DAE solver and symbolic computation

É. Canot (SAGE team)

IRISA (CNRS, Inria Rennes, INSA Rennes & Université de Rennes 1)

RAIM, 20-22 June 2012, Dijon

The SAGE team at IRISA

Simulation and Algorithms on the Grid for Environment
(ALADIN team up to 2003)
J. Erhel, B. Philippe, É. Canot, G. Pichot

- Linear algebra, Sparse algorithms, Eigenvalue problems
- Numerical simulation of (very) large systems, using parallelism
- Global approach for coupled physical problems
- Application to hydrogeology (H2OLab software) and archaeology

→ `http://www.irisa.fr/sage`

UMR IRISA

part 1: Jacobian derivation via symbolic calculation

Using Maple

for automatic differentiation

When a jacobian matrix is needed?

Typically: solution a the non-linear system $R(x) = 0$

- Gradient methods are useful
- $\rightarrow$ Newton and quasi-Newton methods
- $\rightarrow$ efficient methods

Jacobian is required: $J_{i,j} = \frac{\partial R_i}{\partial x_j}$

R(x) in Fortran 90: routine 'deriv()'

```
      ! diffusion ***************************************************

      rhoc_e = alpha*rhot(num)*ct(num) + (1-alpha)*rho_s*cs
      f3 = f1*rhoc_e
      f4 = f2*rhoc_e
      prc_s_mr = 2.0d0*per(num)*rhot(num)*ct(num)/mut(num)
      prc_s_mr_x = prc_s_mr/f3
      prc_s_mr_y = prc_s_mr/f4

      ! avec la vitesse de Darcy (couplage)
      if( j == 1 ) then
! Dirichlet (dessus)
         delta(num) = yprime(num)
      else if( j == ny ) then
! Neumann (dessous)
!   indice 3 impossible -- vy = 0

         if( i == 1 ) then
! Coin inférieur droit  -- vy = 0
!   indice 2 impossible -- vx = 0

            delta(num) =                                yprime(num) &
                       + 2.0d0*(kt(num)+kt(num1))/f3 * y(num)      &
                       - 2.0d0*(kt(num)+kt(num1))/f3 * y(num1)     &
                       - 2.0d0*(kt(num4)+kt(num))/f4 * y(num4)     &
                       + 2.0d0*(kt(num4)+kt(num))/f4 * y(num)

         else if( i == nx ) then
! Coin inférieur gauche -- vy = 0
!   indice 1 impossible -- vx = 0

            delta(num) =                                yprime(num) &
                       - 2.0d0*(kt(num2)+kt(num))/f3 * y(num2)     &
                       + 2.0d0*(kt(num2)+kt(num))/f3 * y(num)      &
                       - 2.0d0*(kt(num4)+kt(num))/f4 * y(num4)     &
                       + 2.0d0*(kt(num4)+kt(num))/f4 * y(num)

      else ! 1 < i and i < nx
         vx = (y(nump1) - y(nump2))/2.0d0
         if( vx < 0.0d0 ) then
            fac1 = 0.0d0
         else
            fac1 = 1.0d0
         end if
```

How to compute J?

By approximation

- via Finite Differences (costly way, unstable, ...)
- by hand, keeping only few terms (not without risk)

Exact computation

- using Automatic Differentiation (ADIFOR, OpenAD, Tapenade, ...)
- using Symbolic Computation (**Maple**, Mathematica, Maxima, ...)

How to compute J? (cont.)

Maple script: definition of $R(x)$

```
with(CodeGeneration):

x := eps*(y[num]-Tf):
phi := (1+erf(x))/2:
phi_prime := epssqrtpi*exp(-x**2):
kt[num] := kl + diffk*phi:
kt[num] := 1/(alpha/kt[num]+(1-alpha)/ks):
ct[num] := cl + diffc*phi + L*phi_prime:
rhot[num] := rhol + (y[nump]/idealgascte/y[num]-rhol)*phi:
rhoc_e := alpha*rhot[num]*ct[num] + (1-alpha)*rhos*cs:
mu_l := 0.4527*(y[num]-Tref+40.0)^(-1.492):
mu_v := 1.22e-5 + 4.0e-8*(y[num]-Tf):
mut[num] := mu_l+(mu_v-mu_l)*phi:
f3 := f1*rhoc_e:
f4 := f2*rhoc_e:
prc_s_mr := 2*per[num]*rhot[num]*ct[num]/mut[num]:
prc_s_mr_x := prc_s_mr/f3:
prc_s_mr_y := prc_s_mr/f4:
#  partie diffusion
# pr j=ny et i =1
kt[num1] := subs( y[num]=y[num1], kt[num] ):
kt[num2] := subs( y[num]=y[num2], kt[num] ):
kt[num3] := subs( y[num]=y[num3], kt[num] ):
kt[num4] := subs( y[num]=y[num4], kt[num] ):
delta1[num] := yprime[num]
             + 2*(kt[num]+kt[num1])/f3*y[num]
             - 2*(kt[num]+kt[num1])/f3*y[num1]
             - 2*(kt[num4]+kt[num])/f4*y[num4]
             + 2*(kt[num4]+kt[num])/f4*y[num]:
#
# pr j=ny et i=nx
delta2[num] := yprime[num]
             + 2*(kt[num]+kt[num2])/f3*y[num]
             - 2*(kt[num]+kt[num2])/f3*y[num2]
             - 2*(kt[num4]+kt[num])/f4*y[num4]
```

How to compute J? (cont.)

