Introduction of AdjointShapeOptimizationFoam

Y. Takagi

November 10, 2012, OF b.@Kansai
Brief history of adjoint method

- 1969 Lions
  - Optimal control of systems governed by partial differential equations
- 1974 Pironneau
  - On optimum design in fluid mechanics
- 1988 Jameson
  - Aerodynamics design via control theory
- 1997 Giles, Pierce
  - Adjoint equations in CFD: duality, boundary conditions and solution behavior
Brief history of adjoint method

• 1997 Anderson, Venkatakrishnan
  – Aerodynamics design optimization on unstructured grids with a continuous adjoint formulation
• 2003 Borrvall, Peterson
  – Topology optimization of fluids in Stokes flow
• 2007 Othmer, Villiers, Weller
  – Implementation of a continuous adjoint for topology optimization of ducted flows
• 2008 Othmer
  – A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows
Formulation of adjoint method

Formulation of adjoint method

• Optimization problem

Minimize $J = J(\alpha, v, p)$ subject to $R(\alpha, v, p) = 0$

where

$J$ : cost function

$\alpha$ : porosity

$v$ : velocity

$p$ : pressure
State equations

Incompressible, steady-state Navier-Stokes equations with porosity

\[(R_1, R_2, R_3)^T = (v \cdot \nabla)v + \nabla p - \nabla \cdot (2vD(v)) + \alpha v\]

\[R_4 = -\nabla \cdot v\]

where \(R\) is the state equations,

\[R = (R_1, R_2, R_3, R_4)^T\]

Introduce a Lagrangian function \(L\),

\[L := J + \int_\Omega (u,q)Rd\Omega\]

\[(u,q) = (u_1, u_2, u_3, q)\] (Lagrangian multipliers)
Variation of Lagrangian function

Total variation of $L$, 

$$ \delta L = \delta_\alpha L + \delta_v L + \delta_p L \quad (\text{Lagrangian multipliers are chosen to satisfy } \delta_v L + \delta_p L = 0) $$ 

$$ = \delta_\alpha L = \delta_\alpha J + \int_\Omega (u,q) \delta_\alpha R d\Omega $$

Then, 

$$ \frac{\partial L}{\partial \alpha_i} = \frac{\partial J}{\partial \alpha_i} + \int_\Omega (u,q) \frac{\partial R}{\partial \alpha_i} d\Omega $$

Without explicit dependence of the cost function on the porosity, 

$$ \frac{\partial J}{\partial \alpha_i} = 0 $$
Sensitivity

By considering the Darcy term in cell $i$, 

$$\frac{\partial R}{\partial \alpha_i} = \begin{pmatrix} v \\ 0 \end{pmatrix} \chi_i$$

Therefore, the desired sensitivity for each cell can be computed by 

$$\frac{\partial L}{\partial \alpha_i} = u_i \cdot v_i V_i$$
Deviation of adjoint equations and boundary conditions

Decompose the cost function $J$ into contributions from the boundary $\Gamma$ and from the interior of $\Omega$,

$$J = \int_{\Gamma} J_{\Gamma} d\Gamma + \int_{\Omega} J_{\Omega} d\Omega$$

\[\text{\ldots (omitted)}\]

Finally, the adjoint Navier-Stokes equations are derived as follows:

$$-2D(u)v = -\nabla q + \nabla \cdot (2\nu D(u)) - \alpha u - \frac{\partial J_{\Omega}}{\partial v}$$

$$\nabla \cdot u = \frac{\partial J_{\Omega}}{\partial p}$$
Specialization to ducted flows

• Adjoint N-S equations:

\[-2D(u)v = -\nabla q + \nabla \cdot (2\nu D(u)) - \alpha u\]
\[\nabla \cdot u = 0\]

• Adjoint BCs for the wall and inlet:

\[u_t = 0, \quad u_n = -\frac{\partial J_G}{\partial p}\]
\[n \cdot \nabla q = 0\]

• Adjoint BCs for the outlet:

\[q = u \cdot v + u_n v_n + \nu (n \cdot \nabla) u_n + \frac{\partial J_G}{\partial n_n}\]
\[0 = v_n u_t + \nu (n \cdot \nabla) u_t + \frac{\partial J_G}{\partial n_t}\]
Example 1: Dissipated power

