PA081: Programování numerických výpočtů

7. Automatické vyhodnocení derivací

jaro 2016

Motivace

- kromě funkce dokážeme vyhodnotit i její derivace
  - první, druhé, ...
  - parciální
- mnohé numerické problémy jsou lépe řešitelné metodami s dostupnou derivací
- metody s derivací zpravidla rychleji konvergují
  - Newton vs. metoda sečen

Motivace

- kromě funkce dokážeme vyhodnotit i její derivace
  - první, druhé, ...
  - parciální
- mnohé numerické problémy jsou lépe řešitelné metodami s dostupnou derivací
- metody s derivací zpravidla rychleji konvergují
  - Newton vs. metoda sečen

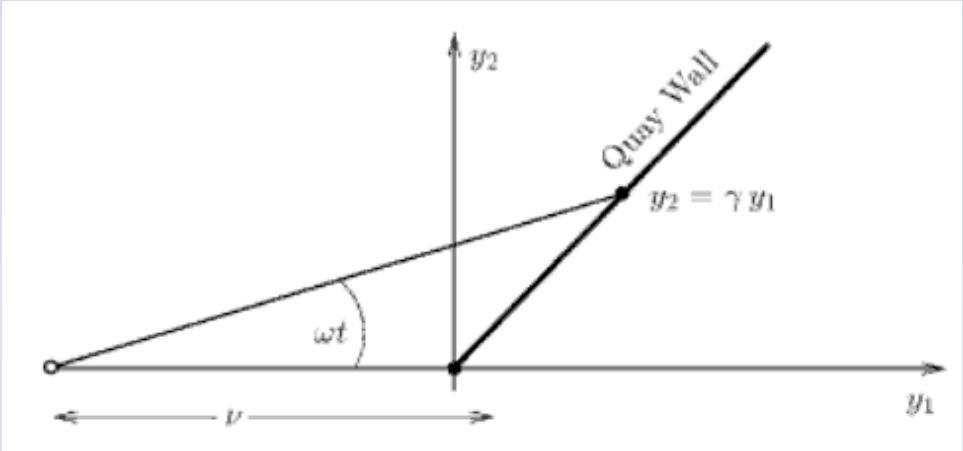
- aproximace konečnou diferencí nestačí
- zpracovávaná funkce nebývá triviální
- analytické vyjádření derivace je náročné
  - zdroj nebezpečných chyb
  - komplikace při změnách

Ruční postup

- funkce je implementována jako program
- hledáme program, který vedle funkční hodnoty spočte i derivace
- `http://www.autodiff.org`

Ruční postup

- funkce je implementována jako program
- hledáme program, který vedle funkční hodnoty spočte i derivace
- `http://www.autodiff.org`
- příklad s majákem



- bod dopadu paprsku na nábřeží

$$y_1 = \frac{\nu \tan \omega t}{\gamma - \tan \omega t} \qquad y_2 = \gamma y_1$$

Ruční postup

- vyjádřeno jako jednoduchý program

$$a := \tan \omega t; \qquad y_1 := va/(y-a); \qquad y_2 := yy_1;$$

- chceme spočítat i rychlost pohybu, tj. derivaci v čase
- výpočet současně s původní funkcí

$$\begin{array}{ll} & \dot{t} := 1 \\ a_1 := \omega t & \dot{a}_1 := \omega \dot{t} \\ a := \tan a_1 & \dot{a} := \dot{a}_1/(\cos a_1)^2 \\ a_2 := y - a & \dot{a}_2 := -\dot{a} \\ y_1 := va/a_2 & \dot{y}_1 := v(\dot{a}a_2 - a\dot{a}_2)/a_2^2 \\ y_2 := yy_1 & \dot{y}_2 := y\dot{y}_1 \end{array}$$

- pouze mechanická aplikace základních pravidel

Automatické derivování

- transformace zdrojového kódu (např. ADIC)
- přetížení operátorů a funkcí (ADOL-C)
- zjednodušený příklad

```
class adouble { double x; double dx; ... }

adouble operator * (adouble a, adouble b) {
    adouble r;
    r.x = a.x * b.x;
    r.dx = a.x * b.dx + a.dx * b.x;
    return r;
}

adouble v1, v2, v3;
...
v1 = v2 * v3;
```

