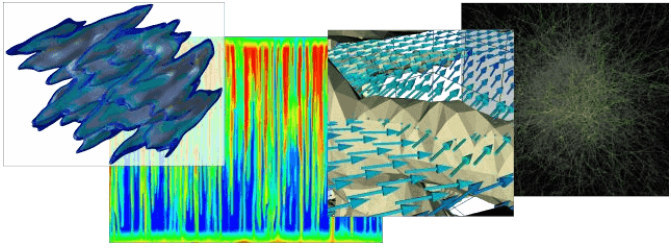
# Introducción al método de elementos finitos

Parte I: `bash`, `g++`, `cmake`, `gnuplot`, `gmsh`, `paraview`



Practique los ejemplos en Gitpod

Open in Gitpod

Disponible en

¡Únete al grupo en Telegram!

# Motivation

- Start with an introduction to the finite element method (FEM) for solving Poisson's equation with piecewise linear "$P_1$" finite elements

# Objetivos de esta introdución

A number of advanced.

- Correr los imágenes prediseñadas con el programa `docker` en sistemas tipo Unix.
- Crear una malla con `gmsh` a través de la API (C/C++, Python, Julia).
- Realizar los ejemplos resueltos del DUNE BOOK.

# Conociendo el sistema operativo Arch Linux

Es una distribución GNU/Linux de propósito general, desarrollada[1] independientemente para procesadores x86-64, que se adhiere a los principios de simplicidad, modernidad, pragmatismo, centrado en usuarios y versatilidad.

Proporciona las últimas versiones estables de la mayoría del software siguiendo el modelo de lanzamiento continuo, no existen versiones como en Ubuntu 20.04, 21.10, 22.04, etc.

Arch está respaldado por `pacman`, un gestor de paquetes ligero, sencillo y rápido, que permite actualizar todo el sistema con una orden. Los scripts PKGBUILD aportados por la comunidad para la elaboración desde las fuentes, como los módulos de DUNE, se encuentran en el *Arch User Repository*.

**Cuadro:** Comparación de la línea de comando del gestión de software (fuente: `wiki.archlinux.org`)

| Acción | Arch | Red Hat/Fedora | Debian/Ubuntu | SLES/openSUSE | Gentoo |
|---|---|---|---|---|---|
| Instala paquetes | `pacman -S` | `dnf install` | `apt install` | `zypper install` | `emerge -a` |
| Elimina paquetes | `pacman -Rs` | `dnf remove` | `apt remove` | `zypper remove` | `emerge -C` |
| Busca paquetes | `pacman -Ss` | `dnf search` | `apt search` | `zypper search` | `emerge -S` |
| Actualiza paquetes | `pacman -Syu` | `dnf upgrade` | `apt update && apt upgrade` | `zypper update` | `emerge -u world` |

[1] El líder del proyecto es `Anthraxx`, desarrollador alemán del kernel `linux-hardened`.

# Filosofía de Arch

**Simplicidad** Es minimalista. Se usa herramienta pequeñas que sigue la filosofía de UNIX, de modo que tengas una base muy pequeña que deje configurar la máquina de manera más cómoda para el usuario.

**Modernidad** Mantener la paquetería lo más actualizada posible sin sacrificar la estabilidad. Cuenta con los compiladores más actuales de `gcc`, `lualatex`, `go`, etc.

**Pragmatismo** La gran cantidad de paquetes y scripts de compilación en los diversos repositorios de Arch Linux ofrecen software gratuito y de código abierto para quienes lo prefieren, así como paquetes de software propietario para quienes adoptan la funcionalidad por encima de la ideología.

**Centrado a las usuarias y usuarios** La distribución está destinada a satisfacer las necesidades de quienes contribuyen a ella, en lugar de intentar atraer a tantos usuarios como sea posible. Está dirigido al usuario competente de GNU/Linux, o cualquier persona con una actitud de hágalo usted mismo que esté dispuesto a leer la documentación y resolver sus propios problemas. Se anima a todos los usuarios a participar y contribuir a la distribución. Informar y ayudar a corregir errores es muy valioso y los parches que mejoran los paquetes o los proyectos principales son muy apreciados: los desarrolladores de Arch son voluntarios y los contribuyentes activos a menudo se convertirán en parte de ese equipo.

**Versatilidad**

En esta ocasión hemos elegido Arch Linux como ambiente de trabajo porque tiene disponible una gran variedad de módulos de DUNE. Es recomendable habilitar el repositorio `arch4edu`[2].

