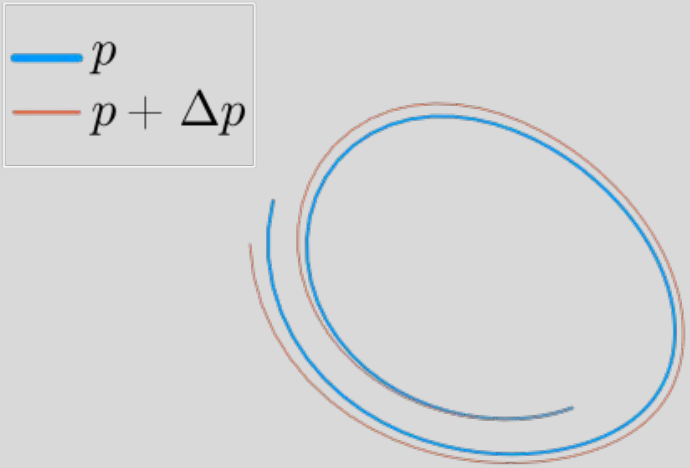
An Introduction to (Local) Sensitivity Analysis for (Ordinary) Differential Equations



Frank Schäfer[1]

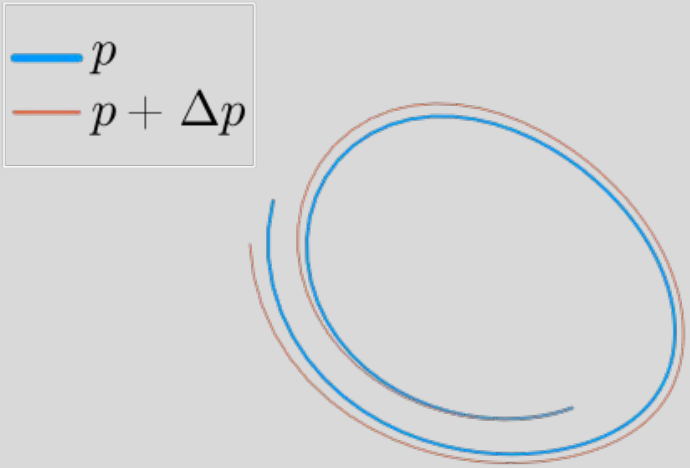
[1]Julia Lab, CSAIL, MIT

18.S096 Special Subject:
Matrix Calculus for Machine Learning and Beyond by Professors Alan Edelman and Steven G. Johnson

An Introduction to (Local) Sensitivity Analysis for (Ordinary) Differential Equations

Frank Schäfer[1]

[1]Julia Lab, CSAIL, MIT



18.S096 Special Subject:
Matrix Calculus for Machine Learning and Beyond by Professors Alan Edelman and Steven G. Johnson

Ordinary Differential Equations (ODEs)

- Initial value problem

$$u(t_0) = u_0, \qquad \frac{\mathrm{d}u}{\mathrm{d}t}(t) = f(t, u(t), p)$$

- How do we solve an ODE?

Toy example: Free falling ball

- ODE with state $u(t) = [z(t), v(t)]$

$$\mathrm{d}z(t) = v(t)\mathrm{d}t,$$
$$\mathrm{d}v(t) = -\mathrm{g}\,\mathrm{d}t$$

$$z(t=0) = z_0 = 5$$
$$v(t=0) = v_0 = -0.1$$

```julia
z0 = 5.0 # initial height
v0 = 0.1 # initial velocity
u0 = [z0, v0] # initial ODE state
p = [10.0] # gravitational constant
tspan = (0.0, 1.0) # integration time
dt = 0.001 # time discretization

# system
function f(u, p, t)
    dz = u[2]
    dv = -p[1]
    [dz, dv]
end
```

- Analytical solution (available in this case)

$$z(t) = z_0 + v_0(t - t_0) - \frac{\mathrm{g}}{2}(t - t_0)^2$$
$$v(t) = v_0 - \mathrm{g}\,(t - t_0)$$

Ordinary Differential Equations (ODEs)

