**Libraries**

1. Adept

Operation: Adept is a combined automatic differentiation and array software library for the C++ programming language. The automatic differentiation capability facilitates the development of applications involving mathematical optimization.

Price: Free

Written in: C++

Example:

#include *<iostream>*

#include *<adept\_arrays.h>*

int main(int argc, **const** char\*\* argv) {

**using** **namespace** adept;

Stack stack; *// Object to store differential statements*

aVector x(3); *// Independent variables: active vector with 3 elements*

x << 1.0, 2.0, 3.0; *// Fill vector x*

stack.new\_recording(); *// Clear any existing differential statements*

adouble J = cbrt(sum(abs(x\*x\*x))); *// Compute dependent variable: 3-norm in this case*

J.set\_gradient(1.0); *// Seed the dependent variable*

stack.reverse(); *// Reverse-mode differentiation*

std::cout << "dJ/dx = "

<< x.get\_gradient() << "**\n**"; *// Print the vector of partial derivatives dJ/dx*

**return** 0;

}

1. Armadillo

Operation: Armadillo is a linear algebra software library for the C++ programming language. It aims to provide efficient and streamlined base calculations, while at the same time having a straightforward and easy-to-use interface. Its intended target users are scientists and engineers.

Price: Free

Written in: C++

Example:

*// Compile with:*

*// $ g++ -std=c++11 main.cpp -o file\_name -O2 -larmadillo*

#include *<iostream>*

#include *<armadillo>*

#include *<math.h>*

int main()

{

*// ^*

*// Position of a particle // |*

arma::vec Pos = {{0}, *// | (0,1)*

{1}}; *// +---x-->*

*// Rotation matrix*

double phi = -3.1416/2;

arma::mat RotM = {{+cos(phi), -sin(phi)},

{+sin(phi), +cos(phi)}};

Pos.print("Current position of the particle:");

std::cout << "Rotating the point " << phi\*180/3.1416 << " deg" << std::endl;

Pos = RotM\*Pos;

Pos.print("New position of the particle:"); *// ^*

*// x (1,0)*

*// |*

*// +------>*

**return** 0;

}

1. Cairo

Operation: Cairo is an open source programming library that provides a vector graphics-based, device-independent API for software developers. It provides primitives for two-dimensional drawing across a number of different back ends. Cairo uses hardware acceleration when available.

Price: Free

Written in: C

Example:

1 #include *<cairo-svg.h>*

2 #include *<stdio.h>*

3

4 int main(int argc, char \*\*argv) {

5 cairo\_t \*cr;

6 cairo\_surface\_t \*surface;

7 cairo\_pattern\_t \*pattern;

8 int x,y;

9

10 surface =

11 (cairo\_surface\_t \*)cairo\_svg\_surface\_create("Cairo\_example.svg", 100.0, 100.0);

12 cr = cairo\_create(surface);

13

14 */\* Draw the squares in the background \*/*

15 **for** (x=0; x<10; x++)

16 **for** (y=0; y<10; y++)

17 cairo\_rectangle(cr, x\*10.0, y\*10.0, 5, 5);

18

19 pattern = cairo\_pattern\_create\_radial(50, 50, 5, 50, 50, 50);

20 cairo\_pattern\_add\_color\_stop\_rgb(pattern, 0, 0.75, 0.15, 0.99);

21 cairo\_pattern\_add\_color\_stop\_rgb(pattern, 0.9, 1, 1, 1);

22

23 cairo\_set\_source(cr, pattern);

24 cairo\_fill(cr);

25

26 */\* Writing in the foreground \*/*

27 cairo\_set\_font\_size (cr, 15);

28 cairo\_select\_font\_face (cr, "Georgia",

29 CAIRO\_FONT\_SLANT\_NORMAL, CAIRO\_FONT\_WEIGHT\_BOLD);

30 cairo\_set\_source\_rgb (cr, 0, 0, 0);

31

32 cairo\_move\_to(cr, 10, 25);

33 cairo\_show\_text(cr, "Hallo");

34

35 cairo\_move\_to(cr, 10, 75);

36 cairo\_show\_text(cr, "Wikipedia!");

37

38 cairo\_destroy (cr);

39 cairo\_surface\_destroy (surface);

40 **return** 0;

41 }

1. CAPD Library

Operation: The CAPD library (Computer Assisted Proofs in Dynamics) is a software library that aims to provide a set of flexible C++ modules designed for rigorous numerics in Dynamical Systems and homology computation. It has been used in a research of chaotic dynamics, bifurcations, heteroclinic/homoclinic solutions and periodic orbits.

