you can import the TxPhysics python modules in the usual manner.

```
% ipython
Python 2.4.3 (#1, Jun 13 2006, 16:41:45)
Type "copyright", "credits" or "license" for more information.

In [1]: import txphysics

In [2]: from txphysics import
txgenelec  txigenelec  txionpack  txradiation  txrand  txsputter  txstopping

In [2]: from txphysics import txstopping

In [3]:
```

You may also import txphysics modules in your python scripts in this way. Consult the Python documentation for more information about the module search path. Java bindings for some of the routines in TxPhysics are also provided in the source code distribution. However, this is an unsupported feature at this time.

4 txstopping – Ion interactions with materials

The txstopping library computes the rate of energy deposition for ions moving in materials, including cold solids and dense plasmas. The total stopping power, or dE/dx , is comprised of contributions from bound electrons, free electrons, and nuclear collisions. Stopping power depends on the type of projectile, projectile energy, target material, and the target temperature.

4.1 Electronic Stopping Power

Electronic stopping powers represent the rate of energy loss from the projectile ion to the target electrons. It is composed of two components; bound and free, corresponding to electrons that are bound to target nuclei and those that are free (plasma and conduction electrons).

4.1.1 Bound Electronic Stopping Power

Bound electron stopping has been studied experimentally and theoretically for many years. In the txstopping library, we have implemented two models of bound electron stopping power. For projectiles with an incident energy of less than 100 MeV/amu, we compute the bound electron stopping by scaling protonic stopping power based on the theory of Brandt-Kitagawa *W. Brandt and M. Kitagawa, Phys. Rev. B. 25 (1982) p. 5631*. In this theory, the ion stopping power scales as $Z_{eff}^{2/3}$, where Z_{eff} is the effective charge state of the projectile ion. The scaling from protonic stopping powers is based on many empirical fits for a large variety of ion/target combinations. A complete description of the scaling theory can be found in the excellent book *J.F. Ziegler, J.P. Biersack and U. Littmark “The Stopping and Range of Ions in Solids” Vol 1, Pergamon Press, New York (1985)*.

We use protonic stopping powers enumerated in the PSTAR data tables from Berger, M.J., Coursey, J.S., Zucker, M.A., and Chang, J. (2005), ESTAR, PSTAR, and ASTAR: Computer Programs for Calculating Stopping-Power and Range Tables for Electrons, Protons, and Helium Ions (version 1.2.3). Available online at <http://physics.nist.gov/Star> [2006, December 12] National Institute of Standards and Technology, Gaithersburg, MD.

For ions incident with energies greater than 100 MeV/amu, we have implemented bound electronic stopping powers based on the CRANGE code developed by B. Weaver at the University of California. See *B. A. Weaver and A. J. Westphal, Nucl. Instrum. and Meth. B, 187 (2002) p. 285* for full details of the algorithms. CRANGE is appropriate for very high energy beams, and include relativistic and ultra-relativistic corrections.

Bound electron stopping is referred to as cold stopping, because in the limit of zero-temperature targets, all electrons are bound to target nuclei. If the target is not at zero temperature, some of the bound electrons are converted to free electrons, and so in general the magnitude of the bound electronic stopping would decrease. This is balanced however by an increase in the bound electronic stopping power because the bound electrons are more tightly bound to the target nuclei, which further strips electrons from the projectile.

4.1.2 Free Electronic Stopping Power

Free electrons are produced in a target when the target atoms become partially or fully ionized, for instance, when heated. Free electron stopping is different from bound electron stopping. We have implemented a model of free electron stopping due to Peter and Meyer-ter-Vehn *T. Peter and J. Meyer-ter-Vehn, Phys. Rev. A., 43(4), 1991, parts I. and II.* which is relevant for dense plasmas. This model solves a linearized Vlasov equation to provide an estimate of the free electron stopping power.

The degree of target ionization is computed by solving both a Saha equation and a coronal equilibrium equation [*D. Mosher, Phys. Rev. A., 10(6), 1974*]. The Saha equation is more accurate at higher densities and lower temperatures, while the coronal equation is better for predicting target ionization at high temperatures and sub-solid densities. We assume that the process with greater recombination dominates, and so choose the lower of the two computed values for a given temperature.

The number of free electrons is directly computed from the target ionization state. Furthermore, the total electronic stopping is made up from contributions from bound and free electrons which is weighted by the target ionization fraction. At $T_e = 0$ target atoms are not ionized at all, and all of the electronic stopping is due to bound electrons. For temperatures above zero, there is a contribution from free electrons.

