Adjoint equations: theory and applications

Jan Pralits

Department of Civil, Chemical and Environmental Engineering
University of Genoa, Italy
jan.pralits@unige.it

January 14, 2014

Lecture Outline

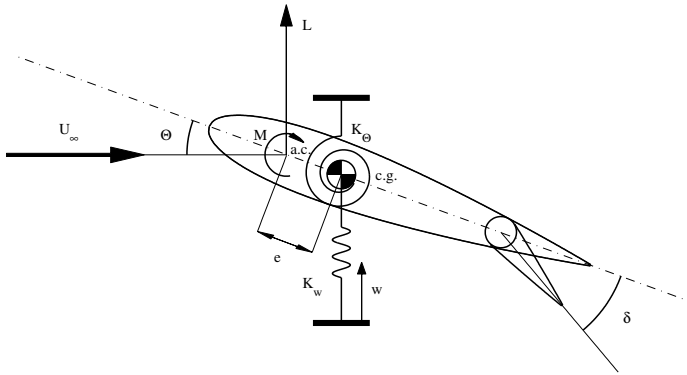
- **Topic** : Adjoint equations: theory and applications
- **Hours** : 2h
- **Content** :
  1. Introduction and Motivation
  2. Sensitivity analysis
  3. Definitions of adjoints
  4. Examples and derivations
  5. Numerical issues and accuracy
- **Aim** : **Overview** of main concepts; Provide you basic knowledge how to derive, implement and verify adjoint equations

Introduction and Motivation I

Let's assume we have a physical model.

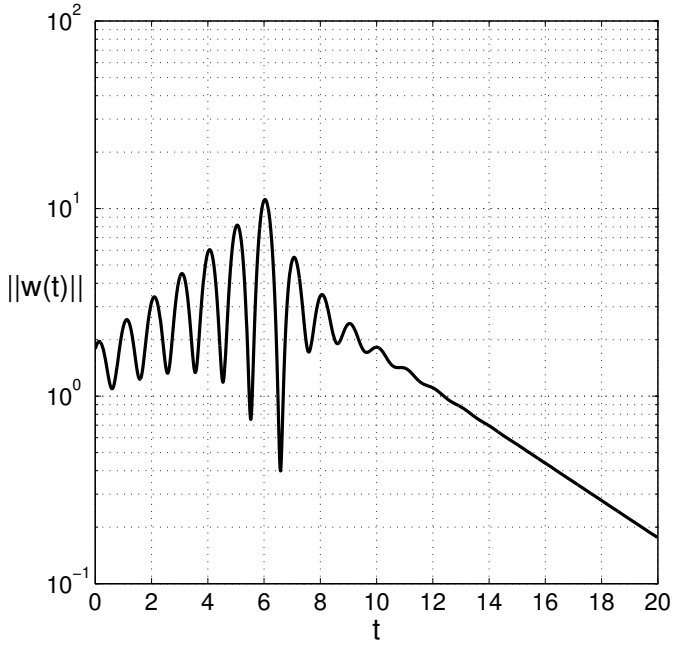
Ex: A linear model of an aeroelasticity problem. The result $x = (w, \Theta)$ gives us the vertical motion $w(y, t)$ and rotation $\Theta(y, t)$.

Determine: torsion and flutter.



$$\dot{x} = Ax + Bu,\ x(0) = x_0,\ 0 \leq t \leq T$$

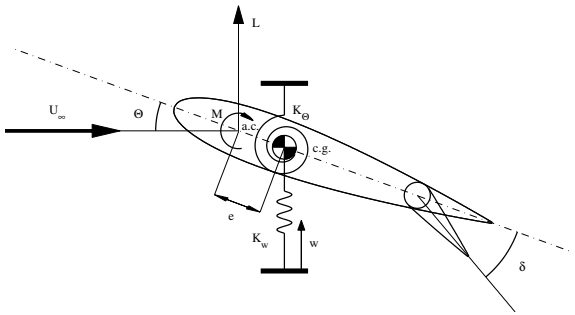
Movie 2

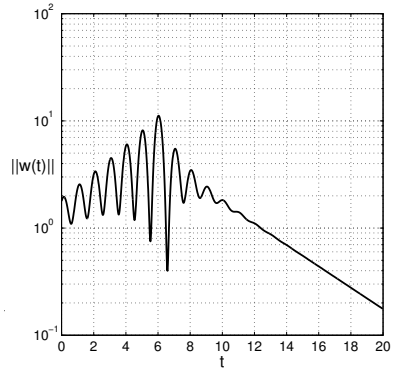


$\|\cdot\|$ means norm in the spanwise direction.

