On the use of DG methods for optimization and uncertainty estimation: CAD-based features and sensitivity analysis

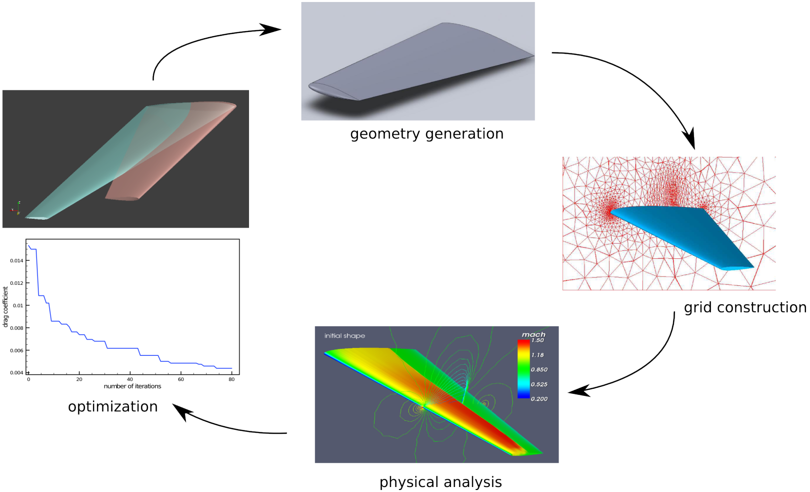
R. Duvigneau

INRIA Sophia-Antipolis Méditerranée, Acumes Team, France

Nachos Seminar, May 2016

CAD features & DG methods

Context: optimum design



Geometrical representations for optimum design

Co-existence of different representations in a typical design loop

- CAD-based description: high-order NURBS (geometry definition)
- mesh-based representation: piecewise linear (PDE solvers)
- ad-hoc parameters: heterogeneous (optimization)

Consequences :

- Difficulty to build et deform grids automatically
- Projections yielding a loss of accuracy
- Introduction of a geometrical error in PDE solvers
- More complex sensitivity analysis
- More complex coupled problems, with moving bodies, refinement, etc

Geometrical representations for optimum design

Co-existence of different representations in a typical design loop

- CAD-based description: high-order NURBS (geometry definition)
- mesh-based representation: piecewise linear (PDE solvers)
- ad-hoc parameters: heterogeneous (optimization)

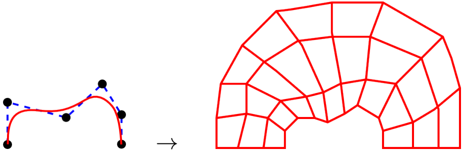
Consequences :

- Difficulty to build et deform grids automatically
- Projections yielding a loss of accuracy
- Introduction of a geometrical error in PDE solvers
- More complex sensitivity analysis
- More complex coupled problems, with moving bodies, refinement, etc

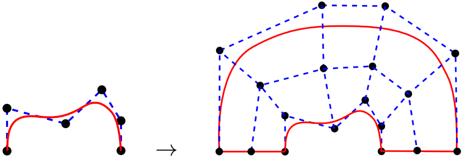
Isogeometric approach

Change of paradigm[1] : solve PDEs on parametric domains

Classical approch:



Isogeometric approch:



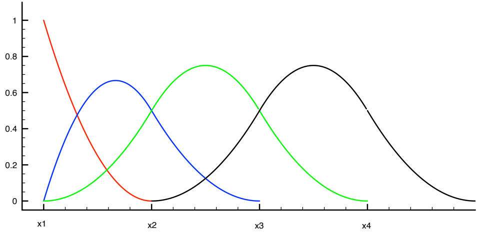
Objective : a unique high-order geometrical representation

1. [Hughes, Cottrell, Bazilevs, *Comp. Meth. Appl. Mech. Eng.* 2005]

CAD bases

B-Spline basis functions $\hat{N}_i^p$ of degree p:

- Defined recursively using a knot vector $[\xi_1, \ldots, \xi_k]$
- Piecewise-polynomials of degree p
- Regularity C^{p-m} at each knot of multiplicity m
- Compact supports $[\xi_i, \xi_{i+p+1})$



NURBS (Non-Uniform Rational Basis Spline) functions:

- Rational extension to represent conic shapes

Curves, surfaces and volumes

- Parametric curves:

$$\boldsymbol{P}(\xi) = (x(\xi), y(\xi), z(\xi)) = \sum_{i=1}^{n} \hat{N}_i^p(\xi)\boldsymbol{P}_i$$

- Parametric surfaces:

$$\boldsymbol{P}(\xi,\eta) = (x(\xi,\eta), y(\xi,\eta), z(\xi,\eta)) = \sum_{i=1}^{n_i}\sum_{j=1}^{n_j} \hat{N}_i^{p_i}(\xi)\,\hat{N}_j^{p_j}(\eta)\,\boldsymbol{P}_{ij}$$

- Parametric volumes:

$$\boldsymbol{P}(\xi,\eta,\zeta) = (x(\xi,\eta,\zeta), y(\xi,\eta,\zeta), z(\xi,\eta,\zeta)) = \sum_{i=1}^{n_i}\sum_{j=1}^{n_j}\sum_{k=1}^{n_k} \hat{N}_i^{p_i}(\xi)\,\hat{N}_j^{p_j}(\eta)\,\hat{N}_k^{p_k}(\zeta)\,\boldsymbol{P}_{ijk}$$

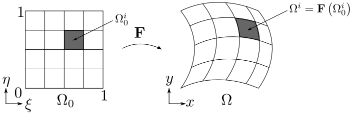
Principles

- Computational domain as a parametric surface (2D) / volume (3D) :

$$\boldsymbol{P}(\xi,\eta) = (x(\xi,\eta), y(\xi,\eta)) = \sum_{i=1}^{n_i} \sum_{j=1}^{n_j} \hat{N}_i^{p_i}(\xi)\, \hat{N}_j^{p_j}(\eta)\, \boldsymbol{P}_{ij}$$

- Non-linear transformation from parametric to physical space:

$$\mathbf{F}: \begin{array}{ccc} \Omega_0 & \rightarrow & \Omega \\ \boldsymbol{\xi} = (\xi,\eta) & \mapsto & \mathbf{x} = (x,y) \end{array}$$