Temperature effects are not simply due a finite free-electron stopping however. The more ionized target atoms become, the more effective bound electrons are at stopping positively charged projectiles. Free electrons also strip electrons bound to the projectile, further increasing the efficiency of bound electronic stopping. This latter effect can be especially important at low incident energies. Thus free electrons not only provide a new term for electronic stopping, but also modify the bound electron stopping.

4.2 Nuclear Stopping Power

Ions moving in solid materials may also lose energy thorough stochastic collisions with target nuclei, called nuclear stopping. Nuclear stopping can be comparable with electronic stopping when the ion energy is below the Bragg peak (approximately 1 MeV/amu). Typically, nuclear stopping is much smaller than electronic stopping at and above the Bragg peak.

The calculation of the nuclear contribution to the stopping power implemented in TxPhysics, is derived from a classical mechanics treatment of scattering of two particles whose interaction is described by a two-body, central force. The two-body force in this case is derived from a Coulomb potential multiplied by a suitable screened “universal” function that accounts for the effects of a dense, background material. The derivation of the nuclear contribution closely follows the treatment in Chapter 2 of Ziegler’s book as summarized on page 53. The routines in the txigenelec directory of the TxPhysics library which are primarily related to calculating the nuclear contribution of the stopping power are contained in dedx_nuclear.c.

4.3 Target specifications

Stopping powers and physical quantities that are based on stopping powers depend critically on the material properties of the target materials. The TxPhysics libraries provide tabulated information on these material properties which can be accessed through interfaces to the data structures held internally (see the API specification). Target materials may be specified by the user using a target number, detailed in table (1). The target numbers correspond to the target numbers used in the CRANGE package, and generally are given by the atomic number of the target material (for single-species targets).

Table 1: Target Numbers

Material	Target Number	Material	Target Number
H	1	Cu	29
He	2	Ge	32
C	6	Ag	47
N	7	Ba	56
O	8	Os	76
Na	11	Pt	78
Al	13	Au	79
Si	14	Pb	82
P	15	U	92
Ar	18	Air	1007
Fe	26	Water	1003
Ni	28	Stainless	10025

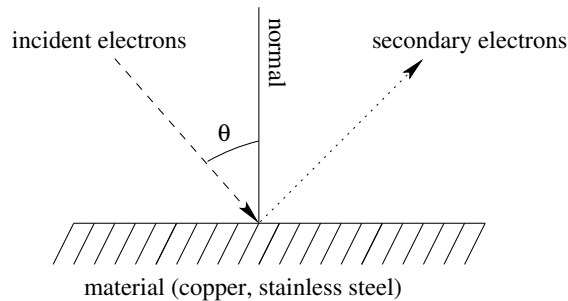


Figure 1: Geometry used by secondary electron calculations in txgenelec and txgenelec

For all txgenelec routines, incident energies are measured in MeV. All stopping powers are returned in units of $A - \text{MeVcm}^2/\text{g}$. Target electron temperatures are measured in units of eV, and free electron densities are in units of cm^{-3} . The full routine signatures and specifications can be found in the API specification.

5 txgenelec – Ion-solid interactions

The txgenelec library computes secondary electron emission and neutral desorption yields due to ion-target interactions. The secondary electron yield for most models depends on the projectile energy and angle of incidence, and the target material. Ion-induced secondary electron emission is proportional to electronic stopping, so the routines in the txstopping library are used to compute secondary electron yields. Similarly, neutral desorption depends on nuclear stopping powers.

5.1 Ion-induced secondary electron emission

(Figure 1 shows the setup assumed for these calculations.) The number of secondary electrons generated by the incident ions is in general a function of the total electronic energy loss per unit length (electronic stopping power) of the target material for the given incident energy and ion type. A number of different ion-induced secondary electron models are implemented in the TxPhysics libraries.

The first model is the constant probability model, in which a single electron is emitted with a user-specified probability for each incident ion independent of all other factors, such as the ion energy and angle of incidence.

The second model implements estimates the secondary electron yield following the experimental fit due to Rothard *et. al.* [*H. Rothard, et al, Phys. Rev. A 41, 2521 (1990)*], namely,

$$E[\gamma] = 0.14C_b \frac{1}{\cos(\theta)} \left(\frac{dE}{dx} \right)_e \quad (1)$$

where $E[\gamma]$ is the expectation of the secondary electron yield for a single incident ion, C_b is an empirically determined constant, $(dE/dx)_e$ is the electronic stopping power, and θ is the angle of incidence measured from the surface normal. An integral number of secondary electrons is emitted, chosen with a probability such that the ensemble average over many trials gives the correct expectation value.

