GetDP — A general software environment for the treatment of discrete problems

Patrick Dular and Christophe Geuzaine

Department of Electrical Engineering

Montefiore Institute B28, Sart Tilman Campus

University of Liège

B-4000 Liège (BELGIUM)

An environment open to various couplings

Any coupling between

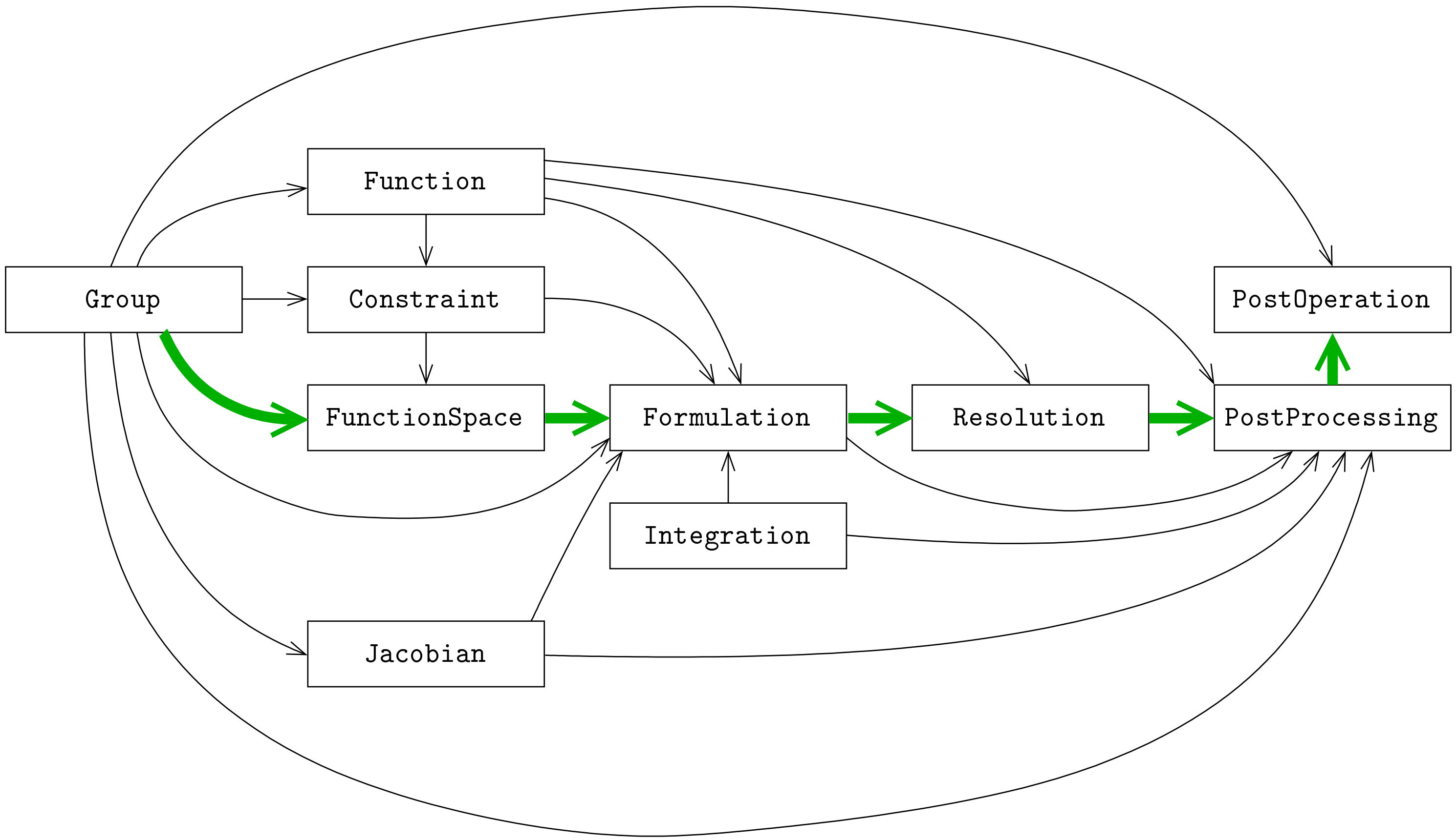
- Physical problems (electromagnetic, thermal, mechanical, ...)
- Numerical methods (finite element methods, integral methods, ...)
- Geometries (1D, 2D, 3D)
- Time states (static, harmonic, transient, eigen values)

How?

- Clear mathematical definitions/structure
- Directly transcribed into 10 interdependent objects