Maple script: jacobian computation

```
# Jacobian
# partie diffusion
d[1] := cj*diff( delta1[num],yprime[num] ) +
        diff( delta1[num],y[num] ):
d[2] := diff( delta1[num],y[num1] ):
d[3] := diff( delta1[num],y[num4] ):
d[4] := diff( delta1[num],y[nump] ):
writeto("jacobian_1.f");
printf("C Contains the Fortran output of the jacobian\n");
printf("C (from the Maple script 'jacobian.mpl')\n");
printf("\n");
Fortran( d[1..4], optimize, coercetypes=false,
limitvariablelength=false, resultname=jacob ):
printf("      END\n");
writeto(terminal);

d[5] := cj*diff( delta2[num],yprime[num] ) +
        diff( delta2[num],y[num] ):
d[6] := diff( delta2[num],y[num2] ):
d[7] := diff( delta2[num],y[num4] ):
d[8] := diff( delta2[num],y[nump] ):
writeto("jacobian_2.f");
printf("C Contains the Fortran output of the jacobian\n");
```

UMR IRISA

Using Maple

By calling only the kernel (avoiding the GUI!)

- define the mathematical expressions using classical variables
- compute derivatives via the '`diff`' command
- generate optimized Fortran code
- output in as many files as needed

Post-processing (specific tools) – the whole process is automatized in a Makefile

- substitution of variable names (some of them are protected, or cannot be used when generate Fortran code)
- conversion from *Fortran fixed syntax* to *Fortran free new one*
- other minor fix, in order to be able to be included in the Fortran 90 source code.

UMR IRISA

Maple output: optimized Fortran code

```
! Contains the Fortran output of the jacobian
! (from the Maple script 'jacobian.mpl')

      t1 = y(num) - Tf
      t3 = erf(eps * t1)
      t4 = 1 + t3
      t6 = kl + diff_k * t4 / 2
      t9 = 1 - alpha
      t11 = t9 / ks
      t12 = alpha / t6 + t11
      t13 = t12 ** 2
      t16 = t6 ** 2
      t19 = sqrt(0.3141592653589793D1)
      t20 = 0.1D1 / t19
      t22 = 0.1D1 / t13 * alpha / t16 * diff_k * t20
      t23 = eps ** 2
      t24 = t1 ** 2
      t26 = exp(-t23 * t24)
      t27 = t26 * eps
      t28 = 0.1D1 / f1
      t29 = 0.1D1 / ideal_gas_cte
      t31 = 0.1D1 / y(num)
      t33 = y(nump) * t29 * t31 - rho_l
      t36 = alpha * (rho_l + t33 * t4 / 2)
      t38 = L * eps_sqrtpi
      t40 = cl + diff_c * t4 / 2 + t38 * t26
      t44 = t36 * t40 + t9 * rho_s * cs
      t45 = 0.1D1 / t44
      t46 = t28 * t45
      t47 = t46 * y(num)
      t51 = 0.1D1 / t12
      t52 = y(num1) - Tf
      t54 = erf(eps * t52)
      t57 = kl + diff_k * (1 + t54) / 2
```

UMR IRISA

How Maple output is included

```
        !-----------------------------------------------------------------
        ! pour l'équation d'énergie

        ! avec la vitesse de Darcy (couplage)
        if( j == 1 ) then
! Dirichlet (dessus)
           jacobian(num,num) = cj
        else if( j == ny ) then
! Neumann (dessous)
!   indice 3 impossible -- vy = 0

           if( i == 1 ) then
! Coin inférieur droit  -- vy = 0
!   indice 2 impossible -- vx = 0

#include "jacobian_1.inc"
              jacobian(num,num)  = jacob(1)
              jacobian(num,num1) = jacob(2)
              jacobian(num,num4) = jacob(3)
              jacobian(num,nump) = jacob(4)


           else if( i == nx ) then
! Coin inférieur gauche -- vy = 0
!   indice 1 impossible -- vx = 0

#include "jacobian_2.inc"
              jacobian(num,num)  = jacob(5)
              jacobian(num,num2) = jacob(6)
              jacobian(num,num4) = jacob(7)
              jacobian(num,nump) = jacob(8)

           else ! if( 1 < i .and. i < nx ) then
              vx = (y(nump1) - y(nump2))/2.0d0
              if( vx < 0.0d0 ) then
                 fac1 = 0.0d0
              else
                 fac1 = 1.0d0
              end if
```