- Construction of analysis-aware domains[2,3] necessary
  - Injectivity preservation
  - Maximize regularity / orthogonality

2. [Xu, Mourrain, Duvigneau, Galligo, Comp. Aided Design 2012]
3. [Xu, Mourrain, Duvigneau, Galligo, J. Comp. Phys. 2013]

Principles of isogeometric analysis

- Definition of the solution space with the same basis (possibly refined) :

$$\Theta(\xi,\eta) = \sum_{i=1}^{n'_i} \sum_{j=1}^{n'_j} \hat{N}_i^{p'_i}(\xi)\, \hat{N}_j^{p'_j}(\eta)\, \Theta_{ij}$$

- Variational formulation to determine degrees of freedom = control points
- Visualization of the solution as a parametric surface / volume

Application to an elliptic problem

- Variational formulation of a heat conduction problem:

$$-\int_{\Omega} \kappa(\mathbf{x}) \boldsymbol{\nabla} T(\mathbf{x}) \cdot \boldsymbol{\nabla} \psi(\mathbf{x}) \, d\Omega + \int_{\partial \Omega_N} \Phi_0(\mathbf{x}) \, \psi(\mathbf{x}) \, d\Gamma = 0 \quad \forall \psi$$

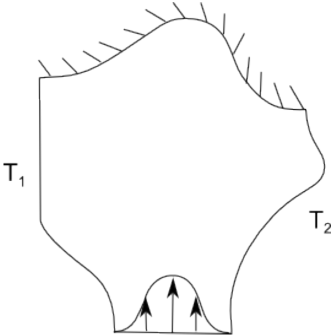
- Discretization yields the linear system $MT = S$:

$$M_{ij,kl} = \int_{\Omega_0} \kappa(\mathbf{F}(\boldsymbol{\xi})) \boldsymbol{\nabla}_{\boldsymbol{\xi}} \hat{N}_{kl}(\boldsymbol{\xi}) \; B(\boldsymbol{\xi})^{\top} \; B(\boldsymbol{\xi}) \; \boldsymbol{\nabla}_{\boldsymbol{\xi}} \hat{N}_{ij}(\boldsymbol{\xi}) \; J(\boldsymbol{\xi}) \; d\hat{\Omega}$$

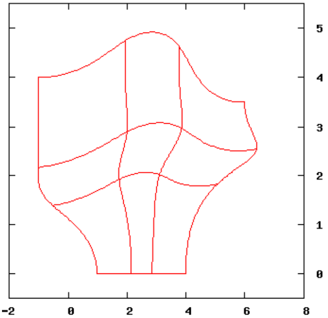
$$S_{ij} = \int_{\partial \Omega_{0\,N}} \Phi_0(\mathbf{F}(\boldsymbol{\xi})) \; \hat{N}_{ij}(\boldsymbol{\xi}) \; J(\boldsymbol{\xi}) \; d\hat{\Gamma}$$

- Integration in parametric space using classical quadrature rules
- Inversion by conjugate gradient

Simple illustration

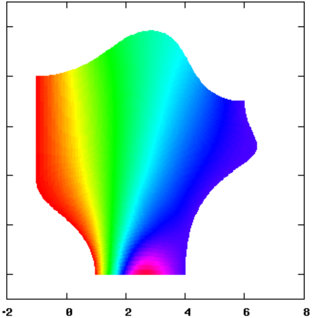


Problem description

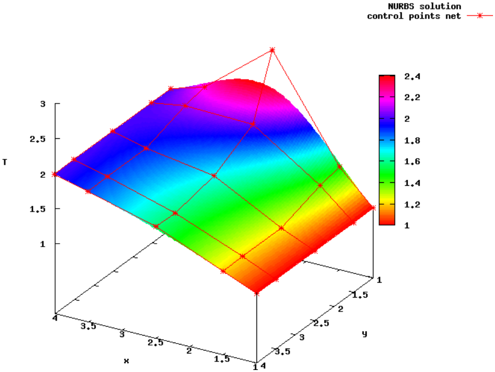


Computational domain
6×6 control points
3×3 knot intervals

Simple illustration

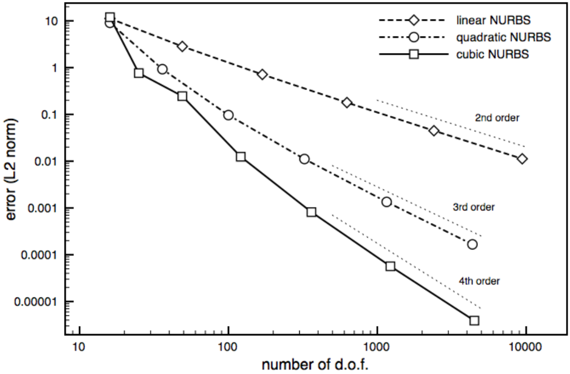


Solution field
in physical space



Solution surface
in parametric space

Convergence study



Computational efficiency[4]

Degree	d.o.f.	Error	CPU (s)
linear	49	$0.28\ 10^{1}$	0.004
linear	625	$0.17\ 10^{0}$	0.072
linear	2401	$0.45\ 10^{-2}$	0.911
linear	9409	$0.11\ 10^{-2}$	12.150
quadratic	36	$0.91\ 10^{0}$	0.004
quadratic	324	$0.11\ 10^{-1}$	0.030
quadratic	1156	$0.13\ 10^{-2}$	0.252
quadratic	4356	$0.16\ 10^{-3}$	2.832
cubic	49	$0.24\ 10^{0}$	0.007
cubic	361	$0.81\ 10^{-3}$	0.078
cubic	1225	$0.56\ 10^{-4}$	0.461
cubic	4489	$0.38\ 10^{-5}$	4.509

4. [Duvigneau, Inria Research Report 6957, 2009]

Application to a hyperbolic system

- Classical conservation law:

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x} f(u) = 0$$

- Variational formulation with SUPG stabilization:

$$\int_\Omega \psi(\mathbf{x})\dot{u}(\mathbf{x})\, d\Omega - \int_\Omega \psi_{,x}(\mathbf{x}) f(u(\mathbf{x}))\, d\Omega + [\psi(\mathbf{x}) f(u(\mathbf{x}))]_{\partial\Omega}$$
$$+ \sum_{k=1}^{n_{el}} \int_{\Omega^k} \left(\psi_{,x}(\mathbf{x}) \frac{\partial f}{\partial u}\right) \tau \left(\frac{\partial f(u)}{\partial x}\right) d\Omega = 0 \quad \forall \psi$$

