报告

对于 RDA 算法的改进

2015.01.15

在前几天研究并解决了一维 LFM 脉冲压缩中的很多问题后，昨天我又对二维频域实现 SRC 的 RD 算法进行了公式推导，详见《2015.01.14. 在二维频域相位相乘实现 SRC，RDA 公式推导》。基于这些结果，我对成像算法进行了改进，并利用点目标仿真来验证成像算法的有效性。

对于成像算法的改进，主要有以下两点：

a) 一是解决了一维 LFM 信号脉冲压缩中的相关问题，这一部分对于成像算法的帮助是：将该成果用于成像算法中的距离压缩部分，保证距离压缩结果的保相性，并且得到了理想的结果；

b) 二是基于昨天对带有 SRC 的 RDA 公式的推导，我可以准确知道每一步处理的相位情况，便于逐步检查成像处理的结果是否符合理论分析；此外，尤其是对于最后的方位压缩，其匹配滤波器的改进（该改进直接影响到能否用于干涉 SAR 处理，详见公式推导的报告）真正保证了成像结果的保相性，并且以推导的公式作为参考，可以检查成像结果的峰值点相位是否符合理论分析，验证其保相性。

总之，基于以上两点重要改进，我对成像算法进行了修改，并且成功验证了修改后的成像程序对于单点及多点目标成像结果的保相性。

下面进行说明。

一、 成像算法的修改

1. 距离压缩滤波器的修改

原来距离压缩滤波器采取方式 3，即直接在频域生成匹配滤波器，但是我认为该方式存在近似，不够准确。所以现在改用方式 2，即时域生成方式：复制脉冲，补零 DFT，再取复共轭得到频域的 MF。该种方式生成的滤波器具有非常好的保相性，而且可以通过在复制脉冲时设置参数实现将峰值点压至我们想要的任意位置。关于该滤波器以及一维压缩情况详见我前几天的报告。

这里采用完全相同的方式，并选择参数使得脉冲压缩峰值点被压至 LFM 脉冲中心。

在成像处理的 MATLAB 程序中，代码如下：

```
% ==========================================================================
% 生成距离向匹配滤波器
% 采用方式 2
% 时域复制脉冲，补零 fft，再取复共轭。
% 同时，在设计复制脉冲时，设计一个适当的时延，使得最终的脉压峰值位于 LFM 脉冲中心；
t_ref = ( -Nr/2 : (Nr/2-1) )/Fr;        % 用来生成距离 MF 的距离时间轴，长度为 Nr（Nr<Nrg）；
t_ref_mtx = ones(Naz,1)*t_ref;      % 矩阵形式
w_range_ref = (abs(t_ref_mtx)) <= ((Tr/2).*ones(Naz,Nr));
% 距离向包络，即距离窗，限制脉冲长度
w_ref = kaiser(Nr,2.5);             % 距离向，构建 Kaiser 窗，此为列向量。
w_ref = ones(Naz,1)*(w_ref.');      % 构成矩阵形式，每一行都相同的加窗。

s_ref = w_range_ref.*exp((1j*pi*Kr).*((t_ref_mtx).^2));      % 复制（发射）脉冲，未加窗。
% s_ref = w_ref.*w_range_ref.*exp((1j*pi*Kr).*((t_ref_mtx).^2)); % 复制（发射）脉冲，加了窗。

% ***************************************************************************
% 对 s_ref 设计一个时间延迟，使得峰值点被压至 LFM 脉冲中心；
% 同时，这部分也完成对 s_ref 的补零；
s_ref = [s_ref,zeros(Naz,Nrg-Nr)];          % 对复制脉冲，末端补零至长度 Nrg；
s_ref = circshift(s_ref,[0 -Nr/2]);         % 向左循环移位（Nr/2）；
% 要设计将峰值点压至 LFM 脉冲中心，既可以用上面的两行代码；
% 也可以用下面的这两行；
% s_ref = fftshift(s_ref,2);
% s_ref = [s_ref(:,1:Nr/2),zeros(Naz,Nrg-Nr),s_ref(:,Nr/2+1:end)];
% ***************************************************************************

S_ref = fft(s_ref,NFFT_r,2);                % 复制脉冲的距离傅里叶变换，零频在两端。
H_range = conj(S_ref);                      % 距离向匹配滤波器，零频在两端。
% 对距离频谱进行：距离压缩
S_range_c = S_range.*H_range;
s_rc = ifft(S_range_c,[],2);            % 完成距离压缩，包络校正和一次运补，回到二维时域。
% s_rc 的长度为：Naz*Nrg。未去除弃置区。
% 这里都统一不考虑去除弃置区问题
% ==========================================================================
```

