Доля П.Г.
Харьковский Национальный Университет
механико – математический факультет
2012 г.

Математические методы компьютерной томографии.

Перед вами брошюра, которая является частью пособия «Математические методы компьютерной томографии». Она знакомит студентов с функциями MATLAB, используемыми при вычислении и визуализации прямого и обратного двумерного преобразования Радона – математического метода, лежащего в основании томографии. Приведен также параграф, в котором содержатся примеры (в основном построения графиков), рассмотренные нами в других частях пособия. Многие описываемые здесь функции относится к пакету расширения `Image Processing Toolbox`. Однако для понимания излагаемого материала необходимы базовые знания по работе в MATLAB.

Часть 2. Преобразование Радона в MATLAB

Оглавление

1. Построение Радоновских образов и прообразов по аналитическим формулам.

В предыдущих частях пособия рассмотрено большое количество примеров построения изображений и радоновских проекций. После получения формул мы часто приводили графический образ. Здесь будут приведены примеры MatLab, с помощью которых выполнялась такая визуализация.

2. Полезные графические функции

Понимание двух функций MatLab `radon` и `iradon`, используемых для вычисления прямого и обратного преобразования Радона, невозможно без

наглядного представления результатов их работы. В настоящем параграфе мы рассматриваем некоторые графические функции, которые наиболее часто используются в этом случае. Вот функции, которые рассматриваются в этом параграфе

```
imread('файл',...)    - чтение графического файла
imfinfo('файл',...)   - информация о графическом файле
image(матрица,...)    - нарисовать образ матрицы
colormap(матрица(m x 3))
imshow(матрица,...)
imagesc(матрица,...)
surface(матрица,...)
surf(матрица,...)
mesh(матрица,...)
```

Графический файл в MATLAB представляется в виде матрицы с элементами, являющимися целыми числами, которые являются индексами (номерами) цветов в матрице называемой картой или палитрой цветов (`colormap`). В MATLAB есть функции чтения графических файлов и отображения полученных матриц. Рассмотрим для примера рисунок, который находится в графическом файле `Pogorelov32x32.bmp`, размером 32 на 32 пикселя.

Команда

```
A=imread('Pogorelov32x32.bmp')
```

читает и печатает матрицу `A` размером 32 x 32. Значения элементов матрицы являются номерами цветов соответствующих пикселей. Вот кусочек этой матрицы

```
 15  15  15  15   0   0   0   0   0   1   6 ...
 15   0   0   6   0   7   7   8   8   8   8 ...
 15  15   0   0   0   1   7   8   7   8   8 ...
 15   7   0   2   1   7   7   8   8   8   8 ...
 15   5   0   7   0   7   7   8   8   8   8 ...
 .....................................................
```

Команда

```
image(A);
```

по этой матрице рисует картинку в графическом окне (см. рисунок слева).

Рисунок не очень хорош и для получения приемлемого изображения нам потребуется изменить параметры его отображения.

Когда матрица содержит целые положительные числа, то ее элементы интерпретируются как индексы в карте цветов. Функция `image(A)` строит ее образ в текущей карте – каждому элементу матрицы будет соответствовать

некоторый прямоугольник на рисунке, закрашенный цветом, номер которого соответствует строке матрицы цветов (палитра). Имеются также дополнительные аргументы, управляющие работой функции `image`.

Полученным образом в графическом окне, можно управлять, используя обычные функции MatLab, управляющие графикой. В частности, в нашем примере рисунок вытянут по горизонтали. Команда

```
axis equal;
```

выравнивает горизонтальные и вертикальные размеры пикселей в графическом окне. Она устанавливает коэффициент сжатия изображения одинаковым по всем осям (см. предыдущий рисунок справа). Тот же результат дает команда

```
axis image;
```

но охватывающий изображение прямоугольник в графическом окне деформируется под размер изображения.

Цвета точек берутся из текущей карты цветов. Карта цветов (палитра) – это матрица, которая содержит 3 столбца и несколько (может быть много) строк. Например, команда

```
gray(5)
```

создает следующую матрицу, которая может быть использована как палитра цветов

```
         0         0         0
    0.2500    0.2500    0.2500
    0.5000    0.5000    0.5000
    0.7500    0.7500    0.7500
    1.0000    1.0000    1.0000
```

Карта цветов `colormap` является $m \times 3$ матрицей вещественных чисел диапазон изменения которых от 0 до 1. Каждая ее строка является RGB вектором, определяющим один цвет. Строка карты цветов с номером k определяет $k-$й цвет в формате $[r(k)\ g(k)\ b(k)]$ с соответствующими интенсивностями красного `r(k)`, зеленого `g(k)` и синего `b(k)` цветов.

Мы видели, что матрица A, прочитанная из файла, содержит числа от 0 до 15, поэтому создадим черно – белую палитру с 16 оттенками серого. Команда

```
colormap(gray(16));
```

преобразует рисунок к следующему виду

Обратите внимание на тип элементов матрицы A в окне Workspace – uint8 (беззнаковые 8 – битные целые могут изменяться в диапазоне от 0 до 255). Если выполнить команду

```
B=A/15;
```

то операция деления даст только 0 или 1, а тип элементов матрицы B не изменится (целое разделить на целое дает целое). Это позволяет создать

черно – белый рисунок. Для этого создадим палитру из двух цветов – черного (ему соответствует строка 0, 0. 0) и белого (ему соответствует строка 1, 1, 1).

```
cm=[0 0 0;1 1 1];   % матрица палитры (два цвета – черный
                      и белый)
image(B);
axis image; colormap(cm);
```

Команда

```
colormap('default')
```

устанавливает карту цветов по – умолчанию.

Графический файл мог сразу быть монохромным. Например, автор создал в `Paint`-е черно-белую картинку 32 x 32 пикселя PeterMono1.bmp

Команда

```
P=imread('PeterMono1.bmp')
```

прочитала и отпечатала матрицу 32 x 32 пикселя, составленную только из нулей или единиц. Команда

```
image(P);axis image; colormap('default');
```

нарисовала очень темную картинку. Но команды

```
cm=[0 0 0;1 1 1];colormap(cm);
```

преобразуют картинку к виду, который показан выше.