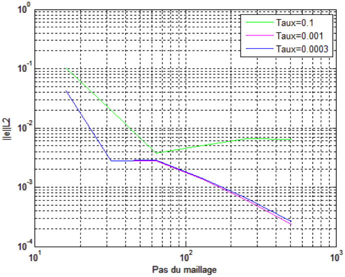
- Integration in parametric space (e.g. SUPG term for a 2D problem) :

$$\int_{\Omega_0} \left[(\hat{N}_{ij\,,\xi} \frac{\partial \xi}{\partial x} + \hat{N}_{ij\,,\eta} \frac{\partial \eta}{\partial x}) \frac{\partial f^1}{\partial u}(\boldsymbol{\xi}) + (\hat{N}_{ij\,,\xi} \frac{\partial \xi}{\partial y} + \hat{N}_{ij\,,\eta} \frac{\partial \eta}{\partial y}) \frac{\partial f^2}{\partial u}(\boldsymbol{\xi}) \right]$$
$$\tau \left[\frac{\partial f^1}{\partial u}(\boldsymbol{\xi})(u_{,\xi} \frac{\partial \xi}{\partial x} + u_{,\eta} \frac{\partial \eta}{\partial x}) + \frac{\partial f^2}{\partial u}(\boldsymbol{\xi})(u_{,\xi} \frac{\partial \xi}{\partial y} + u_{,\eta} \frac{\partial \eta}{\partial y}) \right] J(\boldsymbol{\xi}) d\Omega$$

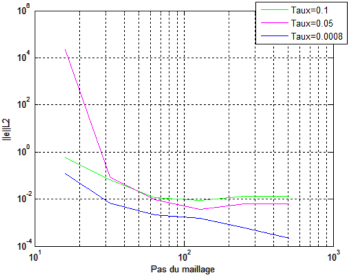
- Runge-Kutta time integration
- τ computed as a characteristic time $\alpha \frac{\Delta x}{c}$

Convergence study

- Linear 1D case
- Strong dependency w.r.t. α stabilization parameter



quadratic B-Spline



Cubic B-Spline

Synthesis

Conclusion regarding isogeometric analysis methods

- Very appealing from conceptual point of view
- More complex to implement
- Local refinement issues (T-Splines)
- Seems to be efficient for elliptic problems
- Tedious for hyperbolic problems

Questions

- Could DG methods handle CAD-based geometries ?
- How ?

Synthesis

Conclusion regarding isogeometric analysis methods

- Very appealing from conceptual point of view
- More complex to implement
- Local refinement issues (T-Splines)
- Seems to be efficient for elliptic problems
- Tedious for hyperbolic problems

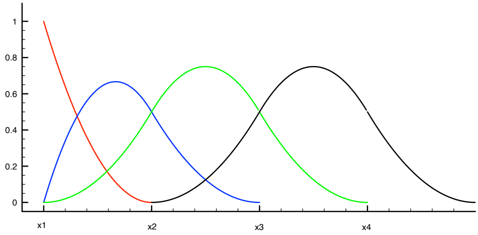
Questions

- Could DG methods handle CAD-based geometries ?
- How ?

Generation of a basis suitable for DG methods

Overview of the problem

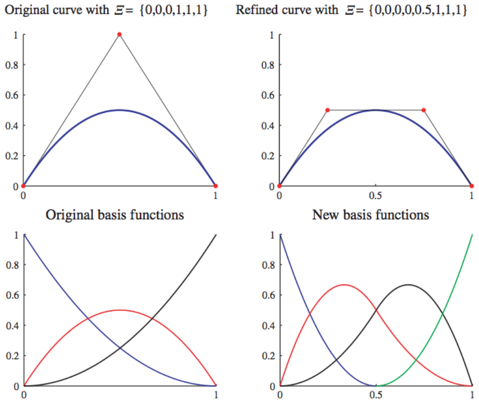
- Start from a B-Spline (of NURBS) definition of the boundary
- Construct a boundary basis suitable for DG without altering the geometry
- Extent to a surface / volume computational domain



Basis transformation

Knot insertion procedure

- A knot can be inserted without modifying the B-Spline / NURBS curve



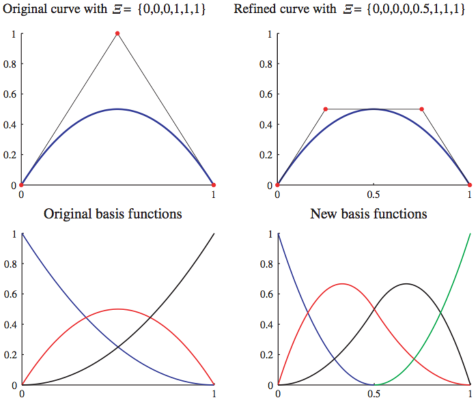
Regularity

- A B-Spline / NURBS curve is C^{p-m} where m is the knot multiplicity

Basis transformation

Knot insertion procedure

- A knot can be inserted without modifying the B-Spline / NURBS curve



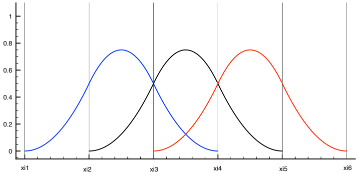
Regularity

- A B-Spline / NURBS curve is C^{p-m} where m is the knot multiplicity

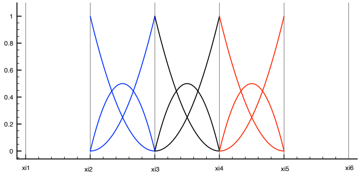
Basis transformation

Generation of a discontinuous basis

- By inserting p knots at existing knots, a discontinuous basis is generated
- The B-Spline / NURBS curve is changed into a set of Bezier / rational Bezier curves
- Geometry unchanged

