Adjoint-Based Derivative Evaluation Methods for Flexible Multibody Systems

Komahan Boopathy
and
Dr. Graeme J. Kennedy

Structures and Multidisciplinary Optimization Laboratory

School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, GA

June 20, 2018

Georgia Tech Aerospace Engineering

Motivation: Need for Derivatives

Finite differences and complex-step

- Simple, robust but inefficient
- Doesn't require knowledge of governing equations

Adjoint and direct method

- Efficient, robust but complicated
- Precise to the tolerance to which the governing ODEs/DAEs/PDEs are solved $(10^{-8}, 10^{-12}, 10^{-16})$
- Direct – efficient for # functions
- Adjoint – efficient for # variables
- Requires knowledge of governing equations (semi-analytic)
  - Calculus
  - Linear algebra
  - Graph theory +
- Adjoint details, to follow...

```
subroutine evalFuncGrad(this, num_dv, x, q, f, dfdx)

  class(solver)                              :: this
  type(scalar), dimension(:), intent(in)    :: x, q
  type(scalar), dimension(:), intent(inout):: dfdx
  type(scalar), intent(out)                  :: f
  ...

  ! Solve the ODE/PDE
  call this % solvePDE(q, x)

  ! Evaluate the function
  call this % evalFunc(q, x, f)

  vars: do m = 1, num_dv
     ! Store the original x(m) value
     xtmp = x(m)

     ! Perturb the m-th index of x
#if defined USE_COMPLEX
     x(m) = cmplx(dble(x(m)), 1.0d-16)
#else
     x(m) = x(m) + dh
#endif
     ! Solve the ODE/PDE
     call this % solvePDE(q, x)

     ! Evaluate the function
     call this % evalFunc(q, x, ftmp)

     ! Restore x
     x(m) = xtmp

     ! Find the FD/CSD derivative
#if defined USE_COMPLEX
     dfdx(m) = aimag(ftmp)/1.0d-16
#else
     dfdx(m) = (ftmp-f)/dh
#endif
  end do vars

end subroutine evalFuncGrad
```

101. Notation:

t	time
k	time index
x	design variables
m	number of design variables
u, v, w	state variables
n	number of state variables
R, S, T	governing (constraint) equations
λ, ψ, ϕ	adjoint variables
F	objective function
$\mathcal{L}$	Lagrangian

1. Objective Function:

A time-averaged objective function is considered (e.g. mean lift, mean potential energy, mean thermal-stress). We can approximate the integral as a discrete sum:

$$F = \frac{1}{T}\int_0^T f_k(u, v, w, x, t)\, dt = \sum_{k=0}^{N} hf_k(u_k, v_k, w_k, x) \tag{1}$$

Other possibilities for the choice of objective function exist too.

3. Lagrangian:

It would be nice and handy to pack all separate equations (objects) into one equation (object) to operate with. We introduce λ_k, ψ_k and ϕ_k as the adjoint variables associated with each of these function objects at the each time step and the Lagrangian function is written as:

$$\mathcal{L} = \sum_{k=0}^{N} h f_k + \sum_{k=0}^{N} h \lambda_k^T R_k + \sum_{k=0}^{N} \psi_k^T S_k + \sum_{k=0}^{N} \phi_k^T T_k. \tag{2}$$

Remark

The Lagrangian is a linear "FAXPY" operator (why not!) operating on vector objects – just like SAXPY, DAXPY, ZAXPY (being Fortranic). Here λ, ϕ, ψ adjoint variables: they manifest themselves as simple-scaling-scalars in linear-algebra and as Lagrange multipliers in the context of optimization.

Remark

When decoupling is possible one or more of the constraint equations fold into other equations and therefore the corresponging adjoint variable need not be included in the formulation of the Lagrangian, but is typically done for the sake of convenience/generality of the implementation.