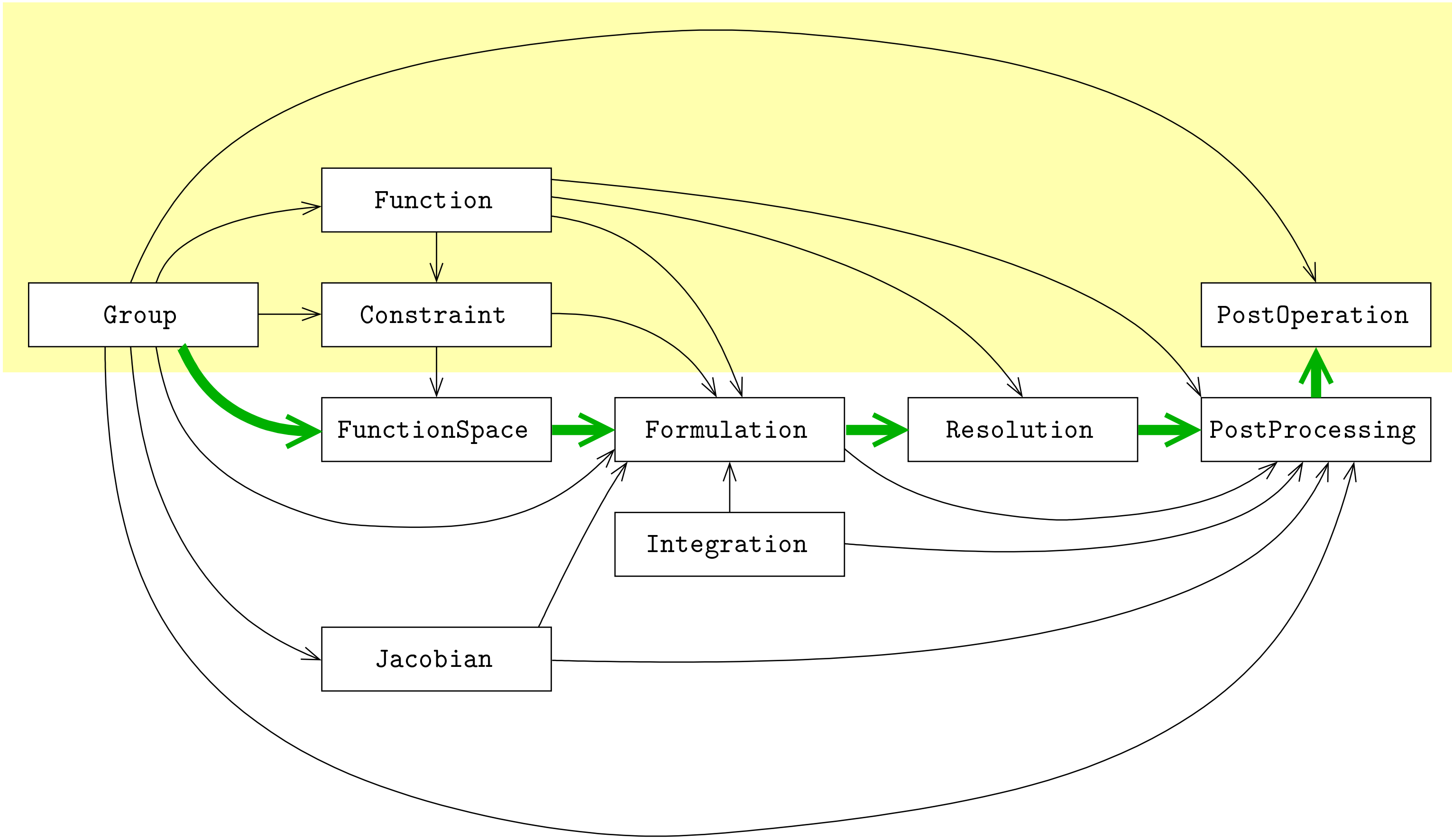
Definition of discrete problems

Copy of the formal mathematical expression of problems in text data files (“.pro files”)



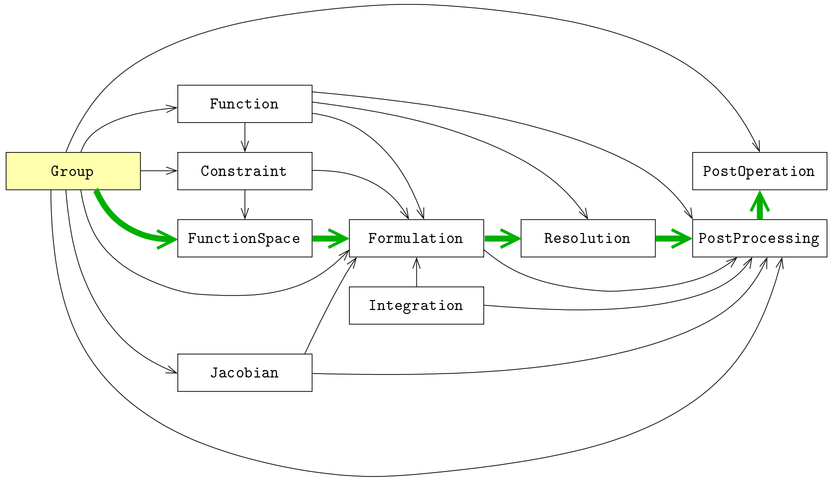
Definition of discrete problems (2)

Particular data of a problem



Method of resolution ("black box")

`Group`: defining topological entities

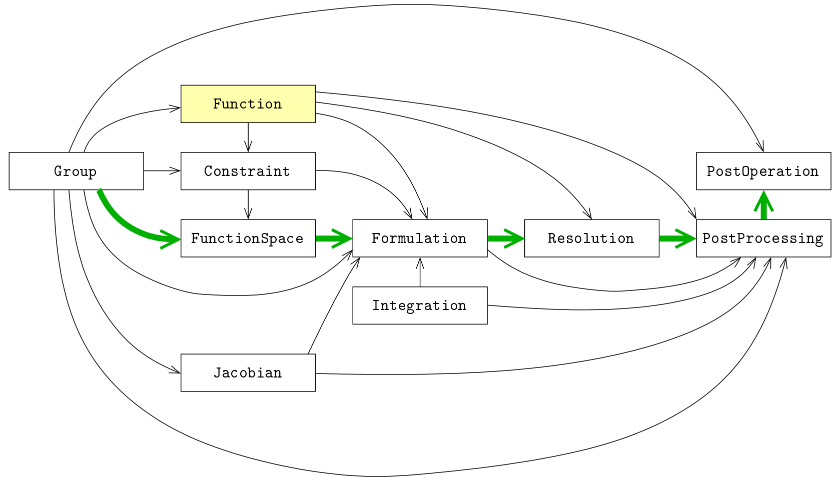


- Regions
- Functions on Regions (nodes, edges, edges of tree, ...)

```
Air = Region[1]; //elementary group (linked with the mesh)
Core = Region[2];
Omega = Region[{Air, Core}];
Nodes = NodesOf[Omega]; //function group
```