- Initial value problem

$$u(t_0) = u_0, \qquad \frac{\mathrm{d}u}{\mathrm{d}t}(t) = f(t, u(t), p)$$

- Simplest numerical solver: Euler's method

$$u_{n+1} = u_n + \Delta t f(t_n, u_n, p), \quad t_n = t_0 + n\Delta t$$

- Tons of more sophisticated methods

  https://docs.sciml.ai/DiffEqDocs/stable/solvers/ode_solve/

- Coding part I

Different methods for sensitivity analysis of ODEs

- 2 main ways of differentiating through an ODE (cf. last lecture for nonlinear problems)
  - **Discrete sensitivity analysis** (AD on solver operations)

    "Exact gradient of the approximation" / "discretize-then-differentiate"

  - **Continuous sensitivity analysis** (custom rules)

    "Approximation of the exact gradient" / "differentiate-then-discretize"

Different methods for sensitivity analysis of ODEs

- 2 main ways of differentiating through an ODE (cf. last lecture for nonlinear problems)
  - **Discrete sensitivity analysis** (AD on solver operations)

    "Exact gradient of the approximation" / "discretize-then-differentiate"

  - **Continuous sensitivity analysis** (custom rules)

    "Approximation of the exact gradient" / "differentiate-then-discretize"

- … with two modes each (forward/tangent and reverse/adjoint)

→ Optimal choice depends on number of states/parameters and system properties

TODAY: **continuous adjoint**

What are sensitivities/derivatives good for?

- Sensitivity analysis
  - How sensitive is the solution to changes in the initial conditions/parameters?
- Parameter estimation
  - What parameters match the observed data?
- Control
  - How can I drive the solution to a certain final state?

What are sensitivities/derivatives good for?

- Sensitivity analysis
  - How sensitive is the solution to changes in the initial conditions/parameters?
- Parameter estimation
  - What parameters match the observed data?
- Control
  - How can I drive the solution to a certain final state?

- Let us consider

$$\frac{\partial z(T)}{\partial g} \left[\text{i.e., } z(T) = G(u(T))\right]$$

of our free falling ball example, where G is a terminal cost.

$\rightarrow$ easy when analytical solution is available

$$z(t) = z_0 + v_0 t - \frac{\mathrm{g}}{2}(t - t_0)^2,$$
$$v(t) = v_0 - \mathrm{g}\,(t - t_0)$$

January 30, 2023

1 Derivation of the continuous-adjoint sensitivity method for ordinary differential equations

Suppose a cost function $G(u, p)$ evaluated on the complete solution $u(t)$ of the ordinary differential equation $u'(t, p) = f(u, p, t)$, $u(t_0) = u_0$, i.e.:

$$G(u, p) = G(u(p), p) = \int_{t_0}^{T} g(u(t, p), p)\, \mathrm{d}t \tag{1}$$

The cost function requires the solution $u(t)$. How do we develop a numerical procedure for the derivative when $u(t)$ can only be obtained numerically? We only assume that we can numerically solve an ODE!

To derive the adjoint equation, introduce the Lagrange multiplier λ. That's a nice mathematical trick: add a zero, and then we can determine later what λ really is

$$\begin{aligned} I(u(p), p) &= G(u(p), p) - \int_{t_0}^{T} \lambda^T (\tfrac{\mathrm{d}u(t,p)}{\mathrm{d}t} - f(u, p, t))\mathrm{d}t \\ &= G(u(p), p) - \int_{t_0}^{T} \lambda^T (u'(t, p) - f(u, p, t))\mathrm{d}t \end{aligned} \tag{2}$$