[2]Administrado por Jingbei Li de la Universidad de Tsinghua.

# Usando el emulador de terminal

```
gitpod ~/dune-basics $ cd
```

# Classes

Listado 1: Programa `hello-linux.cc`.

```
// Tomado de https://stackoverflow.com/a/66161001

#include <sys/utsname.h>
#include <iostream>

// un pequeño ayudante para mostrar el contenido de una estructura utsname:
std::ostream &operator<<(std::ostream &os, const utsname &u)
{
  return os << "sysname : " << u.sysname << '\n'
            << "nodename: " << u.nodename << '\n'
            << "release : " << u.release << '\n'
            << "version : " << u.version << '\n'
            << "machine : " << u.machine << '\n';
}

int main()
{
  utsname result; // declarar la variable para contener el resultado

  uname(&result); // llamar a la función uname() para completar la estructura

  std::cout << result; // mostrar el resultado usando la función ayudante
}
```

# Classes

Listado 2: Programa `dune-basics.cc`.

```
#ifdef HAVE_CONFIG_H
#include "config.h"
#endif
#include <iostream>
#include <dune/common/parallel/mpihelper.hh> // An initializer of MPI
#include <dune/common/exceptions.hh>         // We use exceptions

int main(int argc, char **argv)
{
  try
  {
    // Maybe initialize MPI
    Dune::MPIHelper &helper = Dune::MPIHelper::instance(argc, argv);
    std::cout << "Hello World! This is dune-basics." << std::endl;
    if (Dune::MPIHelper::isFake)
      std::cout << "This is a sequential program." << std::endl;
    else
      std::cout << "I am rank " << helper.rank() << " of " << helper.size()
                << " processes!" << std::endl;
    return 0;
  }
  catch (Dune::Exception &e)
  {
    std::cerr << "Dune reported error: " << e << std::endl;
  }
  catch (...)
  {
    std::cerr << "Unknown exception thrown!" << std::endl;
  }
}
```

# Classes

Listado 3: Programa `dune-basics.cc`.

```
#include <iostream>
#include <dune/common/math.hh>

int main(int argc, char **argv)
{
  std::cout << Dune::StandardMathematicalConstants<double>::pi()
            << "\n"
            << Dune::StandardMathematicalConstants<double>::e()
            << "\n"
            << Dune::factorial<int>(3)
            << "\n"
            << Dune::factorial(5)
            << "\n"
            << Dune::power<int, int>(2, 3)
            << "\n"
            << Dune::binomial(100, 1)
            << "\n"
            << Dune::conjugateComplex(std::complex<double>(0, 1))
            << "\n"
            << Dune::sign<double>(24)
            << "\n"
            << Dune::isFinite(std::complex<double>(0, 1))
            << "\n"
            << Dune::isInf(std::complex<double>(0, 1))
            << "\n"
            << Dune::isNaN(std::complex<double>(0, 1))
            << "\n"
            << Dune::isUnordered(10, 20)
            << "\n";

  return 0;
}
```

# El comando `duneproject`

Es un asistente en el lenguaje `bash` que se encuentra en `/usr/bin/duneproject`…

# Referencias


- Libros

  Oliver Sander. *DUNE — The Distributed and Unified Numerics Environment*. First. Lecture Notes in Computational Science and Engineering 140. Springer International Publishing, 2020. ISBN: 978-3-030-59701-6. DOI: 10.1007/978-3-319-03038-8.

- Artículos

  Peter Bastian y col. "The Dune framework: Basic concepts and recent developments". En: *Computers & Mathematics with Applications* 81.1 (1 de ene. de 2021). Development and Application of Open-source Software for Problems with Numerical PDEs, págs. 75-112. ISSN: 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2020.06.007. URL: https://www.sciencedirect.com/science/article/pii/S089812212030256X.

- Sitios web

  Santiago Torres Arias, Jesús Castro y Andrea Gómez. *Taller de contribución a Arch Linux – Cumbre de Contribuidores de Open Source Software (CCOSS)*. 22 de oct. de 2020. URL: https://sg.com.mx/buzz/ponencias/ccoss-2020/taller-de-contribucion-arch-linux (visitado 10-03-2021).

  The Open Group. *The Evolution of the Unix Time-sharing System*. 15 de ago. de 2021. URL: https://unix.org/what_is_unix/history_timeline.html (visitado 30-05-2021).