Introduction and Motivation II

Let's now assume we want to improve some characteristics.





Ex: Reduce fluctuations $x = (w, \Theta)$ using the flap (avoid flutter).

$$\text{Objective:} \qquad J = \frac{1}{2}\int_0^T \|x(t)\|^2\, dt + l^2\frac{1}{2}\int_0^T \|u(t)\|^2\, dt$$

$$\text{State:} \qquad \dot{x} = Ax + Bu, \quad x(0) = x_0, \quad 0 \le t \le T$$

We can set up a **gradient based** optimization problem:

$$\min_{\forall u} J, \quad \text{using} \quad u^{n+1} = u^n - \rho\left(\frac{\partial J}{\partial u}\right)^n$$

Several techniques exist: Direct (FD), Lagrange multipliers, **adjoint**-based methods...

Summary & Aim

- From the example shown so far it is clear that the **gradient** of J with respect to a chosen variable ξ (∇J_ξ) plays a crucial role for the solution.

  The gradient of an output with respect to an input gives the **sensitivity** which in itself is a useful result.

  $$\delta J = \frac{\partial J}{\partial \xi} \cdot \delta \xi$$

  As we have seen the sensitivity can also be used to solve **optimization** and **control** problems.

- The aim of the lecture is to demonstrate how to compute this quantity in an **efficient** and **accurate** way for a number of different problems, both on a theoretical and practical level.

- The same approach is applicable for problems such as: **optimal control**, **optimal perturbations (nonmodal growth)**

Definitions

Sensitivity analysis (SA) is the study of how the **variation** in the **output** of a mathematical model can be apportioned, to different sources of variation in the **input** of the model.

Definition:
The (**scalar**) directional derivative

$$d(\mathbf{u},\mathbf{p}) \triangleq \frac{\partial J(\mathbf{u})}{\partial \mathbf{u}} \cdot \mathbf{p} = \lim_{\epsilon \to 0^+} \frac{1}{\epsilon}[J(\mathbf{u}+\epsilon\mathbf{p}) - J(\mathbf{u})], \tag{1}$$

The (**vector**) directional gradient

$$\mathbf{g}_i(\mathbf{u}) = d(\mathbf{u},\mathbf{e}^i) = \frac{\partial J(\mathbf{u})}{\partial \mathbf{u}} \cdot \mathbf{e}^i = \lim_{\epsilon \to 0^+} \frac{1}{\epsilon}[J(\mathbf{u}+\epsilon\mathbf{e}^i) - J(\mathbf{u})] \quad \Rightarrow \quad \mathbf{g}(\mathbf{u}) \triangleq \frac{\partial J(\mathbf{u})}{\partial \mathbf{u}} \triangleq \nabla J(\mathbf{u}) \tag{2}$$

In the following we assume that the **dimensions** of $\mathbf{u}$ and $J(\mathbf{u})$ are n and m respectively.

Finite-difference approximation (FD)

First-order finite-difference (FD) formula:

$$d(\mathbf{u}, \mathbf{e}^i) = \frac{J(\mathbf{u} + \epsilon \mathbf{e}^i) - J(\mathbf{u})}{\epsilon} + O(\epsilon). \tag{3}$$

A second-order finite-difference formula:

$$d(\mathbf{u}, \mathbf{e}^i) = \frac{J(\mathbf{u} + \epsilon \mathbf{e}^i) - J(\mathbf{u} - \epsilon \mathbf{e}^i)}{2\epsilon} + O(\epsilon^2). \tag{4}$$

Note:

- This approach is straightforward to implement.
- Using a computer with finite-precision arithmetic is the difficulty to find a suitable value for the step size ϵ. If it is large then the Taylor-series truncation is not valid and when it is small then the subtractive cancellation errors might dominate.
- In order to evaluate $\partial J(\mathbf{u})/\partial \mathbf{u}$ as given by equations (3) and (4) requires $n + 1$ and $2\,n$ function evaluations, respectively, for each m. This **can certainly be computationally expensive** if each evaluation of J requires the solution of an ODE or PDE

