Computational Methods for Oil Recovery

PASI: Scientific Computing in the Americas

The Challenge of Massive Parallelism

Luis M. de la Cruz Salas

Instituto de Geofísica
Universidad Nacional Autónoma de México

January 2011
Valparaíso, Chile

Mathematical and Computational Group (GMMC)

Natural Resources Dept.

- Dr. Ismael Herrera Revilla
- Dra. Graciela Herrera Zamarrón
- Dr. Luis M. de la Cruz Salas
- Dr. Guillermo Hernández García
- Dr. Norberto Vera Guzmán

Students

- Esther Leyva
- Antonio Carrillo
- Ivan Contreras
- Alberto Rosas
- Emilio Zavala
- Ricardo Flores

Computational Group

- Daniel A. Cervantes Cabrera
- Alejandro Salazar Sánchez
- Daniel Monsivais Velázquez
- Renato Leriche Vázquez
- Héctor U. Barrón García
- Ismael Herrera Zamarrón
- Eduardo Murrieta León

http://www.mmc.geofisica.unam.mx

Table of contents

Table of contents

Oil Reservoir Projects

- Funded by PEMEX

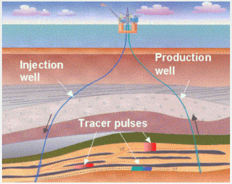
Oil Reservoir Projects

- Funded by PEMEX
- Collaboration with IMP and CIMAT.

Oil Reservoir Projects

- Funded by PEMEX
- Collaboration with IMP and CIMAT.

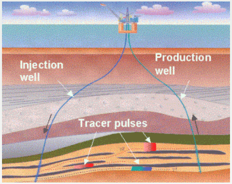
1. WAG injection.



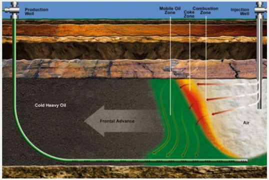
Oil Reservoir Projects

- Funded by PEMEX
- Collaboration with IMP and CIMAT.

1. WAG injection.



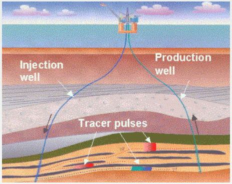
2. AIR injection.



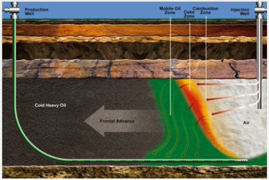
Oil Reservoir Projects

- Funded by PEMEX
- Collaboration with IMP and CIMAT.

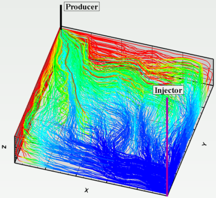
1. WAG injection.



2. AIR injection.



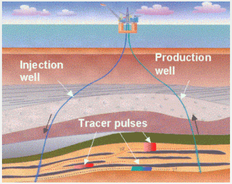
3. SLS method.



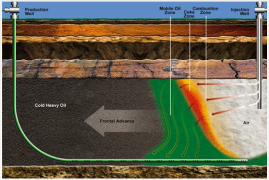
Oil Reservoir Projects

- Funded by PEMEX
- Collaboration with IMP and CIMAT.

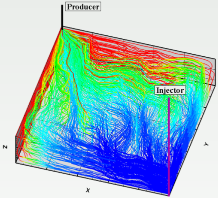
1. WAG injection.



2. AIR injection.



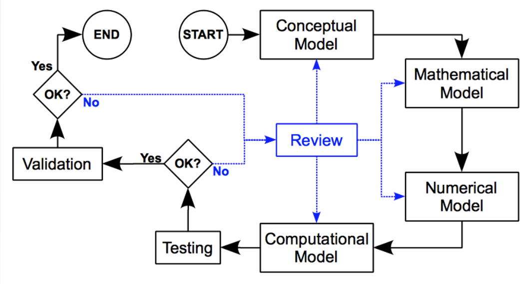
3. SLS method.



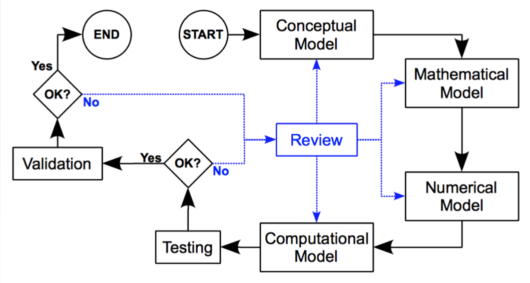
- Oil reservoir simulation is a grand challenge.

- The major goal of reservoir simulation is to predict future performance of reservoir and find ways and means of optimizing the recovery of some of the hydrocarbons under various operating conditions.

