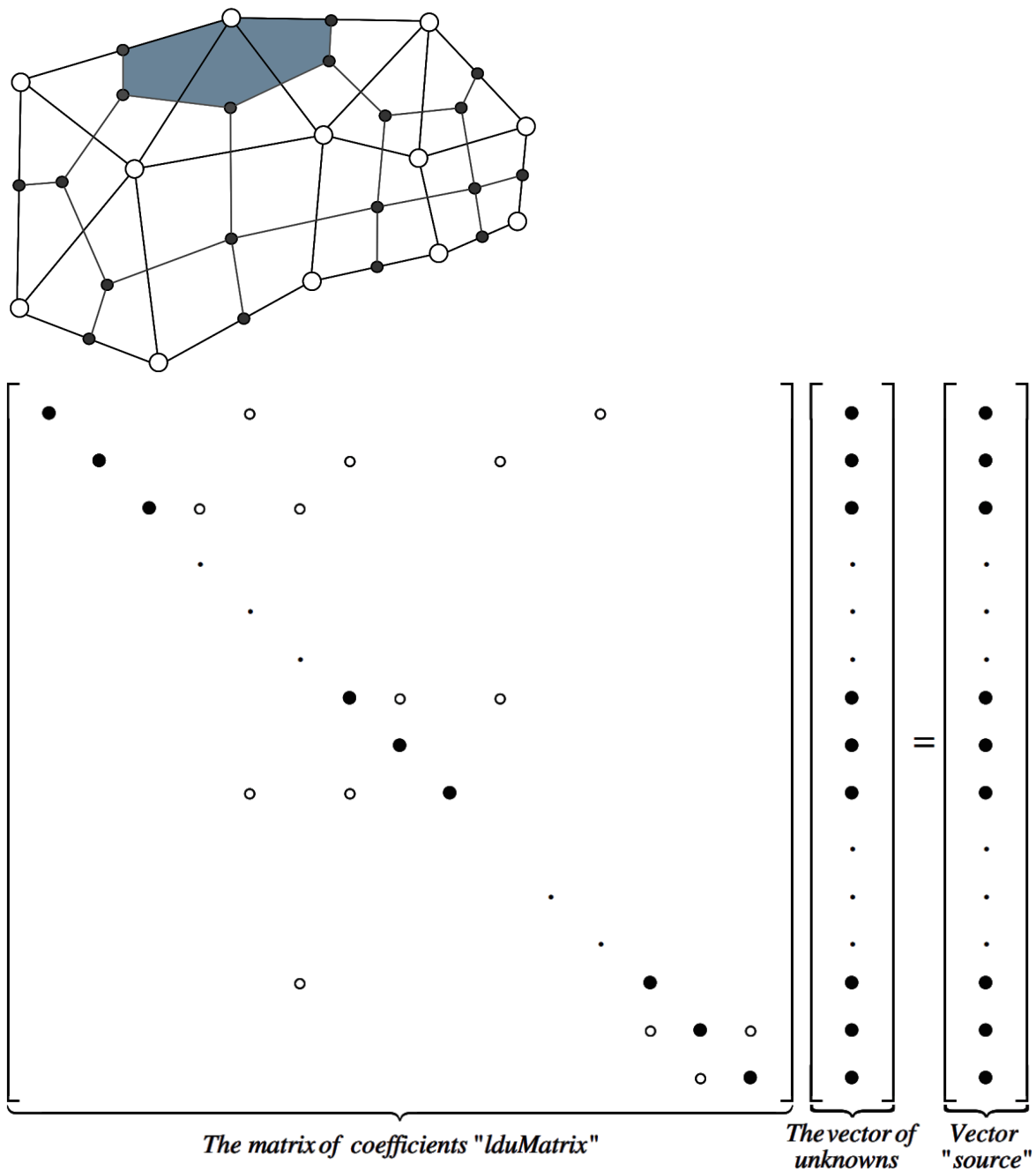


—



Процесс описывает следующее уравнение:

$$\underbrace{\frac{\partial}{\partial t}(\rho\phi)}_{\text{unsteady term}} + \underbrace{\nabla \cdot (\rho\mathbf{v}\phi)}_{\text{convection term}} = \underbrace{\nabla \cdot (\Gamma^\phi \nabla \phi)}_{\text{diffusion term}} + \underbrace{Q^\phi}_{\text{source term}}$$

Так как матрица состоит из производных, то дифференцирование уравнений жидкости, которые даны, это первоочередная задача, возникающая в данной области.