Complex-step derivative (CSD)

If the complex extension $J(\mathbf{z})$ of a real-valued function $J(\mathbf{u})$ is analytic, it can be expanded with a complex Taylor series. The expansion of $J(\mathbf{u} + i\epsilon\mathbf{p})$ can be written

$$J(\mathbf{u} + i\epsilon\mathbf{p}) = J(\mathbf{u}) + i\epsilon\frac{\partial J(\mathbf{u})}{\partial \mathbf{u}} \cdot \mathbf{p} + O(\epsilon^2), \tag{5}$$

and the directional derivative can be found by rearranging the expansion as

$$d(\mathbf{u}, \mathbf{p}) = \frac{\partial J(\mathbf{u})}{\partial \mathbf{u}} \cdot \mathbf{p} = \frac{1}{\epsilon}\Im[J(\mathbf{u} + i\epsilon\mathbf{p})] + H.O.T. \tag{6}$$

Note:

- More "tricky" to implement (in *matlab* it is simple).
- The error scales with ϵ^2.
- There are no cancellation errors.
- This approach requires n computations.
- It is more robust with respect to the choice of the value of ϵ, compared to the FD approach.

Adjoint-based approach I

Case 1: J depends linearly on the solution $\mathbf{x}$ of a linear system, find $\partial J/\partial \mathbf{b}$

$$J(\mathbf{b}) = \mathbf{c}^T \mathbf{x}, \tag{7}$$

$$A\,\mathbf{x} = \mathbf{b}, \tag{8}$$

where A is a known matrix $A \in \mathbb{R}^{n,n}$, $\mathbf{c}$ and $\mathbf{b}$ are the known vectors $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^n$, respectively. Introduce **an additional linear system**

$$A^T \mathbf{y} = \mathbf{c}. \tag{9}$$

... after some linear algebra:

$$J = \mathbf{c}^T \mathbf{x} = (A^T \mathbf{y})^T \mathbf{x} = \mathbf{y}^T A \mathbf{x} = \mathbf{y}^T \mathbf{b} \tag{10}$$

With the new linear system (9), J depends **explicitly** on the vector $\mathbf{b}$, and

$$\frac{\partial J(\mathbf{b})}{\partial \mathbf{b}} = \mathbf{y}. \tag{11}$$

Note: The computational cost is **1** and independent of n since only one solution of equation (9) is needed to evaluate (11).

Adjoint-based approach II

Case 2: J depends quadratically on the solution $\mathbf{x}$ of a linear system.

$$J(\mathbf{b}) = \frac{1}{2}\mathbf{x}^T\mathbf{x}, \tag{12}$$

$$A\,\mathbf{x} = \mathbf{b}, \tag{13}$$

Linearize: $\mathbf{x} \leftarrow \mathbf{x} + \delta\mathbf{x}$ and $\mathbf{b} \leftarrow \mathbf{b} + \delta\mathbf{b}$ into equations (12)-(26) and dropping higher-order δ-terms:

$$\delta J = \mathbf{x}^T\delta\mathbf{x}, \tag{14}$$

$$A\,\delta\mathbf{x} = \delta\mathbf{b}. \tag{15}$$

We now introduce an additional linear system as

$$A^T\mathbf{y} = \mathbf{x}, \tag{16}$$

and the function δJ can be evaluated as

$$\delta J = \mathbf{x}^T\delta\mathbf{x} = (A^T\mathbf{y})^T\delta\mathbf{x} = \mathbf{y}^T A\delta\mathbf{x} = \mathbf{y}^T\delta\mathbf{b}. \tag{17}$$

The gradient is then evaluated easily as

$$\frac{\partial J(\mathbf{b})}{\partial\mathbf{b}} = \mathbf{y}. \tag{18}$$

Note: The computational cost is **2**, independently of n. The two systems to solve are (26) & (16).