(Why is that a zero? Recall $u'(t, p) = f(u, p, t)$). Let us differentiate Eq. (2) with respect to p

$$\frac{\mathrm{d}I(u(p),p)}{\mathrm{d}p} = \frac{\mathrm{d}G(u(p),p)}{\mathrm{d}p}$$

$$= \int_{t_0}^{T} \underbrace{\frac{\mathrm{d}g(u(t,p),p)}{\mathrm{d}u(t,p)}}_{=g_u} \underbrace{\frac{\mathrm{d}u(t,p)}{\mathrm{d}p}}_{=s,\text{sensitivity}} + \underbrace{\frac{\mathrm{d}g(t,p)}{\mathrm{d}p}}_{=g_p} \mathrm{d}t - \int_{t_0}^{T} \lambda^T \left(\underbrace{\frac{\mathrm{d}}{\mathrm{d}p}\frac{\mathrm{d}u(t,p)}{\mathrm{d}t}}_{=\frac{\mathrm{d}}{\mathrm{d}t}\frac{\mathrm{d}u(t,p)}{\mathrm{d}p}=s'} - \underbrace{\frac{\mathrm{d}f(u,p,t)}{\mathrm{d}u(t,p)}}_{=f_u} \frac{\mathrm{d}u(t,p)}{\mathrm{d}p} - \underbrace{\frac{\mathrm{d}f(u,p,t)}{\mathrm{d}p}}_{=f_p} \right) \mathrm{d}t \,. \tag{3}$$

How does $u(t,p)$ change with respect to p? That isn't easy to say.. We aim to isolate s and then set λ so we can drop it! Let's take a look at the second term

$$\int_{t_0}^{T} \lambda^T \left(s' - f_u s - f_p\right) \mathrm{d}t = \int_{t_0}^{T} \lambda^T s' \,\mathrm{d}t - \int_{t_0}^{T} \lambda^T \left(f_u s + f_p\right) \mathrm{d}t \,. \tag{4}$$

We don't want s and s'. To transform the s' into an s, we apply integration by parts

$$\int_{t_0}^{T} \lambda^T s' \,\mathrm{d}t - \int_{t_0}^{T} \lambda^T \left(f_u s + f_p\right) \mathrm{d}t = [\lambda^T(t)s(t)]|_{t_0}^{T} - \int_{t_0}^{T} \lambda'^T s \,dt - \int_{t_0}^{T} \lambda^T \left(f_u s + f_p\right) \mathrm{d}t \,. \tag{5}$$

Insert back into Eq. (3):

$$\frac{\mathrm{d}G(u(p),p)}{\mathrm{d}p} = \int_{t_0}^{T} g_u s + g_p \mathrm{d}t - [\lambda^T(t)s(t)]|_{t_0}^{T} + \int_{t_0}^{T} \lambda'^T s \mathrm{d}t + \int_{t_0}^{T} \lambda^T \left(f_u s + f_p\right) \mathrm{d}t \,. \tag{6}$$

Rearrange terms (group terms in s):

$$\frac{\mathrm{d}G(u(p),p)}{\mathrm{d}p} = \int_{t_0}^{T} g_p + \lambda^T f_p \mathrm{d}t - [\lambda^T(t)s(t)]|_{t_0}^{T} + \int_{t_0}^{T} (\lambda'^T + \lambda^T f_u + g_u) s \,\mathrm{d}t \,. \tag{7}$$

Now, we're free to choose λ! Let us take

$$\lambda'^T = -\lambda^T f_u - g_u \tag{8}$$

$$\lambda(T) = 0 \,, \tag{9}$$

so that the second term and one of the boundary terms vanish. That's an ODE. We call it the "adjoint ODE problem".

Thus, we get

$$\frac{\mathrm{d}G(u(p),p)}{\mathrm{d}p} = \int_{t_0}^{T} g_p + \lambda^T f_p \mathrm{d}t + \lambda^T(t_0)s(t_0) \,. \tag{10}$$

Note that

$$s(t_0) = \frac{\mathrm{d}u(t_0,p)}{\mathrm{d}p} = 0\,. \tag{11}$$

How do you calculate the remaining integral

$$\frac{\mathrm{d}G(u(p),p)}{\mathrm{d}p} = \int_{t_0}^{T} g_p + \lambda^T f_p \mathrm{d}t\,? \tag{12}$$

Note that $\frac{\mathrm{d}G(u(p),p)}{\mathrm{d}p}$ requires $\lambda(t)$, and $\lambda(t)$ requires $u(t)$.

