Matlab - програми обчислення повних 1-D та 2-D сум Фур'є з використанням швидкого перетворення Фур'є.

К.Є. Бабенко

Українська інженерно-педагогічна академія
м. Харків, вул. Університетська, 16
kristina_babenko@mail.ru

Анотація

Задача візуалізації результатів обробки експериментальних даних виникає у різних розділах науки і техніки. Зокрема, в прямому методі Фур'є – одному з найвідоміших методів розв'язання плоскої задачі комп'ютерної томографії важливою для візуалізації результатів методу є задача обчислення сум Фур'є однієї та двох змінних у фіксованій системі точок. Використання в Matlab швидкого перетворення Фур'є (ШПФ) від однієї змінної вимагає [1] парної кількості коефіцієнтів Фур'є, а повна сума Фур'є має непарне їх число. Аналогічна ситуація існує і при обчислення у фіксованій системі точок двовимірних сум Фур'є. Тому для обчислення сум Фур'є з використанням ШПФ у рівновіддалених точках за допомогою системи комп'ютерної математики (СКМ) Matlab, деякі автори [2] використовують неповні суми Фур'є, що погіршує візуалізацію. В даній роботі пропонуються Matlab - програми обчислення з використанням ШПФ повних сум Фур'є однієї та двох змінних.

1. Вступ

1.1. Формулювання проблеми. У теоретичних і практичних дослідженнях з метою візуалізації результату виникає задача обчислення сум Фур'є для функцій однієї та двох змінних у системі рівновіддалених точок. Найбільш широко використовуваним математичним апаратом для розв'язування цієї задачі є ШПФ [2], що дозволяє значно зменшити час необхідних обчислень. Але безпосереднє використання ШПФ для обчислення повних сум Фур'є $s_n(x)=\sum_{k=-n}^{n} c_k e^{ikx}$ неможливе, бо повна сума Фур'є має непарну кількість доданків $N=2n+1$, а ШПФ у СКМ Matlab вимагає, щоб $n=2^k, k\in \mathrm{N}$. У зв'язку з цим деякі автори для цієї мети використовують неповні суми Фур'є $s_n(x)=\sum_{k=-n}^{n-1} c_k e^{ikx}$. Тобто, практика використання Matlab для таких задач вимагає створення пакету програм для обчислення повних сум Фур'є від однієї та двох змінних з допомогою ШПФ.

1.2. ШПФ у системі Matlab.

1.2.1. 1D ШПФ у системі Matlab. Функція обчислення *прямого* ШПФ для однієї змінної у системі Matlab має назву ***fft(x)***. Для заданого у дужках параметра ***x*** (вектора довжини ***n***), ***fft*** повертає вектор ***X***

$$X_k=\sum_{p=1}^{n} x_p \quad e^{-i2\pi(k-1)\frac{(p-1)}{n}}, \quad k=\overline{1,\ n}. \qquad (1)$$

При обчисленні *оберненого* ШПФ для однієї змінної у системі Matlab використовується функція ***ifft(X).*** Для заданого у дужках вектора коефіцієнтів Фур'є ***X*** довжини ***n*** процедура ***ifft*** повертає вектор виду

$$x_p=\frac{1}{n}\sum_{k=1}^{n} X_k \quad e^{i2\pi(k-1)\frac{(p-1)}{n}}, \quad p=\overline{1,\ n}. \quad (2)$$

1.2.2. 2D ШПФ у системі Matlab. Стандартна *Matlab*-функція для обчислення ШПФ від двох змінних має назву ***fft2***. Алгоритм обчислення ***fft2(X)*** полягає у наступному: спочатку здійснюється одновимірне ШПФ кожного стовпчика матриці ***X***, потім здійснюється одновимірне ШПФ всіх рядків отриманої матриці з використанням функції одновимірного ШПФ ***fft***

fft2(X) == fft(fft(X)')'

де знак ' є операцією транспонування матриці ***X***.

