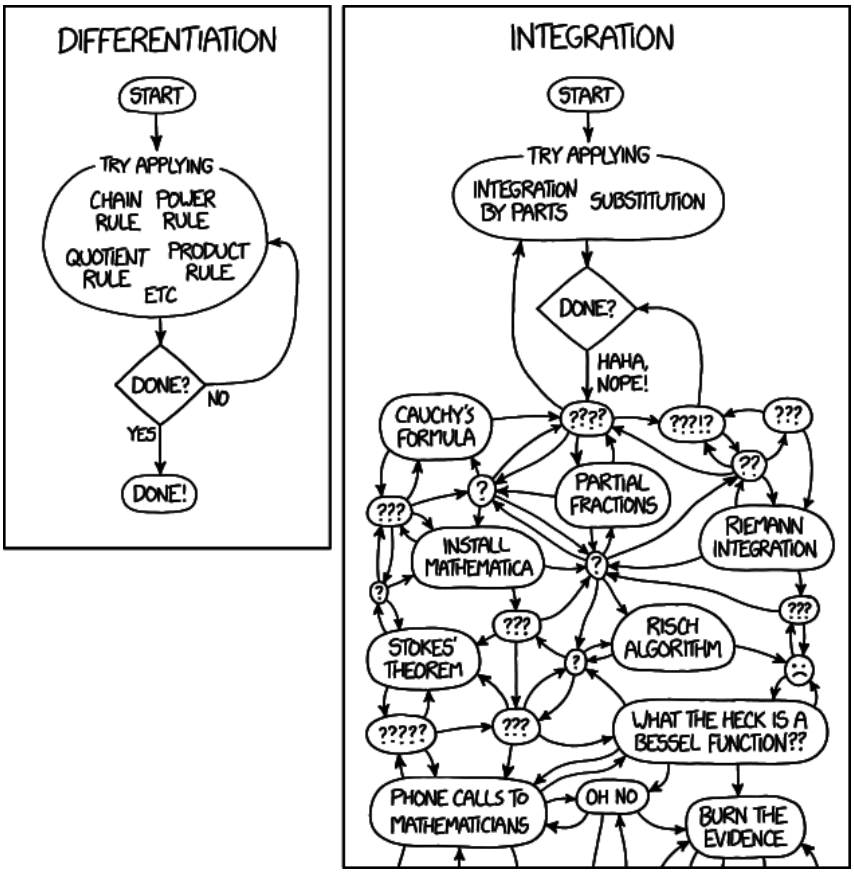
Дифференцирование

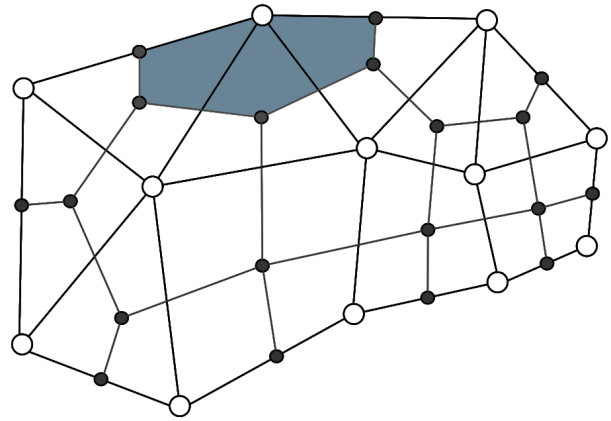
Задача дифференцирования может быть решена разными подходами. Несмотря на то, что задача дифферецирования кажется более простой, чем, например, задача интегрирования, в современных инженерных и научных задачах проблемы, связанные с дифференцированием, актуальны не менее, чем проблемы интегрирования.