2. 方位向匹配滤波器的修改

依据《2015.01.14. 在二维频域相位相乘实现 SRC，RDA 公式推导》，将方位向匹配滤波器修改为：

$$H_{az}\left(f_\eta\right)=\exp\left\{+j\frac{4\pi R_0 f_0 D\left(f_\eta,V_r\right)}{c}-j\frac{4\pi R_0 f_0}{c}\right\}$$
$$=\exp\left\{+j\frac{4\pi R_0\left[D\left(f_\eta,V_r\right)-1\right]f_0}{c}\right\}$$

注意上式与卡明书上公式的区别。

在 MATLAB 中，相应程序代码改为：

```
% ==================================================================
Haz = exp(1j*4*pi.*((D_fn_Vr-1)*R0_RCMC).*f0./c);
% 将 D(f_η,V_r) 换成 [D(f_η,V_r)-1];
% 这里的频率轴和距离多普勒域的方位频谱是对应的。
S_rd_c = S_rd_rcmc.*Haz;          % 乘以匹配滤波器
s_ac = ifft(S_rd_c,[],1);         % 完成方位压缩，变到图像域。结束。
% ==================================================================
```

至此，对原来的 RDA 成像程序的修改就全部完成。下面进行点目标仿真，以考察其保相性。

二、 仿真

1. 情况一：单点目标仿真

（设置该点目标位于场景中心，并将其最近斜距作为 SRC 滤波器的参考距离）

这里为了专门研究成像的保相性问题，我尽可能将场景设计的简单，以免需要排查配准中问题（采用“理论配准”）。

该点目标位于场景中心，其最近斜距为 20km。设置基线距离 B，使得天线 B 到该点目标的距离（20km-5m）比天线 A 到该点目标的距离（20km）小 5m，对应 2 个距离向采样单元。这样天线 A 的成像结果中，点目标中心位于（513,257）；而天线 B 的成像结果中，点目标中心位于（513,255）；此种情况下，可以不用配准，直接取出各自的峰值点相位，检查其保相性；并将天线 A 和天线 B 成像结果的峰值点相位作差，检查其是否符合干涉相位的理论结果。同时，由于这里只有一个点目标，并且正好相差 2 个距离向采样单元，所以可以直接比较配准前后的相位来检查配准的有效性。

1) 点目标原始数据生成的参数设置

```
% 目标 1（场景中心的点目标）
r1 = R0;                          % 目标 1 的最近斜距
x1 = r1*sin(theta);               % 地距平面上的 x 轴坐标
z1 = 0;                           % z 轴坐标（设置为在地平面上）
```

```
y1 = 0 + r1*tan(sita_r_c);                % 目标 1 的方位向距离（地距平面的 y 轴坐标）
target_x = x1;
y_azimuth = y1;
num_target_x = length(target_x);          % 整个场景，x 轴范围的大小
num_y_azimuth = length(y_azimuth);        % 整个场景，y 轴范围的大小
target_z = 0.*ones(num_target_x,num_y_azimuth);
r_range = sqrt(((target_x.')*ones(1,num_y_azimuth)).^2 + (H-target_z).^2);
% 目标的最近斜距，矩阵。
% 计算每个目标各自的波束中心穿越时刻
nc_target = ((ones(num_target_x,1)*y_azimuth) - r_range.*tan(sita_r_c))./Vr;
% 每个目标的波束中心穿越时刻，矩阵
B = x1 - sqrt((20e3-5)^2-H^2);
% 选择基线距，使得目标到天线 B 的最近斜距比到天线 A 的少 5m，对应 2 个距离采样单元。
```