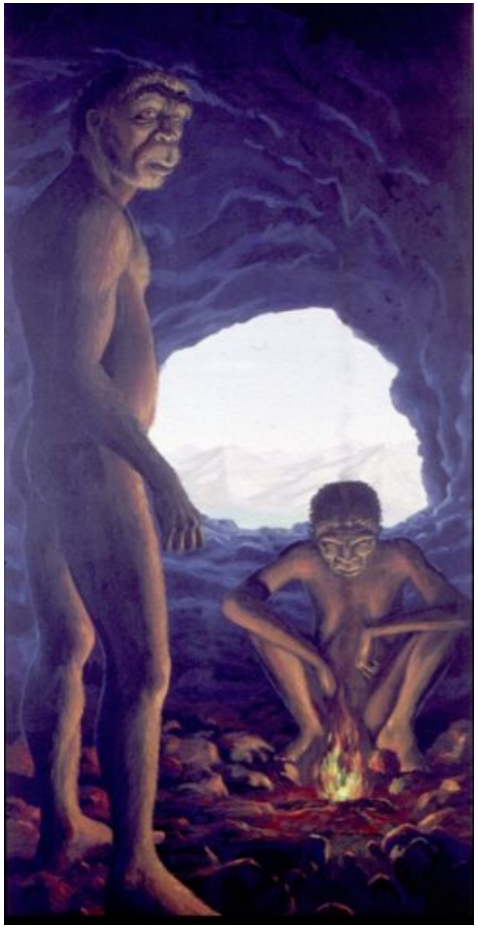
part 2: Application to numerical simulation

Heat and Mass Transfer

in saturated porous media

ARchaeology, PHYsics and MAThematics

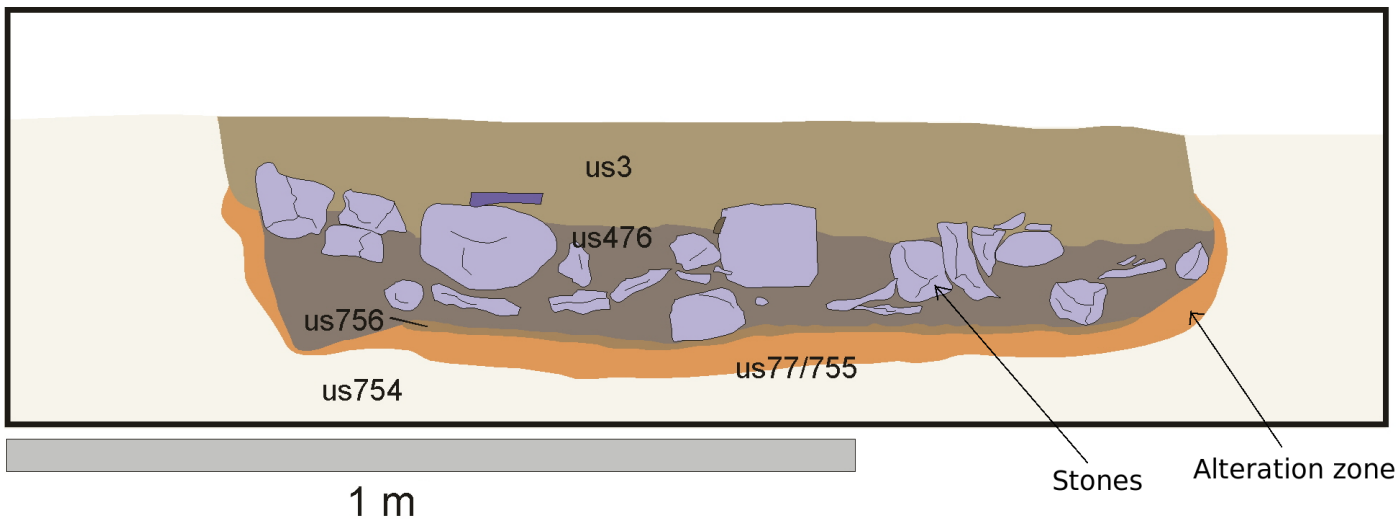
- Rules of human behavior
- Function of the combustion structures
- Reconstitution of the thermal history of each hearth