Price: Free

Written in: C++

1. CherryPy

Operation: CherryPy is an object-oriented web application framework using the Python programming language. It is designed for rapid development of web applications by wrapping the HTTP protocol but stays at a low level and does not offer much more than what is defined in RFC 7231.

Price: Free

Written in: Python

Example:

**import** **cherrypy**

**class** **HelloWorld**(object):

**def** index(self):

**return** "Hello World!"

index.exposed = True

cherrypy.quickstart(HelloWorld())

1. CEGUI

Operation: Crazy Eddie's GUI (CEGUI) is a graphical user interface (GUI) library for the programming language C++. It was designed for the needs of video games, but is usable for non-game tasks, such as applications and tools.

Price: Free

Written in: C++

Example:

CEGUI::Window \*gJumpBtnWindow = NULL;

void CreateJumpButton()

{

gJumpBtnWindow = CEGUI::WindowManager::getSingleton().createWindow("TaharezLook/Button","JumpPushButton"); // Create Window

gJumpBtnWindow->setPosition(CEGUI::UVector2(CEGUI::UDim(0.75,0),CEGUI::UDim(0.50,0)));

gJumpBtnWindow->setSize(CEGUI::USize(CEGUI::UDim(0,50),CEGUI::UDim(0,50)));

gJumpBtnWindow->setText("Jump!");

CEGUI::System::getSingleton().getDefaultGUIContext().getRootWindow()->addChild(gJumpBtnWindow);

}

1. Crypto++

Operation: Crypto++ (also known as CryptoPP, libcrypto++, and libcryptopp) is a free and open-source C++ class library of cryptographic algorithms and schemes written by Wei Dai. Crypto++ has been widely used in academia, student projects, open source and non-commercial projects, as well as businesses.

Price: Free

Written in: C++

Example:

#include "cryptopp\_wrapper.h"

int main()

{

...

cout<<"Space Wrapper TEST!"<<endl<<endl;

SpaceCrypto::CryptBlowFish hello;

hello.setPlainString("HelloWorld!");

hello.setKey("mySecUreKey!!");

std::string crypt;

crypt = hello.Encrypt();

cout<<"Plain Text: HelloWorld!"<<endl;

cout<<"Crypt Text: "<<crypt<<endl;

hello.reset();

hello.setEncString(crypt);

hello.setInputMode(SpaceCrypto::HEX);

hello.setKey("mySecUreKey!!");

hello.setOutputMode(SpaceCrypto::NORMAL);

cout<<"Decrypted Text: "<<hello.Decrypt()<<endl;

}

1. deal.II

Operation: deal.II is a free, open-source library to solve partial differential equations using the finite element method.

Price: Free

Written in: C++

1. Dietlibc

Operation: dietlibc is a C standard library released under the GNU General Public License Version 2, but there are also commercial licences available. It was developed with the help of about 100 volunteers by Felix von Leitner with the goal to compile and link programs to the smallest possible size.

Price: Free

Written in: C

1. DirectFB

Operation: DirectFB (Direct Frame Buffer) is a software library with a small memory footprint that provides graphics acceleration, input device handling and abstraction layer, and integrated windowing system with support for translucent windows and multiple display layers on top of the Linux framebuffer without requiring any kernel modifications.