2) 考察相位情况

天线 A，成像结果的点目标中心位于：（513,257）；

天线 B，成像结果的点目标中心位于：（513,255）；

表 1 情况一，考察成像结果的相位

	点目标 中心坐标	中心处 相位 （rad）	配准后 相位 （rad）	理论相位 $\left(\phi_0-\frac{4\pi R_0 f_0}{c}-\frac{\pi}{4}\right)$ （rad）	理论相位与成像结果的相位之差是否等于2π的整数倍 （$*2\pi$）
天线 A	（513,257）	0.1447		-4.440113283e+06	-706665.9999681966
天线 B	（513,255）	-1.9497	-1.9497	-4.439003254e+06	-706488.9999681892
天线 A 与天线 B 的干涉相位	（513,257）	2.0944		$-\frac{4\pi}{\lambda}\left(R_A-R_B\right)\approx$ -1110.029404268	-177.0000000073590
点目标的后向散射特性：$A_0=\sigma_0\cdot e^{j\phi_0}$，其中$\phi_0=5.1191(rad)$					

上述要注意的是点目标的后向散射特性中的相位ϕ_0，这是在原始回波数据生成时用随机数生成器随机生成的。这些参数在原始数据中保存下来了，因此这里可以用其进行计算。

可以看到，不论是天线 A 还是天线 B 各自的峰值点相位，还是天线 A 和天线 B 成像结果共轭相乘得到的干涉相位，它们都和理论分析符合的非常好，以很高的精度与理论结果满足2π的整数倍。因此，该点目标的保相性是得以保证了的。

2. 情况二：单点目标仿真

（该点目标设计为偏离场景中心一定距离处，但也令其峰值点恰好位于整数采样点处，这样便于成像后提取峰值点相位。由于 SRC 滤波器的参考距离选择在场景中心处，即与情况一相同，该种情况下的点目标二次距离压缩结果是有残余的。我们考察这会对成像结果造成多大影响）

该种情况与情况一略有不同。

先说不同部分：场景点目标被设置在偏离场景中心的位置，其最近斜距设置为（20km+100m），高度为零。特别注意，该点目标的和情况一中的点目标都是设置在地平面上的，其最近斜距不同意味着它们的地距 X 轴坐标不同，也意味着天线到它们的波束下视角不同。此外，该点目标依旧设置在整数采样点处。对于天线 A，其成像点目标中心偏离场景（513,257）恰好 40 个距离向采样单元，位于（513,297）。

相同部分：同样设置基线距离 B，使得天线 B 到该点目标的距离（20km+100m-5m）比天线 A 到该点目标的距离（20km+100m）小 5m，对应 2 个距离向采样单元。原因同情况一。天线 B 的成像结果中，点目标中心位于（513,295）。

如上所述，设置该种情况作为情况一的对比是因为：情况一中，点目标位于场景中心，设置 SRC 滤波器时参考距离也选择场景中心，因此点目标的二次距离压缩是完全的，没有相位残余；而情况二中，点目标偏离场景中心，而 SRC 滤波器的设置同情况一，因此该点目标的二次距离压缩结果是有残余的。我们正是要在此种情况下考察该残余会造成多大程度的影响。