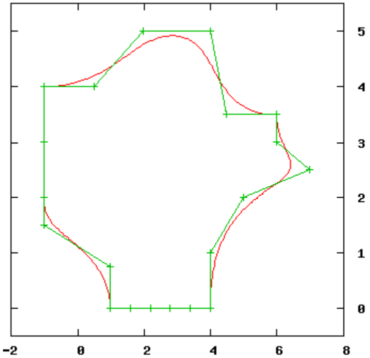


original B-Spline basis



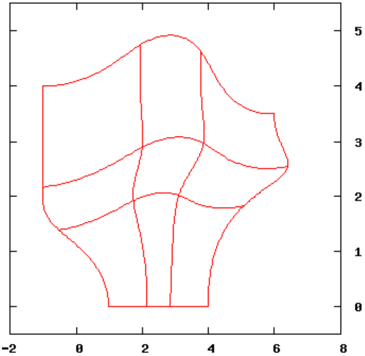
discontinuous Bernstein bases

Illustration



Cubic B-Spline boundaries

knots : $[0\ 0\ 0\ 0\ \frac{1}{3}\ \frac{2}{3}\ 1\ 1\ 1\ 1]$



3×3 Bezier elements

Synthesis

- B-Spline / NURBS basis can be transformed to a set of discontinuous Bernstein / rational Bernstein basis
- A computational domain based on Bezier / rational Bezier elements can be generated:
  - by tensor product $\rightarrow$ structured grid (straightforward for simple problems)
  - triangular or tetrahedral Bezier / rational Bezier grid (not straightforward)
- Note:
  - Bernstein / rational Bernstein basis only required at the boundary
  - Bernstein basis can be transformed to Lagrange basis

DG based on Bernstein basis

Problem

- Unsteady viscous Burgers equation:

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x} f(u) = 0 \quad f(u) = \frac{u^2}{2} - \nu \frac{\partial u}{\partial x}$$

- Initial solution:

$$u_0(x) = \frac{a+b}{2} - \frac{a-b}{2} \tanh\left((a-b)\frac{x}{4\nu}\right)$$

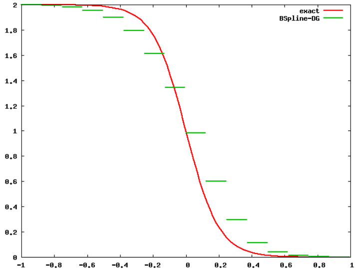
Numerical methods

- Classical DG formulation:

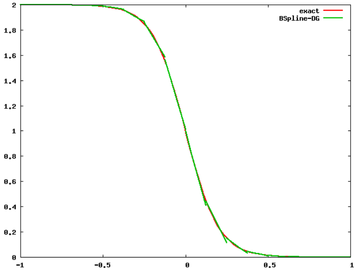
$$\int_{I_j} \frac{\partial u_h(x,t)}{\partial t} v_h(x)\, dx = \int_{I_j} f(u_h(x,t)) \frac{\partial v_h(x)}{\partial x}\, dx + f^\star(x_j^l, t) - f^\star(x_j^r, t)$$

- Local Lax-Friedrichs flux for convective part ; LDG approach for diffusive part
- Explicit RK4 time integration
- Gauss-legendre quadratures
- Bezier representation for u_h
- Least-squares approximation for initial condition

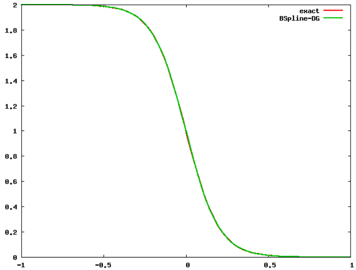
Solution (16 elements)



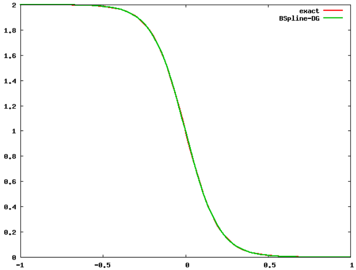
degree 0



degree 1

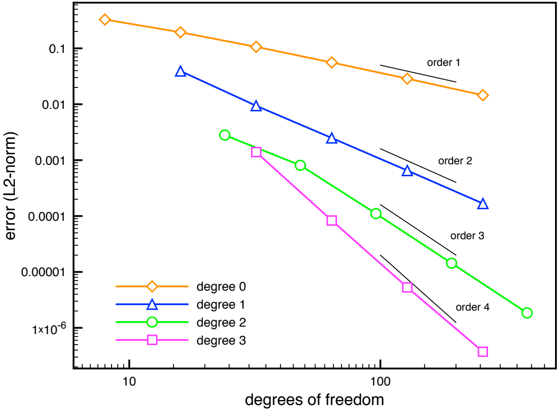


degree 2

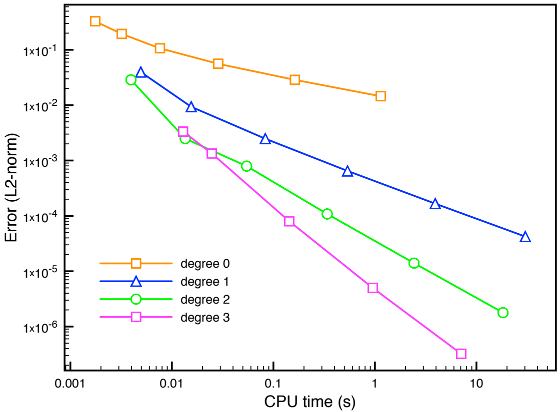


degree 3

Solution accuracy



Accuracy vs CPU time



Synthesis: proposed approach

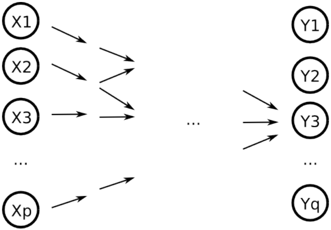
- Transform B-Spline / NURBS boundaries into a set of Bezier / rational Bezier curves by multiple knot insertion
- Generate Bezier elements by tensor product
- Solve PDE system using DG based on Bernstein basis

Extension to 3D Navier-Stokes in progress !