Price: Free

Written in: C

Example:

#include <stdio.h>

#include <unistd.h>

#include <directfb/directfb.h>

static IDirectFB \*dfb = NULL;

static IDirectFBSurface \*primary = NULL;

static int screen\_width = 0;

static int screen\_height = 0;

#define DFBCHECK(x...) \

{ \

DFBResult err = x; \

\

if (err != DFB\_OK) \

{ \

fprintf( stderr, "%s <%d>:\n\t", \_\_FILE\_\_, \_\_LINE\_\_ ); \

DirectFBErrorFatal( #x, err ); \

} \

}

int main(int argc, char \*\*argv) {

DFBSurfaceDescription dsc;

DFBCHECK(DirectFBInit (&argc, &argv));

DFBCHECK(DirectFBCreate (&dfb));

DFBCHECK(dfb->SetCooperativeLevel (dfb, DFSCL\_FULLSCREEN));

dsc.flags = DSDESC\_CAPS;

dsc.caps = DSCAPS\_PRIMARY | DSCAPS\_FLIPPING;

DFBCHECK(dfb->CreateSurface( dfb, &dsc, &primary ));

DFBCHECK(primary->GetSize (primary, &screen\_width, &screen\_height));

DFBCHECK(primary->FillRectangle (primary, 0, 0, screen\_width, screen\_height));

DFBCHECK(primary->SetColor (primary, 0x80, 0x80, 0xff, 0xff));

DFBCHECK(primary->DrawLine (primary, 0, screen\_height / 2, screen\_width - 1, screen\_height / 2));

DFBCHECK(primary->Flip (primary, NULL, 0));

sleep(5);

primary->Release(primary);

dfb->Release(dfb);

return 23;

}

1. Eigen

Operation: Eigen is a high-level C++ library of template headers for linear algebra, matrix and vector operations, geometrical transformations, numerical solvers and related algorithms.

Price: Free

Written in: C++

Example:

#include <iostream>

#include <Eigen/Dense>

using [Eigen::MatrixXd](https://eigen.tuxfamily.org/dox/group__matrixtypedefs.html#ga99b41a69f0bf64eadb63a97f357ab412);

int main()

{

[MatrixXd](https://eigen.tuxfamily.org/dox/group__matrixtypedefs.html#ga99b41a69f0bf64eadb63a97f357ab412) m(2,2);

m(0,0) = 3;

m(1,0) = 2.5;

m(0,1) = -1;

m(1,1) = m(1,0) + m(0,1);

std::cout << m << std::endl;

}

1. Ensmallen

Operation: ensmallen is a library of mathematical optimization algorithms for the C++ programming language. It provides a simple set of abstractions for writing an objective function to optimize. It also provides a large set of optimizers that can be used for virtually any mathematical optimization task.

Price: Free

Written in: C++

Example:

**class** **DifferentiableFunctionType**

{

**public:**

*// Given parameters x, return the value of f(x).*

**double** Evaluate(**const** arma**::**mat**&** x);

*// Given parameters x and a matrix g, store f'(x) in the provided matrix g.*

*// g should have the same size (rows, columns) as x.*

**void** Gradient(**const** arma**::**mat**&** x, arma**::**mat**&** gradient);

*// OPTIONAL: this may be implemented in addition to---or instead*

*// of---Evaluate() and Gradient(). If this is the only function implemented,*

*// implementations of Evaluate() and Gradient() will be automatically*

*// generated using template metaprogramming. Often, implementing*

*// EvaluateWithGradient() can result in more efficient optimizations.*

*//*

*// Given parameters x and a matrix g, return the value of f(x) and store*

*// f'(x) in the provided matrix g. g should have the same size (rows,*

*// columns) as x.*

**double** EvaluateWithGradient(**const** arma**::**mat**&** x, arma**::**mat**&** g);

};

1. Eulim

Operation: Eulim is a Chemistry library written in Ruby under the MIT license. Eulim is a Ruby gem for Chemistry, which supports the calculation of molecular mass of compound, balancing chemical equations, efficient handling of states of chemical species and many more things.

Price: Free

Written in: Ruby

Example:

$ irb

**irb(main):001:0>** require 'eulim'

**irb(main):002:0>** Eulim::Chemistry::Reaction.new(equation: 'KMnO4 + HCl >> KCl + MnCl2 + H2O + Cl2').balanced\_eqn

=> "2KMnO4 + 16HCl >> 2KCl + 2MnCl2 + 8H2O + 5Cl2"