1) 点目标原始数据生成的参数设置

```
% 目标 1（场景中心的点目标）
r1 = R0;                                % 目标 1 的最近斜距
x1 = r1*sin(theta);                     % 地距平面上的 x 轴坐标
z1 = 0;                                 % z 轴坐标（设置为在地平面上）
y1 = 0 + r1*tan(sita_r_c);              % 目标 1 的方位向距离（地距平面的 y 轴坐标）
target_x = sqrt((20e3+100)^2-H^2);
y_azimuth = y1;
num_y_azimuth = length(y_azimuth);  % 整个场景，y 轴范围的大小
target_z = 0.*ones(num_target_x,num_y_azimuth);
r_range = sqrt(((target_x.')*ones(1,num_y_azimuth)).^2 + (H-target_z).^2);
% 目标的最近斜距，矩阵。
% 计算每个目标各自的波束中心穿越时刻
nc_target = ((ones(num_target_x,1)*y_azimuth) - r_range.*tan(sita_r_c))./Vr;
% 每个目标的波束中心穿越时刻，矩阵
```

B = sqrt((20e3+100)^2-H^2) - sqrt((20e3+100 - 5)^2-H^2);

% 选择基线距，使得目标到天线 B 的最近斜距比到天线 A 的少 5m，对应 2 个距离采样单元。

注：加粗部分是与情况一不同的地方。

2) 考察相位情况

天线 A，成像结果的点目标中心位于：（513,297）；

天线 B，成像结果的点目标中心位于：（513,295）；

表 2 情况二，考察成像结果的相位

	点目标中心坐标	中心处相位（rad）	配准后相位（rad）	理论相位 $\left(\phi_0-\frac{4\pi R_0 f_0}{c}-\frac{\pi}{4}\right)$（rad）	理论相位与成像结果的相位之差是否等于 2π 的整数倍（$*2\pi$）
天线 A	（513,297）	-1.9497		-4.462313871e+06	-710198.9999682845
天线 B	（513,295）	2.2391	2.2391	-4.461203842e+06	-710022.9999682714
天线 A 与天线 B 的干涉相位	（513,297）	2.0944		$-\frac{4\pi}{\lambda}\left(R_A-R_B\right)\approx$ -1110.029404268	-177.0000000131580
点目标的后向散射特性：$A_0=\sigma_0\cdot e^{j\phi_0}$，其中 $\phi_0=5.1191(rad)$					

注：情况二和情况一的原始数据是独立生成的，但是两次随机数产生的点目标的后向散射特性中的相位竟然恰好是一样的。这里专门说明，是因为这只是个巧合，并不是故意这样设计的。

根据表 2，我们可以看到其结果类似表 1，仿真相位也非常精确。这里并没有体现出 SRC 滤波残余相位的影响。卡明的书上给出了可以这样进行 SRC 滤波的条件，但我不是太明白，也暂时不知道如何计算这个可以忽略的临界条件。不过在这些仿真中，暂时没有遇到问题。因此，我暂时忽略这个问题，采用该成像算法进行处理。

此外，从情况一和情况二的仿真结果来看，配准也是达到了要求的。关于“理论配准”的相关内容详见文件夹《2014.12.24. 问题排查》中用单点目标和三点目标来考察配准问题的报告。

最后，还要说明：我使用卡明书上给出的方位压缩滤波器，即

$$H_{az}\left(f_\eta\right)=\exp\left\{+j\frac{4\pi R_0 f_0 D\left(f_\eta,V_r\right)}{c}\right\}$$

得到的仿真结果和《2015.01.14. 在二维频域相位相乘实现 SRC，RDA 公式推导》报告中推导出来的结果也是完全吻合的。进行这样的方位压缩，确实丢失了有关目标距离的相位信息，而只剩下$\left(\phi_0\pm\frac{\pi}{4}\right)$，因此该种成像结果是没法用于干涉处理的。这既说明了我昨天推导的公式是正确的，也验证了我的仿真程序是没有问题的。