Не менее актуально дифференцирование при нахождении корня системы линейных уравнений, например, при применении метода Ньютона или метода наименьших квадратов.

Повторение из лекции 11: методы решения СЛАУ

$$\mathbf{f}(\mathbf{x}) = 0, \quad \mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$$

$$\mathbf{f}(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{f}(\mathbf{x}) + \nabla\mathbf{f}(\mathbf{x})\Delta\mathbf{x} + O(\|\Delta\mathbf{x}\|^2)$$

Здесь $\nabla\mathbf{f}(\mathbf{x})$ - градиент функции:

$$\nabla\mathbf{f} = \begin{bmatrix} \frac{\partial\mathbf{f}}{\partial x_1} & \cdots & \frac{\partial\mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

Поскольку функция n -значная и \mathbf{x} – вектор длины n , то матрица получается квадратной. Эту матрицу еще называют *якобиан*.

Невязки - разности значений модели и наблюдений в точках, а минимизируется сумма квадратов этих разностей.

$$\arg \min_{\mathbf{x}} S(\mathbf{x}) = ?$$

$$S(\mathbf{x}) = \sum_{j=1}^m r_j^2(\mathbf{x})$$

$$\nabla S(\mathbf{x}) = \mathbf{0} \Rightarrow f_i(\mathbf{x}) = \sum_{j=1}^m \frac{\partial r_j(\mathbf{x})}{\partial x_i} r_j(\mathbf{x}) = 0, \quad i = 1..n$$

$$\nabla\mathbf{f} \approx A^T A$$

$$A = \begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \cdots & \frac{\partial r_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_m}{\partial x_1} & \cdots & \frac{\partial r_m}{\partial x_n} \end{bmatrix}$$

Более подробно см. лекцию 11, пункт «Задачи, в которых появляются СЛАУ»

Применение дифференцирования (продолжение)

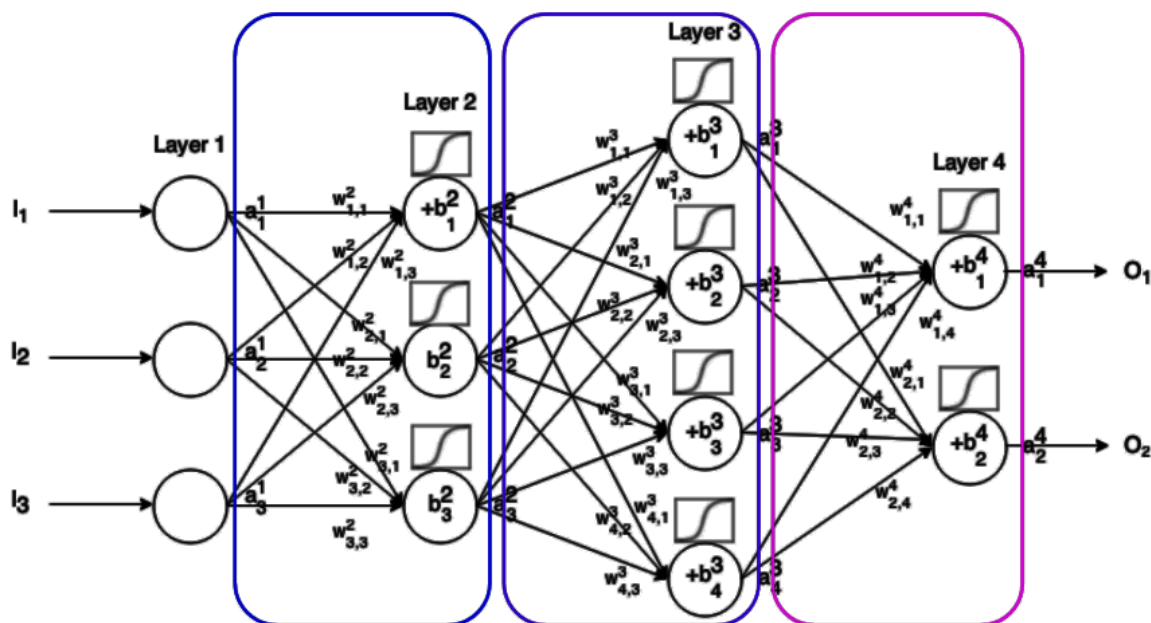
Продолжая тему модели и наблюдательных данных, если модель динамическая, то для составления матрицы A , чтобы подогнать состояние динамической системы в произвольные моменты времени к наблюдательным данным, необходимо на этапе численного интегрирования уравнений динамической системы знать частные производные от всех ее правых частей по всем переменным ее состояния.