- The major goal of reservoir simulation is to predict future performance of reservoir and find ways and means of optimizing the recovery of some of the hydrocarbons under various operating conditions.
- It involves four main interrelated *modeling stages*:



- The major goal of reservoir simulation is to predict future performance of reservoir and find ways and means of optimizing the recovery of some of the hydrocarbons under various operating conditions.
- It involves four main interrelated *modeling stages*:

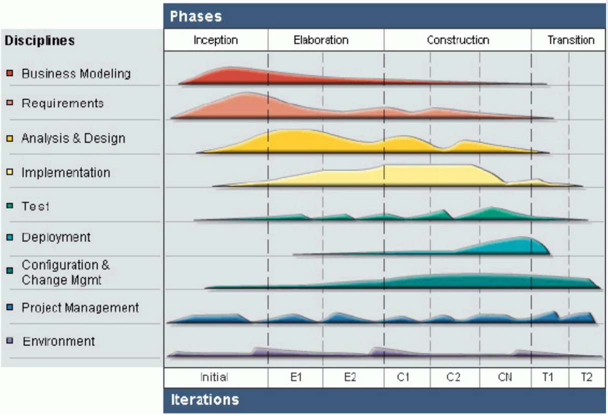


- And requires a combination of skills of physicists, mathematicians, reservoir engineers, and computer scientists.

Software Engineering (IEEE Comput Society's Software Eng. Body of Knowledge)

Application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.

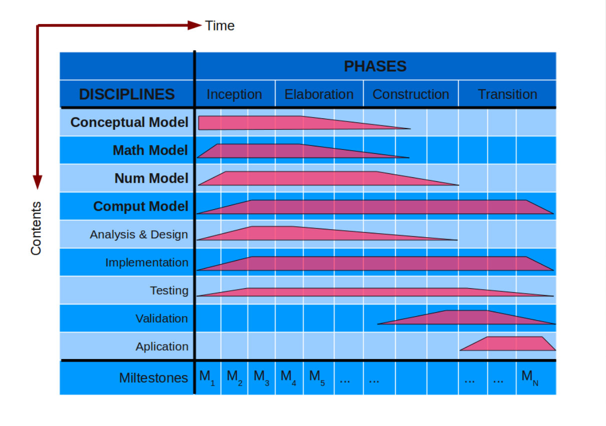
- Unified Process (UP, Booch *et al.* [2])



Software Engineering (IEEE Comput Society's Software Eng. Body of Knowledge)

Application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.

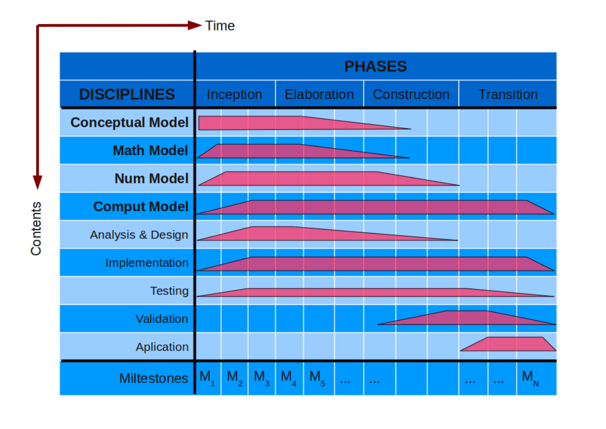
- Unified Process (UP, Booch *et al.* [2])



Software Engineering (IEEE Comput Society's Software Eng. Body of Knowledge)

Application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.

- Unified Process (UP, Booch *et al.* [2])

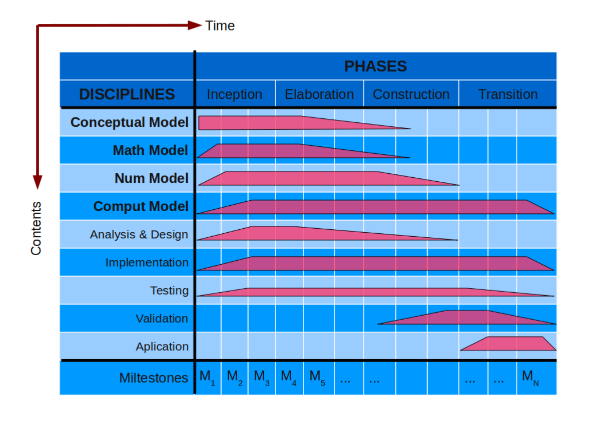


- Requirements
  - Efficiency
  - Accuracy
  - Abstraction

Software Engineering (IEEE Comput Society's Software Eng. Body of Knowledge)

Application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.

- Unified Process (UP, Booch *et al.* [2])



- Requirements
  - Efficiency
  - Accuracy
  - Abstraction
- We get a software:
  - Modular
  - Mantainable
  - Reliable
  - Efficient
  - Productive

Oil production stages

- First stage of oil reservoir production, ***primary recovery***, the oil is extracted by natural drive mechanism.

Oil production stages

- First stage of oil reservoir production, ***primary recovery***, the oil is extracted by natural drive mechanism.
- The reservoir pressure can be maintained, using techniques such as gas or water injection. This is known as ***secondary recovery***.

Oil production stages

- First stage of oil reservoir production, ***primary recovery***, the oil is extracted by natural drive mechanism.
- The reservoir pressure can be maintained, using techniques such as gas or water injection. This is known as ***secondary recovery.***
- Tertiary or **Enhanced Oil Recovery** (EOR) is a generic term that embraces several techniques used to increase the amount of crude oil that can be extracted from an oil field.
    - These techniques are based on the injection of materials not normally present in the reservoir, and is the most advanced stage of the exploitation of a reservoir.

