

---

**ecx**

***Release 0.1.0dev0***

**Milan Skocic**

**Dec 25, 2025**



# CONTENTS

<b>1</b>	<b>Getting Started</b>	<b>3</b>
<b>2</b>	<b>Examples</b>	<b>5</b>
<b>3</b>	<b>APIs</b>	<b>7</b>
<b>4</b>	<b>Changelog</b>	<b>9</b>
	<b>Python Module Index</b>	<b>11</b>
	<b>Index</b>	<b>13</b>



# Modern Fortran

A purple rounded rectangle containing the white text "ECX".

ECX

Electrochemistry for Fortran.



## GETTING STARTED

### 1.1 Introduction

*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.

To use *ecx* within your *fpm* project, add the following lines to your file:

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

### 1.2 Dependencies

```
gcc>=10
gfortran>=10
fpm>=0.7
stdlib>=0.7
```

### 1.3 Installation

A Makefile is provided, which uses *fpm*, for building the library.

- On windows, *msys2* needs to be installed. Add the *msys2* binary (usually C:\msys64\usr\bin) to the path in order to be able to use *make*.
- On Darwin, the *gcc* toolchain needs to be installed.

```
chmod +x configure.sh
./configure.sh
make
make test
make install
make uninstall
```

You need a compiler that can compile the *stdlib*.

## 1.4 License

MIT

## EXAMPLES

## 2.1 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
```

## 2.2 C

```
#include <stdio.h>
#include <stdlib.h>
#include "ecx.h"

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),
```

(continues on next page)

(continued from previous page)

```

        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 \n", creal(z[i]), cimag(z[i]));
    }
    printf("%d %s\n", errstat, errmsg);
    return EXIT_SUCCESS;
}

```

## 2.3 Python

```

from pyecx import eis
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(eis.z("R", w, p))
p = np.asarray([C, 0.0, 0.0])
zc = np.asarray(eis.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()

```

## 3.1 Fortran

<https://milanskocic.github.io/ecx/ford/index.html>

## 3.2 C

```
#ifndef ECX_H
#define ECX_H
#include <complex.h>
#if _MSC_VER
    #define ADD_IMPORT __declspec(dllimport)
    typedef _Dcomplex ecx_cdoube;
    #define ecx_cbuild(real, imag) (_Cbuild(real, imag))
#else
    #define ADD_IMPORT
    typedef double _Complex ecx_cdoube;
    #define ecx_cbuild(real, imag) (real+I*imag)
#endif

extern char* ecx_get_version(void);

ADD_IMPORT extern const double ecx_core_PI;
ADD_IMPORT extern const double ecx_core_T_K;
void ecx_core_nm2eV(double *lambda, double *E, size_t n);
void ecx_core_kTe(double *U, double *kTE, size_t n);

extern double ecx_kinetics_nernst(double E0, int z,
                                double *aox, double *vox, size_t nox,
                                double *ared, double *vred, size_t nred,
                                double T);

extern 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);

extern void ecx_kinetics_bv(double *U, double OCV, double j0, double jdla, double jd1c,
                           double aa, double ac, double za, double zc,
```

(continues on next page)

(continued from previous page)

```
double A, double T, double *i, size_t n);  
  
extern void ecx_eis_z(double *p, double *w, ecx_cdouble *z,  
    char e, size_t k, size_t n,  
    int *errstat, char *(*errmsg));  
  
#endif
```

## 3.3 Python

Python wrapper of the (Modern Fortran) ecx library.

## CHANGELOG

### 4.1 Version 0.1.0-dev

- Implementation of eis + C API
- Python wrappers for eis.



## PYTHON MODULE INDEX

### p

[pyecx](#), 8



## INDEX

### M

module

pyecx, 8

### P

pyecx

module, 8