Sensitivity analysis & DG methods

Sensitivity analysis

- For PDE systems, sensitivity analysis refers to the derivative of an output quantity w.r.t. an input variable
- Mainly used for optimization: evaluate the gradient of a cost functional w.r.t. design parameters
- Preferably use adjoint equation method (independent from design parameters)

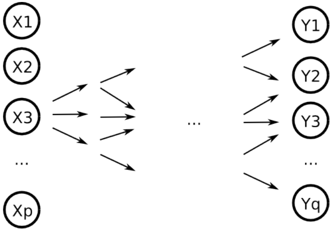


Limitations of adjoint equation method

- Equation dependent on the output of interest
- For unsteady systems, requires storage of unsteady solution for backward time integration
- Restricted to (some) functionals

Alternative: sensitivity equation method

- Obtained by simply differentiating state equations w.r.t. input variables
- Allows to evaluate sensitivity of the whole solution fields $u^{(\alpha)} = \frac{\partial u}{\partial \alpha}$
- Forward time integration
- Several purposes:
  - Optimization
  - Exploration of neighboring solutions
  - Uncertainty propagation
- But equation dependent on the input variable
- Easy parallelization



Continuous vs discrete

- Discretize then differentiate:
  - Consistent with discrete PDE solutions
  - Requires to differentiate discrete quantities (mesh, limiters, etc)

- Differentiate then discretize:
  - More flexible: allows to choose a different numerical scheme, mesh, etc.
  - Non consistent with discrete PDE solutions (for a given mesh)

Sensitivity-based design optimization

- Optimization based on descent methods (steepest-descent, Newton, etc)
- Sensitivity field is used to compute the gradient of the cost function of interest:
  - Inverse problems :

$$J(\alpha) = \frac{1}{2} \int_{\Omega} |u(x) - u^{\star}(x)|^2 dx$$

with $u^{\star}$ target solution

$$\frac{\partial J}{\partial \alpha} = \int_{\Omega} \left(u(x) - u^{\star}(x)\right) u^{(\alpha)}(x) dx$$

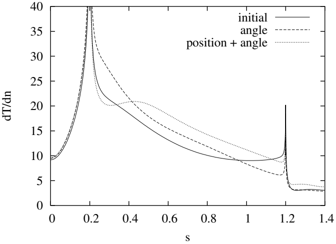
  - Boundary integral :

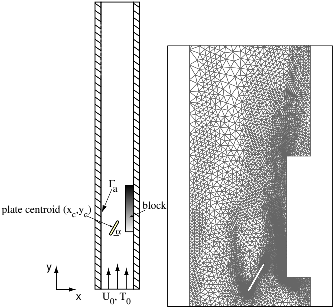
$$J(\alpha) = \frac{1}{2} \int_{\Gamma} \nabla u(s) \cdot \vec{n} ds$$

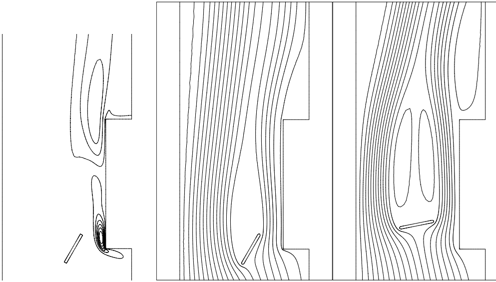
$$\frac{\partial J}{\partial \alpha} = \int_{\Gamma} \nabla u^{(\alpha)}(s) \cdot \vec{n} + \nabla u(s) \cdot \vec{n}^{(\alpha)} ds$$

Application to design optimization

- Forced convection (laminar Navier-Stokes)
- Finite-element analysis adapted to flow and sensitivities
- Shape parameters[3]: location x and y, incidence α







3. [Duvigneau & Pelletier *Num. Heat Transfer* 2006]

Sensitivity-based uncertainty propagation

- We consider a (first-order) Taylor expansion of the quantity g around the expectation value of the uncertain variable α:

$$g(\alpha) = g|_{\mu_\alpha} + \left.\frac{\partial g}{\partial \alpha}\right|_{\mu_\alpha} (\alpha - \mu_\alpha) + O(\delta\alpha^2)$$

- The Taylor expansion is used for a first-order approximation of the variance :

$$\sigma_g^2 = \int_{\Omega_a} g(\alpha)^2 \rho(\alpha) d\alpha - \mu_g^2$$

$$\sigma_g^2 \approx g|_{\mu_\alpha}^2 \underbrace{\int_{\Omega_\alpha} \rho(\alpha) d\alpha}_{=1} + \left.\frac{\partial g}{\partial \alpha}\right|_{\mu_\alpha}^2 \underbrace{\int_{\Omega_\alpha} (\alpha - \mu_\alpha)^2 \rho(\alpha) d\alpha}_{=\sigma_\alpha^2} +$$

$$2g|_{\mu_\alpha} \left.\frac{\partial g}{\partial \alpha}\right|_{\mu_\alpha} \underbrace{\int_{\Omega_\alpha} (\alpha - \mu_\alpha) \rho(\alpha) d\alpha}_{=0} - \mu_g^2$$

$$\sigma_g^2 \approx \left.\frac{\partial g}{\partial \alpha}\right|_{\mu_\alpha}^2 \sigma_\alpha^2$$

Extension to several uncertain parameters and higher order

- For *n* independent Gaussian variables, one obtains:

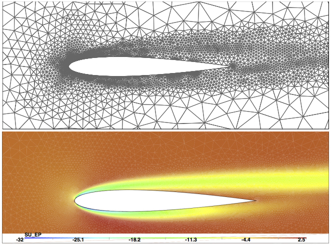
$$\mu_g \approx g(\mu_\alpha) + \frac{1}{2} \sum_{i=1}^{n} \left. \frac{\partial^2 g}{\partial \alpha_i^2} \right|_{\mu_\alpha} \sigma_{\alpha_i}^2$$

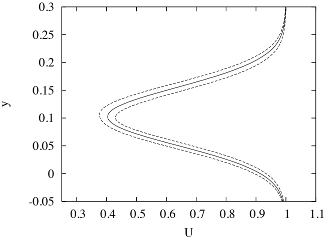
$$\sigma_g^2 \approx \sum_{i=1}^{n} \left. \frac{\partial g}{\partial \alpha_i} \right|_{\mu_\alpha}^2 \sigma_{\alpha_i}^2 + \frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \left. \frac{\partial^2 g}{\partial \alpha_i \partial \alpha_k} \right|_{\mu_\alpha}^2 \sigma_{\alpha_i}^2 \sigma_{\alpha_k}^2$$

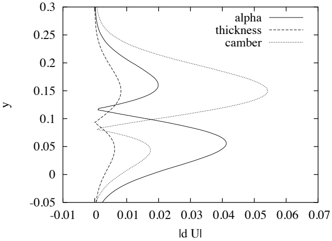
- Extensions to correlated non-Gaussian variables exist.