Comparison of computational effort

Computational effort: the number of function evaluations

Approach	$J(\mathbf{b}) = \mathbf{c} \cdot \mathbf{u}$	$J(\mathbf{b}) = \mathbf{u} \cdot \mathbf{u}$
FD1	$m \cdot (n+1)$	$m \cdot (n+1)$
FD2	$2m \cdot n$	$2m \cdot n$
CSD	$m \cdot n$	$m \cdot n$
ADJ	m	$2m$

Table: Number of function evaluations to compute the gradient using Finite-Difference approximation (FD), Complex step derivative (CSD) and adjoint equations (ADJ). First and second order FD approach are denoted FD1 and FD2. Here, n is the dimension of $\mathbf{b}$ and m the dimension of J.

Note:

- In general $m \ll n$ which makes the adjoint approach much more favorable.
- For cases in which n is small and comparable to m then all approaches have comparable computational effort.
- The CSD has always comparable computational effort compared to FD but is more robust.
- The adjoint approach requires the derivation and implementation of an additional system. (difficult when using commercial software, but an option when using *open source* codes or *in-house* codes.

Definition of adjoint

The adjoint operator was introduced by Lagrange in the 18th century.

$$\int u^* A u \, dx = \int u A^* u^* \, dx, \tag{19}$$

which is valid for u, u^* vanishing at the ends of the integration interval. Relations like equation (19) are usually called **duality relations** or **Lagrange equalities**.

In general, with non-vanishing boundary conditions, and the inner product given as $\langle \cdot, \cdot \rangle$:

$$\langle u^*, Au \rangle = \langle u, A^* u^* \rangle + \text{B.T.} \tag{20}$$

Note:

- A^* is a **linear** operator.
- It is not straight forward to define the adjoint of a **nonlinear** operator. It can however be introduced based on the theory of linear operators. This means that one must **first linearise** the nonlinear operator and **then define its adjoint**.

Matrices

Consider an m-by-n matrix A with complex entries:

$$(A^*)_{ij} = \overline{A_{ji}} \tag{21}$$

or more compact as

$$A^* = (\overline{A})^T = \overline{A^T} = A^H \tag{22}$$

A square matrix A with entries a_{ij} is called

- *Hermitian* or *self-adjoint* if $A = A^*$, i.e., $a_{ij} = \overline{a_{ji}}$.
- *skew Hermitian* or antihermitian if $A = -A^*$, i.e., $a_{ij} = -\overline{a_{ji}}$.
- *normal* if $A^*A = AA^*$.
- *unitary* if $A^* = A^{-1}$.

Even if A is not square, the two matrices A^*A and AA^* are both Hermitian and in fact positive semi-definite matrices. Some further considerations for which holds also for non-square matrices.

- $(A^*)^* = A$ for any matrix A.
- The eigenvalues of A^* are the complex conjugates of the eigenvalues of A.
- $\langle A\mathbf{x}, \mathbf{y}\rangle = \langle \mathbf{x}, A^*\mathbf{y}\rangle$ for any m-by-n matrix A, any vector $\mathbf{x}$ in $\mathbb{C}^n$ and any vector $\mathbf{y}$ in $\mathbb{C}^m$. Here $\langle \cdot, \cdot \rangle$ denotes an inner product on $\mathbb{C}^m$ and $\mathbb{C}^n$.
  (**cf. the adjoint approach to compute a gradient.**)

Some examples I

Case: Previous example, $A\,\mathbf{x} = \mathbf{b}$ $(A^{n\times n}, \mathbf{x}^n, \mathbf{b}^n)$, and $\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^T\mathbf{b}$

$$\langle \mathbf{y}, A\,\mathbf{x} \rangle = \mathbf{y}^T A\,\mathbf{x} = (A^T\mathbf{y})^T\mathbf{x} = \langle A^T\mathbf{y}, \mathbf{x} \rangle = \langle A^*\mathbf{y}, \mathbf{x} \rangle. \tag{23}$$

Let's assume that $A^*\mathbf{y} = \mathbf{c}$

$$\langle \mathbf{y}, \mathbf{b} \rangle = \mathbf{y}^T\mathbf{b} = \mathbf{c}^T\mathbf{x} = \langle \mathbf{c}, \mathbf{x} \rangle. \tag{24}$$

$\mathbf{y}$ is the sensitivity with respect to $\mathbf{b}$. But $\mathbf{c}$ has no physical meaning.