Oil production stages

- First stage of oil reservoir production, ***primary recovery***, the oil is extracted by natural drive mechanism.
- The reservoir pressure can be maintained, using techniques such as gas or water injection. This is known as ***secondary recovery***.
- Tertiary or **Enhanced Oil Recovery** (EOR) is a generic term that embraces several techniques used to increase the amount of crude oil that can be extracted from an oil field.
  - These techniques are based on the injection of materials not normally present in the reservoir, and is the most advanced stage of the exploitation of a reservoir.
- Primary recovery techniques produce 10 – 15 % of the reservoir's oil content. **Combining the processes of secondary and tertiary recovery techniques, it is possible to produce 30 – 60 % of the reservoir's total oil content**.

Table of contents

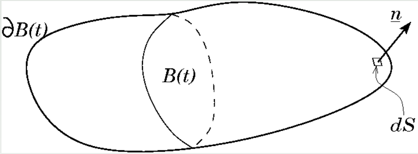
Extensive and Intensive Properties

- In the physical sciences, ***intensive property*** (also called a bulk property, intensive quantity, or intensive variable), is a physical property of a system that does not depend on the system size or the amount of material in the system: it is scale invariant.
  - Density
- By contrast, an ***extensive property*** (also extensive quantity, extensive variable, or extensive parameter) of a system is directly proportional to the system size or the amount of material in the system.
  - Mass

Axiomatic Formulation, (Herrera *et al.* [3, 4]) I

1. To find extensive E and intensive ψ properties :

$$E(t) = \int_{B(t)} \psi(\vec{x}, t) d\vec{x}$$



2. To establish balances:

$$\frac{dE}{dt} = \frac{d}{dt} \int_{B(t)} \psi(\vec{x}, t) d\vec{x} = \int_{B(t)} q(\vec{x}, t) d\vec{x} + \int_{\partial B(t)} \vec{\tau}(\vec{x}, t) \cdot \vec{n} dS \qquad (1)$$

where $q(\vec{x}, t)$ y $\vec{\tau}(\vec{x}, t)$ are the source term in $B(t)$ and the flux vector through the boundary $\partial B(t)$, respectively

Axiomatic Formulation, (Herrera *et al.* [3, 4]) II

- Global balance

$$\int_{B(t)} \left\{ \frac{\partial \psi}{\partial t} + \nabla \cdot (\vec{v}\psi) \right\} d\vec{x} = \int_{B(t)} q d\vec{x} + \int_{B(t)} \nabla \cdot \vec{\tau} d\vec{x} \tag{2}$$

- Local balance

$$\frac{\partial \psi}{\partial t} + \nabla \cdot (\vec{v}\psi) = q + \nabla \cdot \vec{\tau} \tag{3}$$

Conservative form

- Defining a *"flux function"* (see [5]) as $\vec{f} = \vec{v}\psi - \vec{\tau}$ we get:

$$\frac{\partial}{\partial t} \int\limits_{B(t)} \psi d\vec{x} + \int\limits_{B(t)} \nabla \cdot \vec{f} d\vec{x} = \int\limits_{B(t)} q d\vec{x} \tag{4}$$

and therefore

$$\frac{\partial \psi}{\partial t} + \nabla \cdot \vec{f} = q \tag{5}$$

- Equivalently (4) can be written as follows

$$\frac{\partial}{\partial t} \int\limits_{B(t)} \psi d\vec{x} + \int\limits_{\partial B(t)} \vec{f} \cdot \vec{n} dS = \int\limits_{B(t)} q d\vec{x} \tag{6}$$

Table of contents

- In general, the equations governing a mathematical model of a reservoir cannot be solved by analytical methods.

- In general, the equations governing a mathematical model of a reservoir cannot be solved by analytical methods.
- Instead, a numerical model can be produced in a form that is amenable to solution by digital computers.

- In general, the equations governing a mathematical model of a reservoir cannot be solved by analytical methods.
- Instead, a numerical model can be produced in a form that is amenable to solution by digital computers.
- Since the 1950s, numerical models have been used to predict, understand, and optimize complex physical fluid flow processes in petroleum reservoirs.

- In general, the equations governing a mathematical model of a reservoir cannot be solved by analytical methods.
- Instead, a numerical model can be produced in a form that is amenable to solution by digital computers.
- Since the 1950s, numerical models have been used to predict, understand, and optimize complex physical fluid flow processes in petroleum reservoirs.
- Recent advances in computational capabilities have greatly expanded the potential for solving larger problems and hence permitting the incorporation of more physics into the differential equations.

1. Finite Diferences Method (FDM)
    - The FDM can be very easy to implement.
    - Faster than FEM.
    - High accuracy difference schemes can be constructed.
    - In its basic form is restricted to handle only rectangular shapes.
    - Introduce considerable geometrical error and grid orientation effects.
    - Curvilinear coordinates can be used.

1. Finite Diferences Method (FDM)
    - The FDM can be very easy to implement.
    - Faster than FEM.
    - High accuracy difference schemes can be constructed.
    - In its basic form is restricted to handle only rectangular shapes.
    - Introduce considerable geometrical error and grid orientation effects.
    - Curvilinear coordinates can be used.
2. Finite Element Method (FEM)
    - The FEM cand handle complicated geometries.
    - Reduce the grid orientation effects.
    - Solid theoretical foundations.
    - Can manage local grid refinement.
    - The quality of a FEM approximation is often higher than in the corresponding FDM approach.
    - FEM is the method of choice in structural mechanics.
    - It is not easy to implement and is slower than FDM.