`Function`: defining expressions

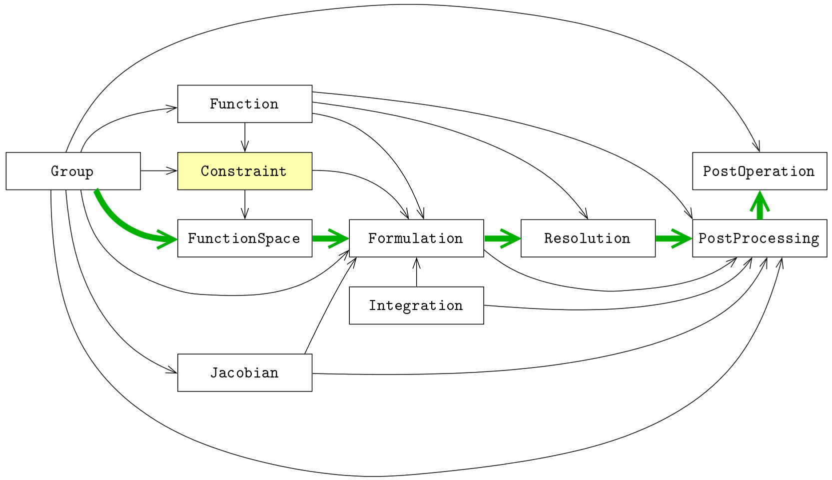
- Piecewise definitions
- Physical characteristics
- Time functions
- Various other functions (natural constraints, ...)




```
mu0 = 4.e-7*Pi; f = 50; //constants
mu[Air] = mu0;
mu[Core] = mu0 + 1/(100+100*$1^6); //argument ($1 <- b)
TimeFct[] = Cos[2*Pi*f*$Time] * Exp[-$Time/0.01]; //current val.
```

`Constraint`: specifying constraints

- Boundary conditions (classical, connection)
- Initial conditions
- Topology of circuits with lumped elements
- Other constraints (on local and global quantities)

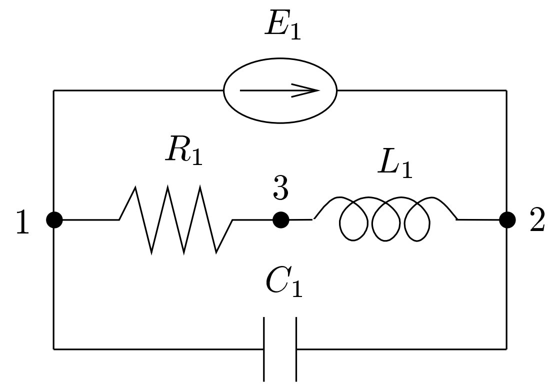



```
{ Name Dirichlet; Type Assign; //boundary conditions
  Case { { Region Surface0; Value 0; }
         { Region Surface1; Value 1; } }
}
```

`Constraint`: specifying constraints (2)

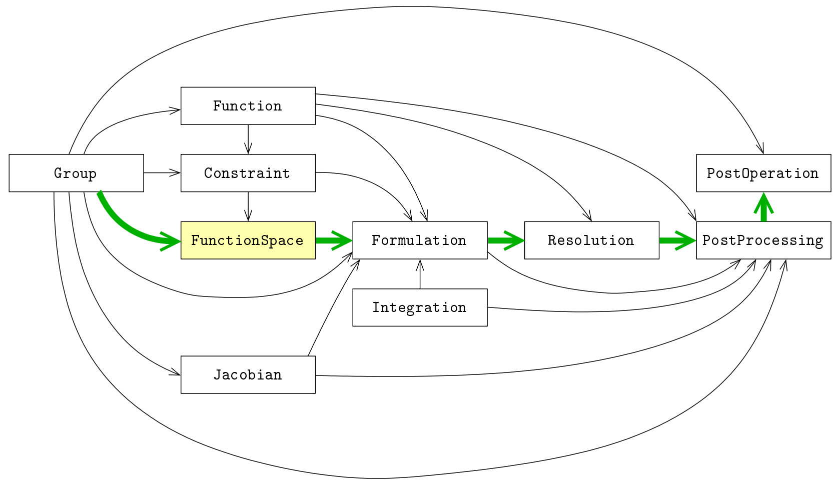
```
{ Name Current; Type Assign; //constraints on global quantities
  Case {
    { Region Inductor1; Value 1000; TimeFunction TimeFct[]; }
  }
}

{ Name ElectricalCircuit; Type Network; //circuit
  Case Circuit1 {
    { Region E1; Branch {1,2}; }
    { Region R1; Branch {1,3}; }
    { Region L1; Branch {3,2}; }
    { Region C1; Branch {1,2}; }
  }
}
```



`FunctionSpace`: **building function spaces**

- Various quantity types (0, 1, 2, 3-forms, scalar, vector)
- Various basis functions (associated with nodes, edges, facets, volumes) of various orders
- Coupling of fields and potentials ($\boldsymbol{t}$-ω, $\boldsymbol{h}$-ϕ, $\boldsymbol{a}$-v, ...)
- Definition of global quantities (fluxes, circulations: current, voltage, m.m.f., ...)
- Essential constraints (boundary and gauge conditions, ...)