The adjoint itself does not have any physical meaning, and is only useful when used as a mathematical tool to solve a specific physical problem.

If we instead relate $\mathbf{c}$ to $J = \mathbf{c}^T\mathbf{x}$, as in the previous example, then the physical interpretation of the results is clear.

Some examples II

Case: A time dependent linear system.

$$J = \mathbf{c}^T \mathbf{x}(T), \tag{25}$$

$$\frac{d\mathbf{x}}{dt} + A\mathbf{x} = 0, \quad \mathbf{x}(0) = \mathbf{x}_0, \quad 0 \leq t \leq T \tag{26}$$

where A is a known matrix $A \in \mathbb{R}^{n,n}$ and $\mathbf{c}$ is a known vector $\mathbf{c} \in \mathbb{R}^n$. Introduce

$$\int_0^T \langle \mathbf{a}(t), \mathbf{b}(t) \rangle dt = \int_0^T \mathbf{a}(t)^T \mathbf{b}(t)\, dt \tag{27}$$

and an adjoint variable $\mathbf{v}(t)$. The adjoint identity is then written

$$\int_0^T \mathbf{v}^T \left(\frac{d\mathbf{x}}{dt} + A\mathbf{x} \right) dt = \int_0^T \left(-\frac{d\mathbf{v}}{dt} + A^T \mathbf{v} \right)^T \mathbf{x}\, dt + \mathbf{v}(T)^T \mathbf{x}(T) - \mathbf{v}(0)^T \mathbf{x}(0) \tag{28}$$

The LHS $= 0$ and so is the RHS. If we define the adjoint equation as

$$-\frac{d\mathbf{v}}{dt} + A^T \mathbf{v} = 0, \quad \mathbf{v}(T) = \mathbf{c}, \quad 0 \leq t \leq T \tag{29}$$

Then by inspection we can write the remaining terms as

$$J = \mathbf{c}^T \mathbf{x}(T) = \mathbf{v}(0)^T \mathbf{x}(0), \tag{30}$$

and the gradient

$$\nabla J = \mathbf{v}(0) \tag{31}$$

Some examples III

Case: Gradient of a cost function for an advection-diffusion problem.

Consider a linear advection-diffusion PDE for the evolution of $u(x,t)$:

$$\frac{\partial u}{\partial t} + \frac{\partial (a\,u)}{\partial x} - \frac{1}{Re}\frac{\partial^2 u}{\partial x^2} = 0, \tag{32}$$

with $a(x)$, and initial and boundary conditions given by $u(x,0) = u_0$ and $u(0,t) = u(1,t) = 0$. Introduce

$$J = \int_0^1 k(x) u(x,T)\, dx, \tag{33}$$

Objective: find $\partial J/\partial u_0$.

Introduce

$$\int_0^T \langle a(x,t), b(x,t)\rangle dt = \int_0^T \int_0^1 a(x,t) b(x,t)\, dx\, dt, \tag{34}$$

and an adjoint variable $v(x,t)$.

We now need to evaluate the adjoint identity...

Some examples IV

The **adjoint identity** can now be written:

$$\int_0^T \int_0^1 v \left(\underbrace{\frac{\partial u}{\partial t} + \frac{\partial (a\,u)}{\partial x} - \frac{1}{Re}\frac{\partial^2 u}{\partial x^2}}_{=0} \right) dxdt = \int_0^T \int_0^1 u \left(\underbrace{-\frac{\partial v}{\partial t} - a\frac{\partial v}{\partial x} - \frac{1}{Re}\frac{\partial^2 v}{\partial x^2}}_{=0} \right) dxdt +$$

$$+ \int_0^1 \left[u(x,T)v(x,T) - u(x,0)v(x,0) \right] dx + \int_0^T \left[\underbrace{a(1)u(1,t)v(1,t)}_{u(1,t)=0} - \underbrace{a(0)u(0,t)v(0,t)}_{u(0,t)=0} \right] dt$$

$$- \frac{1}{Re}\int_0^T \left[\underbrace{v(1,t)\frac{\partial u(1,t)}{\partial x}}_{v(1,t)=0} - \underbrace{v(0,t)\frac{\partial u(0,t)}{\partial x}}_{v(0,t)=0} \right] dt + \frac{1}{Re}\int_0^T \left[\underbrace{u(1,t)\frac{\partial v(1,t)}{\partial x}}_{u(1,t)=0} - \underbrace{u(0,t)\frac{\partial v(0,t)}{\partial x}}_{u(0,t)=0} \right] dt. \qquad (35)$$

The right hand side of (35) is obtained by **integration by parts** in x and t.