2. Matlab-програми для виконання і перевірки ШПФ при обчисленні повних сум Фур'є від однієї змінної

Наведемо програми, що реалізують пряме та обернене швидкі перетворення Фур'є для функцій від однієї змінної у системі Matlab, які дозволяють обчислювати повні суми Фур'є від однієї змінної. Для прикладу здійснимо пряме та обернене ШПФ функції $f(x)=x^2-x-1, \quad x\in [0,\ 1]$. Для визначення цієї функції у системі *Matlab* створимо так званий *M*-файл (у загальному випадку цей файл може бути відсутній, бо функція $f(x)$, взагалі кажучи, є невідомою). Наведемо його зміст (для зручності пояснень будемо нумерувати кожний рядок цього і наступних M-файлів):

```
1.    function f = f(x)
2.    f = x*x-x-1;
```

Для визначення кількості точок функції, в яких буде підраховуватися сума Фур'є, введемо глобальну змінну ***n*** (ця змінна повинна мати вид $n=2^p, \quad p=1,\ 2,\ \ldots$):

```
global n;
```

Для перевірки того, наскільки точно сума Фур'є наближує дану функцію, створимо ще один *M*-файл ***ValueF.m***, що здійснює обчислення значення функції $f(x)$ у ***n*** точках:

```
1.    function ValueF = ValueF
2.    global n;
3.    for p = 1 : n
4.     ValueF(p) = f((p-1)/n);
5.    end;
```

Наприклад, при значенні глобальної змінної ***n=4*** виклик ***ValueF*** повертає вектор:

-1.0000 -1.1875 -1.2500 -1.1875

Для обчислення коефіцієнтів Фур'є (якщо вони невідомі заздалегідь) створимо у системі Matlab файл ***KoefFur.m***. Наведемо його зміст:

```
1.  function KoefFur = KoefFur
2.  global n;
3.  for k = 1 : n
4.   KoefFur(k) = 0;
5.   for p = 1 : n
6.  KoefFur(k) = KoefFur(k) + f((p-1)/n)*exp(-i*2*pi*(k-1)*(p-1)/n);
7.   end;
8.  end;
```

Виклик ***KoefFur*** повертає вектор:

-4.6250 0.2500 0.1250 0.2500

Критерієм перевірки правильності роботи ***KoefFur*** є виклик функції *Matlab* ***fft*** із параметром ***ValueF***:

```
fft(ValueF)
ans =
-4.6250   0.2500   0.1250   0.2500
```

Пряме обчислення суми Фур'є здійснюється у файлі ***SumFur.m***:

```
1.function SumFur = SumFur
2.global n;
3.Q = KoefFur;
4.for p = 1 : n
5. SumFur(p) = 0;
6. for k = 1 : n
7. SumFur(p) = SumFur(p) + Q(k)*exp(i*2*pi*(k-1)*(p-1)/n);
8.  end;
9.  SumFur(p) = SumFur(p)/n;
10. end;
```

Наведемо результати виклику функції ***SumFur:***

-1.0000 -1.1875 -1.2500 -1.1875

Ці значення збігаються із значеннями, що повертає функція ***ValueF.***: пряме обчислення суми Фур'є точно відновлює значення функції $f(x)$ у ***n*** точках.

Для обчислення *прямого* перетворення Фур'є у відповідних точках з допомогою ШПФ створимо файл ***fft1.m***:

```
1.  function fft1 = fft1(x)
2.  n = length(x);          % Length of vector x
3.  m=log2(n);  % Calculate m (number of matrix) such that n=2^m
4.  n2 = n;
5.  for k = 1 : m % Main For (for each matrix)
6.   n1 = n2;
7.   n2 = n2/2;
8.   for j = 1 : n2 % Determining the place of None Zero element
9.   for i = j : n1 : n
10.  q = i + n2;
11.  xt = x(i) - x(q);
12.  x(i) = x(i) + x(q);
13.  x(q) = exp(-2i*pi*(j-1) / n1)*xt; % Multiplication
14.  end;
15. end;
16.end;
17.j = 1;
18.n1 = n - 1;
19.for i = 1 : n1 % Digit reverse counter
20. if  i < j
21.  xt = x(j);   x(j) = x(i);   x(i) = xt;
22. end;
23. k = n / 2;
24. while k < j
25.  j = j - k;
26.  k = k/2;
27. end;
28. j = j + k;
29.end;
30.fft1 = x;
```

Звертаємо увагу, що відповідне алгоритму Кулі і Такі розкладання матриці на добуток більш простих матриць здійснюється одночасно (в одному циклі) з множенням на елемент вхідного вектора (рядки 5-16). Кількість таких матриць дорівнює ***m=log2(n)***. Причому місце ненульових елементів матриці обчислюється не за допомогою логічного оператора ***if*** – ***якщо*** *ненульовий елемент матриці,* ***то*** *здійснити добуток,* а арифметично, шляхом підрахунку кроку вкладених циклів. Такий підхід підвищує ефективність програми.

Останній блок програми (**Digit reverse counter**) необхідний для правильного розташування елементів одержаного вектора.

Результат виконання програми ***fft1(ValueF)-*** вектор

-4.6250 0.2500 0.1250 0.2500,

значення якого повністю збігаються з вектором у файлі ***KoefFur***.

