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.

Fortran API

- **o function get_version()result(fptr)**
  Get the version.
  - **o character(len=:), pointer :: fptr**
    Version of the library.
- **o pure elemental function kTe(T)result(r)**
  Compute the thermal voltage.
  - **o real(dp), intent(in) :: T**
    Temperature in °C.
  - **o real(dp) :: r**
    Thermal voltage in V.
- **o subroutine z(p, w, zout, e, errstat, errmsg)**
  Compute the complex impedance.
  - **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

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

o subroutine mm(p, w, zout, n)
Compute the measurement model.

- **real(dp), intent(in) :: p(:)**
  Parameters.
- **real(dp), intent(in) :: w(:)**
  Angular frequencies in rad.s-1
- **complex(dp), intent(out) :: zout(:)**
  Complex impedance in Ohms.
- **integer(int32), intent(in) :: n**
  Number of voigt elements.

o pure function nernst(E0, z, aox, vox, ared, vred, T)result(E)
Compute the Nernst electrochemical potential in V.

- **real(dp), intent(in) :: E0**
  Standard electrochemical potential in V.
- **integer(int32), intent(in) :: z**
  Number of exchanged electrons.
- **real(dp), intent(in) :: aox(:)**
  Activities of the oxidants.
- **real(dp), intent(in) :: vox(:)**
  Coefficients for the oxidants.
- **real(dp), intent(in) :: ared(:)**
  Activities of the reductants
- **real(dp), intent(in) :: vred(:)**
  Coefficients for the reductants.
- **real(dp), intent(in) :: T**
  Temperature in °C.
- **real(dp) :: E**
  Nernst potential in V.

o pure elemental function sbv(U, OCV, j0, aa, ac, za, zc, A, T)result(I)
Compute Butler Volmer equation without mass transport.

- **real(dp), intent(in) :: OCV**
  Open Circuit Voltage in V.
- **real(dp), intent(in) :: U**
  Electrochemical potential in V.
- **real(dp), intent(in) :: j0**
  Exchange current density in A.cm-2.
- **real(dp), intent(in) :: aa**
  Anodic transfer coefficient.
- **real(dp), intent(in) :: ac**
  Cathodic transfer coefficient.
- **real(dp), intent(in) :: za**
  Number of exchnaged electrons in the anodic branch.

o real(dp), intent(in) :: zc
Number of exchnaged electrons in the cathodic branch.

o real(dp), intent(in) :: A
Area in cm2.

o real(dp), intent(in) :: T
Temperature in °C.

o real(dp) :: I
Current in A.

o pure elemental function bv(U, OCV, j0, jdla, jdlc, aa, ac, za, zc, A, T)result(I)
Compute Butler Volmer equation with mass transport.

o real(dp), intent(in) :: OCV
Open Circuit Voltage in V.

o real(dp), intent(in) :: U
Electrochemical potential in V.

o real(dp), intent(in) :: j0
Exchange current density in A.cm-2.

o real(dp), intent(in) :: jdla
Anodic diffusion limiting current density in A.cm-2.

o real(dp), intent(in) :: jdlc
Cathodic diffusion limiting current density in A.cm-2.

o real(dp), intent(in) :: aa
Anodic transfer coefficient.

o real(dp), intent(in) :: ac
Cathodic transfer coefficient.

o real(dp), intent(in) :: za
Number of exchnaged electrons in the anodic branch.

o real(dp), intent(in) :: zc
Number of exchnaged electrons in the cathodic branch.

o real(dp), intent(in) :: A
Area in cm2.

o real(dp), intent(in) :: T
Temperature in °C.

o real(dp) :: I
Current in A.

C API

- char* **ecx_get_version**(void)
- const double ecx_core_PI
- const double ecx_core_T_K
- void **ecx_core_nm2eV**(double *lambda, double *E, size_t n)
- void **ecx_core_kTe**(double *T, double *kTE, size_t n)
- double **ecx_kinetics_nernst**(double E0, int z, double *aox, double *vox, size_t nox, double *ared, double *vred, size_t nred, double T)
- void **ecx_kinetics_sbv**(double *U, double OCV, double j0, double aa, double ac, double za, double zc, double A, double T, double *i, size_t n)

- void **ecx_kinetics_bv**(double *U, double OCV, double j0, double jdla, double jdlc, double aa, double ac, double za, double zc, double A, double T, double *i, size_t n)
- void **ecx_eis_z**(double *p, double *w, ecx_cdouble *z, char e, size_t k, size_t n, int *errstat, char *(*errmsg))

Python wrappers

- **z**(e:str, w:Union[np.ndarray,array.array,int,float], p:Union[np.ndarray,array.array])->np.ndarray

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)