Чтобы узнать формат графического файла можно использовать команду

```
info = imfinfo('Pogorelov32x32.bmp')
```

Она возвращает структуру с большим количеством полей, содержащую различную информацию о графическом файле. Здесь мы приводим некоторые поля этой структуры

```
info =
              Filename: 'Pogorelov32x32.bmp'
              FileSize: 630
                Format: 'bmp'
         FormatVersion: 'Version 3 (Microsoft Windows 3.x)'
                 Width: 32
                Height: 32
              BitDepth: 4
             ColorType: 'indexed'
    NumColormapEntries: 16
              Colormap: [16x3 double]
             NumPlanes: 1
            ...................
```

Команда `[X,map] = imread(filename)` читает графический файл в матрицу `X`, а его карту цветов в матрицу `map`, элементы которой масштабируются к диапазону [0, 1].

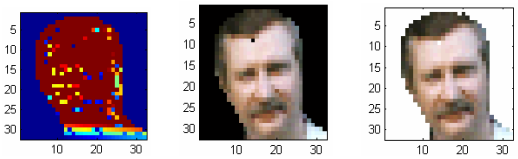
```
[A,m] = imread('Pogorelov32x32.bmp');
image(A);
colormap(m);
```

Команда

```
[A,map,alpha] = imread(...)
```

в третьем параметре возвращает маску (матрицу из логических нулей и единиц), которая используется для определения информации о прозрачности точек образа (какие точки образа отображать на экране, а какие нет; не отображать те, для которых элементы `C` равны 1). Например, этот параметр полезен для рисования образа файла иконки.

```
[A,B,C]=imread('Peter1.ico'); % A содержит числа от 0 до 255.
image(A); axis image;           % левый рисунок
colormap(B);                    % средний рисунок
```



Теперь сделаем фон рисунка белым.

```
B2=B;
B2(256,:)=[1 1 1];      % последний цвет палитры сделать белым цветом
D = ones(size(A)) * (length(B2)-1); %создать матрицу с элементами 255
D(C == 0) = A(C == 0);
image(uint8(D)), colormap(B2);  %правый рисунок
axis image;
```

Поясним работу строки `D(C==0)=A(C==0)`.

Если посмотреть в окне Workspace, то видно, что массив `C` является логическим. И при этом он имеет такой же размер, что и массив `A`. Здесь используется логическое индексирование в форме

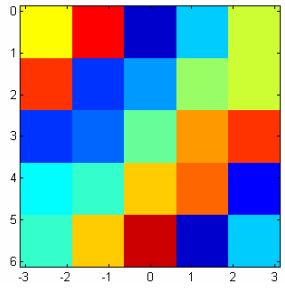
`имя_матрицы(логический массив)`

где логический массив должен иметь тот же размер, что и матрица. Результатом этой операции логического индексирования является вектор, составленный из элементов исходной матрицы, для которых в логическом массиве соответствующие элементы равны логической единице. Этот вектор может стоять как в правой части оператора присваивания, так и в левой. Вначале мы создали `double` массив `D` из чисел 255, а затем создали вектора одинаковой длины `D(C==0)` и `A(C==0)`. Второй вектор содержит только элементы матрицы `A`, для которых в матрице `C` элементы равны нулю. Это значит, что соответствующие точки образа должны быть нарисованы. Значения этого вектора `A(C==0)` присваиваются элементам вектора `D(C==0)`, т.е. соответствующим элементам матрицы `D`. Если в матрице `C` элемент был 1, то соответствующий элемент матрицы `D` не меняется, т.е. остается равным 255. В результате элементы матрицы A, отвечающие отображаемым точкам, скопировались в матрицу D, а отвечающие неотображаемым точкам остались в матрице D равными 255. Кроме того, нам потребовалось заменить последнюю (256 – ю) строку матрицы палитры, чтобы она соответствовала белому цвету. Поскольку функция `image` отображает только матрицы с типом элементов `uint8` (диапазон изменения

чисел от 0 до 255), то нам также потребовалось преобразование типа вещественной матрицы D в тип `uint8`.

Команда `image(x,y,C)`, где `x` и `y` являются двухэлементными векторами, которые определяют диапазон изменения координат `x` и `y`, рисует такой же образ, что и команда `image(C)`. Это бывает полезно, например, в случае, когда вы желаете, чтобы метки осей соответствовали реальным физическим координатам графического образа.

```
A = magic(5);
x = [-2.5 2.5];
y = [0.5 5.5];
image(x,y,A), axis image, colormap(jet(25))
```

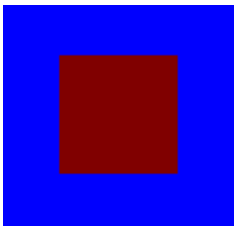
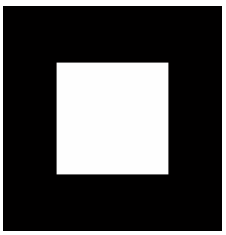


Функция `imshow(M)` тоже строит графический образ полутоновой (т.е. одноцветной с различными градациями яркости) матрицы M элементы которой имеют тип `double`. При этом каждой точке матрице соответствует один пиксель в графическом окне. Вариантов использования этой функции очень много. Она является базовой графической функцией пакета `Image Processing Toolbox`. С различными вариантами ее использования следует ознакомиться по справочной системе.

```
I = zeros(4,4) % генерирование нулевой матрицы размером 4 x 4
I =   0     0     0     0
      0     0     0     0
      0     0     0     0
      0     0     0     0
I(2:3, 2:3) = 1      % изменение значений матрицы
I =   0     0     0     0
      0     1     1     0
      0     1     1     0
      0     0     0     0
imshow(I);           % построение образа матрицы
```

Здесь единицы в матрице `I` соответствуют белым точкам рисунка, а нули – черным. Но точек мало, поэтому рисунок маленький. Если бы мы использовали функцию `image`, то нам надо было бы выполнить преобразование типа, например, так `image(uint8(I))`, но при этом образ матрицы был построен по размеру графического окна. Увеличим размер рисунка, увеличив размер матрицы

```
I = zeros(100,100);
I(25:75, 25:75) = 1;
colormap('default');
imshow(I);      % следующий рисунок слева
colormap(hot); % следующий рисунок справа
```

Слово `hot` представляет имя предопределенной матрицы палитры. Имена готовых матриц палитр можно найти в справочной системе. Вот еще пример использования функции `imshow`.

```
moon = imread('moon.tif');
imshow(moon);
```

или можно сразу передать имя файла

```
imshow('moon.tif');
```

`imshow(I,[low high])` рисует образ I в серых оттенках с диапазоном изменения значений I равным `[low high]`. Значение low и меньшие его рисуются черным цветом, значение high и большие его рисуются белым. Значения между рисуются промежуточными оттенками серого, используя уровни серого определяемые по – умолчанию. Если используется пустая матрица `[]` вместо `[low high]`, то `imshow` использует [min(I(:)) max(I(:))], т.е. минимальное значение в I отображается черным, а максимальное – белым цветами.