**irb(main):003:0>** Eulim::Chemistry::Compound.new("CaCO3")

=> #<Eulim::Chemistry::Compound:0x00000002a65340 @formula="CaCO3", @constituents={"Ca"=>{:element=>#<Eulim::Chemistry::Element:0x00000002c805a8 @name="Calcium", @symbol="Ca", @atomic\_number=20, @atomic\_mass=#<Unitwise::Measurement value=40.078 unit=u>>, :atom\_count=>1}, "C"=>{:element=>#<Eulim::Chemistry::Element:0x00000002c8f6e8 @name="Carbon", @symbol="C", @atomic\_number=6, @atomic\_mass=#<Unitwise::Measurement value=12.0107 unit=u>>, :atom\_count=>1}, "O"=>{:element=>#<Eulim::Chemistry::Element:0x00000002c8dc30 @name="Oxygen", @symbol="O", @atomic\_number=8, @atomic\_mass=#<Unitwise::Measurement value=15.9996 unit=u>>, :atom\_count=>3}}, @molecular\_mass=#<Unitwise::Measurement value=100.0875 unit=u>>

**irb(main):004:0>** Eulim::Chemistry::Reaction.new(equation: '2Na(s) + 2HCl(aq) >> 2NaCl(aq) + H2(g)')

=> #<Eulim::Chemistry::Reaction:0x00000002ce22f8 @equation="2Na(s) + 2HCl(aq) >> 2NaCl(aq) + H2(g)", @species={:reactants=>{"Na"=>{:compound=>#<Eulim::Chemistry::Compound:0x00000002ce1d80 @formula="Na", @constituents={"Na"=>{:element=>#<Eulim::Chemistry::Element:0x00000002c88e10 @name="Sodium", @symbol="Na", @atomic\_number=11, @atomic\_mass=#<Unitwise::Measurement value=22.9897 unit=u>>, :atom\_count=>1}}, @molecular\_mass=#<Unitwise::Measurement value=22.9897 unit=u>>, :stoichiometry=>2, :state=>"solid"}, "HCl"=>{:compound=>#<Eulim::Chemistry::Compound:0x00000002cabdc0 @formula="HCl", @constituents={"H"=>{:element=>#<Eulim::Chemistry::Element:0x000000025e5ab8 @name="Hydrogen", @symbol="H", @atomic\_number=1, @atomic\_mass=#<Unitwise::Measurement value=1.0079 unit=u>>, :atom\_count=>1}, "Cl"=>{:element=>#<Eulim::Chemistry::Element:0x00000002c82c90 @name="Chlorine", @symbol="Cl", @atomic\_number=17, @atomic\_mass=#<Unitwise::Measurement value=35.453 unit=u>>, :atom\_count=>1}}, @molecular\_mass=#<Unitwise::Measurement value=36.4609 unit=u>>, :stoichiometry=>2, :state=>"aqueous"}}, :products=>{"NaCl"=>{:compound=>#<Eulim::Chemistry::Compound:0x00000002c8cda8 @formula="NaCl", @constituents={"Na"=>{:element=>#<Eulim::Chemistry::Element:0x00000002c88e10 @name="Sodium", @symbol="Na", @atomic\_number=11, @atomic\_mass=#<Unitwise::Measurement value=22.9897 unit=u>>, :atom\_count=>1}, "Cl"=>{:element=>#<Eulim::Chemistry::Element:0x00000002c82c90 @name="Chlorine", @symbol="Cl", @atomic\_number=17, @atomic\_mass=#<Unitwise::Measurement value=35.453 unit=u>>, :atom\_count=>1}}, @molecular\_mass=#<Unitwise::Measurement value=58.4427 unit=u>>, :stoichiometry=>2, :state=>"aqueous"},"H2"=>{:compound=>#<Eulim::Chemistry::Compound:0x00000002c6f938 @formula="H2", @constituents={"H"=>{:element=>#<Eulim::Chemistry::Element:0x000000025e5ab8 @name="Hydrogen", @symbol="H", @atomic\_number=1, @atomic\_mass=#<Unitwise::Measurement value=1.0079 unit=u>>, :atom\_count=>2}}, @molecular\_mass=#<Unitwise::Measurement value=2.0158 unit=u>>, :stoichiometry=>1, :state=>"gaseous"}}}, @is\_valid=true, @is\_balanced=true>

1. FLTK

Operation: Fast Light Toolkit (FLTK, pronounced fulltick)[3] is a cross-platform widget (graphical control element) library for graphical user interfaces (GUIs), developed by Bill Spitzak and others. Made to accommodate 3D graphics programming, it has an interface to OpenGL, but it is also suitable for general GUI programming.