Basis functions

$$f(\boldsymbol{x}) = \sum_{i \in E} f_i w_i(\boldsymbol{x})$$

Geometrical entities

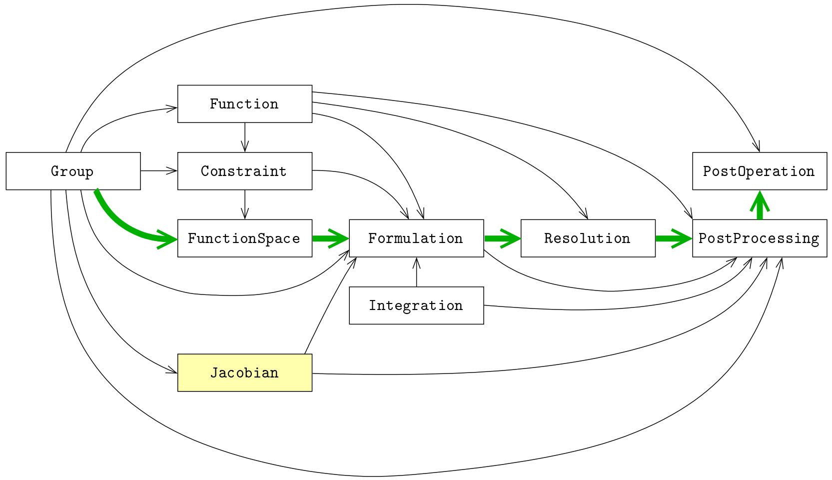
Degrees of freedom

FunctionSpace: building function spaces (2)

```
{ Name H1; Type Form0; //discrete function space for H1
  BasisFunction {
    { Name wi; NameOfCoef fi; Function BF_Node; //order 1
      Support Omega; Entity NodesOf[All]; }
    { Name wi2; NameOfCoef fi2; Function BF_Node_2E;  //order 2
      Support Omega; Entity EdgesOf[All]; }
  }
  Constraint {
    { NameOfCoef fi; //cf. 'Constraint'
      EntityType NodesOf; NameOfConstraint Dirichlet; }
    { NameOfCoef fi2;
      EntityType EdgesOf; NameOfConstraint Dirichlet2; }
  }
}
```

`Jacobian`: defining jacobian methods

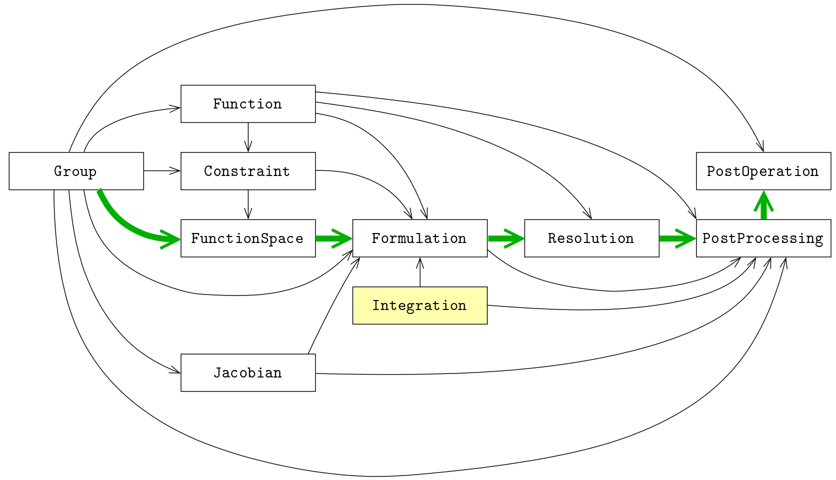
- Mapping from reference to real space
- Geometrical transformations (axisymmetric transformation, infinite domains, ...)




```
{ Name Jacobian1;
  Case { //piecewise defined on groups
    { Region OmegaInf; Jacobian VolSphShell{Rint, Rext}; }
    { Region OmegaAxi; Jacobian VolAxi; }
    { Region All; Jacobian Vol; }
  }
}
```

Integration: defining integration methods

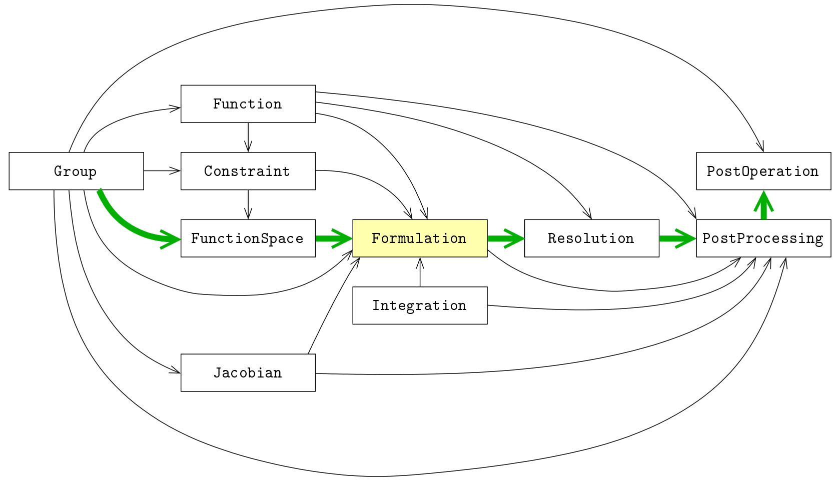
- Various numeric and analytic integration methods
- Criterion-based selection




```
{ Name Integration1; Criterion Test[]; //combination of methods
  Case {
    { Type Gauss;
      Case { { GeoElement Triangle; NumberOfPoints 12; }
             { GeoElement Tetrahedron; NumberOfPoints 15; } } }
    { Type Analytic; }
  }
}
```