3. Finite Volume Method (FVM)
    - Values are calculated at *control volumes*.
    - Conservative method: the flux entering a given volume is identical to that leaving the adjacent volume.
    - Can easily be formulated to allow for unstructured meshes.
    - Used in many computational fluid dynamics packages.
    - FVM is in between FDM and FEM: faster and easier to implement than FEM; and more accurate and versatile than FDM.

3. Finite Volume Method (FVM)
   - Values are calculated at *control volumes*.
   - Conservative method: the flux entering a given volume is identical to that leaving the adjacent volume.
   - Can easily be formulated to allow for unstructured meshes.
   - Used in many computational fluid dynamics packages.
   - FVM is in between FDM and FEM: faster and easier to implement than FEM; and more accurate and versatile than FDM.

- In oil-reservoir problems we usually require a large number of cells ($10^5 - 10^6$), therefore cost of the solution favors simpler and lower order approximation within each cell.

3. Finite Volume Method (FVM)
    - Values are calculated at *control volumes*.
    - Conservative method: the flux entering a given volume is identical to that leaving the adjacent volume.
    - Can easily be formulated to allow for unstructured meshes.
    - Used in many computational fluid dynamics packages.
    - FVM is in between FDM and FEM: faster and easier to implement than FEM; and more accurate and versatile than FDM.

- In oil-reservoir problems we usually require a large number of cells ($10^5 - 10^6$), therefore cost of the solution favors simpler and lower order approximation within each cell.
- About 80% – 90% of the total simulation time is spent on the solution of linear systems.
    - Fast linear solvers are crucial to solve sparse, highly non–symmetric, and ill–conditioned systems.
    - Krylov subspace (preconditioned) algorithms are preferred.

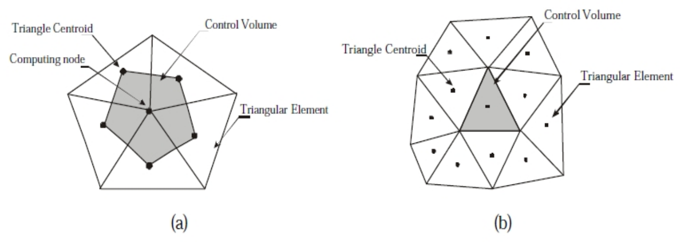
Table of contents

- Finite volume methods are derived on the basis of the conservative form of the balance equations [3, 4, 5].

$$\frac{\partial}{\partial t}\int_{B(t)} \psi d\vec{x} + \int_{B(t)} \nabla \cdot \vec{f} d\vec{x} = \int_{B(t)} q d\vec{x}$$

or

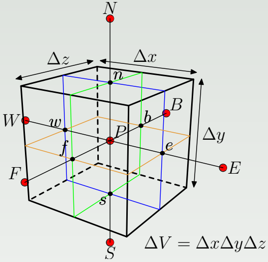
$$\frac{\partial}{\partial t}\int_{B(t)} \psi d\vec{x} + \int_{\partial B(t)} \vec{f} \cdot \vec{n} dS = \int_{B(t)} q d\vec{x}$$



- FVM is *conservative.*

- *Conservative form* of general balance equation:

$$\frac{\partial}{\partial t}\int_{B(t)} \psi d\vec{x} + \int_{B(t)} \nabla \cdot \vec{f} d\vec{x} = \int_{B(t)} q d\vec{x}$$



- Integrating on Δt and taking $B(t) \equiv \Delta V$:

$$\int_{\Delta t} \frac{\partial}{\partial t}\int_{\Delta V} \psi dV dt + \int_{\Delta t}\int_{\Delta V} \nabla \cdot \vec{f} dV dt = \int_{\Delta t}\int_{\Delta V} q dV dt$$

- Notation:

NB	*id*
P	i, j, k
E	$i+1, j, k$
W	$i-1, j, k$
N	$i, j+1, k$
S	$i, j-1, k$
F	$i, j, k+1$
B	$i, j, k-1$

nb	*id*
e	$i+\frac{1}{2}, j, k$
w	$i-\frac{1}{2}, j, k$
n	$i, j+\frac{1}{2}, k$
s	$i, j-\frac{1}{2}, k$
f	$i, j, k+\frac{1}{2}$
b	$i, j, k-\frac{1}{2}$

$$t \equiv n; \quad t + \Delta t \equiv n+1$$

$$\int_{\Delta t} g\; dt \equiv \int_{n}^{n+1} g\; dt$$

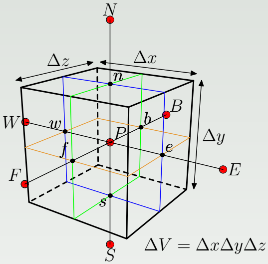
$$\int_{\Delta V} g\; dV \equiv \int_{b}^{f}\int_{s}^{n}\int_{w}^{e} g\; dx\; dy\; dz$$