Cost function:

\[ J := - \int_{\Gamma} d\Gamma \left( p + \frac{1}{2} v^2 \right) v \cdot n \]

\[ J_{\Omega} = 0, \quad J_{\Gamma} = - \left( p + \frac{1}{2} v^2 \right) v \cdot n \]

Derivatives for BCs:

\[ \frac{\partial J_{\Gamma}}{\partial p} = - v \cdot n, \]

\[ \frac{\partial J_{\Gamma}}{\partial v} = - \left( p + \frac{1}{2} v^2 \right) n - (v \cdot n) v \]

Adjoint BCs for the wall and inlet:

\[ \mathbf{u}_t = 0 \quad \text{at wall} \]

\[ \mathbf{u}_n = \begin{cases} 0 & \text{at inlet} \\ v_n & \end{cases} \]

Adjoint BCs for the outlet:

\[ q = \mathbf{u} \cdot \mathbf{v} + u_n v_n + \nu (\mathbf{n} \cdot \nabla) u_n - \frac{1}{2} v^2 - v_n^2 \]

\[ 0 = v_n (\mathbf{u}_t - \mathbf{v}_t) + \nu (\mathbf{n} \cdot \nabla) \mathbf{u}_t \]
adjointShapeOptimization.C

laminarTransport.lookup("lambda") >> lambda;
alpha +=
    mesh.relaxationFactor("alpha")
    *(min(max(alpha + lambda*(Ua & U), zeroAlpha), alphaMax) - alpha);

zeroCells(alpha, inletCells);

// Pressure-velocity SIMPLE corrector
{
    // Momentum predictor
    tmp<fvVectorMatrix> UEqn
    (fvm::div(phi, U)
    + turbulence->divDevReff(U)
    + fvm::Sp(alpha, U)
    + alpha * U);


// Adjoint Pressure-velocity SIMPLE corrector
{
    // Adjoint Momentum predictor

    volVectorField adjointTransposeConvection((fvc::grad(Ua) & U));

    zeroCells(adjointTransposeConvection, inletCells);

    tmp<fvVectorMatrix> UaEqn
    (  
        fvm::div(-phi, Ua)
    - adjointTransposeConvection
    + turbulence->divDevReff(Ua)
    + fvm::Sp(alpha, Ua)
    );

    \[ \n    \n    \]
void Foam::adjointOutletVelocityFvPatchVectorField::updateCoeffs()
{
    if (updated())
    {
        return;
    }

    const fvsPatchField<scalar>& phiap = patch().lookupPatchField<surfaceScalarField, scalar>("phia");

    const fvPatchField<vector>& Up = patch().lookupPatchField<volVectorField, vector>("U");

    scalarField Un(mag(patch().nf() & Up));
    vectorField UtHat((Up - patch().nf()*Un)/(Un + SMALL));

    vectorField Uan(patch().nf()*(patch().nf() & patchInternalField()));

    vectorField::operator=(phiap*patch().Sf()/sqr(patch().magSf()) + UtHat);

    fixedValueFvPatchVectorField::updateCoeffs();
}
void Foam::adjointOutletPressureFvPatchScalarField::updateCoeffs()
{
    if (updated())
    {
        return;
    }

    const fvsPatchField<scalar>& phip = patch().lookupPatchField<surfaceScalarField, scalar>("phi");
  
    const fvsPatchField<scalar>& phiap = patch().lookupPatchField<surfaceScalarField, scalar>("phia");
  
    const fvPatchField<vector>& Up = patch().lookupPatchField<volVectorField, vector>("U");
  
    const fvPatchField<vector>& Uap = patch().lookupPatchField<volVectorField, vector>("Ua");

    operator==((phiap/patch().magSf() - 1.0)*phip/patch().magSf() + (Up & Uap));

    fixedValueFvPatchScalarField::updateCoeffs();
}
Results: adjoint velocity
Result: adjoint pressure
Future work

• Implementation of sensitivity
• BCs for other examples
• Thermal convection problem

• Efficient optimization algorithms to deal with the computed topological and surface sensitivity maps
• Shape update algorithms to translate the shape sensitivities into a new and smooth shape