（我之前都是使用该种方位压缩滤波器，但是对于场景的处理也得到了相应的干涉相位图，这是怎么回事？按照公式推导的结果和这里的点目标仿真结果，场景处理时应该无法得到干涉相位图，问题出在哪儿？）

除了上述单点目标仿真，我还进行了多点目标仿真。由于我接下来要进行点目标的干涉处理，包括单点目标和多点目标的。因此这里我不再对多点目标成像结果的相位进行说明，接下来进行多点目标的干涉处理时，自然会用到它们的峰值点相位来反演高程信息，那时候一并给出多点目标的保相性情况。

至此，修改后的 RDA 算法及 MATLAB 程序的保相性得到了验证，并且被证明能够很好地进行干涉处理；同时，理论配准也被证明是可行的。

因此，到目前为止，我基本可以认为**成像算法、理论配准**都没有问题，可以放心将问题来源锁定在干涉处理的其他环节上了。

WD

2015 年 1 月 15 日 17:29 p.m.

修改后的 RDA 成像程序：

```
function [s_ac,R0_RCMC,Parameter] = RDA_imaging2_v3(raw_data_type)
% 对函数    RDA_imaging_v2（）      进行了以下修改：
% 1）将距离脉压方式改为方式 2，并同时设置复制脉冲的参数，使得峰值点被压至脉冲中心；
% 2）该版程序不考虑去除弃置区问题；
% 3）修改方位向匹配滤波器，使得成像结果能够有更好的保相性；
% 将以上修改后的该版本记为：RDA_imaging2_v3（）
%
% 本函数用来对生成的原始数据成像
% 利用：
% 1）带有 SRC（利用二维频域相位相乘的方法）的 RD 算法
% 2）可选择是否进行两步式运动误差补偿
%
% 载入原始数据时：
% 1）raw_data_type == 1，代表天线 A 对应的原始数据；
% 2）raw_data_type == 2，代表天线 B 对应的原始数据；
%
% 返回值：
% 1）s_ac            是成像结果；
% 2）R0_RCMC     是随距离线变化的斜距，用于后面计算平地相位；
% 2）Parameter      代表成像结果中的一些参数，用于后面进行平地相位计算；
%
% 程序版本截止到：
% 2015.01.15. 17:58 p.m.

%%
% 载入原始数据
load test_raw_data_one_point;

if raw_data_type == 1
    s_echo = s_echoA;                % 载入天线 A 对应的原始数据
end
if raw_data_type == 2
    s_echo = s_echoB;                % 载入天线 B 对应的原始数据
end

%%
% ------------------------------------------------------------------------------
% 计算在 LOS 方向的运动误差，这是进行“运动补偿”的依据
% ------------------------------------------------------------------------------
r = tr_mtx.*c/2;                     % 用于运动误差计算的斜距 r；
% 沿 LOS 方向的运动误差
delta_r = delta_x_t.*( sqrt(r.^2-H^2)./r ) - delta_z_t.*(H./r);  % 沿 LOS 方向，总的运动误差
delta_r = -delta_r*cos(sita_r_c);
```

```
delta_r_R0 = delta_x_t.*( sqrt(R0^2-H^2)/R0 ) - delta_z_t.*(H/R0);
% 沿 LOS 方向，场景中心处的运动误差（空不变误差）
delta_r_R0 = -delta_r_R0*cos(sita_r_c);

%%
% ---------------------------------------------------------------
% 距离压缩，包络校正，一次运动补偿
% ---------------------------------------------------------------
S_range = fft(s_echo,NFFT_r,2);        % 进行距离向傅里叶变换，零频在两端。

% 生成距离向匹配滤波器
% ====================================================
% 采用方式 2
% 时域复制脉冲，补零 fft，再取复共轭。
% 同时，在设计复制脉冲时，设计一个适当的时延，使得最终的脉压峰值位于 LFM 脉冲中心；
t_ref = ( -Nr/2 : (Nr/2-1) )/Fr;          % 用来生成距离 MF 的距离时间轴，长度为 Nr（Nr<Nrg）；
t_ref_mtx = ones(Naz,1)*t_ref;            % 矩阵形式
w_range_ref = (abs(t_ref_mtx)) <= ((Tr/2).*ones(Naz,Nr));