`Formulation`: **building equations**

- Various formulation types: FEM, BEM, circuit equations, ...
- Symbolic expression of equations: volume and surface integral terms, collocation
- Involves local, global and integral quantities based on function spaces

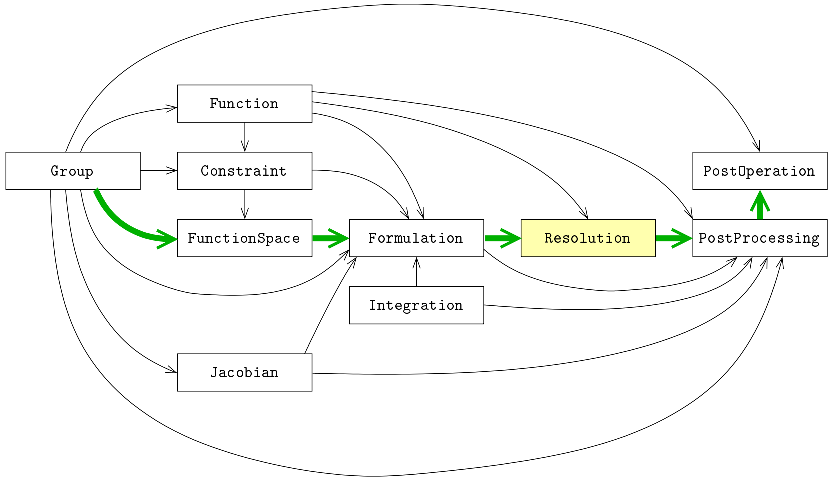


$$\partial_t^2(\epsilon \boldsymbol{e}, \boldsymbol{e}') + (\mu^{-1}\mathbf{curl}\,\boldsymbol{e}, \mathbf{curl}\,\boldsymbol{e}') = 0$$

```
Equation {
  Galerkin { DtDt [ epsilon[] * Dof{e} , {e} ]; ... }
  Galerkin { [ 1/mu[] * Dof{Curl e} , {Curl e} ]; ... }
}
```

`Resolution`: solving systems of equations

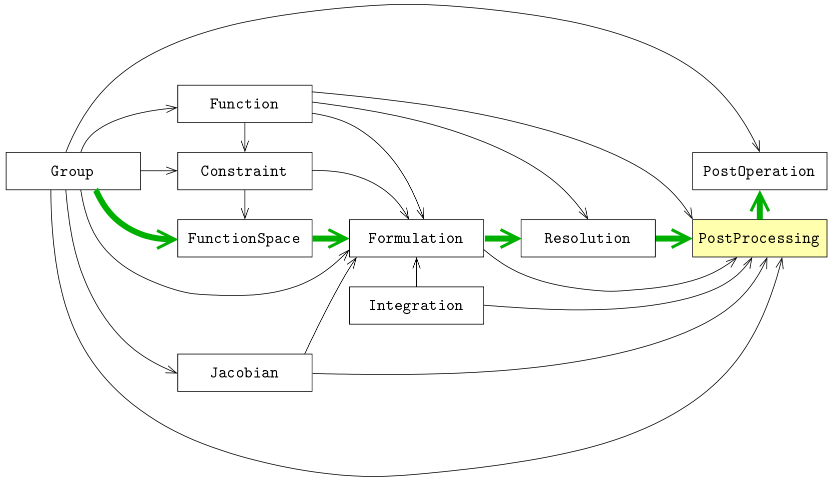
- Description of a sequence of operations
- Time loops (with time step adaptation)
- Nonlinear iterative loops (e.g. fixed point or Newton-Raphson methods)
- Coupled problems (e.g. magneto-thermal coupling)
- Linking of various resolution steps (e.g. pre-computation of source fields)




```
Operation{
  InitSolution[A];
  TimeLoopTheta[tmin,tmax,dt,1]{
    Generate[A]; Solve[A];
    If[Save[]]{ SaveSolution[A]; }
  }
}
```

PostProcessing: exploiting computational data

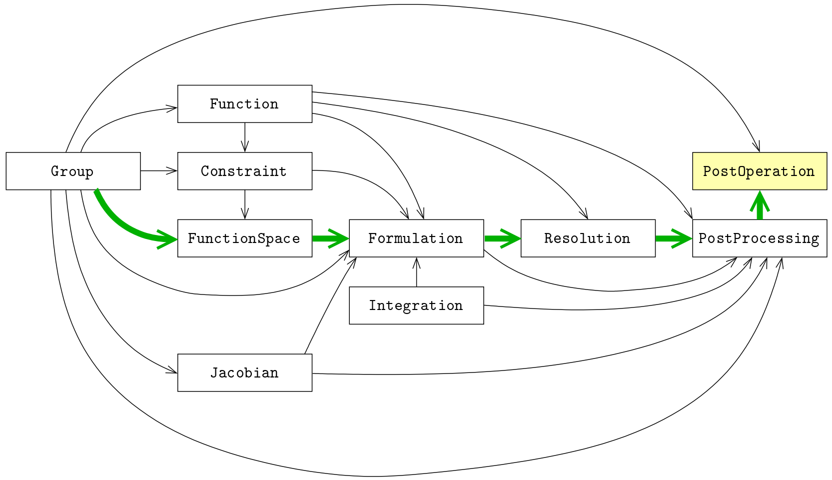
- “Front-end” to computational data
- Piecewise definition of any quantity of interest
- Local or integral evaluation




```
Quantity {
  { Name B;
    Value { Local { [ -mu[]*{Grad phi} ]; In Omega; }
            Local { [ -{dGreen} ]; In Gamma; } }
}
```