Automatické derivování

- aktivní proměnná
  - nezávislé proměnné, podle kterých chceme derivovat
  - všechny další proměnné na nich během výpočtu závisející
- vyhrazený typ `adouble`
- přiřazení `adouble` → `double` je chybné
  - ztrácíme informaci o derivacích tam, kde je potřebná
  - explicitní přístup metodou `value()`
- více nezávislých proměnných
  - s každou aktivní proměnnou udržujeme vektor parciálních derivací

Typický program

```
#define NUMBER_DIRECTIONS 3
#define ADOLC_TAPELESS
#include <adolc/adouble.h>

func(adouble[], const adouble[]);

adouble x[3];
adouble y[4];
...
for (i=0; i<3; i++)
    for (j=0; i<3; i++)
        x[i].setADValue(j,i==j ? 1.0 : 0.0);

func(y,x);

for (i=0; i<4; i++) {
    cout << y.getValue() << ": ";
    for (j=0; j<3; j++) cout << y.getADValue(j) << ", ";
    cout << endl;
}
```

Není to tak jednoduché

- nediferencovatelné funkce (`fmod`)
  - nelze použít
- absolutní hodnota (`fabs`)
  - v 0 pravá nebo levá derivace ± 1
  - případně položíme uměle rovnu 0
- minimum a maximum (`fmin`, `fmax`)
  - lze nahradit $(a + b \pm |a - b|)/2$
- nedefinované na celém $\mathbb{R}$ (`sqrt`, `pow`, `tan`, ...)
  - derivaci položíme uměle `NaN`, resp. `Inf`

Zpětné vyhodnocení

- množství potřebných operací $O(mk)$ pro $f: \mathbb{R}^m \to \mathbb{R}^n$
  - m počet nezávislých proměnných
  - n počet závislých proměnných
  - k počet kroků výpočtu
- nepříjemné s rostoucím m
- velká třída problémů velkým m ale malým n
  - optimalizace funkce více proměnných ($n = 1$)

Zpětné vyhodnocení

- množství potřebných operací $O(mk)$ pro $f: \mathbb{R}^m \to \mathbb{R}^n$
  - m počet nezávislých proměnných
  - n počet závislých proměnných
  - k počet kroků výpočtu
- nepříjemné s rostoucím m
- velká třída problémů velkým m ale malým n
  - optimalizace funkce více proměnných ($n = 1$)
- počítali jsme citlivost pomocných proměnných na změny vstupu
- obrácený postup, citlivost výstupu (závislých proměnných) na změny pomocných
- výpočet od konce

Zpětné vyhodnocení

- triviální příklad $y = \sin x^2$
- tedy po krocích $a_1 = x^2$, $y = \sin a_1$
- formálním derivováním

$$\partial y / \partial a_1 = \cos a_1 \qquad \partial a_1 / \partial x = 2x$$

- dostáváme $\partial y / \partial x = 2x \cos a_1$

Zpětné vyhodnocení

- triviální příklad $y = \sin x^2$
- tedy po krocích $a_1 = x^2$, $y = \sin a_1$
- formálním derivováním

$$\partial y / \partial a_1 = \cos a_1 \qquad \partial a_1 / \partial x = 2x$$

- dostáváme $\partial y / \partial x = 2x \cos a_1$
- označujeme $\bar{a} = \partial y / \partial a$
- obecně v konkrétním kroku $a_i = f(a_j, a_k, \dots)$ derivujeme

$$\partial a_i / \partial a_j = g(a_j, a_k, \dots)$$

- spolu se známým $\bar{a}_i = \partial y / \partial a_i$ dostaneme

$$\bar{a}_j = \partial y / \partial a_j = g(a_j, a_k, \dots)\bar{a}_i$$

Zpětné vyhodnocení

- triviální příklad $y = \sin x^2$
- tedy po krocích $a_1 = x^2$, $y = \sin a_1$
- formálním derivováním

$$\partial y / \partial a_1 = \cos a_1 \qquad \partial a_1 / \partial x = 2x$$

- dostáváme $\partial y / \partial x = 2x \cos a_1$
- označujeme $\bar{a} = \partial y / \partial a$
- obecně v konkrétním kroku $a_i = f(a_j, a_k, \dots)$ derivujeme