Угадывать палитру или использовать палитру графического файла не всегда удобно. Поэтому в MatLab есть функция `imagesc`, которая масштабирует матрицу под текущую палитру и затем рисует ее образ.

Команда `imagesc(A)` отображает матрицу как графический образ. Каждый элемент матрицы соответствует прямоугольному участку образа. Значения элементов являются индексами строк текущей матрицы палитры, а тройка чисел этой строки в формате RGB определяет цвет такого прямоугольника. Команда `imagesc(x,y,A)` отображает матрицу A в виде набора цветов (графического образа) на координатной сетке с диапазоном определяемым векторами `x` и `y`.

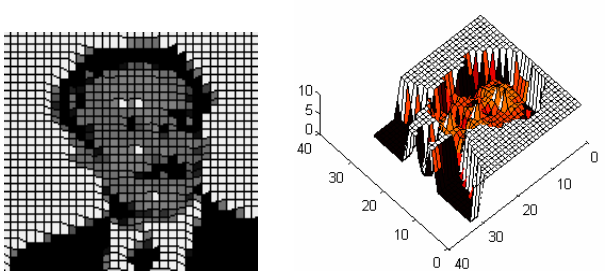
Иногда желательно отобразить матрицу как поверхность над прямоугольной областью. Это можно сделать с помощью функции `surface(Z)`, где матрица `Z` содержит вещественные числа, которые будут интерпретироваться как высота поверхности над плоскостью XY. Но матрицы, прочитанные из

графического файла, содержали только целые числа. Здесь нам может потребоваться преобразование типа. Например, команда

```
A = imread('Pogorelov32x32.bmp');
C = double(A)/16;
```

создаст вещественную матрицу `double(A)`, каждый элемент которой будет вещественным числом, которые после деления на 16 создадут матрицу `C` с действительными элементами в диапазоне от 0 до 1.

```
C=double(A)/16;
surface(C);    %  рисует поверхность матрицы с double
                  элементами.
surface(C*10); colormap(hot);
```



Аналогично работает команда `surf(Z)`, которая создает $3-$x мерный график поверхности по `z` координатам точек, заданных в матрице `Z` используя `x=1:n` и `y=1:m`, где `[m,n]=size(Z)`. Высоты точек поверхности `Z` являются вещественными числами, а цвет поверхности пропорционален высоте точек.

```
A = imread('Pogorelov32x32.bmp')
C=double(A)/16;
surfc(C);
colormap(gray(17));
```

Полезной может быть также функция построения каркасного изображения поверхности. Один из наиболее часто используемых ее синтаксисов следующий

```
mesh(X,Y,Z);
```

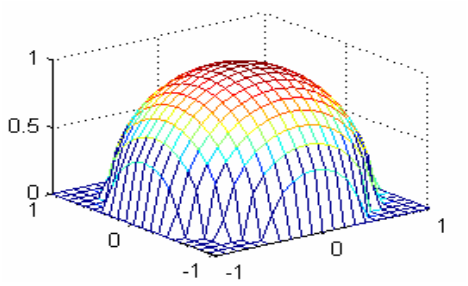
где матрицы `X,Y,Z` должны быть одинакового размера. График поверхности строится над прямоугольной областью плоскости *XY*, которая разбивается сеткой с размером таким же, как у матриц. В узлах сетки значения функции равны соответствующим значениям элементов матрицы Z. Матрицы X и Y содержат x и y координаты узлов сетки. В следующем примере мы строим единичную полусферу по уравнению

$$z=\begin{cases}\sqrt{1-x^2-y^2}, & x^2+y^2\le 1\\ 0, & x^2+y^2>1\end{cases},$$

где $-1\le x\le 1$, $-1\le y\le 1$.

```
[X Y]=meshgrid(-1:0.1:1);
XY=X.^2+Y.^2;
```

```
C=XY<=1;
XYZ=zeros(size(XY));   % нулевая матрица
XYZ(C)=1-XY(C);        % логическое индексирование
Z=sqrt(XYZ);
mesh(X,Y,Z);
```



Здесь мы использовали логическое индексирование так же, как описано выше в этом параграфе.

3. Прямое преобразование Радона. Функция radon.

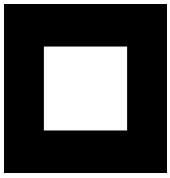
Функция прямого преобразования Радона `radon` для своего выполнения требует установки двух основных параметров – исходного изображения `I` и углов `theta`. Она имеет следующий упрощенный синтаксис вызова

```
R = radon(I);
R = radon(I,theta);
```

Исходное изображение задается в форме матрицы `I`. Аргумент `theta` задает вектор углов, для которых должно выполняться преобразование. Если аргумент `theta` не указан, то он принимается равным `0:179`. Возвращаемое значение `R` представляет собой матрицу, в которой каждый столбец является преобразованием Радона для каждого из углов, содержащихся в векторе `theta`. Матрица проекций `R` имеет формат данных `double`.