Application to uncertainty estimation

- Airfoil NACA 0012 ($Re = 2000$)
- Finite-element analysis adapted to flow and sensitivities
- Shape uncertainty[3]: thickness (1%), incidence (0.5°), camber (1%)







3. [Duvigneau & Pelletier *Int. J. Comp. Fluid Dyn.* 2006]

Sensitivity equation method

Problem

- Unsteady viscous Burger equation:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2} \quad \forall (x,t) \in [x_L, x_R] \times [0, T]$$

- Initial solution:

$$u(x,0) = u_0(x) \quad \forall x \in [x_L, x_R]$$

- Boundary condition:

$$u(x_L,t) = u_L(t) \quad u(x_R,t) = u_R(t) \quad \forall t \in [0, T]$$

Sensitivity equation method

Problem

- Conservative form:

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x} f(u) = 0 \quad \forall (x,t) \in [x_L, x_R] \times [0, T]$$

with:

$$f(u) = \frac{u^2}{2} - \nu \frac{\partial u}{\partial x}$$

- First-order system form (LDG approach $q = \sqrt{\nu}\ \frac{\partial u}{\partial x}$):

$$\boxed{\begin{array}{ll} \frac{\partial}{\partial t} u + \frac{\partial}{\partial x} f = 0 & f = \frac{u^2}{2} - \sqrt{\nu} q \\ q + \frac{\partial}{\partial x} g = 0 & g = -\sqrt{\nu} u. \end{array}}$$

Sensitivity equation method

Principle of the method

- Sensitivity variable:

$$u^{(\alpha)} = \frac{\partial u}{\partial \alpha}$$

- Formal differentiation of state equation w.r.t. α:

$$\frac{\partial}{\partial \alpha}\left(\frac{\partial u}{\partial t}\right) + \frac{\partial}{\partial \alpha}\left(u\frac{\partial u}{\partial x}\right) = \frac{\partial}{\partial \alpha}\left(\nu\frac{\partial^2 u}{\partial x^2}\right) \quad \forall (x,t) \in [x_L, x_R] \times [0, T]$$

- By switching derivatives with respect to α and x or t:

$$\frac{\partial u^{(\alpha)}}{\partial t} + u^{(\alpha)}\frac{\partial u}{\partial x} + u\frac{\partial u^{(\alpha)}}{\partial x} = \nu\frac{\partial^2 u^{(\alpha)}}{\partial x^2} + \nu^{(\alpha)}\frac{\partial^2 u}{\partial x^2} \quad \forall (x,t) \in [x_L, x_R] \times [0, T]$$

- Initial condition for sensitivity:

$$u^{(\alpha)}(x,0) = u_0^{(\alpha)}(x) \quad \forall x \in [x_L, x_R]$$

- Boundary condition for sensitivity:

$$u^{(\alpha)}(x_L,t) = u_L^{(\alpha)}(t) \quad u^{(\alpha)}(x_R,t) = u_R^{(\alpha)}(t) \quad \forall t \in [0, T]$$

Sensitivity equation method

Principle of the method

- First-order system form (LDG approach $q^{(\alpha)} = \sqrt{\nu}\ \frac{\partial u^{(\alpha)}}{\partial x} + \frac{\nu^{(\alpha)}}{2\sqrt{\nu}}\ \frac{\partial u}{\partial x}$):

$$
\boxed{
\begin{array}{ll}
\frac{\partial}{\partial t} u^{(\alpha)} + \frac{\partial}{\partial x} f^{(\alpha)} = 0 & f^{(\alpha)} = u u^{(\alpha)} - \sqrt{\nu} q^{(\alpha)} - \frac{\nu^{(\alpha)}}{2\sqrt{\nu}} q \\
q^{(\alpha)} + \frac{\partial}{\partial x} g^{(\alpha)} = 0 & g^{(\alpha)} = -\sqrt{\nu} u^{(\alpha)} - \frac{\nu^{(\alpha)}}{2\sqrt{\nu}} u.
\end{array}
}
$$

Sensitivity equation method

Principle of the method

One has to solve the extended system:

$$\frac{\partial}{\partial t} w + \frac{\partial}{\partial x} \phi(w) = 0$$

For the extended variables and fluxes:

$$w = \begin{pmatrix} u \\ q \\ u^{(\alpha)} \\ q^{(\alpha)} \end{pmatrix} \qquad \phi = \begin{pmatrix} f \\ g \\ f^{(\alpha)} \\ g^{(\alpha)} \end{pmatrix}$$

Sensitivity equation method

Some properties

- Same type of PDE system as original problem (e.g. hyperbolic)
- Sensitivity system has:
  - the same flux Jacobian matrix: $f^{(\alpha)} = \frac{\partial f(u)}{\partial \alpha} = \frac{\partial f}{\partial u}\frac{\partial u}{\partial \alpha} = \frac{\partial f}{\partial u} u^{(\alpha)}$
  - the same eigenvalues
  - the same eigenvectors

  Consequences:
  - same stability conditions
  - same time-marching approach
  - same implicit part if an implicit scheme is used

High-order sensitivity

Principle of the method

- One introduces a couple of parameters α_1, α_2 and second-order sensitivities:

$$u^{(\alpha_1,\alpha_2)} = \frac{\partial^2 u}{\partial \alpha_1 \partial \alpha_2}$$