- Approximation of the integrals:

$$\int_{n}^{n+1} \frac{\partial}{\partial t} \int_{\Delta V} \psi dV dt \approx \left(\psi_{\boldsymbol{P}}^{n+1} - \psi_{\boldsymbol{P}}^{n} \right) \Delta V$$

$$\int_{n}^{n+1} \int_{\Delta V} \nabla \cdot \vec{f} dV dt \approx \int_{n}^{n+1} \mathcal{F}(\vec{f}_{\boldsymbol{nb}}) dt$$

$$\int_{\Delta t} \int_{\Delta V} q dV dt \approx \int_{n}^{n+1} \bar{Q} \Delta V dt$$



- Theta scheme:

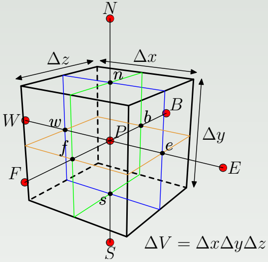
$$\int_{n}^{n+1} \mathcal{F}\, dt = \left(\theta \mathcal{F}^{n+1} + (1-\theta) \mathcal{F}^{n} \right) \Delta t, \qquad 0 \leq \theta \leq 1$$

Explicit (Forward–Euler)	$\theta = 0$	$\mathcal{F}^{n} \Delta t.$
Implicit (Backward–Euler)	$\theta = 1$	$\mathcal{F}^{n+1} \Delta t.$
Crank–Nicolson	$\theta = 1/2$	$(\mathcal{F}^{n} + \mathcal{F}^{n+1}) \Delta t/2.$

- Recall that $\vec{f} = \vec{v}\psi - \vec{\tau}$:

$$\mathcal{F}(\vec{f}_{nb}) \approx \int_{\Delta V} \nabla \cdot \vec{f} dV =$$

$$\int_w^e \int_s^n \int_b^f \Big(\frac{\partial(v_x\psi - \tau_x)}{\partial x} + \frac{\partial(v_y\psi - \tau_y)}{\partial y} + \frac{\partial(v_z\psi - \tau_z)}{\partial z} \Big) dxdydz$$

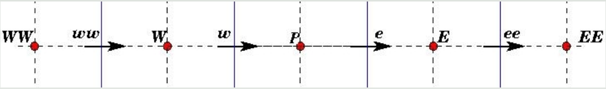


- Discretized flux function:

$$\mathcal{F}(\vec{f}_{nb}) = \big[(v_x\psi - \tau_x)_e - (v_x\psi - \tau_x)_w\big] A_x +$$
$$\big[(v_y\psi - \tau_y)_n - (v_y\psi - \tau_y)_s\big] A_y + \big[(v_z\psi - \tau_z)_f - (v_z\psi - \tau_z)_b\big] A_z$$

where $A_x = \Delta y \times \Delta z$, $A_y = \Delta x \times \Delta z$, $A_z = \Delta x \times \Delta y$, represents the area of the faces.

- Advective $\vec{v}\psi$ and Diffusive terms $\vec{\tau}$ need to be approximated on the faces.



- Diffusive terms
  - Central differences: e.g. $(\tau_x)_e = D \frac{\partial \psi}{\partial x}\Big|_e = D \frac{\psi_P - \psi_E}{\Delta x_e}$
- Advective terms
  - Average : $(v_x \psi)_e = (v_x)_e (\lambda \psi_E + (1-\lambda)\psi_P)$, where $\lambda = \frac{x_e - x_P}{\Delta x_e}$
  - Upstream:

Upwind

if $((v_x)_e > 0)$ **then**
  $\psi_e = \psi_P$
else
  $\psi_e = \psi_E$
end if

QUICK (second order upstream)

if $((v_x)_e > 0)$ **then**
  $\psi_e = h(\psi_W, \psi_P, \psi_E)$
else
  $\psi_e = h(\psi_P, \psi_E, \psi_{EE})$
end if

- Implicit, non-linear:

$$\boldsymbol{a}_P^{n+1}\psi_P^{n+1} = \boldsymbol{a}_E^{n+1}\psi_E^{n+1} + \boldsymbol{a}_W^{n+1}\psi_W^{n+1} + \boldsymbol{a}_N^{n+1}\psi_N^{n+1} + \boldsymbol{a}_S^{n+1}\psi_S^{n+1} + \boldsymbol{a}_F^{n+1}\psi_F^{n+1} + \boldsymbol{a}_B^{n+1}\psi_B^{n+1} + q_P^n$$

- Implicit linear:

$$\boldsymbol{a}_P^{n}\psi_P^{n+1} = \boldsymbol{a}_E^{n}\psi_E^{n+1} + \boldsymbol{a}_W^{n}\psi_W^{n+1} + \boldsymbol{a}_N^{n}\psi_N^{n+1} + \boldsymbol{a}_S^{n}\psi_S^{n+1} + \boldsymbol{a}_F^{n}\psi_F^{n+1} + \boldsymbol{a}_B^{n}\psi_B^{n+1} + q_P^n$$