The left hand side is zero by definition of equation (32) $\rightarrow$ RHS $= 0$.

Impose boundary conditions: direct and adjoint

Some examples V

The remaining terms from the adjoint identity can now be written

$$\int_0^1 [v(x,T)u(x,T) - v(x,0)u(x,0)]\,dx = 0. \tag{36}$$

Identify equation (33) in the remaining terms (36). Choice: $v(x,T) = k(x)$

$$\underbrace{\int_0^1 k(x)u(x,T)\,dx}_{=J} - \int_0^1 v(x,0)u(x,0)\,dx = 0. \tag{37}$$

Rewrite

$$J = \int_0^1 k(x)u(x,T)\,dx = \int_0^1 v(x,0)u(x,0)\,dx \triangleq \langle \frac{\partial J}{\partial u(x,0)}, u(x,0)\rangle, \tag{38}$$

the gradient of J with respect to $u(x,0)$ can be identified as

$$\frac{\partial J}{\partial u(x,0)} = v(x,0). \tag{39}$$

Some examples VI

This is called **adjoint identity approach** and the computational cost is **1**.

The system of equations needed to compute the gradient can now be summarized as follows
state equation:

$$\frac{\partial u}{\partial t} + \frac{\partial (a\, u)}{\partial x} - \frac{1}{Re}\frac{\partial^2 u}{\partial x^2} = 0,$$

with initial and boundary conditions given by $u(x,0) = u_0$ and $u(0,t) = u(1,t) = 0$.
cost function:

$$J = \int_0^1 k(x) u(x,T)\, dx.$$

adjoint equations:

$$-\frac{\partial v}{\partial t} - a\frac{\partial v}{\partial x} - \frac{1}{Re}\frac{\partial^2 v}{\partial x^2} = 0,$$

with initial and boundary conditions given by $v(x,T) = k(x)$ and $v(0,t) = v(1,t) = 0$.
gradient:

$$\frac{\partial J}{\partial u(x,0)} = v(x,0).$$

Note: The adjoint equations are integrated backwards in time with an initial condition at $t = T$.

Accuracy of the adjoint

The accuracy of the adjoint it is often referred to how accurate the adjoint identity

$$\langle u^*, Au \rangle = \langle u, A^* u^* \rangle + \text{B.T.}$$

is satisfied once the equations have been discretised.

We only need to consider the terms left once the adjoint equations have been defined.
Ex:

$$\int_0^1 [u(x,T)v(x,T) - u(x,0)v(x,0)]\, dx = 0.$$

A discrete version of this equation can be written

$$\text{error} = |\langle u(x,T), v(x,T) \rangle_{\Delta h} - \langle u(x,0), v(x,0) \rangle_{\Delta h}|, \tag{40}$$

using the values of u and v at $t = 0$ and $t = T$, and

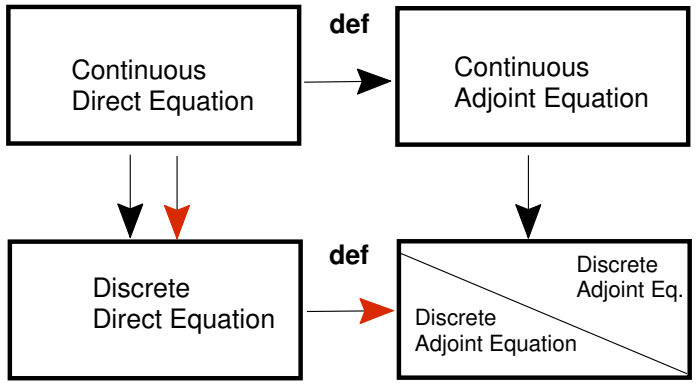
$\langle , \rangle_{\Delta h}$ which is a discretised version of the inner product $\langle , \rangle$.