Пример 1. Создадим матрицу, состоящую из единиц и нулей.

```
I = zeros(100,100); I(25:75, 25:75) = 1; imshow(I);
```



Здесь единицы в матрице `I` соответствуют белым точкам рисунка, а нули – черным. Теперь получим матрицу проекций, т.е. выполним преобразование Радона.

```
R=radon(I);
```

Поскольку матрица R содержит вещественные элементы, а не типа `uint8`, то перед построением ее образа она должна быть масштабирована под текущую палитру. Используем для этого функцию `imagesc(R)`

```
imagesc(R); colormap(hot); % рисунок слева
```

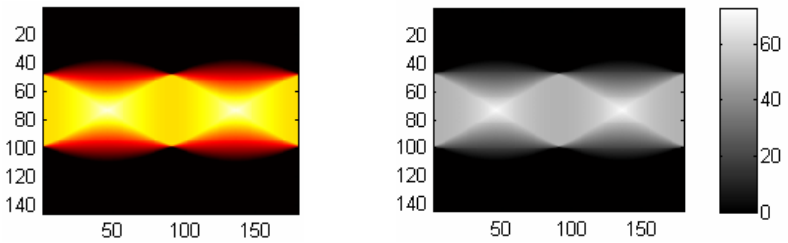


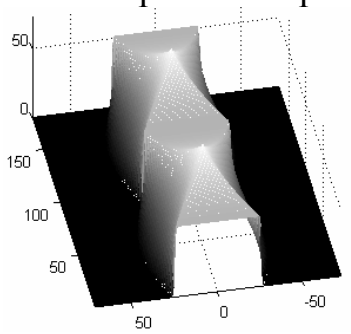
Рисунок справа получен после выполнения команд

```
colormap(gray(129)); colorbar;
```

Команда `colorbar` рисует в графическом окне справа от образа текущую палитру цветов. Команды

```
[R,xp]=radon(I); mesh(0:179,xp,R); colormap(gray);
```

строят поверхность радоновского образа – матрицы R



□

Пример 2. Построим ПР характеристической функции (ХФ) единичного круга. Вначале создадим матрицу, значения которой представляют ХФ единичного круга. Разберем методику на матрице небольшого размера

```
[X,Y]=meshgrid(-1:0.5:1)
XY=X.^2+Y.^2
C=XY<=1                 % создаем логическую матрицу
Z=ones(size(C))         % единичная матрица вещественная
ZF=zeros(size(C))
```

Все строки мы не завершаем точкой с запятой. Это приводит к выводу матриц в командное окно. Для краткости мы здесь не приводим этих матриц. Теперь посмотрим, как выделить нужные элементы. Команда `ZF(C)` выделяет вектор – столбец элементов нулевой матрицы `ZF`, для которых в логической матрице `C` значения равны единице.

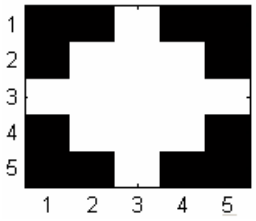
```
ZF(C)'    % отображаем транспонированный вектор - столбец
1  1  1  1  1  1  1  1  1  1  1  1  1
```

Элементам нулевой матрицы `ZF`, стоящим на таких местах, что в логической матрице `C` стоят единицы, присваиваем значения матрицы `Z` из аналогичных ячеек (с теми же номерами строк и столбцов). Остальные значения матрицы `ZF` не меняются. Это делает команда

```
ZF(C)=Z(C)
ZF =
     0     0     1     0     0
     0     1     1     1     0
     1     1     1     1     1
     0     1     1     1     0
     0     0     1     0     0
```

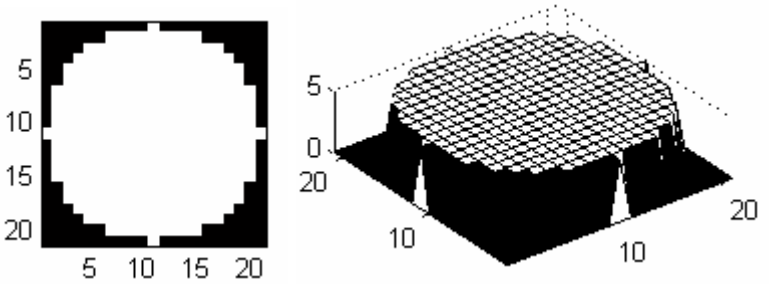
Отобразим матрицу `ZF`, например, командой

```
imagesc(ZF); colormap(gray);
```



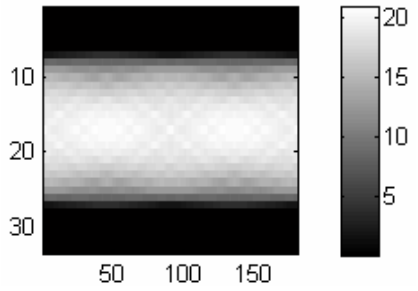
Теперь сделаем то же, но для матриц большего размера.

```
[X,Y]=meshgrid(-1:0.1:1);
XY=X.^2+Y.^2;
C=XY<=1;
Z=ones(size(C));
ZF=zeros(size(C));
ZF(C)=Z(C);
imagesc(ZF); colormap(gray); axis image; % левый рисунок
surf(ZF*5); axis image; colormap(gray);  % правый рисунок
```



Теперь выполним ПР для характеристической функции единичного круга, используя матрицу `ZF`.

```
R=radon(ZF,0:1:179);
imagesc(R); colormap(gray); colorbar;
```



По горизонтали у нас отложены углы в градусах, а по вертикали значения номеров строк матрицы `R` (их количество `n` определено по умолчанию). Если выполнить команду

```
max(max(R))
ans =   20.9740
```

то найдем максимальное значение в матрице `R`. Функция `colorbar` показала нам, что самым темным точкам образа соответствуют нулевые значения матрицы `R`, а самым светлым – значения близкие к 20.

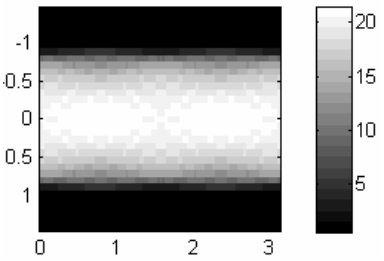
Команда

```
[R,xp] = radon(ZF,1:179);
xp'
-16   -15   -14   -13  ...  14    15    16
```

показывает, что вычисления выполнены для 33 точек (значения *s* вдоль проекции при фиксированном угле) изменяется от -16 до +16. Кроме того ясно, что начало координат XY изображения находится в центре матрицы.