The distribution of energies of emitted secondary electrons is computed according to the empirical two-parameter fit described in [*Stoltz et. al. "Modeling ion-induced electrons in the High Current Experiment", Phys. Plasmas 13, 056702 (2006)*]. The velocity components (scaled β) are uniformly distributed in solid angle.

5.2 Neutral desorption

TxPhysics contains an implementation of a model for neutral desorption based on the empirical values obtained by Molvik *et. al.* [Molvik *et al.*, "Gas desorption and electron emission from 1 MeV potassium ion bombardment of stainless steel", 10.1103/PhysRevSTAB.7.093202]. Neutral yield is proportional to nuclear stopping, using a model very similar to that for ion-induced secondary electron emission due to Rothard above,

$$\gamma_N = C_b \left(\frac{dE}{dx} \right)_n^{1.5} \quad (2)$$

where γ_N is the yield of neutral atoms desorbed, $C_b = 3000$. is an empirically determined constant, and $(dE/dx)_n$ is the nuclear component of the stopping power.

6 txsputter – Physical sputtering

Ions impacting solid materials may eject atoms of the target material itself. This process is known as “physical sputtering” or simply “sputtering”, and depends on the incident ion energy and angle of incidence, as well as the projectile-target material combination. The sputtering process is strongly dependent on the nuclear component of the stopping power and therefore the routines to calculate the sputtering yield make calls to routines in the txstopping library.

More detailed information on sputtering can be found in the following papers:

- P. Sigmund, *Phys. Rev. Vol 184*, 2 “Theory of Sputtering. I. Sputtering Yield of Amorphous and Polycrystalline Targets” p383 (1969).
- Y. Yamamura, H. Tawara, *Atomic Data and Nuclear Data Tables Vol 62*, “Energy Dependence of Ion-induced Sputtering Yields From Monatomic Solids at Normal Incidence” p. 149-253 (1996).
- Y. Yamamura *Nuclear Instruments and Methods in Physics Research B2*, “A Simple Analysis of the Angular Dependence of Light-Ion Sputteringormal Incidence” p. 578-582 (1984).

The sputtering yield algorithms implemented in TxPhysics are based on a modified threshold model by Yamamura and depends strongly on the nuclear component of the stopping power. The model also depends on the mass ratio and surface binding energy of target material. The implementation includes enhancements to the yield due to angular dependence of the incoming projectile. The angular dependence is based on another model by Yamamura as can be found in the reference above. The model includes two fit parameters which are $\theta_{opt}=1.4486$ and $f=1.8$.

The Yamamura model gives a non-integral yield based on the parameters provided. We provide routines that also return an integer yield, chosen with probability such that the ensemble average of many identical trials produces the desired expectation value. We provide implementations of the Yamamura model for both single incident ions, and for an array of incident ions, each with its own energy and angle of incidence. These routines return the total sputtering yield, the energy of each sputtered atom, and velocity components for each sputtered atom. Currently, velocities are assigned as themal, ie. Gaussian in each direction, but with a non-negative component in the (outward) normal direction.

It is also possible to compute the sputtering yield for alloys using these routines, by specifying the different elemental materials and the mass stoichiometry of the alloy target. Sputtering yields for each individual material element are calculated and then weighted by the stoichiometry. See the API documentation for complete details of the sputtering routine parameterizations.

7 txegenelec — Electron-induced secondary electron emission

The txegenelec portion of the TxPhysics library contains routines for modeling electron-induced secondary electron yield. The setup used by txegenelec is shown in Figure 1. Presently, these routines contain material models for copper and stainless steel only. These routines are based on the models developed at Lawrence Berkeley National Laboratory. See *M. A. Furman and M. Pivi, Phys. Rev. ST Accel. Beams 5, 124404 (2002)* for complete details. TxPhysics provides routines for computing secondary electron yields for both singly incident and arrays of incident primary electrons. See the API documentation for a complete description of the routine signatures.

For each of the electron-induced secondary electron routines, the total yield is returned, as well as the dimensionless components of normalized velocity; beta_nor, beta_tan, and beta_z. The velocity components are returned as arrays of length 10, corresponding to the model parameters.

8 txionpack — Impact and field ionization routines

The txionpack library contains routines to calculate the rates for field (tunneling) ionization of specific neutral atoms and ions, and to calculate rates of ionization due to collisions between electrons or protons and neutral atoms. This functionality can then be used to implement ionization of neutral fluids, atoms, and ions in simulation codes dealing with such fluids and particles. Conceptually, the process modeled by txionpack corresponds to the schematic in Figure 2. The explicitly modeled projectile particles (electrons or protons) are able to ionize a specific background gas.