UMR IRISA

Foreword

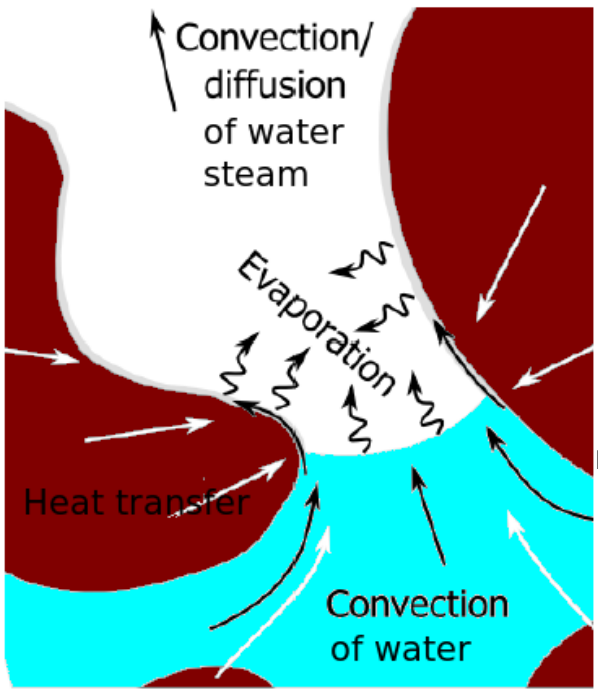
A real prehistoric fire occupation

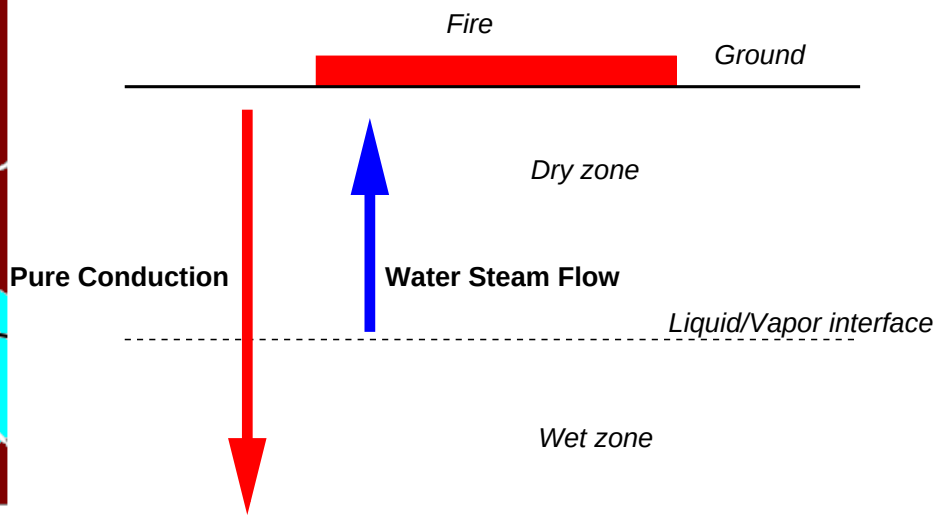


- Determine the shape of the occupations?
- Determine their mode of functioning?
- What was their utility?
- What was their minimal duration of burning?

UMR IRISA

Forced evaporation in soil



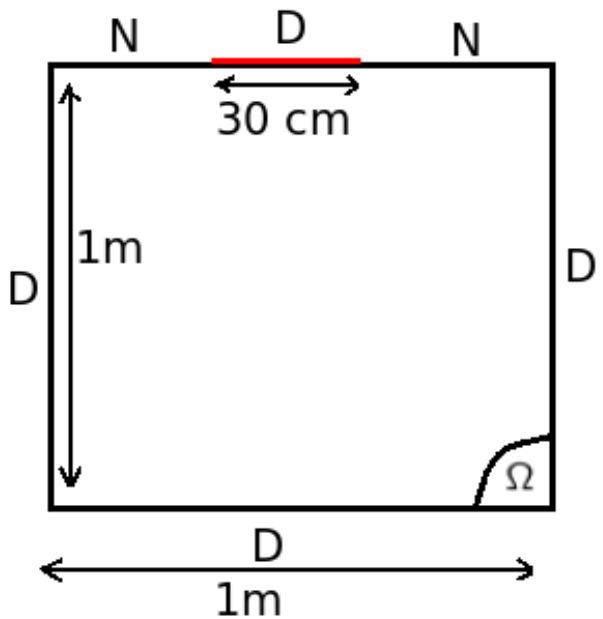


Set of equations

$$(\rho C)_e(T)\frac{\partial T(x,t)}{\partial t} + div(q) = \frac{k_e(\rho C)_f}{\mu_f}\nabla P.\nabla T \quad \text{in} \quad \Omega \times (0, t_{end}],$$
$$q = -k_e(T)\nabla T(x,t) \quad \text{in} \quad \Omega \times (0, t_{end}].$$

+ Initial conditions
+ Boundary conditions

$$(\rho C)_e = \phi(\rho C)_f + (1-\phi)(\rho C)_s$$
and $$\frac{1}{k_e} = \frac{\phi}{k_f} + \frac{1-\phi}{k_s}$$



RISA

Set of equations (cont.)

Momentum (Darcy law):

$$\vec{\nabla} P = -\frac{\mu_f}{K} \vec{V_f}$$

Mass conservation:

$$\frac{\partial(\phi \rho_f)}{\partial t} + \text{div}(\rho_f V_f) = 0$$

Fluid constitutive law:

$$\rho_f = F(p, T)$$

UMR IRISA

Set of equations (cont.)

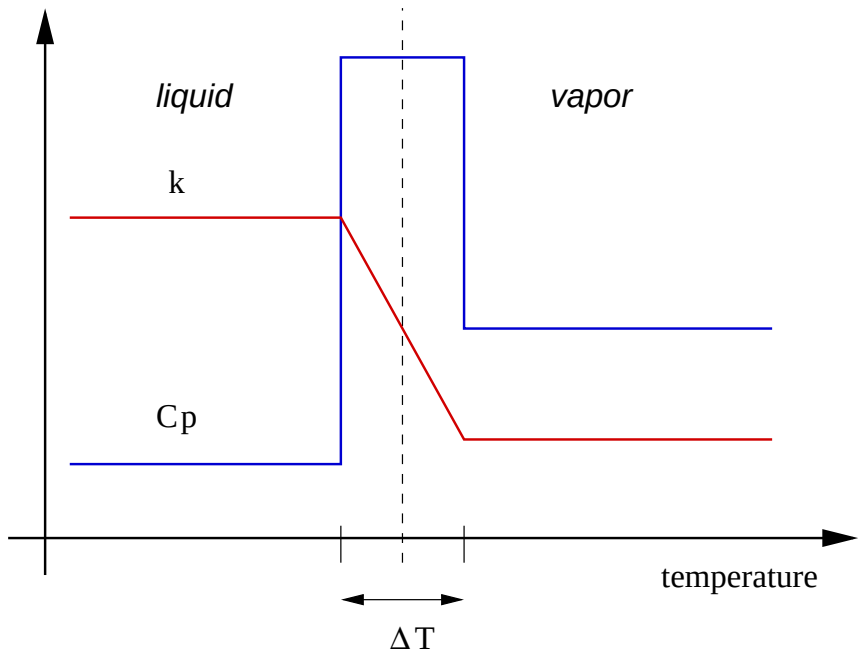
Flow equation:

$$\begin{aligned}\mathrm{div}(\nabla P_f) = & \frac{\phi \mu_f}{K \rho_f} \frac{\partial \rho_f}{\partial t} + \frac{1}{\mu_f} \nabla \mu_f . \nabla P_f \\ & - \frac{1}{\rho_f} \nabla \rho_f . \nabla P_f - \frac{1}{K} \nabla K . \nabla P_f\end{aligned}$$