% 距离向包络，即距离窗，限制脉冲长度
w_ref = kaiser(Nr,2.5);                   % 距离向，构建 Kaiser 窗，此为列向量。
w_ref = ones(Naz,1)*(w_ref.');            % 构成矩阵形式，每一行都相同的加窗。

s_ref = w_range_ref.*exp((1j*pi*Kr).*((t_ref_mtx).^2));      % 复制（发射）脉冲，未加窗。
% s_ref = w_ref.*w_range_ref.*exp((1j*pi*Kr).*((t_ref_mtx).^2)); % 复制（发射）脉冲，加了窗。

% ************************************************************************
% 对 s_ref 设计一个时间延迟，使得峰值点被压至 LFM 脉冲中心；
% 同时，这部分也完成对 s_ref 的补零；
s_ref = [s_ref,zeros(Naz,Nrg-Nr)];        % 对复制脉冲，末端补零至长度 Nrg；
s_ref = circshift(s_ref,[0 -Nr/2]);       % 向左循环移位（Nr/2）；
% 要设计将峰值点压至 LFM 脉冲中心，既可以用上面的两行代码；
% 也可以用下面的这两行；
% s_ref = fftshift(s_ref,2);
% s_ref = [s_ref(:,1:Nr/2),zeros(Naz,Nrg-Nr),s_ref(:,Nr/2+1:end)];
% ************************************************************************

S_ref = fft(s_ref,NFFT_r,2);              % 复制脉冲的距离傅里叶变换，零频在两端。
H_range = conj(S_ref);                    % 距离向匹配滤波器，零频在两端。
% ====================================================
% 计算用于 包络校正 和 一次运动误差 的参考函数

% 包络校正，参考函数
He_fr = exp(1j*4*pi/c.*delta_r_R0.*fr_mtx); % 距离零频在中心
```

```
He_fr = fftshift(He_fr,2);                    % 距离零频在两端

% 一次运动补偿，参考函数
Hc1 = exp(1j*4*pi/lamda.*delta_r_R0);    % 距离零频在中心
Hc1 = fftshift(Hc1,2);                    % 距离零频在两端
% ================================================
% 对距离频谱进行：距离压缩 + 包络校正 + 一次运动补偿
S_range_c = S_range.*H_range;
% S_range_c = S_range.*H_range.*He_fr.*Hc1;    % 零频在两端。
s_rc = ifft(S_range_c,[],2);          % 完成距离压缩，包络校正和一次运补，回到二维时域。
% s_rc 的长度为：Naz*Nrg。未去除弃置区。
% 这里都统一不考虑去除弃置区问题
% ================================================

%%
% ----------------------------------------------------------------
% 变换到二维频域，进行 SRC
% ----------------------------------------------------------------
s_rc = s_rc.*exp(-1j*2*pi*fnc.*(ta.'*ones(1,Nrg)));       % 数据搬移
S_2df = fft(s_rc,NFFT_a,1);        % 方位向傅里叶变换，到距离多普勒域。
S_2df = fft(S_2df,NFFT_r,2);       % 距离向傅里叶变换，到二维频域
% !!! 注意：距离向零频在两端。!!!
% ================================================================
% 设置方位频率轴——这是关键点
fa = fnc + fftshift(-NFFT_a/2:NFFT_a/2-1)/NFFT_a*Fa;      % 方位频率轴如此设置。
% ================================================================
D_fn_Vr = sqrt(1-lamda^2.*(fa.').^2./(4*Vr^2));              % 大斜视角下的徙动因子
K_src = 2*Vr^2*f0^3.*D_fn_Vr.^3./(c*R0*(fa.').^2);          % 列向量