- Second-order sensitivity system:

$$\boxed{\begin{aligned}
&\frac{\partial}{\partial t} u^{(\alpha_1,\alpha_2)} + \frac{\partial}{\partial x} f^{(\alpha_1,\alpha_2)} = 0 \\
&\quad f^{(\alpha_1,\alpha_2)} = u u^{(\alpha_1,\alpha_2)} + u^{(\alpha_2)} u^{(\alpha_1)} \\
&\qquad\qquad - \sqrt{\nu} q^{(\alpha_1,\alpha_2)} - \frac{\nu^{(\alpha_2)}}{2\sqrt{\nu}} q^{(\alpha_1)} - \frac{\nu^{(\alpha_1)}}{2\sqrt{\nu}} q^{(\alpha_2)} - \frac{\nu^{(\alpha_1,\alpha_2)}}{4\sqrt{\nu}^3} q \\
&\quad q^{(\alpha_1,\alpha_2)} + \frac{\partial}{\partial x} g^{(\alpha_1,\alpha_2)} = 0 \\
&\quad g^{(\alpha_1,\alpha_2)} = -\sqrt{\nu} u^{(\alpha_1,\alpha_2)} - \frac{\nu^{(\alpha_2)}}{2\sqrt{\nu}} u^{(\alpha_1)} - \frac{\nu^{(\alpha_1)}}{2\sqrt{\nu}} u^{(\alpha_2)} - \frac{\nu^{(\alpha_1,\alpha_2)}}{4\sqrt{\nu}^3} u
\end{aligned}}$$

High-order sensitivity

Principle of the method

One has to solve the extended system:

$$\frac{\partial}{\partial t} w + \frac{\partial}{\partial x} \phi(w) = 0$$

For the extended variables and fluxes:

$$w = \begin{pmatrix} u \\ q \\ u^{(\alpha_1)} \\ q^{(\alpha_1)} \\ u^{(\alpha_2)} \\ q^{(\alpha_2)} \\ u^{(\alpha_1,\alpha_1)} \\ q^{(\alpha_1,\alpha_1)} \\ u^{(\alpha_1,\alpha_2)} \\ q^{(\alpha_1,\alpha_2)} \\ u^{(\alpha_2,\alpha_2)} \\ q^{(\alpha_2,\alpha_2)} \end{pmatrix} \qquad \phi = \begin{pmatrix} f \\ g \\ f^{(\alpha_1)} \\ g^{(\alpha_1)} \\ f^{(\alpha_2)} \\ g^{(\alpha_2)} \\ f^{(\alpha_1,\alpha_1)} \\ g^{(\alpha_1,\alpha_1)} \\ f^{(\alpha_1,\alpha_2)} \\ g^{(\alpha_1,\alpha_2)} \\ f^{(\alpha_2,\alpha_2)} \\ g^{(\alpha_2,\alpha_2)} \end{pmatrix}$$

$\rightarrow$ parallel solving strategy required for efficiency !

Numerical resolution

DG method

- Classical DG formulation:

$$\int_{I_j} \frac{\partial w_h(x,t)}{\partial t} v_h(x)\, dx = \int_{I_j} \phi(w_h(x,t)) \frac{\partial v_h(x)}{\partial x}\, dx + \phi^\star(x_j^l, t) - \phi^\star(x_j^r, t)$$

- Local Lax-Friedrichs flux for convective part ; LDG approach for diffusive part
- Explicit RK4 time integration
- Gauss-legendre quadratures
- Bezier representation for w_h
- Least-squares approximation for initial conditions
- One code ligne to add for each sensitivity ! (flux expression)

Test problem

Problem definition

- Unsteady viscous Burger equation:

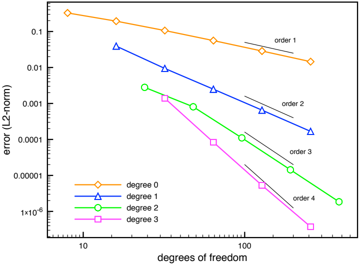
$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2} \quad \forall (x,t) \in [-1,1] \times [0,0.5]$$

- Exact solution:

$$u(x,t) = \frac{a+b}{2} - \frac{a-b}{2} \tanh\left((a-b)\frac{x - \frac{1}{2}(a+b)t}{4\nu} \right)$$

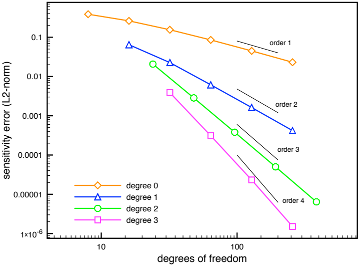
- Two sensitivity parameters : ν (diffusion coef.) and a (value at $-\infty$)