Procedure:

- solve primal ODE $u'(t,p) = f(u,p,t)$, $u(t_0) = u_0$ forward in time.
- solve adjoint ODE $\lambda' = -{f_u}^T\lambda - g_u^T$, $\lambda(T) = 0$ backward in time (cf. reverse-mode, adjoint methods).
- integrate Eq. (12).

Note that ODEs are reversible! So we do not necessarily have to store the full ODE in memory but can compute it backward in time from the final time $u(T)$. Therefore, we can solve the primal and the adjoint ODE together in lockstep. (Unfortunately, ODE solvers are not fully reversible in general, so this approach tends to be unstable. We can use checkpoints to reset the ODE solution. Other approaches: interpolating adjoints, where we recompute the ODE solution forward in time (usually) between checkpoints.)

Note that we can also transform the last equation into an ODE. An integral

$$F(T) = \int_{t_0}^{T} f(t)\mathrm{d}t \tag{13}$$

can be written as an ODE

$$w' = f(t), \qquad w(t_0) = 0\,. \tag{14}$$

So that

$$F(T) = \int_{t_0}^{T} f(t)\mathrm{d}t = w(T)\,. \tag{15}$$

For instance, if we apply Euler's method (see the beginning of the lecture):

$$w(t+\Delta t) = w(t) + \Delta t f(t) \tag{16}$$

$$w(t+\Delta t) = w(t) + \Delta t f(t) + \Delta t f(t+\Delta t) \tag{17}$$

$$\dots \tag{18}$$

$$w(t+n\Delta t) = w(t) + \Delta t \sum_{i=0}^{n-1} f(t+i\Delta t) \tag{19}$$

$$F(T) = \int_{t_0}^{T} f(t)\mathrm{d}t \approx \Delta t \sum_{i}^{n} f(t + i\Delta t) \tag{20}$$

Thus, we define

$$w' = g_p + \lambda^T f_p, \qquad w(T) = 0 \tag{21}$$

so that

$$\frac{\mathrm{d}G(u(p),p)}{\mathrm{d}p} = -w(t_0)\,. \tag{22}$$

Therefore, we can solve an augmented ODE system:

$$\begin{aligned} u' &= f(u,p,t), & u(T) &= uT \\ \lambda' &= -{f_u}^T\lambda - g_u^T, & \lambda(T) &= 0 \\ w' &= g_p + \lambda^T f_p, & w(T) &= 0 \end{aligned} \tag{23}$$

```
cd(@__DIR__)
using Pkg;
Pkg.activate(".")
Pkg.instantiate()

z0 = 5.0 # initial height
v0 = 0.1 # initial velocity
u0 = [z0, v0] # initial ODE state
p = [10.0] # gravitational constant
tspan = (0.0, 1.0) # integration time
dt = 0.001 # time discretization

# system
function f(u, p, t)
    dz = u[2]
    dv = -p[1]
    [dz, dv]
end

### solving the ODE!

# hand-written Euler solver
function my_euler(f, u0, tspan, p, dt)
    u = u0
    t = tspan[1]:dt:tspan[2] # time grid
    for ti in t[1:end-1]
        u = u .+ f(u, p, ti) .* dt
    end
    u
end

# Jacobian ∂f/∂u
function fu(u, p, t)
    [0 1
     0 0]
end

# Parameter Jacobian ∂f/∂p
function fp(u, p, t)
    # 2x1 matrix
    [0;; -1]'
end

### Continuous-adjoint sensitivity analysis
function f_augmented(z, p, t)
    u, λu, λp = z

    dz = u[2]
    dv = -p[1]
    du = [dz, dv]
```

```
    dλu = -λu' * fu(u, p, t)
    dλp = -λu' * fp(u, p, t)

    (du, dλu', dλp')
end

uend = my_euler(f, u0, tspan, p, dt)
z0 = (uend, [1.0, 0.0], zeros(1))
my_euler(f_augmented, z0, reverse(tspan), p, -dt)
```

MIT OpenCourseWare
https://ocw.mit.edu

18.S096 Matrix Calculus for Machine Learning and Beyond
Independent Activities Period (IAP) 2023

For information about citing these materials or our Terms of Use, visit: https://ocw.mit.edu/terms.