Apparent Heat Capacity Method

Bonacina et al., 1970

- easy to implement
- leads to stiff system
- optimum value for parameter ΔT?



Smoothed Functions

$$C_f = C_v + (C_l - C_v)\,\sigma(T) + L\frac{d\sigma}{dT} \tag{1}$$

$$k_f = k_v + (k_l - k_v)\,\sigma(T) \tag{2}$$

where

$$\sigma(T) = \frac{1}{2}\left(1 + \mathrm{erf}(\epsilon(T - T_f))\right) \tag{3}$$

and

$$\frac{d\sigma}{dT} = \left(\epsilon\pi^{-1/2}\right)\exp[-\epsilon^2(T - T_f)^2] \tag{4}$$

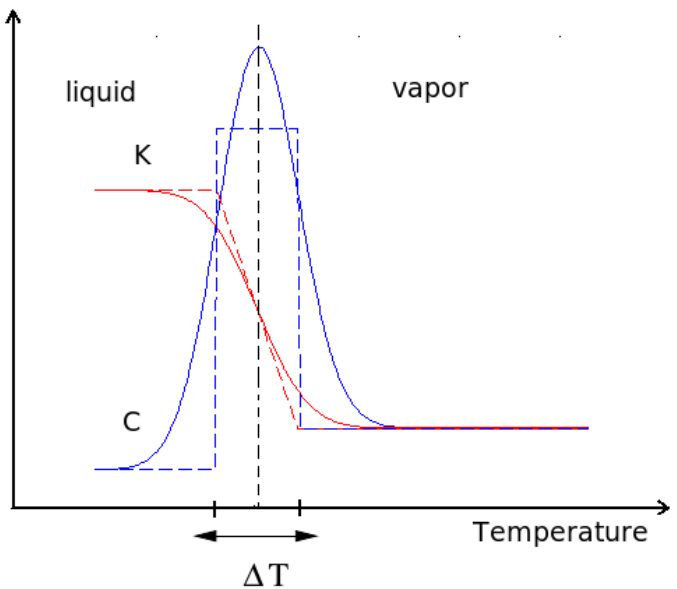
in which $\epsilon = 1/\sqrt{2}\Delta T$

Smoothed Functions: Schematic View

$$\rho_f = \rho_l + (\rho_v - \rho_l)\,\sigma(T)$$

$\rho_v \longrightarrow$ ideal gas law

$$\mu_f = \mu_l + (\mu_v - \mu_l)\,\sigma(T)$$



System of equations: DAE system

After spatial discretization only (Method of lines):

$$\begin{cases} \dfrac{\partial T}{\partial t} = f(t, x, T, P) \\ \gamma \dfrac{\partial T}{\partial t} + \theta \dfrac{\partial P}{\partial t} = g(t, x, T, P) \end{cases}$$

where $\gamma(T, P)$ and $\theta(T)$ may vanish. $(\theta \approx \frac{\partial \rho_f}{\partial P})$

This PDE system is algebraic of index one (when θ is null).

UMR IRISA

Complete model: best approach

- all equations and properties are treated as in AHC
- same equations for the whole domain (no front tracking)
- time integration is performed by the BDF method (quasi-Newton method is used to deal with the nonlinearity)
- use of approx. Jacobian reduces comput. cost (but see after)

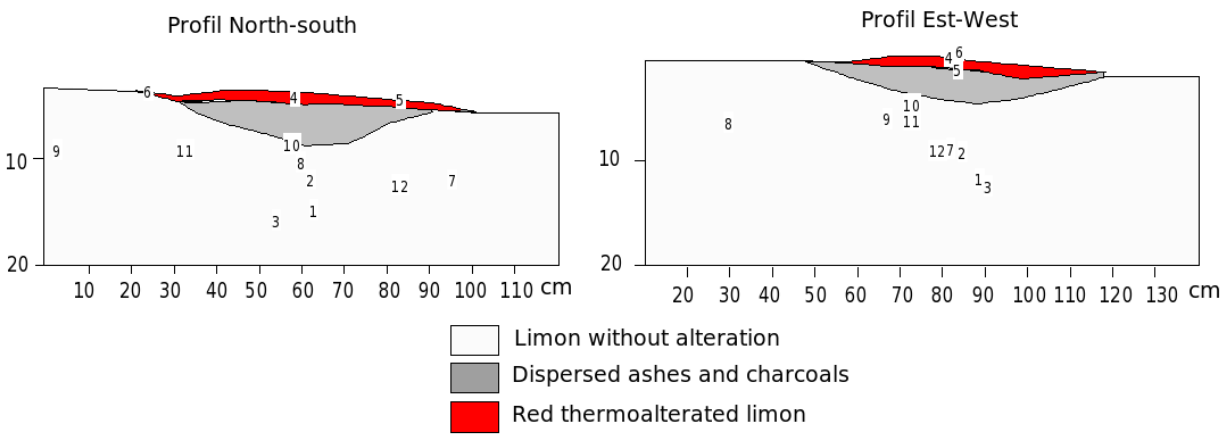
Solver DAE (DASSL of SLATEC)
CPU time: 10 min with sparse Jacobian (50 000 unknowns)
(instead of 6 hrs with dense Jacobian)

