Типовик по линейной алгебре №3, задание 4 «Алгебраические операции с тензорами.»

Латыпов Владимир Витальевич,
ИТМО КТ M3138, ***вариант 10***

10 июня 2022 г.

1. Формулировка условия

Утверждение 1. Условие таково:

Тензор α^{ijk} (3 раза контравариантный) задан трехмерной матрицей третьего порядка $A = \|\alpha^{ijk}\|$.

- Вычислить матрицу транспонированного тензора $\beta^{ijk} = \alpha^{kji}$.
- Вычислить матрицу полностью симметричного тензора $\alpha^{(ijk)}$.
- Вычислить матрицу полностью антисимметричного тензора $\alpha^{[ijk]}$.
- Вычислить матрицу тензора $\alpha^{(i|j|k)}$, симметризованного по индексам i и k.
- Вычислить матрицу тензора $\alpha^{i[jk]}$, антисимметризованного по индексам j и k.

$$A = \left(\begin{array}{ccc|ccc|ccc} -2 & 3 & 4 & 3 & 6 & 0 & 2 & 1 & 3 \\ 3 & -1 & -4 & 2 & 4 & -6 & 1 & 0 & 2 \\ -1 & 2 & 2 & 1 & -2 & 3 & 1 & 0 & 4 \end{array}\right) \tag{1}$$

2. Транспонируем

Заметим, что для применения перестановки достаточно совершить одну транспозицию, поменяв первую координату с третьей (для матрицы это строчка и слой соответственно), то есть фиксируя вторую (столбец).

Тогда выпишем двухмерные матрицы, имеющие константный столбец и, транпонировав их, вернём на место.

Для надёжности будем испольовать `matrixcalc.org`.

$$\alpha^{?1?} = \begin{pmatrix} -2 & 3 & 2 \\ 3 & 2 & 1 \\ -1 & 1 & 1 \end{pmatrix} \to \begin{pmatrix} -2 & 3 & -1 \\ 3 & 2 & 1 \\ 2 & 1 & 1 \end{pmatrix} \tag{2}$$

$$\alpha^{?2?} = \begin{pmatrix} 3 & 6 & 1 \\ -1 & 4 & 0 \\ 2 & -2 & 0 \end{pmatrix} \to \begin{pmatrix} 3 & -1 & 2 \\ 6 & 4 & -2 \\ 1 & 0 & 0 \end{pmatrix} \tag{3}$$

$$\alpha^{?3?} = \begin{pmatrix} 4 & 0 & 3 \\ -4 & -6 & 2 \\ 2 & 3 & 4 \end{pmatrix} \to \begin{pmatrix} 4 & -4 & 2 \\ 0 & -6 & 3 \\ 3 & 2 & 4 \end{pmatrix} \tag{4}$$

И, собственно, записываем полученную новую гадость назад, причём в том же порядке, в котором вынимали старую...

$$B = \left(\begin{array}{ccc|ccc|ccc} -2 & 3 & 4 & 3 & -1 & -4 & -1 & 2 & 2 \\ 3 & 6 & 0 & 2 & 4 & -6 & 1 & -2 & 3 \\ 2 & 1 & 3 & 1 & 0 & 2 & 1 & 0 & 4 \end{array}\right) \tag{5}$$

3. Симметрирование

Здесь посчитаем по определению, однако ручками звучит очень больно. Поэтому:

```
def full_symmetrize_tensor(t):
        n = len(t.shape)
        return Fraction(1, math.factorial(n)) * sum([
        (t + Fraction()).transpose(p)
        for p in permutations(range(n))
        ])
```

Получаем вывод:

```
Symmetrized:
[
    [
        [Fraction(-2, 1) Fraction(3, 1) Fraction(5, 3)]
        [Fraction(3, 1) Fraction(7, 3) Fraction(1, 6)]
        [Fraction(5, 3) Fraction(1, 6) Fraction(2, 1)]
    ]
    [
        [Fraction(3, 1) Fraction(7, 3) Fraction(1, 6)]
        [Fraction(7, 3) Fraction(4, 1) Fraction(-8, 3)]
        [Fraction(1, 6) Fraction(-8, 3) Fraction(5, 3)]
        ]
    [
        [Fraction(5, 3) Fraction(1, 6) Fraction(2, 1)]
        [Fraction(1, 6) Fraction(-8, 3) Fraction(5, 3)]
        [Fraction(2, 1) Fraction(5, 3) Fraction(4, 1)]
```

```
        ]
    ]
```