Continuous vs. Discrete Adjoint Equations

The adjoint equations can be derived using two different approaches.
Both with advantages and disadvantages.
By definition we have

$$\langle u^*, Au \rangle = \langle u, A^* u^* \rangle + \text{B.T.}$$

- **Continuous approach** $\rightarrow$: The adjoint equations are derived by definition using the continuous direct equations.
  - \+ Straightforward derivation, reuse old code when programming
  - – Accuracy depends on discretization, difficulties with boundary conditions
- **Discrete approach** $\rightarrow$: The adjoint equations are derived from the discretized direct equations.
  - \+ Accuracy can be achieved close to machine precision, and can be independent of discretization !!
  - – Tricky derivation, usually requires making a new code, or larger changes of an existing code.



Here "def" means definition of the adjoint operator.
In the top row it is on continuous form while in the bottom row it is on discrete form.

Derivation of the adjoint equation I

Consider the following optimization problem (ODE) where ϕ is the state and g the control.

$$\frac{d\phi(t)}{dt} = -A\phi(t) + Bg(t), \qquad \text{for} \qquad 0 \leq t \leq T,$$

with initial condition

$$\phi(0) = \phi_0$$

We can now define an optimization problem in which the goal is to find an optimal $g(t)$ by minimizing the following objective function

$$J = \frac{\gamma_1}{2}[\phi(T) - \Psi]^2 + \frac{\gamma_2}{2}\int_0^T g(t)^2 dt,$$

Derivation of the adjoint equation II

Continuous approach

We can solve this problem using an **adjoint identity** approach or by introducing **Lagrange multipliers**.

$$\int_0^T a[\frac{d\phi}{dt} + A\phi - Bg]\,dt = \int_0^T [-\frac{da}{dt} + A^* a]\phi\,dt - \int_0^T aBg\,dt + a(T)\phi(T) - a(0)\phi(0).$$

If we now define the adjoint equation as $-da/dt = -A^* a$ with an arbitrary initial condition $a(T)$ then the identity reduces to

$$\text{LHS} = -\int_0^T aBg\,dt + a(T)\phi(T) - a(0)\phi(0)$$

By definition the Left Hand Side is identically zero but this is exactly what must be checked numerically, i.e. error$=$ |LHS|.

Derivation of the adjoint equation III

The gradient of J w.r.t. g can be derived considering the J is nonlinear in ϕ and g. We linearise by $\phi \to \phi + \delta\phi$, $g \to g + \delta g$ and then write the linearised objective function as

$$\gamma_1[\phi(T) - \Psi]\delta\phi(T) = \delta J - \gamma_2 \int_0^T g\delta g \, dt,$$

If we choose $a(T) = \gamma_1[\phi(T) - \Psi]$ then the equation for δJ can be substituted into the expression for the adjoint identity. If you further define the adjoint equations, remember that $\delta\phi(0) = 0$, then the final identity is written

$$\delta J = \int_0^T [\gamma_2 g + B^* a]\delta g \, dt$$

The adjoint equations and gradient of J w.r.t. g are written

$$-\frac{da}{dt} + A^* a, \;\; a(T) = \gamma_1[\phi(T) - \Psi], \quad \text{and} \quad \nabla J_g = \gamma_2 g + B^* a.$$

The so called optimality condition is given by $\nabla J_g = 0$.

Derivation of the adjoint equation IV

- The accuracy of the adjoint solution is important since it quantfies a "gradient" in the optimization problem.
- The "error" must be evaluated to quantify the accuracy the adjoint solution.
- Note that the adjoint solution depends on the resolution (Δt), and likewise the accuracy.
- **Can we do better ?**

Derivation of the adjoint equation V

Discrete approach

A discrete version of the direct equation is written

$$\frac{\phi^{i+1} - \phi^i}{\Delta t} = -A\phi^i + Bg^i, \quad \text{for} \quad i = 1, ..., N-1,$$

where N denotes the number of discrete points on the interval $[0, T]$, Δt is the constant time step, and

$$\phi^1 = \phi_0,$$

is the initial condition. This can be written as a discrete evolution equation

$$\phi^{i+1} = L\phi^i + \Delta t B g^i, \quad \text{for} \quad i = 1, ..., N-1.$$

with $L = I - \Delta t A$, and I the identity matrix. A discrete version of the objective function can be written

$$J = \frac{\gamma_1}{2}(\phi^N - \Phi)^2 + \frac{\gamma_2}{2}\sum_{i=1}^{N-1} \Delta t (g^i)^2.$$