- Explicit:

$$\boldsymbol{a}_P^{n}\psi_P^{n+1} = \boldsymbol{a}_E^{n}\psi_E^{n} + \boldsymbol{a}_W\psi_W^{n} + \boldsymbol{a}_N^{n}\psi_N^{n} + \boldsymbol{a}_S^{n}\psi_S^{n} + \boldsymbol{a}_F^{n}\psi_F^{n} + \boldsymbol{a}_B^{n}\psi_B^{n} + q_P^n$$

Linear Systems

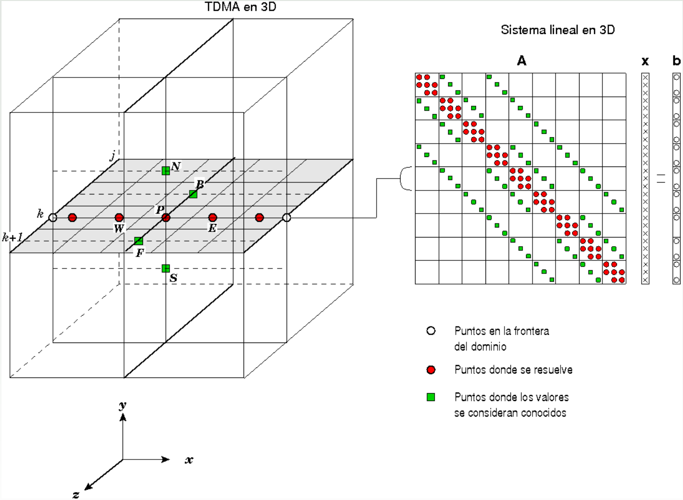
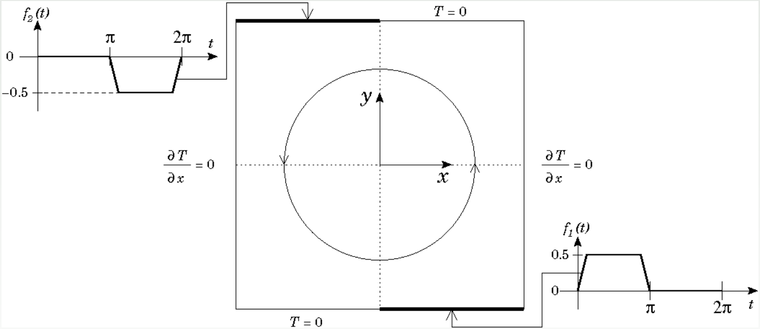
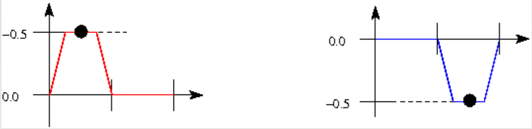


Table of contents

Template Units for Numerical Applications

- Natural convection in box-shaped containers





$$f_1(t) = \begin{cases} 0.5\sin^2(4t) & \text{for} \quad 0 \le t < \pi/8 \\ 1 & \text{for} \quad \pi/8 \le t < 7\pi/8 \\ 0.5\sin^2(4t-3\pi) & \text{for} \quad 7\pi/8 \le t < \pi \\ 0 & \text{for} \quad \pi \le t < 2\pi \end{cases}$$

$$f_2(t) = \begin{cases} 0 & \text{for} \quad 0 \le t < \pi \\ -0.5\sin^2(4t-3\pi) & \text{for} \quad \pi \le t < 9\pi/8 \\ 1 & \text{for} \quad 9\pi/8 \le t < 15\pi/8 \\ -0.5\sin^2(4t-6\pi) & \text{for} \quad 15\pi/8 \le t < 2\pi \end{cases}$$

Navier-Stokes equations

- Mass balance:

$$\frac{\partial u_j}{\partial x_j} = 0$$

- Momentum balance (Navier-Stokes):

$$\rho_0 \left[\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} \right] = -\frac{\partial p}{\partial x_i} + \mu \frac{\partial^2 u_i}{\partial x_j \partial x_j} + \rho b_i$$

- Energy balance:

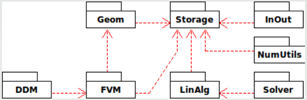
$$\frac{\partial T}{\partial t} + u_j \frac{\partial T}{\partial x_j} = \alpha \frac{\partial^2 T}{\partial x_j \partial x_j}$$

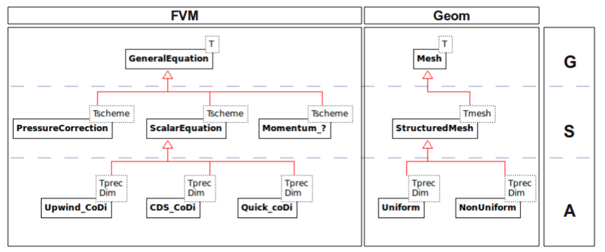
- Equations of state:

$$\rho = \rho_0 \left[1 - \beta (T - T_0) \right], \quad \beta = -\frac{1}{\rho_0} \left(\frac{\partial \rho}{\partial T} \right)_{T=T_0}$$

Template Units for Numerical Applications I

- TUNA use several C++ template techniques (Blitz++).