Это записывается в виде многомерной матрицы

$$\left(\begin{array}{ccc|ccc|ccc} -2 & 3 & \frac{5}{3} & 3 & \frac{7}{3} & \frac{1}{6} & \frac{5}{3} & \frac{1}{6} & 2 \\ 3 & \frac{7}{3} & \frac{1}{6} & \frac{7}{3} & 4 & -\frac{8}{3} & \frac{1}{6} & -\frac{8}{3} & \frac{5}{3} \\ \frac{5}{3} & \frac{1}{6} & 2 & \frac{1}{6} & -\frac{8}{3} & \frac{5}{3} & 2 & \frac{5}{3} & 4 \end{array}\right) \tag{6}$$

4. Альтенирование

По факту — тут достаточно найти значение для одного элемента, индексы которого — перестановка, а потом поставить нули везде, где не она и полученное значение со знаком $(-1)^{\varepsilon(\sigma_1)+\varepsilon(\sigma_2)}$, где вторая — чётной очередной перестановки, а первая — исходной.

Но проще использовать автоматизированну. протестированную версию...

Стало лениво перепечатывать в $LaTeX$, поэтому автоматизируем это:

```
def fraction_as_latex(f):
    if f.denominator == 1:
        return str(f.numerator)
    else:
        return f"\\frac{{{f.numerator}}}{{{f.denominator}}}"

def format_3d_matrix(m): # M's internal (right index sequence
    s = m.shape
    for i in range(s[0]):
        line = []
        for k in range(s[2]):
            for j in range(s[1]):
                line.append(fraction_as_latex(m[i][j][k]))
        print(f"{' & '.join(line)}\\\\")
```

Аналогично симметрированию, но умножаем на -1, если нечётная перестановка.

```
def full_alternate_tensor(t):
    n = len(t.shape)
        return
    Fraction(1, math.factorial(n)) *
```

```
sum([
    Fraction(perm_parity(list(p))) *
    (t + Fraction()).transpose(p)
    for p in permutations(range(n))
    ])
```

Получаем:

$$\left(\begin{array}{ccc|ccc|ccc} 0 & 0 & 0 & 0 & 0 & \frac{5}{6} & 0 & \frac{-5}{6} & 0 \\ 0 & 0 & \frac{-5}{6} & 0 & 0 & 0 & \frac{5}{6} & 0 & 0 \\ 0 & \frac{5}{6} & 0 & \frac{-5}{6} & 0 & 0 & 0 & 0 & 0 \end{array}\right) \tag{7}$$

5. Симметрирование по части индексов

Для части индексов пришлось немного повозиться, но суть та же:

```
def slice_around(indexes, n):
    # Create slices around indexes
    slice_between = [-1] + indexes + [n]
    slices = []
    for i in range(len(slice_between) - 1):
        slices.append(list(range(slice_between[i] + 1, slice_l
    return slices

# print(slice_around([0, 3, 4], 6))

def interchange_arrays(first, second):
    res = [first[0]]
    for i in range(0, len(second)):
        res.append(second[i])
        res.append(first[i + 1])
    return res

def flatten(lst):
    return [item for sublist in lst for item in sublist]

def permute_index_set(n, permuting_index_set):
    slices = slice_around(permuting_index_set, n)

    raw_ps = permutations(permuting_index_set)
    for p in raw_ps:
```

```
yield flatten(interchange_arrays(slices, [[v] for v i
```

Теперь просто в качестве перестановок передаём вызов permute_index_set от нужных аргументов.

Получаем ответ на нашем тензоре:

$$\left(\begin{array}{ccc|ccc|ccc} -2 & 3 & 4 & 3 & \frac{5}{2} & -2 & \frac{1}{2} & \frac{3}{2} & \frac{5}{2} \\ 3 & \frac{5}{2} & -2 & 2 & 4 & -6 & 1 & -1 & \frac{5}{2} \\ \frac{1}{2} & \frac{3}{2} & \frac{5}{2} & 1 & -1 & \frac{5}{2} & 1 & 0 & 4 \end{array}\right) \tag{8}$$

6. Альтенирование по части индексов

Применяем уже написанные функции:

$$\left(\begin{array}{ccc|ccc|ccc} 0 & 0 & 1 & 0 & 0 & \frac{-1}{2} & -1 & \frac{1}{2} & 0 \\ 0 & \frac{-3}{2} & \frac{-5}{2} & \frac{3}{2} & 0 & -3 & \frac{5}{2} & 3 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & \frac{-1}{2} & 0 & \frac{3}{2} & \frac{-1}{2} & \frac{-3}{2} & 0 \end{array}\right) \tag{9}$$

7. Послесловие

Кажется, быстрее было сделать всё ручками... Особенно — учитывая возню с тем, что естественный способ нумерации размерностей в программировании отличается от индексации многомерных матриц, и с этим тоже много возни...

Но это для слабаков