$$\frac{d\mathbf{x}}{d\mathbf{P}} = \begin{bmatrix} \frac{dx^{(1)}}{dx_0^{(1)}} & \cdots & \frac{dx^{(1)}}{dx_0^{(n)}} & \frac{dx^{(1)}}{dp_1} & \cdots & \frac{dx^{(1)}}{dp_m} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{dx^{(n)}}{dx_0^{(1)}} & \cdots & \frac{dx^{(n)}}{dx_0^{(n)}} & \frac{dx^{(n)}}{dp_1} & \cdots & \frac{dx^{(n)}}{dp_m} \end{bmatrix}$$

$$\left(\frac{d\mathbf{x}}{d\mathbf{P}} \right)^* = \frac{d\mathbf{f}}{d\mathbf{P}} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \frac{d\mathbf{x}}{d\mathbf{P}}$$

$$\frac{d\mathbf{x}}{d\mathbf{P}}(t_0) = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & 0 \end{bmatrix}$$

В задачах, связанных с нейронными сетями, в период обучения даются некоторые входные данные и некоторые выходные, и требуется подстроить параметры нейронов (веса), чтобы входные данные соответствовали выходным. Для всего этого требуется подсчитывать большое количество градиентов, которые отражают ход вычислений между нейронами.



Итак, можно сказать, что современные компьютеры большую часть времени занимаются решением СЛАУ, либо дифференцированием для их составления, рассмотрим способы выполнения дифференцирования.

Виды дифференцирования

Рассмотрим функцию:

$$f(x) = \cos 2x + 2 \sin^2 x$$

1. Символьное дифференцирование дает результат: $f'(x) = 0$. Это математический способ, который можно выполнять на бумаге или на компьютере. Несмотря на простой алгоритм,

символьным дифференцированием занимаются преимущественно системы компьютерной алгебры, а программы для инженерных и научных расчетов занимаются им в меньшей степени.

2. Численное дифференцирование. Здесь функция $f(x)$ для алгоритмов является черным ящиком.

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h)$$

Схему можно немного модернизировать и повысить точность результата:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2)$$

3. Автоматическое дифференцирование. Данный способ заключается в применении правила взятия полной производной до тех пор, пока его можно применять.

$$f'(x) = \sin 2x \cdot 2 + 2(2 \sin x \cdot \cos x)$$

В случае применения для заданной функции видов дифференцирования, указанных в п.2 и п.3, в результате не получится 0.

Поговорим о преимуществах и недостатках каждого из видов дифференцирования.

Символьное дифференцирование

Достоинство: наиболее точное (потенциально)

Недостатки:

- не устраняет дублирование вычислений

Пример:

simplify	$\frac{\partial}{\partial x} \frac{1}{\cos(\log(2(x^2 + 3x + 5)))}$
----------	---

Results:

$$\frac{(2x+3) \tan(\log(2(x(x+3)+5))) \sec(\log(2(x(x+3)+5)))}{x(x+3)+5}$$

$$\frac{(2x+3) \tan(\log(2(x^2+3x+5))) \sec(\log(2(x^2+3x+5)))}{x^2+3x+5}$$

$$\frac{(2x+3) \tan(\log(x^2+3x+5) + \log(2)) \sec(\log(x^2+3x+5) + \log(2))}{x^2+3x+5}$$

Не учитывается, что выражения могут повторяться, что можно устранить дублирование введением временных переменных

- сложно формулируется при наличии ветвлений и циклов

Если в формуле есть суммы, ветвления внутри сумм или просто ветвления, то есть они являются частью вычислительной схемы, то символьное дифференцирование становится непрактичным.