Solution accuracy

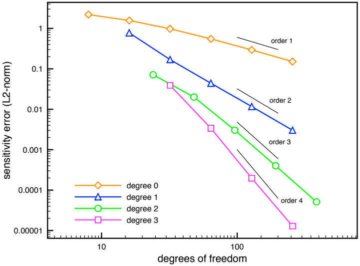


error for u

First-order sensitivity accuracy

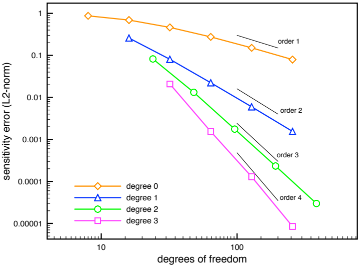


error for $u^{(a)}$

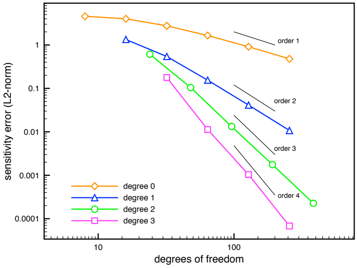


error for $u^{(\nu)}$

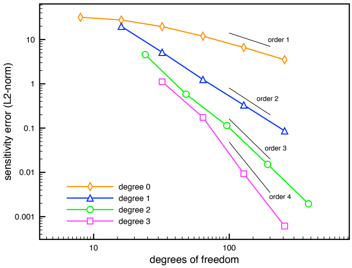
Second-order sensitivity accuracy



error for $u^{(a,a)}$

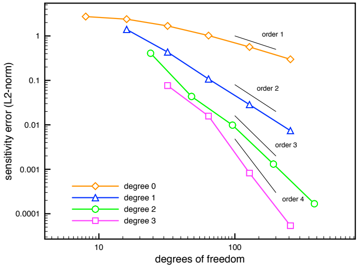


error for $u^{(a,\nu)}$

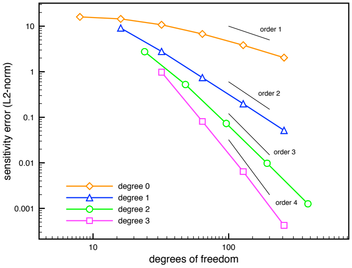


error for $u^{(\nu,\nu)}$

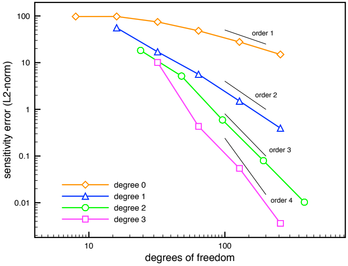
Third-order sensitivity accuracy



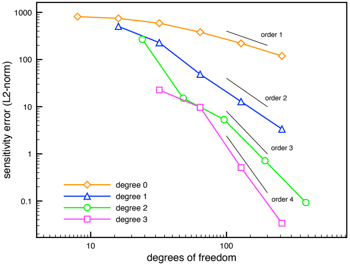
error for $u^{(a,a,a)}$



error for $u^{(a,a,\nu)}$

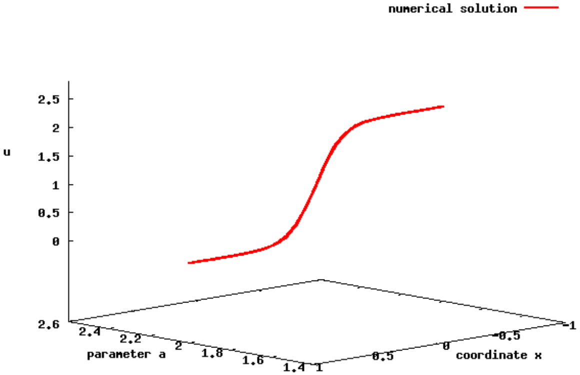


error for $u^{(a,\nu,\nu)}$



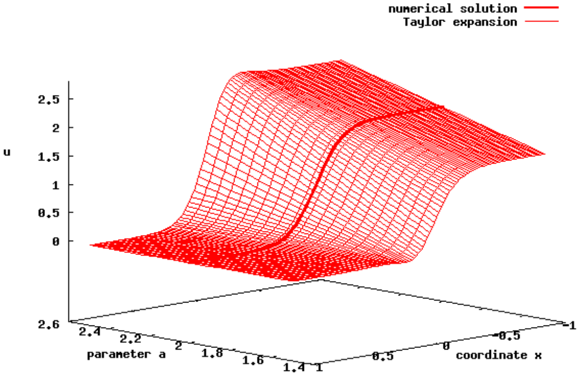
error for $u^{(\nu,\nu,\nu)}$

A simple illustration



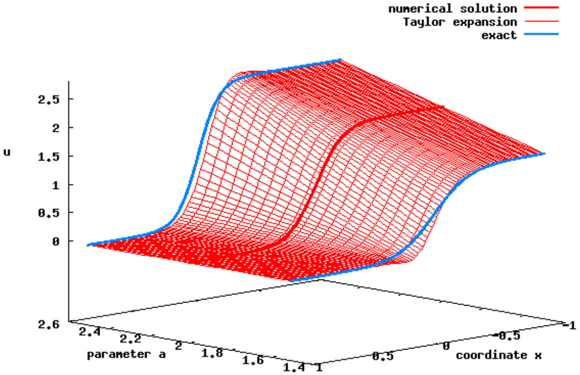
Solution of Burgers equation

A simple illustration



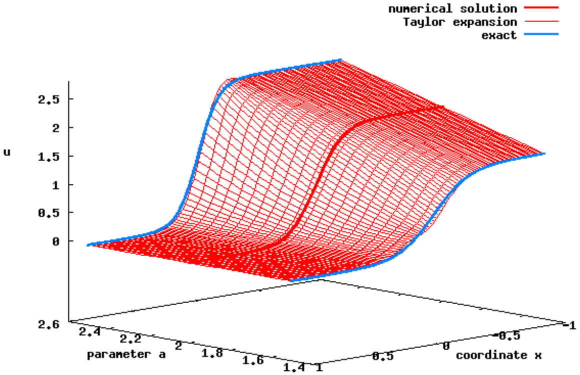
Linear taylor expansion

A simple illustration



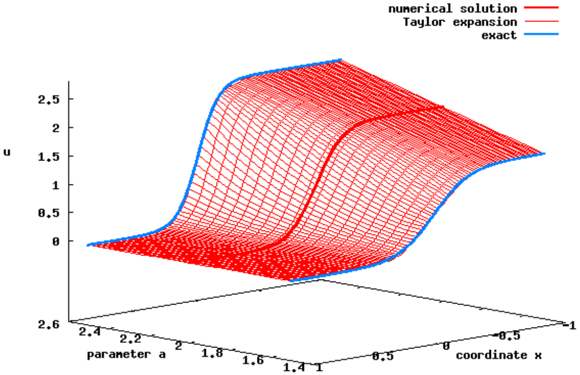
Linear taylor expansion

A simple illustration



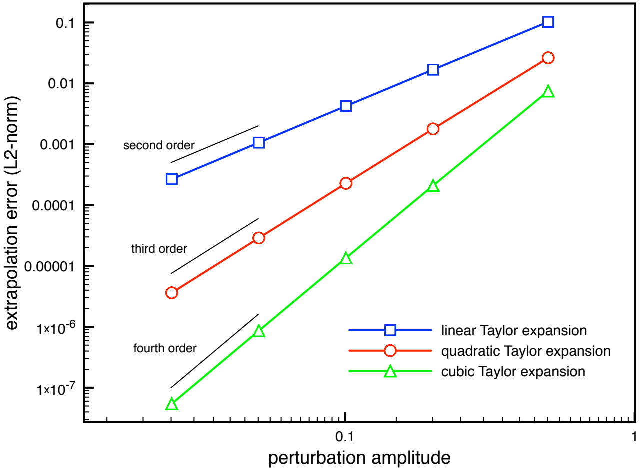
Quadratic taylor expansion

A simple illustration



Cubic taylor expansion

A simple illustration



Extrapolation error in the direction $(1, 0.1)$

Synthesis

- Sensitivity equation can be efficiently implemented in existing DG code
- High-order accuracy for sensitivity variables
- Parallelization strategy for computational efficiency to be explored