Results for 1D, 2D, 3D-axisymmetric

UMR IRISA

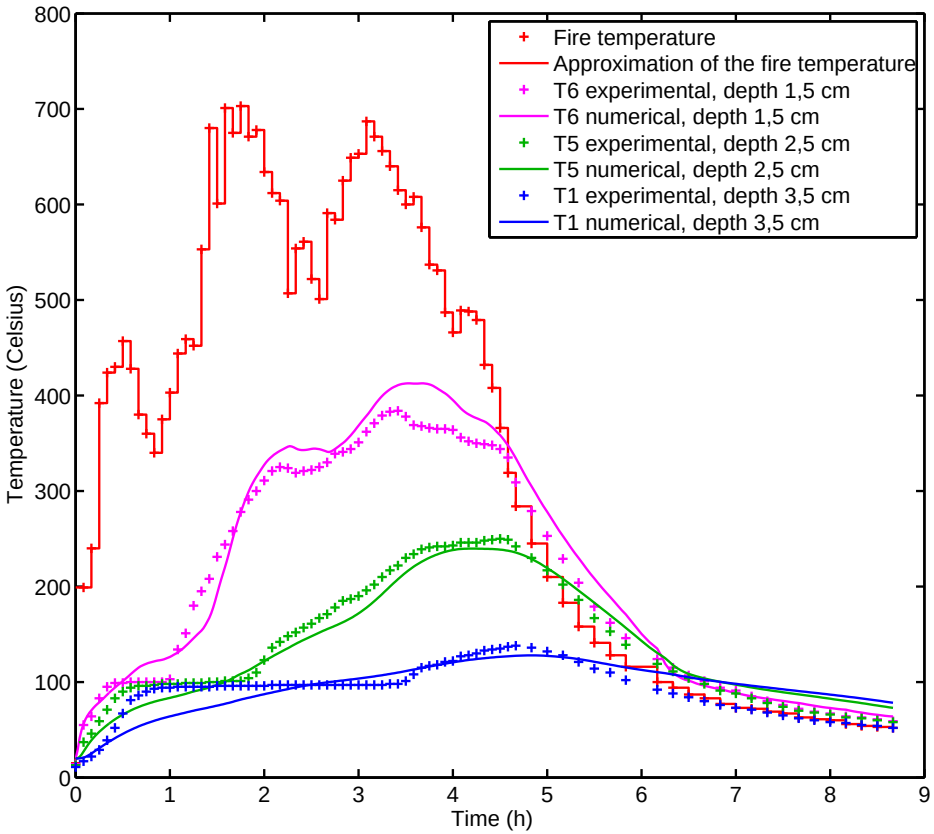
Simulation of a real fire experiment (I)

Experience with a real fire lighted on a limon-clay soil.



Simulation of a real fire experiment (II)

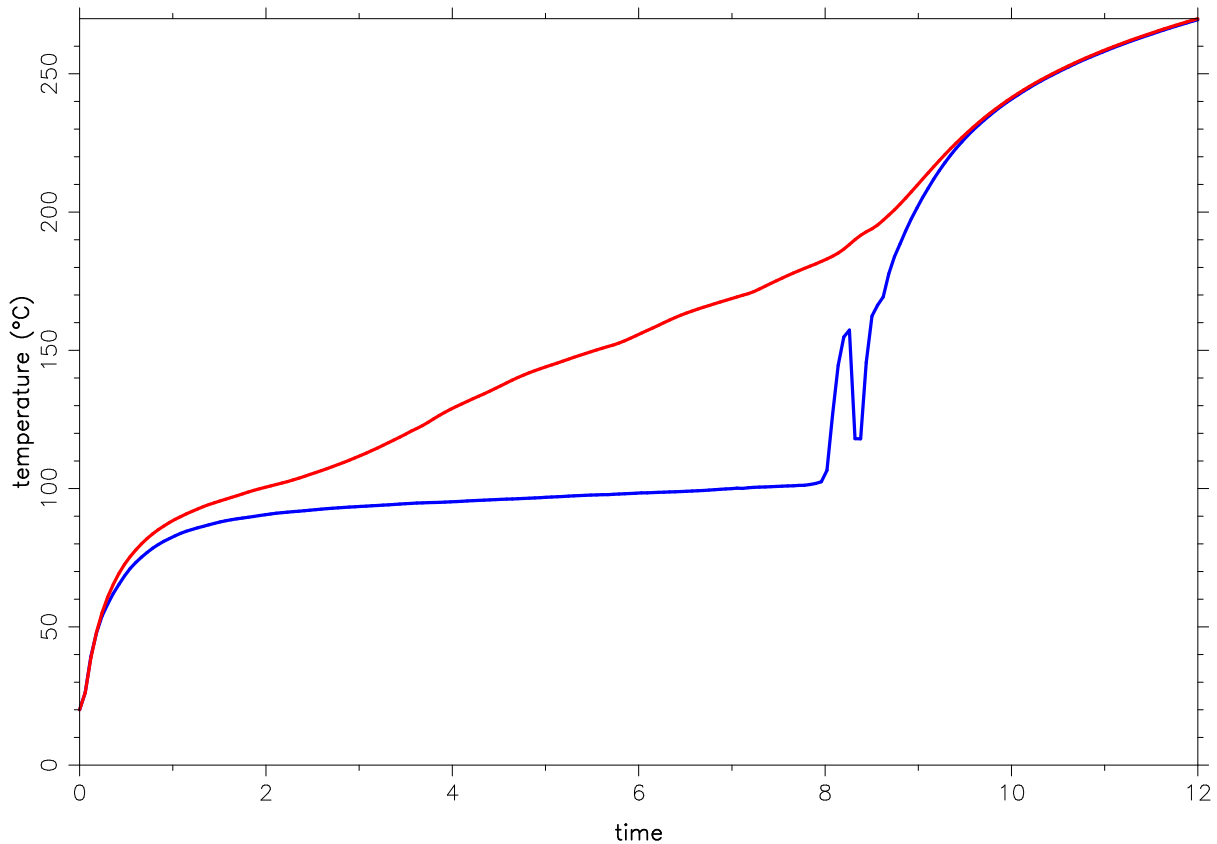
Simulation vs experiment. Domain $0,5\ m \times 0,5\ m$.