Наведемо *Matlab*-програму, що реалізує *оберненеШПФ*:

```
1.  function ifft1 = ifft1(x)
2.  n = length(x);
3.  m=log2(n);
4.  n2 = n;
5.  for k = 1 : m
6.   n1 = n2;
7.   n2 = n2/2;
8.   for j = 1 : n2
9.   for i = j : n1 : n
10.  q = i + n2;
11.  xt = x(i) - x(q);
12.  x(i) = x(i) + x(q);
13.  x(q) = exp(2i*pi*(j-1) / n1)*xt;
14.  end;
15. end;
16. end;
17. j = 1;
18. n1 = n - 1;
19. for i = 1 : n1
20. if  i < j
21.  xt = x(j);   x(j) = x(i);   x(i) = xt;
22. end;
23. k = n / 2;
24. while k < j
25.  j = j - k;
26.  k = k/2;
27. end;
28. j = j + k;
29. end;
30. for i = 1 : n
31. x(i) = x(i)/n;
```

```
32.  end;
33.  ifft1 = x;
```

Таким чином, наведена програма аналогічна ***fft*** за виключенням зміни знака аргумента експоненти, що обчислюється у рядку 13. У рядках 30-32 здійснюється ділення отриманого вектора ***x*** на його довжину ***n*** (відповідно формулі $x_p = \frac{1}{n}\sum_{k=1}^{n} X_k \quad e^{i2\pi(k-1)\frac{(p-1)}{n}}, \quad p = \overline{1, \quad n}$.

Критерієм перевірки правильності роботи програм ***fft1*** та ***ifft1*** може бути наступний виклик

```
ifft1(fft1(ValueF))
ans =
-1.0000   -1.1875   -1.2500   -1.1875
```

3. Matlab-програми обчислення 2-D сум Фур'є за допомогою ШПФ

3.1. Загальні формули для обчислення 2-D сум Фур'є.

Хай треба обчислити суму Фур'є

$$S_n(x,y) = \sum_{p=-n/2}^{n/2} \sum_{q=-n/2}^{n/2} C_{pq} \quad e^{i2\pi(px+qy)},$$

$$C_{pq} = \int_0^1\int_0^1 f(x,y) \quad e^{-i2\pi(px+qy)} dxdy$$

у точках $x_k = \frac{k-1}{n}, \quad k = \overline{1, \quad n}$, $y_l = \frac{l-1}{n}, \quad l = \overline{1, \quad n}$,

тобто обчислити n^2 сум ($k = \overline{1, \quad n}, \quad l = \overline{1, \quad n}$)

$$S_n\left(\frac{k-1}{n}, \frac{l-1}{n}\right) = \sum_{p=-n/2}^{n/2} \sum_{q=-n/2}^{n/2} C_{pq} \quad e^{i2\pi\left(p\frac{k-1}{n} + q\frac{l-1}{n}\right)}.$$

Цю задачу зведемо до задачі обчислення сум

$$S_n\left(\frac{k-1}{n}, \frac{l-1}{n}\right) =$$

$$= (-1)^{k+l}\left(\begin{array}{l} C_{n+1,\, n+1} + \sum_{p=1}^{n}\sum_{q=1}^{n} C_{pq} \quad e^{i2\pi\left((p-1)\frac{k-1}{n} + (q-1)\frac{l-1}{n}\right)} + \\ + \sum_{p=1}^{n} C_{p,\, n+1} e^{i2\pi(p-1)\frac{k-1}{n}} \quad + \sum_{q=1}^{n} C_{n+1,\, q} \quad e^{i2\pi(q-1)\frac{l-1}{n}} \end{array}\right)$$

$C_{pq}, p = \overline{1, \quad n+1}, \quad q = \overline{1, \quad n+1}$ – коефіцієнти Фур'є

$$C_{pq} = \int_0^1\int_0^1 f(x,y) \quad e^{-i2\pi\left(\left(p-\frac{n}{2}-1\right)x + \left(q-\frac{n}{2}-1\right)y\right)} \quad dx \quad dy.$$

2.2 Matlab-програми для виконання і перевірки ШПФ при обчисленні сум Фур'є від двох змінних.

Файл ***f2.m*** використовується для обчислення значення функції двох змінних, за допомогою якої визначаються коефіцієнти Фур'є (якщо вони невідомі заздалегідь):

```
1.  function f2 = f2(x, y)
2.  f2 = x*(1-x)*y*(1-y);
```