Мы начинали с ХФ единичного круга, которую представили матрицей над прямоугольником [-1 1] x [-1 1]. Для приведения графика проекции к реальным физическим координатам мы должны показать диапазон изменения угла $[0, \pi]$, а радиального параметра $[-\sqrt{2}, \sqrt{2}]$.

```
RU=uint8(R);
max(max(RU))
ans =  21
image([0 pi],[-sqrt(2) sqrt(2)],RU);
axis image; colormap(gray(21));
```



□

Рассмотрим подробнее синтаксис функции `radon`. Он может быть одним из следующих:

```
R=radon(I, theta)
R=radon(I, theta, n)
[R, xp]=radon(...)
```

Функция `R=radon(I, theta)` выполняет преобразование Радона полутонового изображения, задаваемого матрицей `I`, и помещает результат в матрицу проекций `R`. Преобразование Радона представляет собой вычисление проекций изображения на прямые, задаваемые углами в градусах, отсчитываемых против часовой стрелки от горизонтали. Эти углы передаются в виде вектора в параметре `theta`. Если `theta` – скаляр, то `R` является вектор – столбцом, содержащим преобразование Радона для угла `theta`. Если `theta` – вектор, то `R` представляет собой матрицу, в которой каждый столбец является преобразованием Радона для одного из углов, содержащихся в векторе `theta`. Если при вызове функции параметр `theta` опущен, то в `theta` записываются значения углов от 0 до 179 с шагом в 1°.

В формате `R=radon(I,theta,n)` функция выполняет преобразование Радона изображения `I`, значения каждой проекции вычисляется в `n` точках, и матрица `R` имеет `n` строк. Если при вызове функции параметр `n` опущен, то он устанавливается равным

```
2*ceil(norm(size(I)-floor((size(I)-1)/2)-1))+3.
```

Если дополнительно определить выходной параметр `xp`

```
[R, xp]=radon(...),
```

то в него помещаются значения координат, в которых вычислялись значения проекции. Значения в *k*-й строке `R` соответствуют координате `xp(k)`.

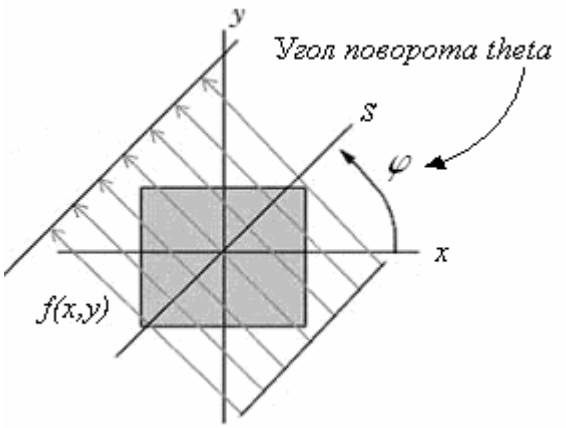
Матрица проекций R может рассматриваться как полутоновое изображение и имеет формат представления данных `double`.

Напомним, что ПР двумерной функции `f(x,y)` на ось `s` представляет собой линейный интеграл

$$R(s,\varphi)=\int_{-\infty}^{\infty} f\left(s\cos\varphi - t\sin\varphi,\ s\sin\varphi + t\cos\varphi\right)dt$$

где оси s и t задаются поворотом на угол φ против часовой стрелки

$$\begin{bmatrix} s \\ t \end{bmatrix} = \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

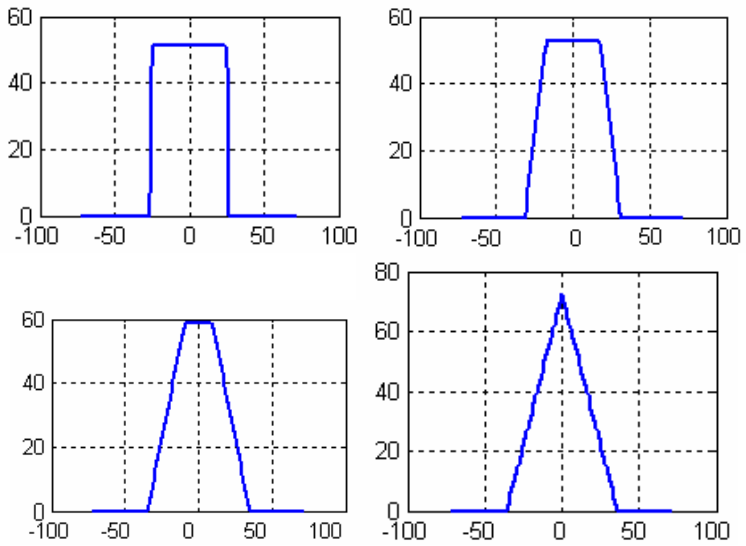


Исходное полутоновое изображение `I` рассматривается как двумерная функция. Начало координат в системе `s t` соответствует центральному пикселю изображения `I` в пиксельной системе координат. Центральный пиксель `I` можно определить согласно выражению `floor((size(I) + 1)/2)`.

Рассмотрим, как в преобразованиях Радона вычисляются проекции. Рассмотрим проекции под углом 0°, 15°, 30° и 45° квадрата из примера 1.

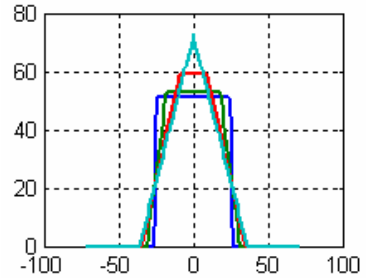
```
I = zeros(100,100); I(25:75, 25:75) = 1; imshow(I);
```

```
[R,xp] = radon(I,[0 15 30 45]);
% Построение графиков для углов 0°, 15°, 30° и 45°.
figure; plot(xp,R(:,1),'LineWidth',2);
title('R_{45^o}(s)');grid on;
figure; plot(xp,R(:,2),'LineWidth',2);
title('R_{45^o}(s)');grid on;
figure; plot(xp,R(:,3),'LineWidth',2);
title('R_{45^o}(s)');grid on;
figure; plot(xp,R(:,4),'LineWidth',2);
title('R_{45^o}(s)');grid on;
```