Non homogeneous permeability: motivations

- The presence of a long plateau in some temperature curves is a big issue (cf. the last experiment)
  $\rightarrow$ introduction of non-uniform permeability in the soil
- When big contrast in permeability is used (over than 100), the approximated Jacobian computation lead to unstabilities
  $\rightarrow$ exact derivation of the Jacobian, via Maple, with optimized Fortran ouput.

Unstable result from use of approx. jacobian

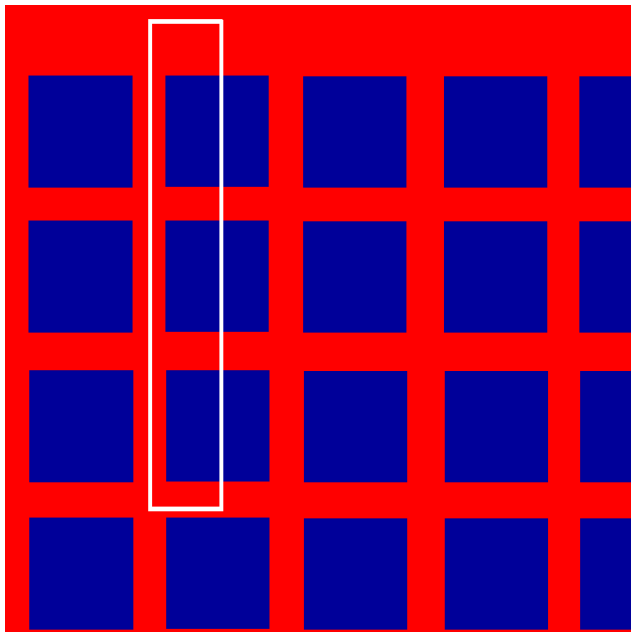


→ Wrong simulation: blue curve must be strictly increasing!

Jacobian derivation via Maple

- Maple source script:
  - definition of vector function: 180 lines
  - jacobian computation: 210 lines
- Generated Fortran code: 1900 lines (in 12 files)
  to be compared to the '`deriv()`' routine: 110 lines
- Perl script for variables substitution: 10 lines
- Performance: *(code 3D-axi; mesh 120x200)*
  - approx. jacobian: binary size = 99.8 kb; CPU = 22.0 s
  - exact jacobian: binary size = 131.6 kb; CPU = 35.8 s

Non homogeneous permeability: illustration



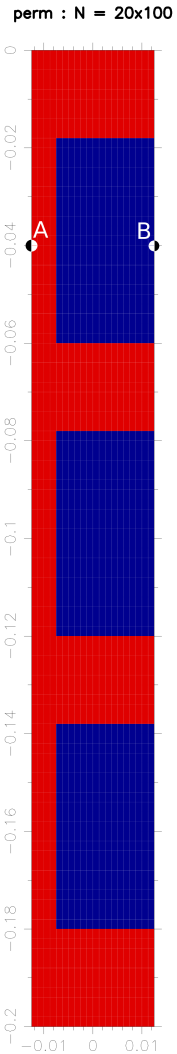
Tiling of square blocks with high-contrast permeability (ratio: 1000)

- typical block size: few millimeters
- double vertical symmetry $\rightarrow$ restricted computational domain