PostOperation: exporting results

- Evaluation of post-processing quantities (e.g. maps, sections, local or global evaluation, ...)
- Operations on post-processing quantities (sorting, smoothing, adaptation, ...)
- Various output formats (e.g. space or time oriented, text, binary, ...)




```
Print[ B, OnElementsOf Omega, File "b.pos", Format Gmsh ];
Print[ B, OnLine {{0,0,0}{1,0,0}} {100}, File "b.txt" ];
```

Examples

Magnetostatics

$$\mathbf{curl}\,\boldsymbol{h} = \boldsymbol{j}, \quad \operatorname{div}\boldsymbol{b} = 0 \quad \text{and} \quad \boldsymbol{b} = \mu\boldsymbol{h} + \mu_0\boldsymbol{h}_m$$

$$\begin{tikzcd}
\phi \arrow[r, "\mathbf{grad}_h"] & \boldsymbol{h} \arrow[r, "\mathbf{curl}_h"] \arrow[d, leftrightarrow, "\mu"] & \boldsymbol{j} \arrow[r, "\operatorname{div}_h"] & 0 \\
0 & \boldsymbol{b} \arrow[l, "\operatorname{div}_e"'] & \boldsymbol{a} \arrow[l, "\mathbf{curl}_e"'] &
\end{tikzcd}$$

- Weak form of Gauss' law:

$$(\boldsymbol{b}, \mathbf{grad}\,\phi') + \langle \boldsymbol{n}\cdot\boldsymbol{b}, \phi' \rangle = 0 \quad \forall \phi' \in H_0^1(\Omega)$$

- Weak form of Ampère's law:

$$(\boldsymbol{h}, \mathbf{curl}\,\boldsymbol{a}') + \langle \boldsymbol{n}\times\boldsymbol{h}, \boldsymbol{a}' \rangle = (\boldsymbol{j}, \boldsymbol{a}') \quad \forall \boldsymbol{a}' \in \boldsymbol{H}_0(\mathbf{curl};\Omega)$$

Magnetostatics: b-conforming

Vector potential formulation

$$\boldsymbol{b} = \mathbf{curl}\,\boldsymbol{a}$$

$$\Downarrow$$

Weak form of Ampère's law

$$\Downarrow$$

$$(\mu^{-1}\mathbf{curl}\,\boldsymbol{a}, \mathbf{curl}\,\boldsymbol{a}') = (\boldsymbol{j}, \boldsymbol{a}'), \quad \forall \boldsymbol{a}' \in \boldsymbol{H}_0(\mathbf{curl}; \Omega)$$

NB: gauge for $\boldsymbol{a}$, ...

Magnetostatics: b-conforming (2)

```
Group {
  Core = #1; Inductor = #2; SkinInductor = #3, Air = #4;
  Omega = Region[{Core, Inductor, Air}];
}
Function {
  mu0 = 4.e-7 * Pi; mur = 1000;
  mu[ Core ] = mur * mu0;
  mu[ Region[{Air, Inductor}] ] = mu0;
  j[ Inductor ] = ...; //to be defined
}
Constraint {
  { Name a;
    Case {
      { Region CL_a0; Value 0; }
    }
  }
}
```

Magnetostatics: b-conforming (3)

```
FunctionSpace {
  { Name Hcurl; Type Form1; //vector potential
    BasisFunction {
      { Name se;  NameOfCoef ae; Function BF_Edge; Support Omega;
        Entity EdgesOf[All]; } //associated with the edges of the mesh
    }
    Constraint { //essential constraint + gauge (unicity)
      { NameOfCoef ae;  EntityType EdgesOf; NameOfConstraint a; }
      { NameOfCoef ae;  EntityType EdgesOfTreeIn;
        EntitySubType StartingOn; NameOfConstraint Gauge; }
    }
  }
}
```

Magnetostatics: b-conforming (4)

```
Formulation {
  { Name MagSta_a; Type FemEquation;
    Quantity {
      { Name a; Type Local; NameOfSpace Hcurl; }
    }
    Equation {
      Galerkin { [ 1/mu[] * Dof{Curl a} , {Curl a} ];
                 In Omega; Integration I1; Jacobian JVol; }
      Galerkin { [ -j[] , {a} ];
                 In Inductor; Integration I1; Jacobian JVol; }
    }
  }
}
```

Magnetostatics: b-conforming (5)

```
Resolution {
  { Name MagSta_a;
    System {
      { Name A; NameOfFormulation MagSta_a; }
    }
    Operation { Generate[A]; Solve[A]; SaveSolution[A]; }
  }
}
PostProcessing {
  { Name test; NameOfFormulation MagSta_a;
    Quantity {
      { Name a; Value { Local{ [ {a} ]; In Omega; } } }
      { Name normb; Value { Local{ [ Norm[{d a}] ]; Omega; } } }
    }
  }
}
```

Magnetostatics: b-conforming (6)

Magnetodynamics?

Additional term in the formulation:

```
Galerkin { DtDof [ sigma[] * Dof{a} , {a} ];
           In Core; Integration I1; Jacobian JVol; }
```

New resolution:

```
{ Name MagDyn_a_t; //time domain
  System {
    { Name A; NameOfFormulation MagDyn_a; }
  }
  Operation {
    InitSolution[A]
    TimeLoopTheta[0,20/50,0.1/50,1] { //tmin,tmax,dt,theta
      Generate[A]; Solve[A]; SaveSolution[A];
    }
  }
}
```

Magnetostatics: h-conforming

Magnetic field conforming formulation

$$\boldsymbol{h} = \boldsymbol{h}_s + \boldsymbol{h}_r, \quad \text{with} \quad \mathbf{curl}\, \boldsymbol{h}_s = \boldsymbol{j} \quad \text{and} \quad \boldsymbol{h}_r = -\mathbf{grad}\, \phi$$

$$\Downarrow$$

Weak form of Gauss law

$$\Downarrow$$

$$(\mu(-\mathbf{grad}\, \phi + \boldsymbol{h}_s), \mathbf{grad}\, \phi') = 0, \quad \forall \phi' \in H_0^1(\Omega)$$

NB: choice of source field $\boldsymbol{h}_s$, treatment of multiply connected Ω, ...