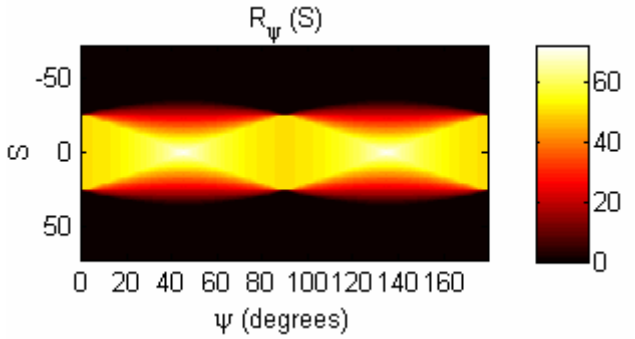
Все графики можно показать в одном окне

```
plot(xp,R(:,1),xp,R(:,2),xp,R(:,3),xp,R(:,4),'LineWidth
',2); grid on;
```



Но это не всегда удобно. Преобразования Радона при большом числе углов часто строится в виде изображения матрицы. Мы это уже много раз делали

```
theta = 0:179;
[R,xp] = radon(I,theta);
imagesc(theta,xp,R);
title('R_{\psi} (S)');
xlabel('\psi (degrees)'); ylabel('S');
set(gca,'XTick',0:20:180);
colormap(hot); colorbar;
```



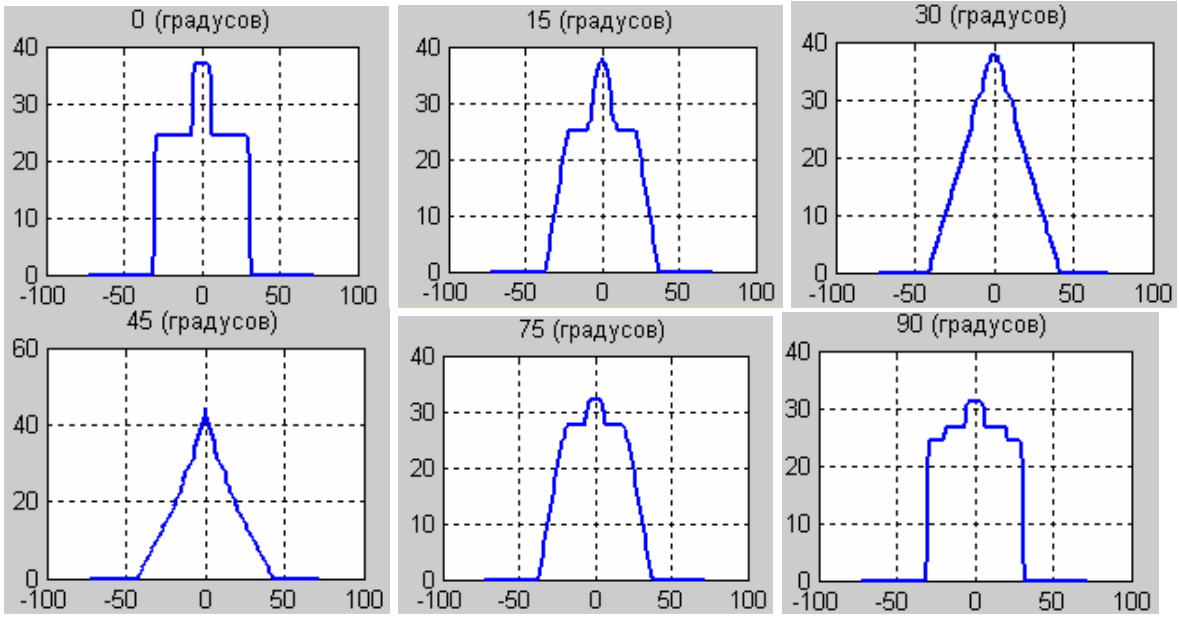
Пример. Сформируем небольшое тестовое изображение, состоящее из трех прямоугольников различной яркости, и построим его ПР.

```
% Пример демонстрирует преобразование Радона.
% и формирование изображения.
I=zeros(100, 100); I(20:80,20:80)=0.4;
I(30:70,45:55)=0.6; I(45:55,45:55)=1;
imshow(I);
```

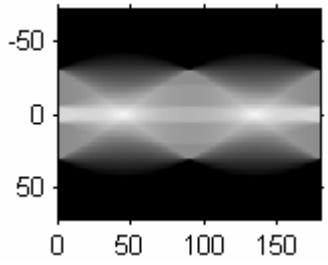
```
[R, xp]=radon(I, 0:179); % Прямое преобразование Радона.
% Вывод на экран проекций на оси X, Y и под углом в 45°.
```

Проекции данного изображения на ось X, и на оси с углами под углом в 15°, 30°, 45°, 75° и на ось Y приведены соответственно на следующих рисунках. По горизонтали на всех графиках отложен радиальный параметр s.

```
figure, plot(xp,R(:,1),'LineWidth',2);
title('0 (градусов)'); grid on;
figure, plot(xp,R(:,16),'LineWidth',2);
title('15 (градусов)'); grid on;
figure, plot(xp,R(:,31),'LineWidth',2);
title('30 (градусов)'); grid on;
figure, plot(xp,R(:,46),'LineWidth',2);
title('45 (градусов)'); grid on;
figure, plot(xp,R(:,76),'LineWidth',2);
title('75 (градусов)'); grid on;
figure, plot(xp,R(:,91),'LineWidth',2);
title('90 (градусов)'); grid on;
```




```
figure, imshow( R,[],'XData',0:179, 'YData', xp);
```



4. Обратное преобразование Радона. Функция iradon.

Функция прямого преобразования Радона `radon` для своего выполнения требует установки двух основных параметров – исходного изображения `I` и углов `theta`.

```
R = radon(I,theta);
```

Функция `iradon` реализует восстановление изображения `I` на основе проекционных данных, которые содержатся в массиве `R`.

```
IR = iradon(R,theta);
```

Параметр `theta` описывает углы (в градусах), под которыми получена каждая проекция.