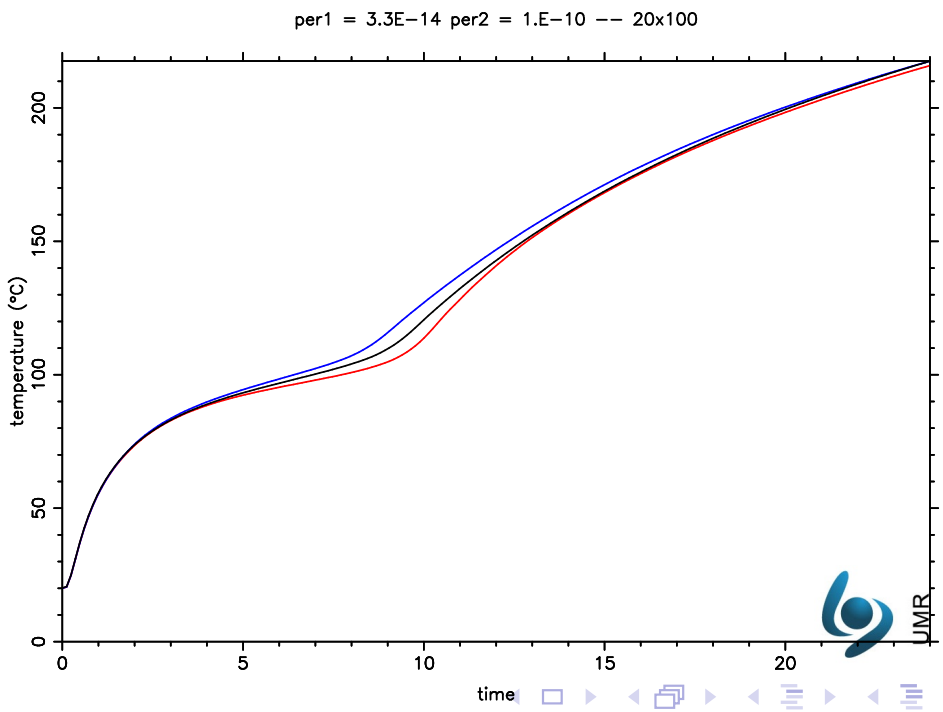
UMR IRISA

Non homogeneous permeability : simulation

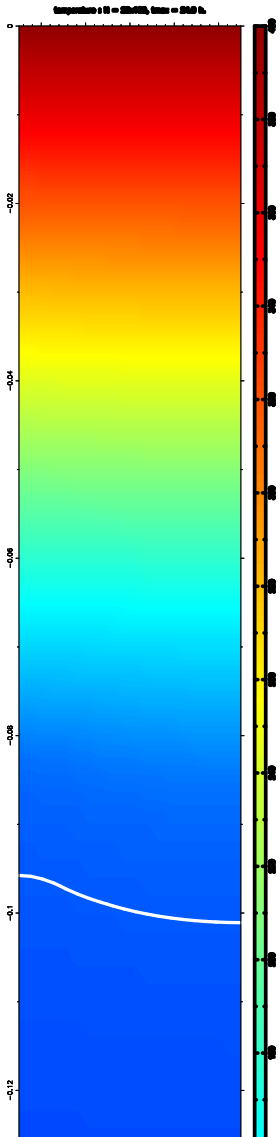
domain size : 2.5 cm $\times$ 20 cm

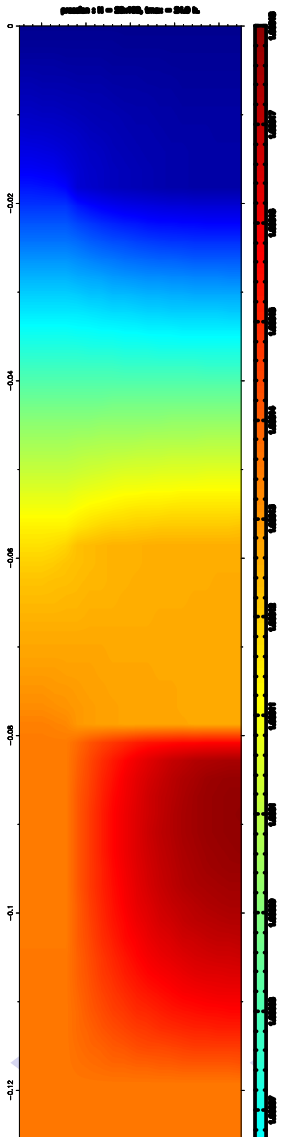


Temperature histories : blue curve (point B), red curve (point A), black curve (homogeneous case)



Temperature and pressure fields





Conclusions and perspectives

- Interesting features are shown in the numerical simulation.
- Global approach is robust.
- Better stability whith exact jacobian matrix (Maple).

- Studying the stability of the model and the sensitivity of the used parameters.
- Extension of this model to the 3D geometry.
- Answering the questions of archaeologists (inverse pb).

UMR IRISA

Associated publications

- Muhieddine M., Canot É., March R., 2012, *Heat transfer modeling in saturated porous media and identification of the thermophysical properties of the soil by inverse problem*, Int. J. for Applied Numer. Math., vol. 62, pp 1026-1040.
- Muhieddine M., Canot É., March R., Delannay R., 2011, *Coupling heat conduction and water-steam flow in a saturated medium*, Int. J. Num. Meth. in Engng, vol. 85, pp 1390-1414.
- Muhieddine M., 2009, *Simulation des structures de combustion préhistoriques*, Thèse de doctorat en informatique, Université de Rennes 1, 16 Octobre.
- Muhieddine M., Canot É., March R., 2009, *Various Approaches for Solving Problems in Heat Conduction with Phase Change*, Int. J. on Finite Volumes, vol. 6, n. 1.

UMR IRISA