Price: Free

Written in: C++

Example:

// hello.cxx (example1)

#include <fltk/Window.h>

#include <fltk/Widget.h>

#include <fltk/run.h>

using namespace fltk;

int main(int argc, char \*\*argv) {

[Window](https://www.fltk.org/doc-2.0/html/classfltk_1_1Window.html) \*window = new [Window](https://www.fltk.org/doc-2.0/html/classfltk_1_1Window.html)(300, 180);

window->[begin](https://www.fltk.org/doc-2.0/html/classfltk_1_1Group.html#acc8de5563f6f88be0b1a69549ab58f00)();

[Widget](https://www.fltk.org/doc-2.0/html/classfltk_1_1Widget.html) \*box = new [Widget](https://www.fltk.org/doc-2.0/html/classfltk_1_1Widget.html)(20, 40, 260, 100, "Hello, World!");

box->box([UP\_BOX](https://www.fltk.org/doc-2.0/html/namespacefltk.html#a888fb756f3b30353184defdd7712c753));

box->labelfont(HELVETICA\_BOLD\_ITALIC);

box->labelsize(36);

box->labeltype(SHADOW\_LABEL);

window->[end](https://www.fltk.org/doc-2.0/html/classfltk_1_1Group.html#a7143e366d94862af722276ea98182e37)();

window->[show](https://www.fltk.org/doc-2.0/html/classfltk_1_1Window.html#a8f986e19a11c4c97ed8e6ad3d0e648b7)(argc, argv);

return [run](https://www.fltk.org/doc-2.0/html/namespacefltk.html#a66fd84d46d1bfef4d3ceaf1faecfaa0c)();

}

1. Nana

Operation: Nana is a cross-platform C++ library for creating graphical user interfaces. It uses a platform-independent API and currently supports Windows and Linux(X11) platforms.

Price: Free

Written in: C++

Example:

#include <nana/gui.hpp>  
#include <nana/gui/widgets/button.hpp>  
  
int main()  
{  
    using namespace nana;  
    form fm;  
    fm.caption(L"Hello, World!");  
    button btn(fm, rectangle{20, 20, 150, 30});  
    btn.caption(L"Quit");  
    btn.events().click(API::exit);  
    fm.show();  
    exec();  
}

1. GNU C

Operation: The GNU C Library, commonly known as glibc, is the GNU Project's implementation of the C standard library. Despite its name, it now also directly supports C++ (and, indirectly, other programming languages). It was started in the early 1990s by the Free Software Foundation (FSF) for their GNU operating system.

Price: Free

Written in: C

Example: Here is an example program showing the use of the system database inquiry functions. The program prints some information about the user running the program.

#include <grp.h>

#include <pwd.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdlib.h>

int

main (void)

{

uid\_t me;

struct passwd \*my\_passwd;

struct group \*my\_group;

char \*\*members;

/\* Get information about the user ID. \*/

me = getuid ();

my\_passwd = getpwuid (me);

if (!my\_passwd)

{

printf ("Couldn't find out about user %d.\n", (int) me);

exit (EXIT\_FAILURE);

}

/\* Print the information. \*/

printf ("I am %s.\n", my\_passwd->pw\_gecos);

printf ("My login name is %s.\n", my\_passwd->pw\_name);

printf ("My uid is %d.\n", (int) (my\_passwd->pw\_uid));

printf ("My home directory is %s.\n", my\_passwd->pw\_dir);

printf ("My default shell is %s.\n", my\_passwd->pw\_shell);

/\* Get information about the default group ID. \*/

my\_group = getgrgid (my\_passwd->pw\_gid);

if (!my\_group)

{

printf ("Couldn't find out about group %d.\n",

(int) my\_passwd->pw\_gid);

exit (EXIT\_FAILURE);

}

/\* Print the information. \*/

printf ("My default group is %s (%d).\n",

my\_group->gr\_name, (int) (my\_passwd->pw\_gid));

printf ("The members of this group are:\n");

members = my\_group->gr\_mem;

while (\*members)

{

printf (" %s\n", \*(members));

members++;

}

return EXIT\_SUCCESS;

}

1. Libuv

Operation: Libuv is a multi-platform C library that provides support for asynchronous I/O based on event loops. It supports epoll(4), kqueue(2), Windows IOCP, and Solaris event ports. It is primarily designed for use in Node.js but it is also used by other software projects.