Magnetostatics: h-conforming (2)

```
FunctionSpace {
  { Name H1; Type Form0; //scalar potential
    BasisFunction {
      { Name sn; NameOfCoef phin; Function BF_Node; Support Omega;
        Entity NodesOf[All]; } //associated with the nodes of Omega
    }
    Constraint { //essential constraint
      { NameOfCoef phin; EntityType NodesOf; NameOfConstraint phi; }
    }
  }
}
```

Magnetostatics: h-conforming (3)

```
Formulation {
  { Name MagSta_phi; Type FemEquation;
    Quantity {
      { Name phi; Type Local; NameOfSpace H1; }
      { Name hs; Type Local; NameOfSpace Hcurl_s; } //patience...
    }
    Equation {
      Galerkin { [ mu[] * {hs} , {Grad phi} ];
                 In Omega; Integration I1; Jacobian JVol;  }
      Galerkin { [ mu[] * Dof{Grad phi} , {Grad phi} ];
                 In Omega; Integration I1; Jacobian JVol;  }
    }
  }
}
```

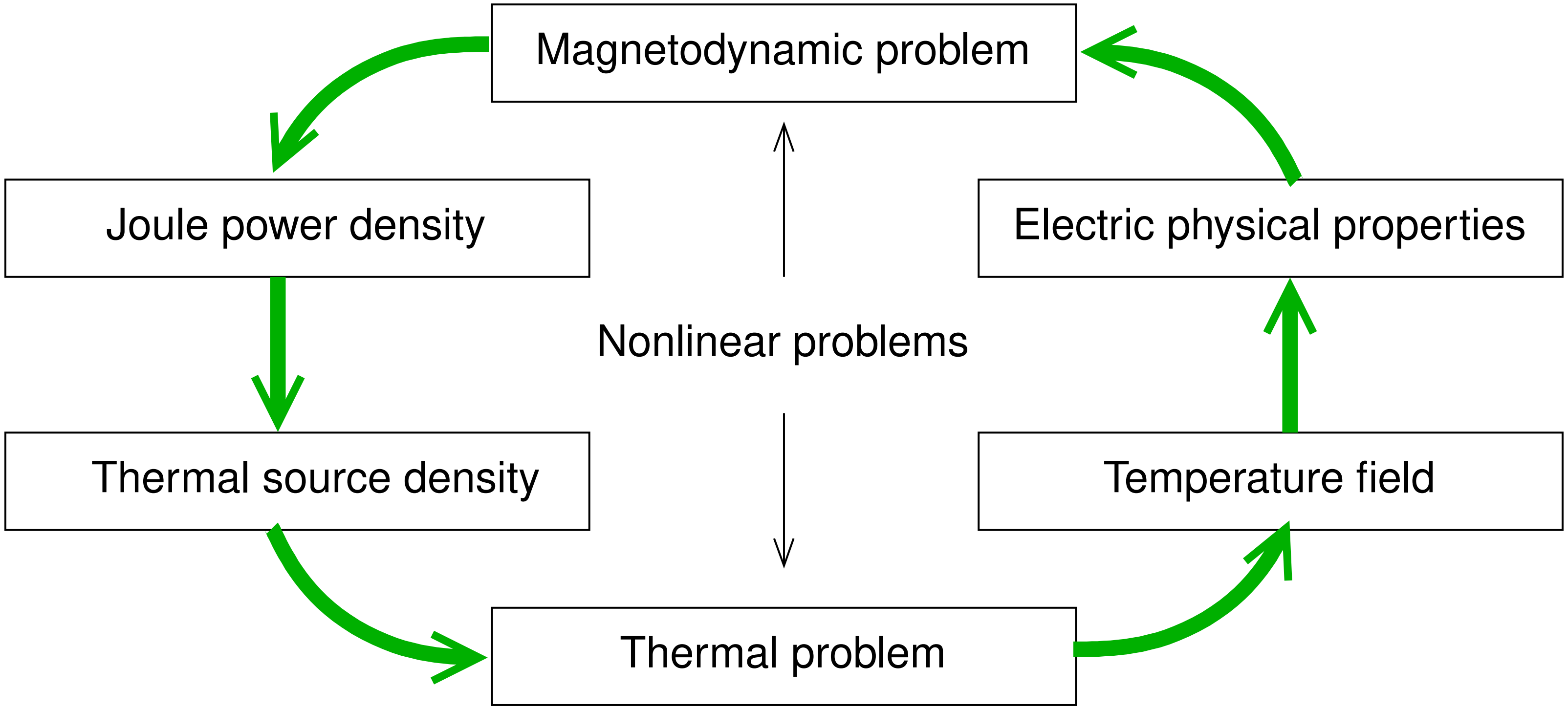
Magnetostatics: h-conforming (4)

```
FunctionSpace {
  { Name Hcurl_s; Type Form1; //space for the source field
    BasisFunction {
      { Name se; NameOfCoef he; Function BF_Edge; Support Inductor;
        Entity EdgesOf[All, Not SkinInductor]; }
      { Name sc; NameOfCoef Ic; Function BF_GradGroupOfNodes;
        Support Transition; Entity GroupsOfNodesOf[Cut]; }
      { Name sc; NameOfCoef Icc; Function BF_GroupOfEdges;
        Support Inductor; Entity ...; }
    }
    Constraint {
      { NameOfCoef he; EntityType EdgesOfTreeIn;
        EntitySubType StartingOn; NameOfConstraint Gauge; }
      { NameOfCoef Ic; EntityType GroupsOfNodesOf; NameOfConstraint I; }
      { NameOfCoef Icc; EntityType GroupsOfNodesOf; NameOfConstraint I; }
    }
  }
}
```