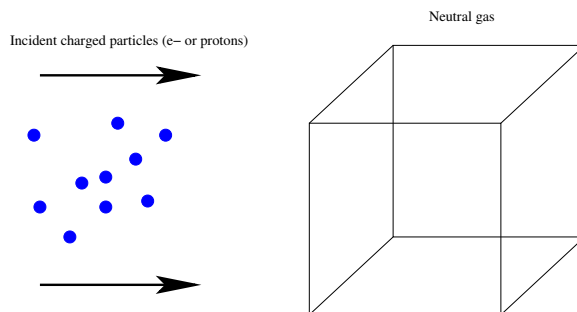


Figure 2: Charged particles moving through a uniform, background gas

Total ionization cross sections for impact ionization routines are estimated using a parameterized fit (see *Martin Reiser, Theory and Design of Charged Particle Beams [Wiley, New York, 1994]*), using data from *Rieke and Prepejchal, Phys. Rev. A 6, p. 1507 (1972)*.

Tunneling ionization rates are computed for a variety of processes based on the theoretical work of Keldysh *L. V. Keldysh, Ionization in the field of a strong electromagnetic wave. Sov. Phys. JETP, 20:1307, 1965*.

9 txradiation – Routines for computing radiated power

The txradiation library computes power radiated by ions in a plasma using a coronal model [*D. Mosher, Phys. Rev. A., 10(6), 1974*]. There are two components to the radiated power, Continuum and Line. Continuum radiation includes radiation due to both Brehmstrahlung and recombination. It is given by

$$P_c = 1.5 \times 10^{-32} \theta^{1/2} n_e n_i (\langle Z^2 \rangle + \langle Z^2 \xi_{Z-1} / \theta \rangle) \quad (3)$$

in W/cm^3 . Here θ is the electron temperature in eV, n_e and n_i are the electron and ion densities respectively in $\#/cm^3$, and the quantities in the angle brackets are the Brehmstrahlung and recombination radiation powers averaged over ionization state. Values for ξ_Z and other quantities are tabulated by *Carlson, Atomic Data, 2(1), 1970*.

The line component of the total radiated power is given by

$$P_l = 3.5x10^{-25}\theta^{-1/2}n_en_i \langle e^{-E_{N,N+1}^Z/\theta} \rangle \quad (4)$$

in W/cm^3 , where the quantity in the angle brackets is related to the oscillator strength of the (dominant) transition resonance from principal quantum state $N \rightarrow N + 1$ or an electron in state Z .

10 txrand — Routines for generating pseudo-random numbers

In order to provide consistent results across platforms, TxPhysics includes its own random number generator, a version of the “Mersenne Twister” algorithm. See *M. Matsumoto and T. Nishimura, “Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator”, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, pp. 3-30 (1998)* for details of the algorithms used to generate pseudo-random numbers using this method.

By default, TxPhysics uses a built-in, thread-unsafe version of its random number generator. If you will be using TxPhysics in a threaded application, TxPhysics can create a new random number generator dynamically for each thread. In addition, every function in TxPhysics which uses random numbers comes in two forms, a thread-unsafe version and a thread-safe version. The thread-unsafe version uses the built-in random number generator, whereas the thread-safe version requires you to pass in a pointer to your private random number generator data (arranged as an array of unsigned long integers) and a pointer to a random function which takes an array of unsigned long integers as its argument and returns a double precision random number. You are free to use your own random number generator, if it can specify its data as an array of unsigned longs and you have a random function which takes an array of unsigned long integers as its argument.

11 Troubleshooting

- Problem: The build in the Python directory exits with the error
ld: multiple definitions of symbol _derf_
Solution: The g2c library is being linked statically. Make sure that libg2c.so or libg2c.dylib is present in your g2c library directory.
- Problem: On the Mac OS X, the build in the Python directory does not produce shared libraries
Solution: Remove the ltmain.sh in the config directory, replace it with a copy of ltmain.sh appropriate for your system, copy an appropriate version of libtool.m4 to acinclude.m4 in the cmee directory, in the cmee directory run ‘autoreconf’, configure the project again, make clean and make the project.

11.1 Python on AIX

Calling the TxPhysics routines from Python is possible only on platforms where a shared library can be built. This is not possible on some AIX platforms. The Python bindings for TxPhysics have not been tested under AIX.