Background

4. Solution:

For each timestep k:

Forward Mode

- Find state variables $\mathbf{u}_K$
- Newton's method based on linearization of governing equations: $\frac{\partial \mathbf{R}_k^p}{\partial \mathbf{u}_k^p} \Delta \mathbf{u}_k^p = -R_k$; $\mathbf{u}_k^p = \mathbf{u}_k^p + \Delta \mathbf{u}_k^p$
- Involves p linear solutions

Reverse Mode

- Find adjoint variables: λ, ψ and ϕ and and total derivative $\frac{df}{dx}$
- $1/p$ of computational time per function
- Linear problem of solving stationary points of the Lagrangian w.r.t. the states: $\frac{\partial \mathcal{L}}{\partial u_k}, \frac{\partial \mathcal{L}}{\partial v_k}, \frac{\partial \mathcal{L}}{\partial w_k} = 0$

```
subroutine evalFuncGrad(this, num_dv, x, q, f, dfdx)

  class(solver)                              :: this
  type(scalar), dimension(:), intent(in)    :: x, q
  type(scalar), dimension(:), intent(inout):: dfdx
  type(scalar), intent(out)                  :: f
  ...

  ! Solve the ODE/PDE
  call this % solvePDE(q, x)

  ! Evaluate the function
  call this % evalFunc(q, x, f)

  vars: do m = 1, num_dv
     ! Store the original x(m) value
     xtmp = x(m)

     ! Perturb the m-th index of x
#if defined USE_COMPLEX
     x(m) = cmplx(dble(x(m)), 1.0d-16)
#else
     x(m) = x(m) + dh
#endif
     ! Solve the ODE/PDE
     call this % solvePDE(q, x)

     ! Evaluate the function
     call this % evalFunc(q, x, ftmp)

     ! Restore x
     x(m) = xtmp

     ! Find the FD/CSD derivative
#if defined USE_COMPLEX
     dfdx(m) = aimag(ftmp)/1.0d-16
#else
     dfdx(m) = (ftmp-f)/dh
#endif
  end do vars

end subroutine evalFuncGrad
```

Georgia Tech Aerospace Engineering

5. Total Derivative:

The total derivative can be written as a linear combination of contributions from the function of interest and the constraints:

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial F}{\partial x} = \sum_{k=0}^{N} h \frac{\partial f_k}{\partial x}^T + \sum_{k=0}^{N} h \frac{\partial R_k}{\partial x}^T \lambda_k + \sum_{k=0}^{N} \frac{\partial S_k}{\partial x}^T \psi_k + \sum_{k=0}^{N} \frac{\partial T_k}{\partial x}^T \phi_k. \quad (3)$$

Remark

The objective function value F and the gradient $\frac{dF}{dx}$ goes to the optimization code.

Flexible Multibody Dynamics

Context of Structural Dynamics and Time Marching

t	time	t	time
k	time index	k	time index
x	design variables	x	design variables
m	number of design variables	m	number of design variables
u, v, w	state variables	$q, \dot{q}, \ddot{q}$	position, velocity and acceleration states
n	number of state variables	n	number of state variables
R, S, T	constraint governing equations	R, S, T	constraint governing equations
λ, ψ, ϕ	adjoint variables	λ, ψ, ϕ	adjoint variables
F	objective function	F	objective function
$\mathcal{L}$	Lagrangian	$\mathcal{L}$	Lagrangian

The residual of the governing equations from structural dynamics can be posed as:

$$R_k = R_k(\underline{\ddot{q}_k}, \dot{q}_k, q_k, x) = 0. \tag{4}$$

Example

$R = m\ddot{q} + c\dot{q} + kq = 0$, x $= [m, c, k]$ can be the design variables of choice, the objective be minimizing the time-averaged potential energy $F = \frac{1}{T}\int_0^T f_k \, dt$, where $f_k = \frac{1}{2}kq^2$ is the instantaneous PE.

Georgia Tech Aerospace Engineering

Further Directions

Adjoint

- AUTOMATIC evaluation of adjoint – yes, we can!
  - Based on `type(integrator)` – leaverage smart data structures and modern constructs, graph theory
  - NOT automatic differentiation (AD)!
- Applications:
  1. Optimization
  2. Error estimation
  3. Mesh refinement
  4. ...

Direct

Explore time dependent direct sensitivities

Georgia Tech Aerospace Engineering