Derivation of the adjoint equation VI

An adjoint variable a^i is introduced defined on $i = 1, ..., N$ and we write the adjoint identity as

$$a^{i+1} \cdot L\phi^i = (L^\star a^{i+1}) \cdot \phi^i, \quad \text{for} \quad i = 1, ..., N-1.$$

We then introduce the definition of the state equation on the left hand side of and impose that

$$a^i = L^\star a^{i+1} \quad \text{for} \quad i = N-1, ..., 1.$$

This is the **discrete adjoint equation**. Using the discrete direct and adjoint yields

$$a^{i+1} \cdot (\phi^{i+1} - \Delta t B g^i) = a^i \cdot \phi^i, \quad \text{for} \quad i = 1, ..., N-1.$$

which must be valid for any ϕ and a. An error can therefore be written as

$$\text{error} = |a^N \cdot \phi^N - a^1 \cdot \phi^1 - \sum_{i=1}^{N-1} \Delta t \, a^{i+1} \cdot B g^i|.$$

This can be compared with the error for the continuous formulation from the previous derivation

$$\text{error} = |a(T)\phi(T) - a(0)\phi(0) - \int_0^T aBg \, dt|$$

Derivation of the adjoint equation VII

Comparison

Error from the discrete approach

$$\text{error} = |a^N \cdot \phi^N - a^1 \cdot \phi^1 - \sum_{i=1}^{N-1} \Delta t \, a^{i+1} \cdot Bg^i|$$

Error from the continuous approach

$$\text{error} = |a(T)\phi(T) - a(0)\phi(0) - \int_0^T aBg \, dt|$$

The convergence of the physical problem and the accuracy of the gradient can be considered two different issues.

- In the continuous approach the **adjoint solution depends on the discretization scheme used and the error will decrease as the spatial and temporal resolution increase**. In this case it is difficult to distinguish between different errors in the case an optimization problem fail to converge.
- The advantage of the **discrete approach is that the adjoint solution will be numerically "exact", independently of the spatial and temporal resolution**.

Derivation of the adjoint equation VIII

The discrete optimality condition is then derived. Since J is nonlinear with respect to ϕ and g we must first linearize. This can be written

$$\delta J = \gamma_1(\phi^N - \Phi) \cdot \delta\phi^N + \gamma_2 \sum_{i=1}^{N-1} \Delta t g^i \cdot \delta g^i.$$

We now choose the terminal condition of the adjoint as $a^N = \gamma_1(\phi^N - \Phi)$ and substitute this expression into the discrete adjoint identity. This is written

$$\gamma_1(\phi^N - \Phi) \cdot \delta\phi^N = a^1 \cdot \delta\phi^1 + \sum_{i=1}^{N-1} \Delta t\, a^{i+1} \cdot B\delta g^i$$

By inspection one can see that the left hand side is identical to the first term in the expression for δJ, and $\delta\phi^1 = 0$. Rearranging the terms, we get

$$\delta J = \sum_{i=1}^{N-1} \Delta t \left(\gamma_2 g^i + B^\star a^{i+1}\right) \cdot \delta g^i,$$

from which we get the discrete optimality condition

$$g^i = -\frac{1}{\gamma_2} B^\star a^{i+1} \quad \text{for} \quad i = 1, .., N-1.$$

Note that if B is a matrix then $B^\star = B^T$.

Checkpointing algorithm

- When the adjoint equation is forced in time by the direct solution (ex. quadratic objective function), then this poses storage requirements (hard ware).
- Checkpointing: This consists of sampling the direct solution at given rate and then recompute the direct solution for short time intervals when needed. This means in theory that one more solution of the direct system has been added to the computational effort.
- However, since it is common to use parallel computing, and processors is becoming a smaller problem on can do something to obtain the minimal required computational time.
- This is done by recomputing the direct solution, in parallel, while computing the adjoint.