Обчислення значення функції ***f2(x, y)*** у ***n×n*** точках здійснюється у файлі ***ValueF2.m***

```
function ValueF2 = ValueF2
1.      global n;
2.      for k = 1 : n
3.      for l = 1 : n
4.      ValueF2(k, l) = f2((k-1)/n, (l-1)/n);
5.      end;
6.      end;
```

У результаті роботи цієї програми на екрані повинна з'явитися така матриця чисел (при значенні **n=4**):

```
ValueF2=
0        0        0        0
0   0.0352   0.0469   0.0352
0   0.0469   0.0625   0.0469
0   0.0352   0.0469   0.0352
```

Ця матриця використовується для перевірки точності наближення функції від двох змінних її Фур'є сумою (у випадку, якщо така функція нам відома).

Для обчислення коефіцієнтів Фур'є (якщо вони не задані заздалегідь) створимо у системі Matlab файл ***KoefFur2.m:***

```
1.      function KoefFur2 = KoefFur2
2.      global n;
3.      syms x;
4.      syms y;
5.      for p = 1 : n + 1
6.      for q = 1 : n + 1
7.      KoefFur2(p,        q)        =
double(int(int(f2(x,y)*exp(-2*i*pi*((p-n/2-1)*x+(q-n/2-
1)*y)), x, 0, 1), y, 0, 1));
8.      end;
9.      end;
```

Після виконання цієї програми на екрані повинна з'явитися наступна матриця числових значень

```
KoefFur2=
0.0002   0.0006   -0.0021   0.0006   0.0002
0.0006   0.0026   -0.0084   0.0026   0.0006
```

-0.0021	*-0.0084*	*0.0278*	*-0.0084*	*-0.0021*
0.0006	*0.0026*	*-0.0084*	*0.0026*	*0.0006*
0.0002	*0.0006*	*-0.0021*	*0.0006*	*0.0002*

Пряме обчислення суми Фур'є здійснюється у файлі ***SumFur2.m:***

```
1.  function SumFur2 = SumFur2
2.  global n;
3.  %tic;
4.  QQ = KoefFur2;
5.  for k = 1 : n
6.  for l = 1 : n
7.  SumFur2(k, l) = 0;
8.  for p = 1 : n+1
9.  for q = 1 : n+1
10. SumFur2(k, l) = SumFur2(k, l) + QQ(p, q)*exp(i*2*pi*((k-1)*(p-n/2-1) +(l-1)*(q-n/2-1))/n);
11. end;
12. end;
13. end;
14. end;
15. %toc;
```

Обчислення суми Фур'є за допомогою ШПФ здійснюється у файлі ***FastSumFur2.m:***

```
1.  function FastSumFur2 = FastSumFur2
2.  global n;
3.  %tic;
4.  QQ = KoefFur2;
5.  for p = 1 : n
6.  for q = 1 : n
7.  C1(p, q) = QQ(p, q);
8.  end;
9.  end;
10. for p = 1 : n
11. C2(p) = QQ(p, n+1);
12. end;
13. for q = 1 : n
14. C3(q) = QQ(n+1, q);
15. end;
16. Bas1 = fft2(C1);
17. Bas2 = fft(C2);
18. Bas3 = fft(C3);
19. for k = 1 : n
20. for l = 1 : n
21. FastSumFur2(k, l) = (-1)^(k+l)*(Bas1(k, l) + Bas2(l) + Bas3(k) + QQ(n+1, n+1));
22. end;
23. end;
24. %toc;
```

У рядку 16 наведеної програми використовується Matlab-функція ШПФ для двох змінних ***fft2***. Після цієї програми ***FastSumFur2.m*** на екрані повинна з'явитися наступна матриця ***FastSumFur2=***

0.0016	*0.0077*	*0.0097*	*0.0077*
0.0077	*0.0369*	*0.0466*	*0.0369*
0.0097	*0.0466*	*0.0589*	*0.0466*
0.0077	*0.0369*	*0.0466*	*0.0369*

яка збігається з матрицею у файлі ***SumFur2.***

Автор виражає подяку проф.. О.М.Литвину і доц.. В.І.Межуєву за допомогу при створенні цих програм.

Список літературних джерел

1. Потемкин В. Г. Система инженерных и научных расчетов Matlab 5.x. В 2-х т. – М.: ДИАЛОГ-МИФИ, 1999. – Т. 1. – 366 с.; т. 2. – 304 с.
2. Gottlieb D., Gustafsson B. On the direct Fourier method for computer tomography. Upsala University, Department of Scientific Computing, March 28, 1998. p.1-31. http//www.tdb.uu.se/archive/reports/index.html , № 207.