Пример 1. Определи прообраз функции

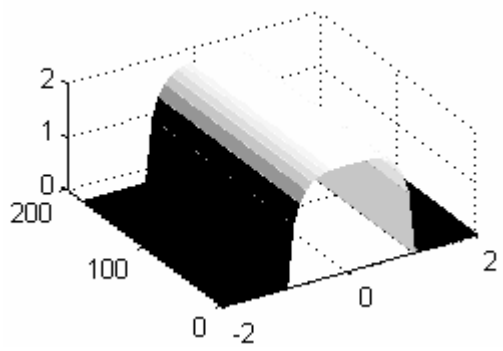
$$R(s,\varphi)=\begin{cases}2\sqrt{1-s^2}, & |s|\le 1\\ 0 \qquad\quad , & |s|>1\end{cases}$$

Как он выглядит, мы уже знаем – это характеристическая функция единичного круга

$$f(x,y)=\begin{cases}1, & x_0^2+y_0^2\le 1\\ 0 \ , & x_0^2+y_0^2>1\end{cases}$$

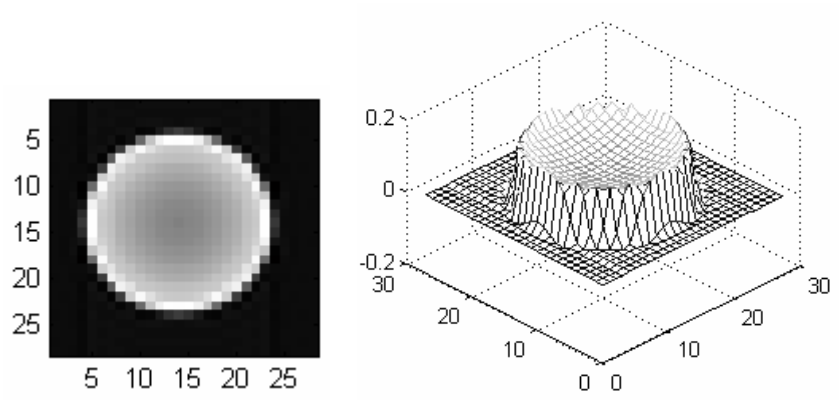
Проверим это с помощью функции `iradon`. Создадим матрицу R, соответствующую функции $R(s,\varphi)$.

```
[Ph,S]=meshgrid( 0:180, -2:0.1:2);
C=abs(S)<=1;
RSPh=zeros(size(C));
S2=S.^2;
RSPh(C)=2.*sqrt(1-S2(C).^2);
mesh(S,Ph,RSPh);
```



Теперь выполним обратное ПР.

```
I=iradon(RSPh,0:180);
imagesc(I); colormap(gray);  % образ слева
mesh(I);                     % график справа
```



Результат похож на ХФ круга, но точность недостаточная и, кроме того, требуется приведение к физическим координатам.

Рассмотрим подробнее синтаксис функции `iradon`. Он может быть одним из следующих:

```
I=iradon(P, theta)
I=iradon(P, theta, interp, filter, d, n)
[I, h]=iradon(...)
```

Функция `I=iradon(P,theta)` выполняет реконструкцию изображения `I` по его проекционным данным, которые содержатся в массиве `P`. Строки `P` являются данными параллельно-лучевых проекций. В функции `iradon` центр вращения является центральной точкой проекций и определяется по выражению `ceil(size(P,1)/2)`. Параметр `theta` описывает углы (в градусах), под которыми получена каждая проекция. Этот параметр может представлять собой вектор, содержащий углы или скаляр, описывающий угол между проекциями `D_theta`. Когда `theta` представлен вектором, тогда он должен содержать углы с равномерной разбивкой. Когда `theta` представлен скаляром, описывающим D_theta, тогда проекции берут согласно углам `theta=m*D_theta`, где `m=0,1,2,...,size(P,2)-1`. Когда исходными данными является пустая матрица ([]), тогда параметр `D_theta` по умолчанию равен `180/size(P,2)`.

Функция `iradon` использует алгоритм фильтрации обратных проекций для выполнения инверсного преобразования Радона. Фильтр проектируется непосредственно в частотной области и умножается на функцию преобразования Фурье проекций. Для ускорения вычислений функции преобразования Фурье проводятся специальные преобразования над проекциями.

Функция `I=iradon(P,theta,interp,filter,d,n)` содержит описание параметров, которые используются при инверсных преобразованиях Фурье. Существует также возможность точного определения некоторых комбинаций последних четырех аргументов. Для пропущенных параметров функция `iradon` по умолчанию устанавливает некоторые значения.

Параметр interp определяет тип интерполяции, который используется в backprojection. Приведем список доступных опций:

'nearest' - интерполяция по ближайшей окрестности;

'linear' - линейная интерполяция (по умолчанию);

'spline' - сплайновая интерполяция

Параметр filter описывает какой тип фильтра используется для частотной фильтрации. Параметр filter представляет собой строку, в которой описаны несколько стандартных фильтров:

'Ram-Lak' - усеченный фильтр Рама-Лака (устанавливается по умолчанию). Частотный отзыв этого фильтра равен | f |. Одним из недостатков фильтра Рама-Лака является то, что он чувствительный к шуму на проекциях. Поэтому он используется в комбинациях с другими фильтрами.

'Shepp-Logan' - фильтр Шепа-Логана, умноженный на фильтр Рама-Лака через фазовую функцию.

'Cosine' - косинусный фильтр, умноженный на фильтр Рама-Лака через косинусную функцию.

'Hamming' - фильтр Хэмминга, умноженный на фильтр Рама-Лака через окно Хэмминга.

'Hann' - фильтр Ханна, умноженный на фильтр Рама-Лака через окно Хана.

Параметр d представляет собой скаляр в диапазоне (0, 1] и служит для модификации фильтра в плане масштабирования по частотной оси. По умолчанию он равен 1. Когда d меньше 1, тогда фильтр сжимает частотный диапазон до [0,d], нормирует частоты; все частоты, которые больше значения d, приравниваются к 0.

Параметр n представляет собой скаляр, описывающий число строк и столбцов в восстановленном изображении. Когда параметр n не описан, тогда размеры определяются исходя из длины проекций. n=2*floor(size(P, 1)/(2*sqrt(2))).

После определения параметра n, функция iradon восстанавливает изображение, не изменяя масштаб данных. Когда проекции были вычислены с помощью функции radon, тогда размеры восстановленного и исходного изображений могут не совпадать.