Примеры:

$$\frac{f_{\text{fig-pm}}}{m} = \mu \text{Re} \left[\sum_{n=2}^{n_{\text{max}}} R^n \sum_{m=0}^n (\bar{C}_{nm} - i\bar{S}_{nm}) \nabla \bar{V}_{nm}(r, \lambda, \phi) \right]$$

$$\bar{V}_{nm}(r, \lambda, \phi) = N_{nm} \frac{\cos m\lambda + i \sin m\lambda}{r^{n+1}} P_n^m(\sin \phi)$$

$$N_{nm} = \sqrt{\frac{(n-m)!(2n+1)!(2-\delta_{0m})}{(n+m)!}}$$

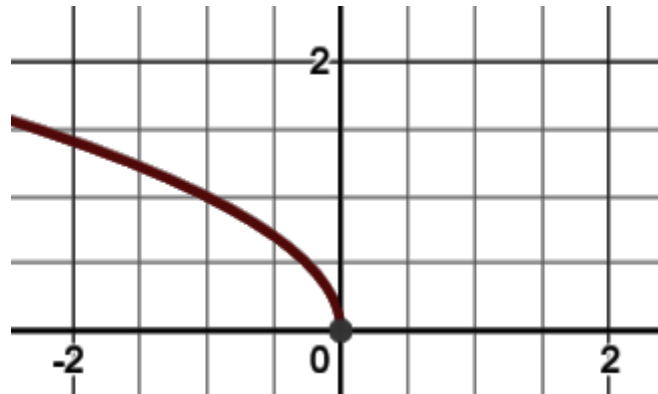
Численное дифференцирование

Достоинство: просто в реализации

Недостатки:

- наиболее затратно по времени
- может выходить за область определения f

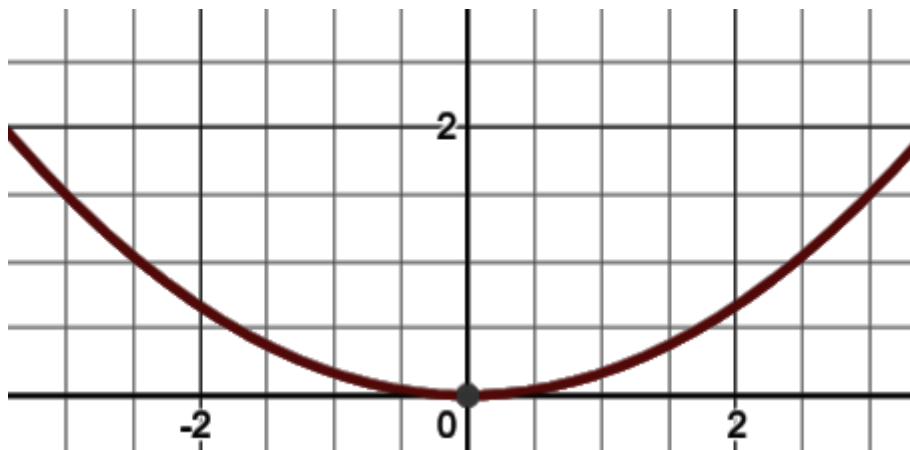
Например, $f(x) = \sqrt{-x}$. При нахождении производной в точке 0, посчитать производную по формуле численного дифференцирования невозможно.



Решением может быть проверка области определения и изменение схемы взятия производной или приравнивание ее к 0.

Особенность:

- выдает ненулевой результат в локальном экстремуме f



Если для данной параболы искать производную в точке 0 по формуле, указанной ниже, то значение будет ненулевым.

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h)$$

Можно использовать измененную схему:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2)$$

Но в большинстве численных методов используется именно первая схема.

Тем не менее, эта особенность не является проблемой в некоторых ситуациях, а может быть даже и преимуществом, потому что происходит косвенный учет членов второго порядка, что можно назвать компенсацией того, что численные методы отбрасывают члены второго порядка, и, оказавшись в точке локального экстремума, не будут «считать», что выход из точки не приведет к изменению функции.