Template Units for Numerical Applications II

http://mmc.geofisica.unam.mx/ (then go to my homepage).

```
Examples : programs that use TUNA
 |--- tuna-cfd-rules.in => rules to compile the examples
 |--- 01StructMesh => uniform structured meshes
 |--- 02NonUniformMesh => non uniform structured meshes
 |--- 03Laplace => Solution of Laplace equation
 |--- 04HeatDiffusion => Solution of heat conduction problems
 |--- 05ConvDiffForced => Solution of forced convection
 |--- 06ConvDiff => Solution of natural convection problems
 |--- 07ConvDiffLES => Solution of turbulent natural convection
 |--- README.pdf => Explanation of the examples of each directory

1) Unpack TUNA and change to the TUNA dir:
% tar zxvf TUNA.tar.gz
% cd TUNA


2) Blitz++: http://www.oonumerics.org/blitz/
```

Template Units for Numerical Applications III

```
- Unpack with: tar zxvf blitz-09.tar.gz
- Change to blitz-0.9 with: cd blitz-0.9
- Config blitz with: ./configure --prefix=$PWD/../BLITZ
- Compile and install blitz with: make install

These instruction will install Blitz in the TUNA/BLITZ directory

3) Run the examples
- Change to the Examples directory: cd Examples
- Edit the files tuna-cfd-rules.in
  Change the environment variable BASE according to your paths.
  (e.g. BASE = /home/luiggi/TUNA)

- Then, e.g. change to the 06ConvDiff dir:
  % cd 06ConvDiff
  % make     <---  this creates: convdiff1 and convdiff2

- Visualization: CXXFLAGS: Add -DWITH_GNUPLOT or -DWITH_DX.
```

Example: Natural Convection I

```
#include "Meshes/Uniform.hpp"
#include "Storage/DiagonalMatrix.hpp"
#include "Equations/ScalarEquation.hpp"
#include "Schemes/CDS_CoDi.hpp"
#include "Equations/Momentum_XCoDi.hpp"
#include "Schemes/CDS_XCoDi.hpp"
#include "Equations/Momentum_YCoDi.hpp"
#include "Schemes/CDS_YCoDi.hpp"
#include "Equations/Momentum_ZCoDi.hpp"
#include "Schemes/CDS_ZCoDi.hpp"
#include "Equations/PressureCorrection.hpp"
#include "Schemes/Simplec.hpp"
#include "Solvers/TDMA.hpp"
#include "Utils/inout.hpp"
#include "Utils/num_utils.hpp"
#include "Utils/GNUplot.hpp"
using namespace Tuna;

typedef TunaArray<double,3>::huge ScalarField3D;
```

Example: Natural Convection II

```
DiagonalMatrix<double, 3> A(num_nodes_x, num_nodes_y, num_nodes_z);
ScalarField3D             b(num_nodes_x, num_nodes_y, num_nodes_z);

StructuredMesh<Uniform< double, 3> > mesh(length_x, num_nodes_x,
                                          length_y, num_nodes_y,
                                          length_z, num_nodes_z);

ScalarField3D T(mesh.getExtentVolumes());
ScalarField3D p(mesh.getExtentVolumes());
ScalarField3D u(mesh.getExtentVolumes());      // u-velocity
ScalarField3D v(mesh.getExtentVolumes());      // v-velocity
ScalarField3D w(mesh.getExtentVolumes());      // w-velocity

Range all = Range::all();
T(T.lbound(firstDim), all, all) = left_wall;   // Left
T(T.ubound(firstDim), all, all) = right_wall;    // Rigth
```

Example: Natural Convection III

```
ScalarEquation<CDS_CoDi<double,3> > energy(T, A, b, mesh.getDeltas());
energy.setDeltaTime(dt);
energy.setNeumann(TOP_WALL);
energy.setNeumann(BOTTOM_WALL);
energy.setDirichlet(LEFT_WALL, left_wall);
energy.setDirichlet(RIGHT_WALL, right_wall);
energy.setNeumann(FRONT_WALL);
energy.setNeumann(BACK_WALL);
energy.setUvelocity(us);
energy.setVvelocity(vs);
energy.setWvelocity(ws);
energy.print();

Momentum_XCoDi<CDS_XCoDi<double, 3> > mom_x(us, A, b, mesh.getDeltas());

Momentum_YCoDi<CDS_YCoDi<double, 3> > mom_y(vs, A, b, mesh.getDeltas());

Momentum_ZCoDi<CDS_ZCoDi<double, 3> > mom_z(ws, A, b, mesh.getDeltas());

PressureCorrection<Simplec<double, 3> > press(pp, A, b, mesh.getDeltas());
```

Example: Natural Convection IV

```
template<typename Tprec, int Dim>
class CDS_CoDi : public ScalarEquation< CDS_CoDi< Tprec, Dim > >
{
public:
  typedef Tprec prec_t;
  typedef typename TunaArray<prec_t, Dim >::huge ScalarField;

  CDS_CoDi() : ScalarEquation<CDS_CoDi<prec_t, Dim> >() { }
  ~CDS_CoDi() { };

  inline void calcCoefficients1D();
  inline void calcCoefficients2D();
  inline void calcCoefficients3D();
  inline void printInfo() { std::cout << " CDS_CoDi "; }
};
```

$$\boldsymbol{a}_P^n \psi_P^{n+1} = \boldsymbol{a}_E^n \psi_E^{n+1} + \boldsymbol{a}_W^n \psi_W^{n+1} + \boldsymbol{a}_N^n \psi_N^{n+1} + \boldsymbol{a}_S^n \psi_S^{n+1} + \boldsymbol{a}_F^n \psi_F^{n+1} + \boldsymbol{a}_B^n \psi_B^{n+1} + q_P^n$$