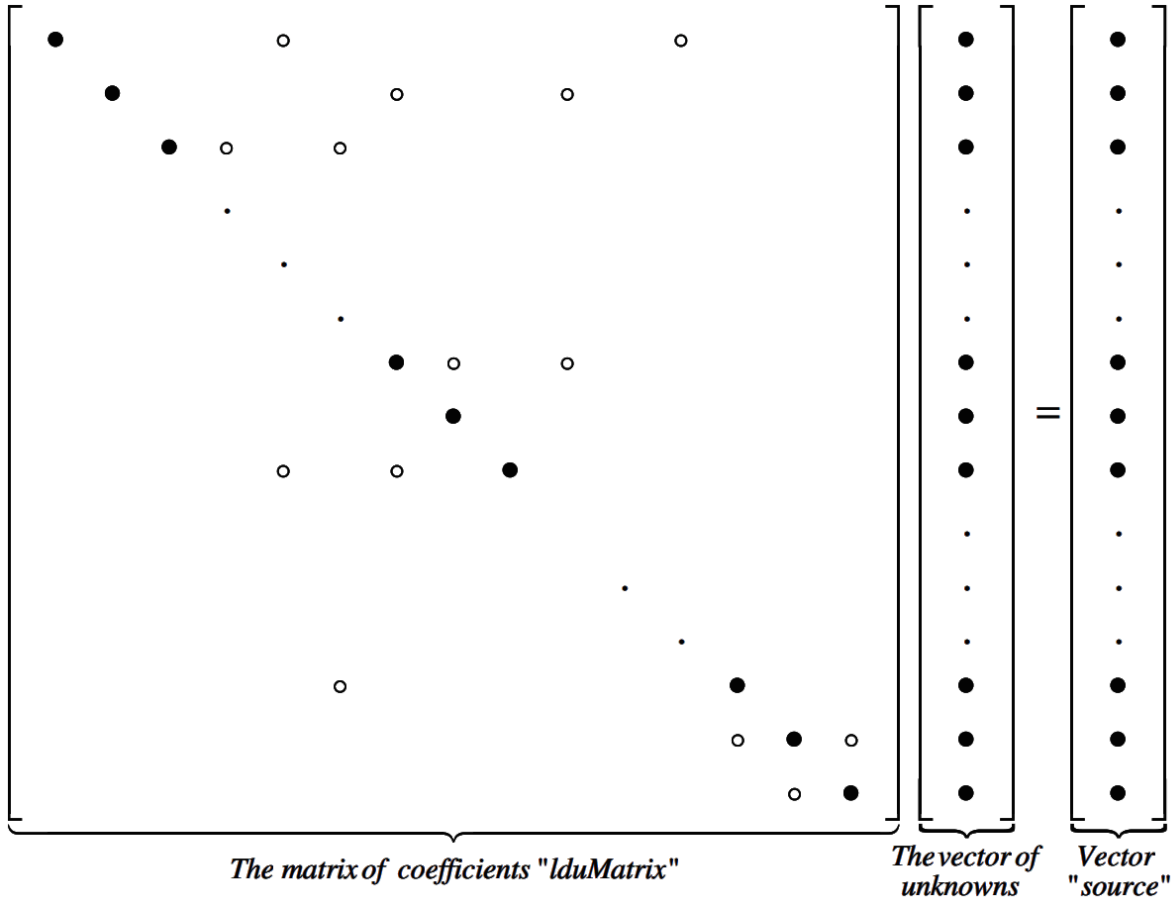


Рассмотрим несколько задач, где возникает необходимость дифференцирования.

Применение дифференцирования

В задаче гидродинамики для расчета течения жидкости в некоторой области, дискретизированной сеткой, необходимо решать систему уравнений, в матрице которой находятся производные потока жидкости по числовым характеристикам в узлах сетки (например, по давлению жидкости).





Процесс описывает следующее уравнение:

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

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 про методы решения СЛАУ

$$\mathbf{f}(\mathbf{x}) = 0, \;\; \mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$$
$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x_0}) + \nabla\mathbf{f}(\mathbf{x} - \mathbf{x_0}) + O(||\mathbf{x} - \mathbf{x_0}||^2)$$

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

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

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

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

$$\begin{aligned} \arg\min_{\mathbf{x}} S(\mathbf{x}) &= ? \\ S(\mathbf{x}) &= \textstyle\sum_{j=1}^{m} r_j^2(\mathbf{x}) \end{aligned}$$

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

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

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

> Более подробно см. пункт «Задачи, в которых появляются СЛАУ» в вышеуказанной лекции

Применение дифференцирования (продолжение)