K_src_1 = 1./K_src;                % 列向量。为了后面能使用矩阵乘法，这里先求倒数
H_src = exp(-1j*pi.*K_src_1*(fr.^2));          % 二次距离压缩滤波器。距离向，零频在中间。
% 这是矩阵，大小 Naz*Nrg
H_src = fftshift(H_src,2);      % （左右半边互换）距离向，零频在两端。 !!! 这很关键!!!
S_2df_src = S_2df.*H_src;
% 这一步点乘时，要注意两者的距离向频率轴应该对应上，不然会出错!!
% 这就是为什么上面的 H_src 要 fftshift 的原因!!

S_rd = ifft(S_2df_src,[],2);          % 完成二次距离压缩（SRC），回到距离多普勒域。

%%
% ----------------------------------------------------------------
% 距离多普勒域，进行距离徙动校正
% ----------------------------------------------------------------
% 每一个最近斜距（R0）都随着距离门的不同而改变。
tr_RCMC = 2*R0/c + ( -Nrg/2 : (Nrg/2-1) )/Fr;     % 用于 RCMC 的时间轴（其实和 tr 相同）；
```

```
R0_RCMC = (c/2).*tr_RCMC;
% 随距离线变化的 R0，记为 R0_RCMC，用来计算 RCM 和 Ka。
delta_Rrd_fn = ((1-D_fn_Vr)./D_fn_Vr)*R0_RCMC;    % 更精确的 RCM 值。

num_range = c/(2*Fr);     % 一个距离采样单元，对应的长度
delta_Rrd_fn_num = delta_Rrd_fn./num_range;
% 每一个方位向频率，其 RCM 对应的距离采样单元数

R = 8;   % sinc 插值核长度
S_rd_rcmc = zeros(NFFT_a,Nrg);       % 用来存放 RCMC 后的值

h = waitbar(0,'RCMC, please wait');
for p = 1 : NFFT_a
    for q = 1 : Nrg     % 此时距离向的长度是 Nrg。
        delta_Rrd_fn_p = delta_Rrd_fn_num(p,q);
        Rrd_fn_p = q + delta_Rrd_fn_p;

        Rrd_fn_p = rem(Rrd_fn_p,Nrg);
       % 由于 RCM 的长度会超过 Nrg，所以这样处理一下。

        Rrd_fn_p_zheng = ceil(Rrd_fn_p);              % ceil，向上取整。
        ii = ( Rrd_fn_p-(Rrd_fn_p_zheng-R/2):-1:Rrd_fn_p-(Rrd_fn_p_zheng+R/2-1)    );
        rcmc_sinc = sinc(ii);
        rcmc_sinc = rcmc_sinc/sum(rcmc_sinc);     % 插值核的归一化
        % ii 是 sinc 插值过程的变量;
        % g(x)=sum(h(ii)*g_d(x-ii)) = sum(h(ii)*g_d(ll));

        % 由于 S_rd 只有整数点取值，且范围有限。因此插值中要考虑它的取值溢出边界
          问题。
        % 这里我采取循环移位的思想，用来解决取值溢出问题。
        if (Rrd_fn_p_zheng-R/2) > Nrg       % 全右溢
            ll = (Rrd_fn_p_zheng-R/2-Nrg:1:Rrd_fn_p_zheng+R/2-1-Nrg);
        else
            if (Rrd_fn_p_zheng+R/2-1) > Nrg       % 部分右溢
                ll_1 = (Rrd_fn_p_zheng-R/2:1:Nrg);
                ll_2 = (1:1:Rrd_fn_p_zheng+R/2-1-Nrg);
                ll = [ll_1,ll_2];
            else
                if (Rrd_fn_p_zheng+R/2-1) < 1       % 全左溢（不可能发生，但还是要考虑）
                    ll = (Rrd_fn_p_zheng-R/2+Nrg:1:Rrd_fn_p_zheng+R/2-1+Nrg);
                else
                    if (Rrd_fn_p_zheng-R/2) < 1            % 部分左溢