Example: Natural Convection V

```
template<class Tprec, int Dim>
inline void CDS_CoDi<Tprec, Dim>::calcCoefficients2D() {
    prec_t Gdy_dx = Gamma * dy / dx, Gdx_dy = Gamma * dx / dy;
    prec_t dxy_dt = dx * dy / dt;
    aE = 0.0; aW = 0.0; aN = 0.0; aS = 0.0; aP = 0.0; sp = 0.0;
    for (int i =  bi; i <= ei; ++i)
      for (int j = bj; j <= ej; ++j) {
        aE (i,j) = Gdy_dx - u(i  , j) * dy * 0.5 ;
        aW (i,j) = Gdy_dx + u(i-1, j) * dy * 0.5 ;
        aN (i,j) = Gdx_dy - v(i, j  ) * dx * 0.5;
        aS (i,j) = Gdx_dy + v(i, j-1) * dx* 0.5;
        aP (i,j) = aE (i,j) + aW (i,j) + aN (i,j) + aS (i,j) + dxy_dt;
        sp (i,j) = phi_0(i,j) * dxy_dt ;
      }
    applyBoundaryConditions2D();
}
```

$$\boldsymbol{a}_P^n \psi_P^{n+1} = \boldsymbol{a}_E^n \psi_E^{n+1} + \boldsymbol{a}_W^n \psi_W^{n+1} + \boldsymbol{a}_N^n \psi_N^{n+1} + \boldsymbol{a}_S^n \psi_S^{n+1} + \boldsymbol{a}_F^n \psi_F^{n+1} + \boldsymbol{a}_B^n \psi_B^{n+1} + q_P^n$$

Example: Natural Convection VI

```
for(iteration = 1; iteration <= max_time_steps; ++iteration) {
    sorsum = SIMPLEC(energy, mom_x, mom_y, mom_z, press, max_iter, tolerance);
}

template<class T_e, class T_x, class T_y, class T_z, class T_p>
double SIMPLEC(T_e &energy, T_x &mom_x, T_y &mom_y, T_z &mom_z, T_p &press,
               int max_iter, double tolerance)
{
    double sorsum = 10.0, tol_simplec = 1e-02;
    int counter = 0;

    while ( (sorsum > tol_simplec) && (counter < 20) ) {
       energy.calcCoefficients();
       Solver::TDMA3D(energy, tolerance, max_iter);
       errorT = energy.calcErrorL2();
       energy.update();

       mom_x.calcCoefficients();
       Solver::TDMA3D(mom_x, tolerance, max_iter);
       errorX = mom_x.calcErrorL2();
       mom_x.update();
```

Example: Natural Convection VII

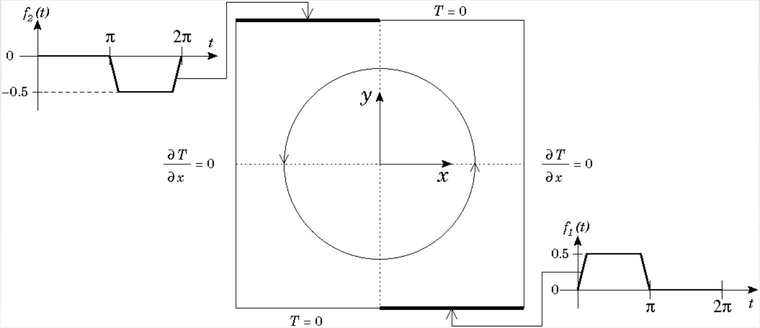
```
        mom_y.calcCoefficients();
        Solver::TDMA3D(mom_y, tolerance, max_iter);
        errorY = mom_y.calcErrorL2();
        mom_y.update();

        mom_z.calcCoefficients();
        Solver::TDMA3D(mom_z, tolerance, max_iter);
        errorZ = mom_z.calcErrorL2();
        mom_z.update();

        press.calcCoefficients();
        Solver::TDMA3D(press, tolerance, max_iter);
        press.correction();
        sorsum = fabs( press.calcSorsum() );

        ++counter;
    }
    return sorsum;
}
```

Example: Natural Convection VIII



References I

[1] R.A. Tapia and C. Lanius,
Computational Science: Tools for a Changing World
http://ceee.rice.edu/Books/CS/chapter1/intro52.html, 2001.

[2] I. Jacobson and G. Booch and J. RumbaughPrimer,
The Unified Software Development Process,
Addison – Wesley, 1999.

[3] I. Herrera and M. B. Allen and G. F. Pinder,
Numerical modeling in science and engineering,
John Wiley & Sons., USA, 1988.

[4] I. Herrera and G. F. Pinder,
General principles of mathematical computational modeling,
John Wiley, in press.

References II

[5] R.J. Leveque,
Finite Volume Method for Hyperbolic Problems,
Cambridge University Press, 2004.