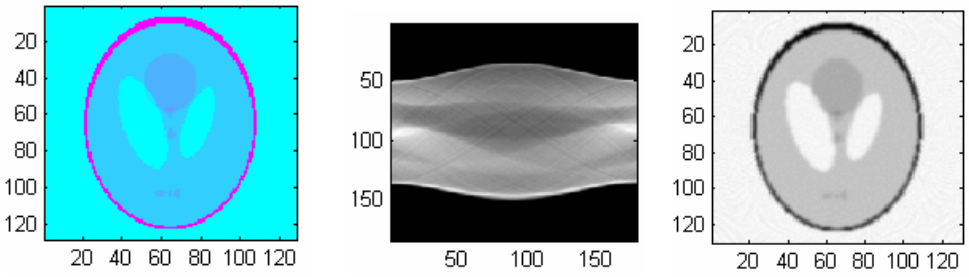
Функция [I,h]=iradon(...) возвращает частотный отклик фильтра в вектор h.

Требования к исходным данным. Все исходные и результирующие аргументы должны быть представлены в формате double.

Пример. Рассмотрим пример восстановления изображений на основе данных параллельных проекций. В качестве тестового изображения используется готовая модель MatLab изображения головы человека, которая в приложении

Image Processing Toolbox генерируется функцией phantom (этой функцией создается подходящая матрица заданного размера). Это изображение демонстрирует различные особенности, которые присутствуют в реальных томографических изображениях.

```
% phantom(n) создает матрицу n x n с с элементами от 0 до 1
% в форме эллипсов разной толщины
P=phantom(128);
imagesc(P);   % образ матрицы P (рисунок слева)
colormap('Cool'); axis image;
R=radon(P, 0:179);
imagesc(R); % образ матрицы R (рисунок в центре)
colormap(gray);
I=iradon(R, 0:179,'nearest', 'Hann');
figure,
G=flipud(gray);% перевернуть матрицу палитры 'gray' сверху вниз
imagesc(I);    % образ матрицы I (рисунок справа)
colormap(G);
```

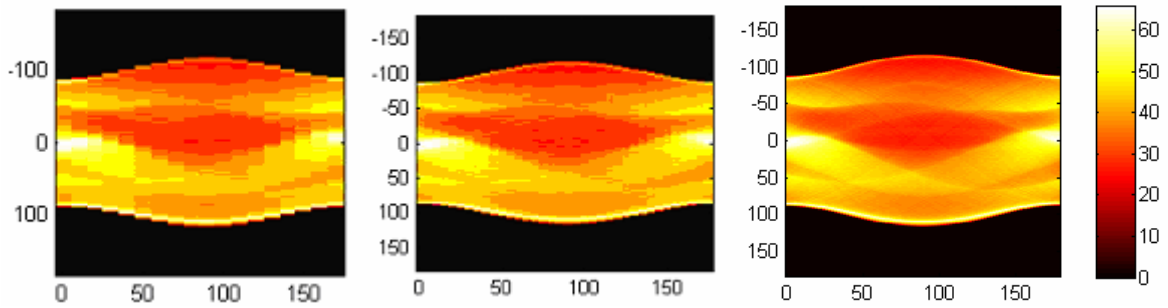


Вычислим преобразований Радона исходного изображения (рисунок вверху слева) для трех различных значений theta. R1 имеет 18 проекций, R2 – 36 проекций и R3 – 90 проекций.

```
theta1 = 0:10:170; [R1,xp] = radon(P,theta1);
theta2 = 0:5:175;  [R2,xp] = radon(P,theta2);
theta3 = 0:2:178;  [R3,xp] = radon(P,theta3);
```

Визуализируем результаты преобразований Радона. На изображении внизу показаны матрицы R1, R2, R3.

```
figure, imagesc(theta1,xp,R3); colormap(hot);
figure, imagesc(theta2,xp,R3); colormap(hot);
figure, imagesc(theta3,xp,R3);
colormap(hot); colorbar;
xlabel('\theta'); ylabel('x\prime');
```



Отметим, что первая строка матрицы преобразований соответствует проекции под углом 0°, что соответствует интеграции данных в

горизонтальном направлении. Проекция под углом 90° соответствует интеграции данных в вертикальном направлении.

Восстановление изображения модели головы человека на основе проекционных данных с последующей их визуализацией реализуется следующим образом.

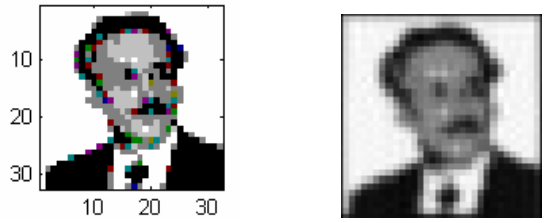
```
I1 = iradon(R1,10); % задаем шаг приращения угла 10
I2 = iradon(R2,5);
I3 = iradon(R3,2);
G=flipud(gray);% перевернуть матрицу палитры 'gray' сверху вниз
figure, imshow(I1); colormap(G);
figure, imshow(I2); colormap(G);
figure, imshow(I3); colormap(G);
```

На изображении внизу показано результаты реконструкции с использованием различного количества проекций. Отметим, что изображение `I1` реконструировано с использованием 18 проекций, изображение `I2` – с использованием 36 проекций, а изображение `I3` – с использованием 90 проекций. Как видно из реконструированных изображений количество проекционных данных пропорционально качеству восстанавливаемых изображений

Результаты применения обратного преобразования Радона к модели головы человека.

Пример

```
[A,m] = imread('Pogorelov32x32.bmp');
image(A);colormap(m);
R=radon(A,0:180);
I=iradon(R, 0:180);
imshow( I,[]);
```



Функция `iradon` восстанавливает полутоновое изображение `IR` из матрицы проекций `R`, которые получены под соответствующими углами `theta`. Значения углов в векторе `theta` определяют направление осей. В общей сложности, функция `iradon` позволяет задать четыре параметра. Кроме описанных двух параметров, функция `iradon` задает также тип интерполяции, которая используется при восстановлении исходного

изображения. Еще один параметр дает возможность определить фильтр предварительной обработки проекций, из которых будет восстанавливаться изображение. Если эти параметры не указываются, то используется соответствующее значение по умолчанию.