ecx

M. Skocic

Table of Contents

NAME
ecx - library for electrochemistry

LIBRARY
ecx (**-libecx**, **-lecx**)

SYNOPSIS
```
ecx (Fortran): use ecx
ecx (C): include "ecx.h"
ecx (python): import pyecx
```

DESCRIPTION
ecx a Fortran library for providing a collection of routines for electrochemistry. A C API allows usage from C, or can be used as a basis for other wrappers. A Python wrapper allows easy usage from Python.

It covers:

o kinetics
Nernst, Butler-Volmer

o electrochemical
Impedance, Admittance, Circuit Elements, Equivalent Circuits

o photoelectrochemistry
Photocurrent, Band-gap, space charge.

The C API is defined by adding a prefix to the functions from the Fortran API due to the lack of module/namespace feature in the C language. The functions are therefore following this template: (c_prefix)fortran_func.

- (ecx_)get_version
- (ecx_core_)kTe
- (ecx_eis_)z
- mm
- (ecx_kinetics_)nernst
- (ecx_kinetics_)sbv
- (ecx_kinetics_)bv
- (ecx_eis_)z

NOTES
To use ecx within your fpm <https://github.com/fortran-lang/fpm> project, add the following lines to your file:

```
[dependencies]
ecx = { git="https://github.com/MilanSkocic/ecx.git" }
```

EXAMPLE
Example in Fortran:

```
program example_in_f
    use iso_fortran_env
    use ecx
    implicit none

    real(real64) :: w(3) = [1.0d0, 1.0d0, 100.0d0]
    real(real64) :: r = 100.0d0
    real(real64) :: p(3) = 0.0d0
```

```
        character(len=1) :: e
        integer :: errstat
        complex(real64) :: zout(3)
        character(len=:), pointer :: errmsg

        p(1) = r
        e = "R"
        call z(p, w, zout, e, errstat, errmsg)
        print *, zout
        print *, errstat, errmsg
    end program
```

Example in C:

```
    int main(void){
        int errstat, i;
        double w[3] = {1.0, 1.0, 1.0};
        double p[3] = {100.00, 0.0, 0.0};
        ecx_cdouble z[3] = {ecx_cbuild(0.0,0.0),
                            ecx_cbuild(0.0, 0.0),
                            ecx_cbuild(0.0, 0.0)};

        char *errmsg;

        ecx_eis_z(p, w, z, 'R', 3, 3, &errstat, &errmsg);

        for(i=0; i<3;i++){
            printf("%f %f 0, creal(z[i]), cimag(z[i]));
        }
        printf("%d %s0, errstat, errmsg);
        return EXIT_SUCCESS;
    }
```

Example in Python:

```
    import numpy as np
    import pyecx
    import matplotlib.pyplot as plt

    R = 100
    C = 1e-6
    w = np.logspace(6, -3, 100)

    p = np.asarray([R, 0.0, 0.0])
    zr = np.asarray(pyecx.z("R", w, p))
    p = np.asarray([C, 0.0, 0.0])
    zc = np.asarray(pyecx.z("C", w, p))
    zrc = zr*zc / (zr+zc)
    print("finish")

    fig = plt.figure()
    ax = fig.add_subplot(111)

    ax.set_aspect("equal")
    ax.plot(zrc.real, zrc.imag, "g.", label="R/C")

    ax.invert_yaxis()
```

```
        plt.show()
```

SEE ALSO
       complex(7), **gsl**(3), **catanh**(3), **gnuplot**(1), **ecx_get_version**(3)

NAME
ecx_get_version - version getter for the library (C API)

LIBRARY
Electrochemistry library - (**-libecx**, **-lecx**)

SYNOPSIS
```
char* ecx_get_version(void);
```

DESCRIPTION
This function returns the version of the ecx library.

RETURN VALUE
char *

SEE ALSO
ecx(3), **ecx_get_version**(3)

NAME

ecxcli(1) - Command line for ecx

SYNOPSIS

ecxcli *SUBCOMMAND* [*OPTIONS ...*] *ARGS ...*

DESCRIPTION

ecxcli is command line interface for computing electro- chemical properties:

> **o EIS**   Electrochemical Impedance **Z=f**(w)
>
> **o Kinetics**
>> **j=f**(U)
>
> **o PEC**   **Iph=f**(hv, U)

It can also provide the molar masses, isotope compositions and nuclide compositions.

SUBCOMMANDS

o all   Get the whole periodic table.

o saw   Get the standard atomic weight.

Enter **ecxcli** *SUBCOMMAND* **--help** for detailed descriptions.

OPTIONS

o --abridged, -a
> Use the abridged value.

o --uncertainty, -u
> Use the uncertainty.

o --pprint
> Nice formatting.

o --mass, -z
> Get the mass number.

VALID FOR ALL SUBCOMMANDS

o --help
> Show help text and exit

o --verbose
> Display additional information when available.

o --version
> Show version information and exit.

NAME
get_version - version getter for the library

LIBRARY
Electrochemistry library - (**-libecx**, **-lecx**)

SYNOPSIS
```
function get_version() result(fptr)
```

```
extern char* ecx_get_version(void) (C API)
```

DESCRIPTION
This function returns the version of the ecx library.

RETURN VALUE
character(len=:), pointer :: *fptr*

char * (C *API*)

SEE ALSO
ecx(3)

NAME
      kTe - thermal voltage

LIBRARY
      Electrochemistry library - (**-libecx**, **-lecx**)

SYNOPSIS
```
pure elemental function kTe(T) result(r)
```

DESCRIPTION
      Compute the thermal voltage: kTe[V] = kB[eV] * (T[degC]+273.15)

      Parameters:

      o T   Temperature in degC

RETURN VALUE
      real(dp) :: *r*
            Thermal voltage in Volts.

EXAMPLE
      Calling:

```
value = kTe(T)
```

SEE ALSO
      ecx(3)

NAME

z - complex impedance

LIBRARY

Electrochemistry (**-libecx**, **-lecx**)

SYNOPSIS

```
subroutine z(p, w, zout, e, errstat, errmsg)
```

DESCRIPTION

Compute the complex impedance for the element *e*.

Parameters:

> o **real(dp), intent(in) :: p(:)**
> > Parameters defining the element *e*

> o **real(dp), intent(in) :: w(:)**
> > Angular frequencies in rad.s-1

> o **character(len=1), intent(in) ::** *e*
> > Electrochemical element: R, C, L, Q, O, T, G

> o **complex(dp), intent(out) :: zout(:)**
> > Complex impedance in Ohms.

> o **integer(int32), intent(out) ::** *errstat*
> > Error status

> o **character(len=:), intent(out), pointer ::** *errmsg*
> > Error message

Z_R(*w*) = R = **p**(1)

Z_C(*w*) = **-j**/(Cw) = **-j**/(**p**(1)**w*)

Z_L(*w*) = jLw = j****j****p(1)**w

RETURN VALUE

None

EXAMPLE

Calling:

```
call z(p, w, zout, e, errstat, errmsg)
```

SEE ALSO

ecx(3)