Magnetostatics: h-conforming (5)

```
Formulation {
  { Name MagSta_hs; Type FemEquation;
    Quantity {
      { Name hs; Type Local; NameOfSpace Hcurl_hs; }
    }
    Equation {
      Galerkin { [ Dof{Curl hs} , {Curl hs} ];
                   In Omega; Integration I1; Jacobian JVol; }
      Galerkin { [ -j[] , {d hs} ];
                   In Omega; Integration I1; Jacobian JVol; }
    }
  }
}
```

Magneto-thermal coupling: step by step



Magneto-thermal coupling: step by step (2)

Magnetodynamic formulations

- Adapted function spaces for the fields and potentials involved
- Boundary conditions
- Electric circuit coupling, prescribed currents or voltages
- Nonlinear magnetic characteristics

e.g.

$$(\mu^{-1}\mathbf{curl}\,\boldsymbol{a}, \mathbf{curl}\,\boldsymbol{a}')_{\Omega} + (\sigma\partial_t\boldsymbol{a}, \boldsymbol{a}')_{\Omega_c} + (\sigma\mathbf{grad}\,v, \boldsymbol{a}')_{\Omega_c} = 0, \quad \forall\boldsymbol{a}' \in \boldsymbol{H}_0(\mathbf{curl};\Omega)$$

$$\partial_t(\mu\boldsymbol{h}, \boldsymbol{h}')_{\Omega} + (\sigma^{-1}\mathbf{curl}\,\boldsymbol{h}, \mathbf{curl}\,\boldsymbol{h}')_{\Omega_c} = 0, \quad \forall\boldsymbol{h}' \in \boldsymbol{H}_0(\mathbf{curl};\Omega)$$

Magneto-thermal coupling: step by step (3)

Thermal formulation

- For example temperature T formulation
- Essential boundary conditions for T
- Natural boundary conditions for convection and radiation heat flows
- Nonlinear thermal characteristics

$$(\kappa\,\mathbf{grad}\,T, \mathbf{grad}\,T')_\Omega - (\rho c_p\,\partial_t T, T')_\Omega + (p_q, T')_\Omega$$

$$+\langle \eta (T - T_0), T' \rangle_{\Gamma_{\text{conv}}} + \langle \epsilon \sigma_s (T^4 - T_0^4), T' \rangle_{\Gamma_{\text{rad}}} = 0\,, \quad \forall T' \in H_0^1(\Omega)$$

Magneto-thermal coupling: step by step (4)

Movement of regions

- Addition of transport term (e.g. modified Ohm's law: $\boldsymbol{j} = \sigma(\boldsymbol{e} + \boldsymbol{v} \times \boldsymbol{b})$)

e.g.

$$-(\sigma \boldsymbol{v} \times \mathbf{curl}\, \boldsymbol{a}, \boldsymbol{a}')_{\Omega_v}$$

$$-(\mu \boldsymbol{v} \times \boldsymbol{h}, \mathbf{curl}\, \boldsymbol{h}')_{\Omega_v}$$

e.g.

$$-(\rho c_p \boldsymbol{v} \cdot \mathbf{grad}\, T, T')_{\Omega_v}$$

Magneto-thermal coupling: step by step (5)

Magneto-thermal coupling

- Heat source term $p_q = \frac{1}{2}\sigma^{-1}j^2$
- Temperature dependent electric and magnetic characteristics $\mu(T)$ and $\sigma(T)$

e.g.

$$j = \sigma \|\partial_t \boldsymbol{a} + \mathbf{grad}\, v\|$$

$$j = \|\mathbf{curl}\, \boldsymbol{h}\|$$

Magneto-thermal coupling: step by step (6)

Resolutions

- Magnetodynamic resolution in time or frequency domain
- Thermal resolution in steady state or in time domain

```
Resolution { //magnetodynamic freq + thermal static
  { Name Magnetothermal_h_T;
    System {
      { Name Mag; NameOfFormulation MagDyn_h; Frequency 50; }
      { Name The; NameOfFormulation The_T }
    }
    Operation {
      IterativeLoop[16,1.e-4,1] { //max_its, stop, relax
        GenerateJac[Mag]; SolveJac[Mag]; GenerateJac[The]; SolveJac[The];
      }
      SaveSolution[Mag]; SaveSolution[The];
    }
  } }
```

Conclusions

Conclusions

- Structured and concise software environment
  - Numerical modeling of physical problems (but no physics inside the code!)
  - User-friendly language for the definition of methods and problems
  - Convenient for research, application or teaching
  - Shared for collaboration (freely available on the Internet, documentation, mailing lists)

Conclusions (2)

- Adapted for stage-by-stage use
  - From 1D to 3D models
  - From linear to nonlinear problems
  - From static to dynamic problems
  - From single physical or numerical models to coupled ones
- Open to new developments
  - Open for adding new functionalities in a progressive way
  - Interactive use possible through Gmsh or home-made application
  - Integration into higher level optimization packages

Conclusions (3)

For more information, please visit http://www.geuz.org/getdp/