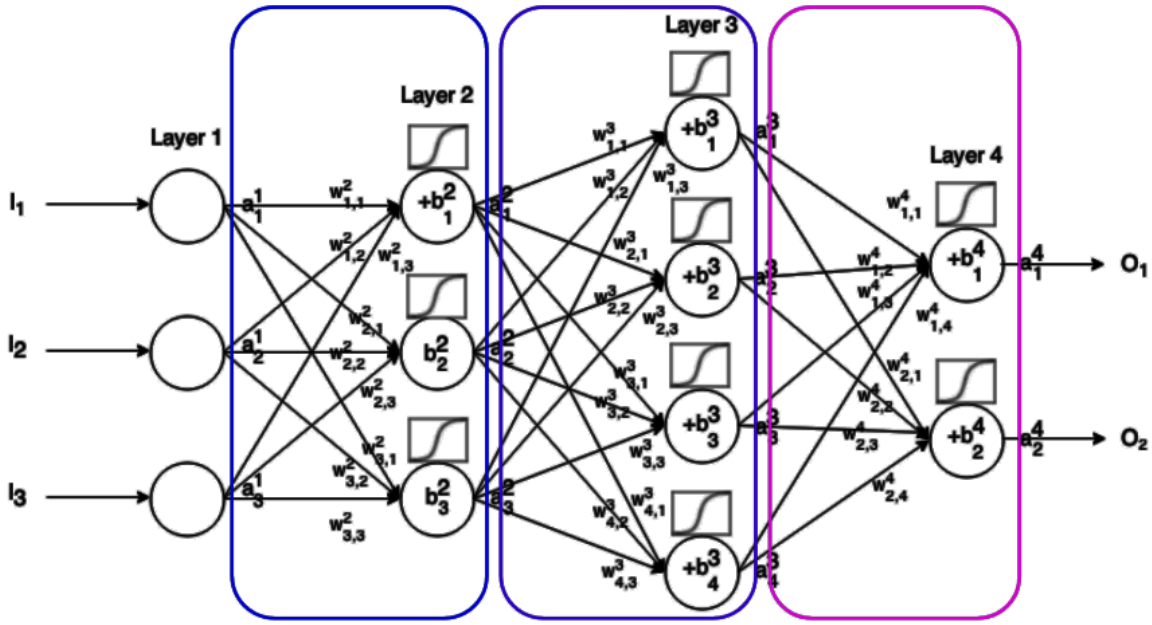
Продолжая тему модели и наблюдательных данных: если модель динамическая, то для составления матрицы A, чтобы подогнать состояние динамической системы в произвольные моменты времени к наблюдательным данным, необходимо на этапе численного интегрирования уравнений динамической системы знать частные производные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

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\mathbf{P}} = \begin{bmatrix} \frac{\mathrm{d}x^{(1)}}{\mathrm{d}x_0^{(1)}} & \cdots & \frac{\mathrm{d}x^{(1)}}{\mathrm{d}x_0^{(n)}} & \frac{\mathrm{d}x^{(1)}}{\mathrm{d}p_1} & \cdots & \frac{\mathrm{d}x^{(1)}}{\mathrm{d}p_m} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\mathrm{d}x^{(n)}}{\mathrm{d}x_0^{(1)}} & \cdots & \frac{\mathrm{d}x^{(n)}}{\mathrm{d}x_0^{(n)}} & \frac{\mathrm{d}x^{(n)}}{\mathrm{d}p_1} & \cdots & \frac{\mathrm{d}x^{(n)}}{\mathrm{d}p_m} \end{bmatrix}$$

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

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

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.



Можно с большой долей уверенности сказать, что современные компьютеры большую часть времени занимаются либо решением СЛАУ, либо дифференцированием для их составления. Рассмотрим способы выполнения дифференцирования.

Виды дифференцирования

Рассмотрим функцию:

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

1. Символное 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.

$$f'(x) = \frac{f(x+h)-f(x)}{h} + O(h) \qquad (1)$$

Схему можно немного модернизировать и повысить точность результата:

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

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 пп. 2 и 3, в результате не получится 0.

Поговорим о преимуществах и недостатках каждого из видов дифференцирования.

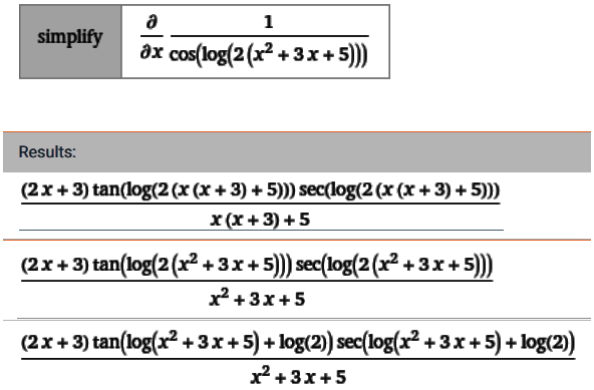
Символьное дифференцирование

Достоинство: наиболее точное (потенциально)

Недостатки:

- не устраняет дублирование вычислений

Пример:



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:

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

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

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

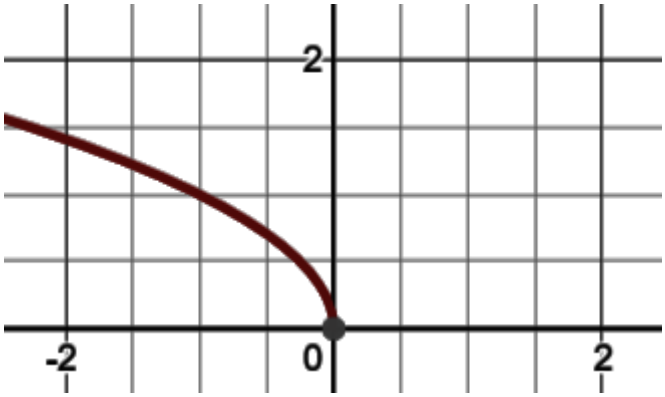
Численное дифференцирование

Достоинство: просто в реализации

Недостатки:

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

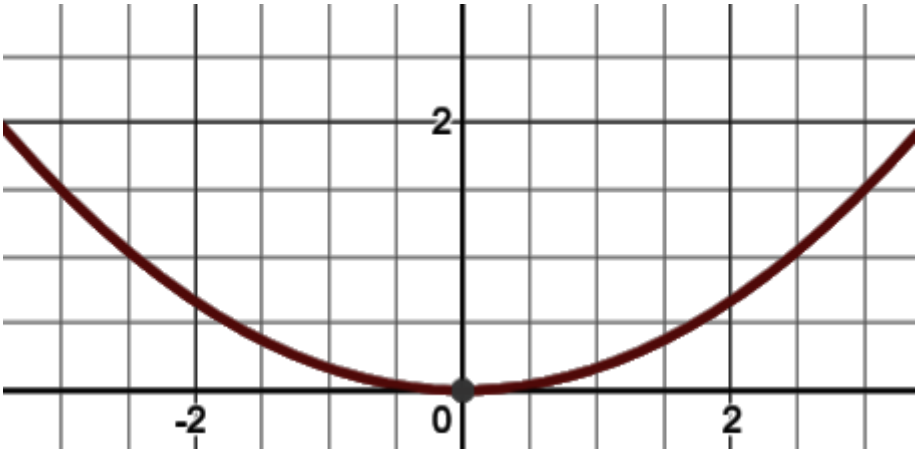
Например, $f(x) = \sqrt{-x}$. При нахождении производной в точке 0 посчитать производную по формуле численного дифференцирования невозможно.



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 (1), указанной ниже, то значение будет ненулевым.

Можно использовать измененную схему (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:

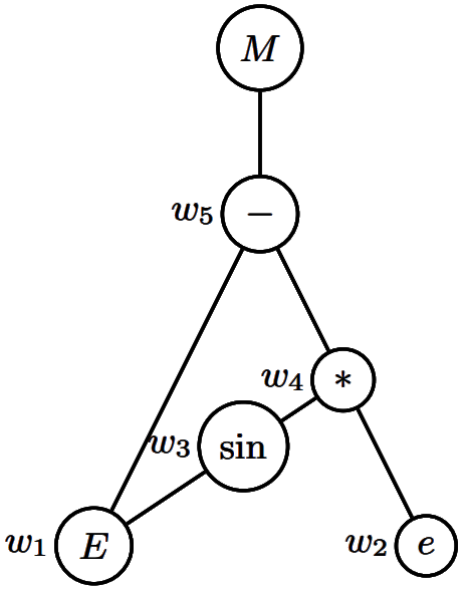
$$M = E - e \sin E$$

Программный код, соответствующий этой формуле:

```
func M(E, e):
  w1 = E
  w2 = e
  w3 = sin(w1)
  w4 = w3 * w2
  w5 = w1 - w4
  M = w5
  return(M)
```

Здесь w_i — временные переменные.

Схематично процедуру можно изобразить так:



Дифференцирование будем производить по обоим входным параметрам.

$$\text{func } \mathrm{d}M(E,\ e):$$
$$\frac{\partial w_1}{\partial E} = 1$$
$$\frac{\partial w_1}{\partial e} = 0$$
$$w_1 = E$$

$$\frac{\partial w_2}{\partial E} = 0$$
$$\frac{\partial w_2}{\partial e} = 1$$
$$w_2 = e$$

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

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

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

$$\frac{\partial M}{\partial E} = \frac{\partial w_5}{\partial E}$$
$$\frac{\partial M}{\partial e} = \frac{\partial w_5}{\partial e}$$
$$M = w_5$$
$$\text{return}(M,\ \frac{\partial M}{\partial E},\ \frac{\partial M}{\partial e})$$

Функция $\mathrm{d}M$ 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. Для практических нужд существует множество программных пакетов, реализующих автоматическое дифференцирование.