$$\partial a_i / \partial a_j = g(a_j, a_k, \dots)$$

- spolu se známým $\bar{a}_i = \partial y / \partial a_i$ dostaneme

$$\bar{a}_j = \partial y / \partial a_j = g(a_j, a_k, \dots)\bar{a}_i$$

- tyto konstrukce jsou pouze ilustrativní, zjednodušené a v principu nekorektní

Zpětné vyhodnocení

- např. funkce $y = \sin(x_1/x_2)e^{x_3}$

$$
\begin{aligned}
a_1 &:= x_1/x_2 \\
a_2 &:= \sin a_1 \\
a_3 &:= e^{x_3} \\
y &:= a_2 a_3 \\
\bar{y} &:= 1 \\
\bar{a}_2 &:= a_3 \bar{y} \\
\bar{a}_3 &:= a_2 \bar{y} \\
\bar{x}_3 &:= e^{x_3} \bar{a}_3 \\
\bar{a}_1 &:= \bar{a}_2 \cos a_1 \\
\bar{x}_2 &:= -\bar{a}_1/x_2 \\
\bar{x}_1 &:= \bar{a}_2/x_2
\end{aligned}
$$

- v $O(nk)$ operacích máme všechny $\partial y/\partial x_i$

Stopa výpočtu

- derivace se vyhodnocují od konce
  - tj. až poté, co proběhl výpočet funkce
  - proto „zpětné“ (reverse) vyhodnocení
- výpočet funkce je nutné nějak zaznamenat
- ADOL-C: datová struktura (i soubor) páska (tape)
- záznam posloupnosti operací, nikoli konkrétních hodnot
  - lze recyklovat pro jiný vstup
  - případné rozdílnosti vyhodnocení (`a>b ? c : d`) jsou detekovány
- použije se pro vyhodnocení funkce, gradientu/Jakobiánu, Hessiánu, ...

Instrumentace pro zpětné výhodnocení

```
#include <adolc/adouble.h>

void func(
    double px[4],    /* vstup */
    double py[4],    /* vystup */
) {
    adouble x[4],y[4],aux;
    trace_on(0);                              /* paska 0 */
    for (i=0; i<4; i++) x[i] <<= px[i];
    aux = x[1] * x[4] + exp(x[2]);            /* vlastni vypocet */
    ...
    y[4] = sin(aux);
    for (i=0; i<4; i++) y[i] >>= py[i];
    trace_off();
}
```

Drivery

- ADOL-C poskytuje celou řadu „driver“ funkcí
  - vstupem je vždy páska a konkrétní vstupní hodnoty
  - driver provede výpočet na základě záznamu na pásce
- funkční hodnota funkce $\mathbb{R}^n \to \mathbb{R}^m$
  - `function(short tag, int m, int n, double x[n],double y[m])`
- gradient funkce $\mathbb{R}^n \to \mathbb{R}$
  - `gradient(short tag, int n, double x[n],double g[n])`
- Jakobián funkce $\mathbb{R}^n \to \mathbb{R}^m$
  - `jacobian(short tag, int m, int n, double x[n],double J[m][n])`
- Hessián (matice druhých derivací) funkce $\mathbb{R}^n \to \mathbb{R}$
  - `hessian(short tag, int m, int n, double x[n],double H[n][n])`

Drivery

- optimalizované varianty pro Newtonovu metodu
- řešení různých typů diferenciálních rovnic
- speciální varianty pro řídké Jakobiány a Hessiány
- tenzory vyšších derivací
- implicitně definované funkce

$$G(\mathbf{x}, \mathbf{y}) = 0$$

- obecné – plná kontrola nad zpracováním pásky

Rekapitulace zpětného vyhodnocení

- aktivní proměnné deklarovat jako `adouble`
- výpočet označit `trace_on()` ... `trace_off()`
- zavolat instrumentovanou funkci jednou
  - s typickými hodnotami vstupů
  - vyprodukuje záznam výpočtu – pásku
- opakovaně volat potřebné drivery
  - případně na různé vstupy
- pro jinou posloupnost výpočtu zavolat instrumentovanou funkci s jinou páskou

Další materiál

- rozsáhlá problematika, řada implementací
- včetně obsáhlé teorie
- `http://www.autodiff.org`
  - přehled nástrojů, FAQ, ...
- `https://projects.coin-or.org/ADOL-C`
  - implementac a dokumentace ADOL-C
  - včetně řady příkladů
- Griewank, Walter. *Evaluating Derivatives.* 2008