Автоматическое дифференцирование

Недостаток: сложно в реализации

Достоинства:

- наиболее быстрое за счет исключения дублирующих вычислений
- легко формируется при наличии циклов, ветвлений, подпроцедур, временных переменных и пр.

Пример

Уравнение Кеплера:

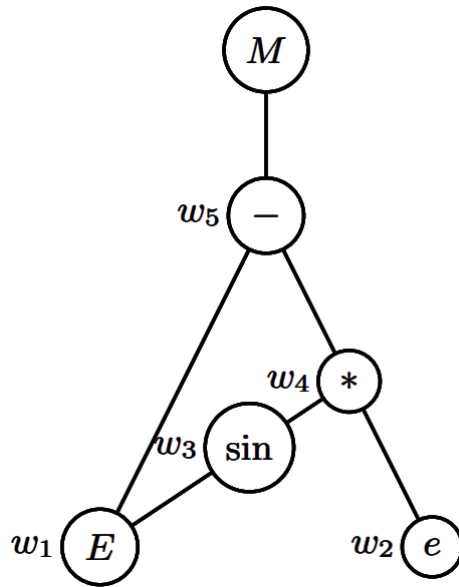
$$M = E - e \sin E$$

Программный код, соответствующий этой формуле:

```
func  $M(E, e)$ :
     $w_1$  =  $E$ 
     $w_2$  =  $e$ 
     $w_3$  = sin( $w_1$ )
     $w_4$  =  $w_3 * w_2$ 
     $w_5$  =  $w_1 - w_4$ 
     $M$  =  $w_5$ 
    return( $M$ )
```

Здесь w_i – временные переменные.

Схематично процедуру можно изобразить так:



Дифференцирование будем производить по обоим входным параметрам.

```
func dM(E, e):
```

$$\frac{\partial w_1}{\partial E} = 1$$

$$\frac{\partial w_1}{\partial e} = 0$$

$$w_1 = E$$

$$\frac{\partial w_2}{\partial E} = 0$$

$$\frac{\partial w_2}{\partial e} = 1$$

$$w_2 = e$$

$$\frac{\partial w_3}{\partial E} = \cos(w_1) * \frac{\partial w_1}{\partial E}$$

$$\frac{\partial w_3}{\partial e} = \cos(w_1) * \frac{\partial w_1}{\partial e}$$

$$w_3 = \sin(w_1)$$

$$\frac{\partial w_4}{\partial E} = \frac{\partial w_3}{\partial E} * w_2 + w_3 * \frac{\partial w_2}{\partial E}$$

$$\frac{\partial w_4}{\partial e} = \frac{\partial w_3}{\partial e} * w_2 + w_3 * \frac{\partial w_2}{\partial e}$$

$$w_4 = w_3 * w_2$$

$$\frac{\partial w_5}{\partial E} = \frac{\partial w_1}{\partial E} - \frac{\partial w_4}{\partial E}$$

$$\frac{\partial w_5}{\partial e} = \frac{\partial w_1}{\partial e} - \frac{\partial w_4}{\partial e}$$

$$w_5 = w_1 - w_4$$

$$\frac{\partial f_1}{\partial E} = \frac{\partial w_5}{\partial E}$$

$$\frac{\partial f_2}{\partial E} = \frac{\partial w_5}{\partial E}$$

$$M = w_5$$

```
return(M,  $\frac{\partial M}{\partial E}$ ,  $\frac{\partial M}{\partial e}$ )
```


Функция dM перед присваиванием в каждую из временных переменных делается присваивание двух (их может быть и больше) частных производных. Также видно, что для вычисления новых частных производных можно использовать уже посчитанные ранее, а также сами значения временных переменных.

Данная схема называется прямым автоматическим дифференцированием, оно уместно в ситуациях, когда присутствует достаточно большое количество входных данных и умеренное количество параметров.

Есть и схема обратного автоматического дифференцирования, которая актуальна для задач, связанных с нейронными сетями, где количество параметров достаточно большое.

Существуют способы комбинировать прямое и обратное автоматическое дифференцирования, так как схемы не гарантируют оптимальной работы алгоритма.

Есть и более глубокие аспекты, связанные с дифференцированием программного кода, например, дифференцирование с рекурсией. С практической точки зрения существует множество пакетов для автоматического дифференцирования.