Price: Free

Written in: C

Example:

#include *<stdio.h>*

#include *<stdlib.h>*

#include *<uv.h>*

int main() {

uv\_loop\_t \*loop = malloc(**sizeof**(uv\_loop\_t));

uv\_loop\_init(loop);

printf("Now quitting.**\n**");

uv\_run(loop, UV\_RUN\_DEFAULT);

uv\_loop\_close(loop);

free(loop);

**return** 0;

}

1. LWJGL

Operation: The Lightweight Java Game Library (LWJGL) is an open-source Java software library for video game developers. It exposes high performance cross-platform libraries commonly used in developing video games and multimedia titles, such as Vulkan, OpenGL, OpenAL and OpenCL. It further provides access to controllers such as gamepads, steering wheels and joysticks in a platform-neutral way.

Price: Free

Written in: Java, C, Kotlin

Example:

/\*\*

\* Handles mouse input.

\*/

public void handleMouseInput() throws IOException

{

super.handleMouseInput();

int i = Mouse.getEventDWheel();

if (i != 0 && this.needsScrollBars())

{

int j = (((GuiContainerCreative.ContainerCreative)this.inventorySlots).itemList.size() + 9 - 1) / 9 - 5;

if (i > 0)

{

i = 1;

}

if (i < 0)

{

i = -1;

}

this.currentScroll = (float)((double)this.currentScroll - (double)i / (double)j);

this.currentScroll = MathHelper.clamp\_float(this.currentScroll, 0.0F, 1.0F);

((GuiContainerCreative.ContainerCreative)this.inventorySlots).scrollTo(this.currentScroll);

}

}

1. Mlpack

Operation: Mlpack is a machine learning software library for C++, built on top of the Armadillo library. Mlpack has an emphasis on scalability, speed, and ease-of-use. Its aim is to make machine learning possible for novice users by means of a simple, consistent API, while simultaneously exploiting C++ language features to provide maximum performance and maximum flexibility for expert users.

Price: Free

Written in: C++

Example: A simple program to compute the covariance of a data matrix ("data.csv"), assuming that the data is already centered, and save it to file.

// Includes all relevant components of mlpack.

#include <mlpack/core.hpp>

// Convenience.

using namespace mlpack;

int main()

{

// First, load the data.

arma::mat data;

// Use data::Load() which transposes the matrix.

data::Load("data.csv", data, true);

// Now compute the covariance. We assume that the data is already centered.

// Remember, because the matrix is column-major, the covariance operation is

// transposed.

arma::mat cov = data \* trans(data) / data.n\_cols;

// Save the output.

data::Save("cov.csv", cov, true);

}

1. MOEA Framework

Operation: The MOEA Framework is an open-source evolutionary computation library for Java that specializes in multi-objective optimization. It supports a variety of multi-objective evolutionary algorithms (MOEAs), including genetic algorithms, genetic programming, grammatical evolution, differential evolution, and particle swarm optimization. As a result, it has been used to conduct numerous comparative studies to assess the efficiency, reliability, and controllability of state-of-the-art MOEAs.

Price: Free

Written in: Java

Example: Statistical comparison of algorithms.

String problem = "UF1";

String[] algorithms = { "NSGAII", "GDE3", "eMOEA" };

//setup the experiment

Executor executor = new Executor()

    .withProblem(problem)

    .withMaxEvaluations(10000);

Analyzer analyzer = new Analyzer()

    .withProblem(problem)

    .includeHypervolume()

    .showStatisticalSignificance();

//run each algorithm for 50 seeds

for (String algorithm : algorithms) {

    analyzer.addAll(algorithm,

        executor.withAlgorithm(algorithm).runSeeds(50));

}

//print the results

analyzer.printAnalysis();