                        ll_1 = (Rrd_fn_p_zheng-R/2+Nrg:1:Nrg);
                        ll_2 = (1:1:Rrd_fn_p_zheng+R/2-1);
```

```
                    ll = [ll_1,ll_2];
                else
                    ll = (Rrd_fn_p_zheng-R/2:1:Rrd_fn_p_zheng+R/2-1);
                end
            end
        end
        rcmc_S_rd = S_rd(p,ll);
        S_rd_rcmc(p,q) = sum( rcmc_sinc.*rcmc_S_rd );
    end
    waitbar(p/NFFT_a);
end
close(h);
% S_rd_rcmc  就是 RCMC 后的距离多普勒域频谱。

%%
%{
% ------------------------------------------------------------------
%  回到时域，进行二次运动补偿
% ------------------------------------------------------------------
%  由于去除了弃置区，因此这里用于二次运动补偿的斜距 r 要重新计算
r = ones(Naz,1)*R0_RCMC;
%  下面计算对应去除弃置区后，引入的运动误差
delta_x_t = a*sin(2*pi*w/La*Vr.*(ta.'*ones(1,Nrg)));          %  这是沿地距 x 轴的运动误差
% delta_x_t = 0;
% delta_z_t = a*sin(2*pi*w/La*Vr.*(ta.'*ones(1,Nrg)));       %  这是沿 z 轴的运动误差
delta_z_t = 0;

%  计算，沿 LOS 方向的运动误差
delta_r = delta_x_t.*( sqrt(r.^2-H^2)./r ) - delta_z_t.*(H./r);  %  沿 LOS 方向，总的运动误差
delta_r = -delta_r*cos(sita_r_c);
delta_r_R0 = delta_x_t.*( sqrt(R0^2-H^2)/R0 ) - delta_z_t.*(H/R0);
%  沿 LOS 方向，场景中心处的运动误差（空不变误差）
delta_r_R0 = -delta_r_R0*cos(sita_r_c);

%  下面在二维时域进行二次运动误差补偿
Hc2 = exp(1j*4*pi/lamda.*(delta_r-delta_r_R0));    %  二次运动补偿，参考函数
s_rd_rcmc = ifft(S_rd_rcmc,[],1);                  %  将 RCMC 后的结果变换到时域
s_rd_rcmc_MoCo = s_rd_rcmc.*Hc2;                   %  进行二次运动补偿

S_rd_rcmc_MoCo = fft(s_rd_rcmc_MoCo,[],1);   %  二次运动补偿后，变换到距离多普勒域
%}
```

```
%%
% --------------------------------------------------------------------
%  方位压缩
% --------------------------------------------------------------------
fa_azimuth_MF = fa;             %  方位频率轴，采用和 RCMC 中所用的频率轴相同。
Haz = exp(1j*4*pi.*((D_fn_Vr-1)*R0_RCMC).*f0./c);     %  将  D  换成（D-1）;
%  这里的频率轴和距离多普勒域的方位频谱是对应的。

S_rd_c = S_rd_rcmc.*Haz;     %  乘以匹配滤波器（这是没有进行二次运动补偿时采用的）
% S_rd_c = S_rd_rcmc_MoCo.*Haz;        %  乘以匹配滤波器
s_ac = ifft(S_rd_c,[],1);                          %  完成方位压缩，变到图像域。结束。

%  作图
%  图 7——成像结果
figure;
imagesc(abs(s_ac));
title('点目标成像');
xlabel('距离时域（采样点）');
ylabel('方位时域（采样点）');

%%
%  需要返回的一些参数
Parameter = [ Naz;                  % Parameter  的第一行代表  Naz
              H;                    % Parameter  的第二行代表  H
              lamda ];              % Parameter  的第三行代表  lamda


end
```