文章编号:1000-0887(2013)05-0506-12

基于 CT 图像重建的多重集合分裂可行性问题应用分析*

王培元, 周海云

(军械工程学院 基础部数学教研室,石家庄 050003)

摘要: 为了较好地应用 CQ 算法解决稀疏角度 CT 图像重建的问题,提出了一种新的实时的分块逐次混合算法.首先将稀疏角度 CT 图像重建的问题转化成分裂可行性问题.其次,通过分析非空闭凸集 C 和 Q 的不同的定义,在 N 维实空间中分别针对不同的 CQ 算法给出了 7 种不同的实现方案.通过试验,分别对不同算法及其方案的重建精度和收敛速度进行了对比分析,并对多重集合分裂可行性问题算法中约束权因子的选取及其对输出的影响进行了研究,从而给出了 CQ 算法在稀疏角度 CT 图像重建问题中应用的最佳凸集定义方案.以此为基础,给出了所提出算法的最佳实现方案.试验结果表明,该算法收敛速度快,重建精度高,为多重集合分裂可行性问题及其改进算法在该重建问题上的应用提供了参考.

关 键 词: CQ 算法; 多重集合分裂可行性问题; 非空闭凸集; 图像重建

中图分类号: TP301.6 **文献标志码:** A

DOI: 10.3879/j.issn.1000-0887.2013.05.009

引 言

分裂可行问题(SFP)是最优化理论中的一个重要问题,在信号处理和图像重建等领域有着广泛的应用[1-2].1994 年 Censor 和 Elfving 首次引入了分裂可行性问题的概念[3],但是其中的算法及后来的算法[4-5]在每次迭代中需要计算矩阵逆.2002 年,Byrne[6]提出了 CQ 算法,克服了在迭代中求矩阵逆这一缺点.之后,人们基于 CQ 算法又作了很多的改进[7-8],但大多都是停留在对算法格式的改进和收敛性的证明上.2005 年,Censor 和 Elfving 等又在文献[9]中把分裂可行性问题推广到了多重集合分裂可行性问题(multiple-sets split feasibility problem, MSSFP).除了 Censor 等[10-11]曾将相关问题引入到 IMRT 中,目前专门研究 CQ 算法的不同凸集定义方式及其对图像重建效果的影响还比较少.另外,由于解析方法不能很好地解决稀疏角度 CT 图像重建问题,因此 CQ 迭代算法的应用也成为必然.

在本文中,将稀疏角度的 CT 图像重建问题转化为分裂可行性问题,并将相关算法作用到 R^N 空间中进行求解,我们发现对非空闭凸集 C 和 Q 进行不同的定义,如将松弛的 CQ 算法[8]中 C_k 和 Q_k 关于半空间的定义引入到实际应用中,以及对约束权因子 α_i 和 β_j 的不同选取,重建结果是不同的.通过对不同算法及其不同的凸集定义方式所获得的重建结果进行比较,确定了在

* **收稿日期:** 2012-11-16; **修订日期:** 2013-04-12
基金项目: 国家自然科学基金资助项目(11071053)
作者简介: 王培元(1985—),男,河北献县人,博士生(通讯作者. E-mail: wangpy629@163.com).

不同非空闭凸集定义方式下各种算法的优劣,从而在不引入外部变量的前提下,提出了一种新的分块逐次混合算法,进而给出了其最佳实现方案.其重建结果表明,该算法收敛速度快,重建精度较高,而且具备较好的实时性.

1 问题及算法描述

1.1 分裂可行性问题

分裂可行性问题[3](split feasibility problem),即 SFP,是指对于 Hilbert 空间 H_1 和 H_2,C,Q 分别是 H_1 和 H_2 的非空闭凸子集.$\boldsymbol{A}:H_1 \to H_2$ 为有界线性算子.寻找满足下面条件的 $\boldsymbol{x}$:

$$\boldsymbol{x} \in C, \qquad \boldsymbol{Ax} \in Q. \tag{1}$$

将上述问题等价于一个不动点问题我们可以得到

$$\boldsymbol{x} = P_C(\boldsymbol{I} - \gamma \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{I} - P_Q)\boldsymbol{A})\boldsymbol{x} = T\boldsymbol{x}, \qquad \boldsymbol{x} \in C, \tag{2}$$

进而,Byrne 提出的 CQ 算法[6]为

算法 1 设 $\boldsymbol{x}_0$ 是任意的,且 $k = 0,1,\cdots$,计算

$$\boldsymbol{x}_{n+1} = P_C(\boldsymbol{x}_n - \gamma \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{I} - P_Q)\boldsymbol{A}\boldsymbol{x}_n), \qquad \boldsymbol{x} \in C, \tag{3}$$

其中,若 $C \subseteq R^N$,$Q \subseteq R^M$,$\boldsymbol{A}$ 为 $M \times N$ 实矩阵.$\gamma \in (0,2/L)$,L 为矩阵 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 的最大特征值,$\boldsymbol{I}$ 为单位矩阵,P_C,P_Q 分别为到 C 和 Q 的正交投影.下面我们不加证明地引入如下定理[6]:

定理 1.1 设 $\boldsymbol{A}(C) = \{\boldsymbol{Ax} \mid \boldsymbol{x} \in C\}$,如果 $\boldsymbol{A}(C) \cap Q \neq \varnothing$,当 $0 < \gamma < 2/\|\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\|$ 时,由算法 3 生成的序列 $\{\boldsymbol{x}_n\}$ 收敛到(SFP)的一个解.

此外,Byrne 还给出了分块迭代 CQ 算法[6](block-iterative CQ,BICQ).设 $\boldsymbol{A}_j$ 为一个 $M_j \times N$ 矩阵,并且 Q_j 为 R^{M_j} 的非空闭凸子集,其中 $j = 1,2,\cdots,J$.寻找 $\boldsymbol{x} \in R^N$,满足 $\boldsymbol{x} \in C$ 和 $\boldsymbol{A}_j\boldsymbol{x} \in Q_j$.

算法 2 设 $\boldsymbol{x}_0$ 是任意的,$n = 0,1,\cdots$,并且 $j(n) = n(\mathrm{mod}\ J) + 1$,计算

$$\boldsymbol{x}_{n+1} = P_C(\boldsymbol{x}_n - \gamma_{j(n)} \boldsymbol{A}_{j(n)}^{\mathrm{T}}(\boldsymbol{I} - P_{Q_{j(n)}})\boldsymbol{A}_{j(n)}\boldsymbol{x}_n), \tag{4}$$

其中,$\gamma_j \in (0,2/L_j)$,L_j 是 $\boldsymbol{A}_j^{\mathrm{T}}\boldsymbol{A}_j$ 的最大特征值.

上述算法的 C 和 Q 为全空间,在计算投影时比较困难,下面我们给出 C 和 Q 为半空间时的算法[8].

算法 3 设 $\boldsymbol{x}_0$ 是任意的,$n = 0,1,\cdots$,计算

$$\boldsymbol{x}_{n+1} = P_{C_n}(\boldsymbol{I} - \gamma \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{I} - P_{Q_n})\boldsymbol{A})\boldsymbol{x}_n, \qquad \boldsymbol{x} \in C, \tag{5}$$

其中,$\gamma \in (0,2/L)$,L 为矩阵 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 的最大特征值.对于凸集的定义是

$$C_n = \{\boldsymbol{x} \in R^N \mid c(\boldsymbol{x}_n) + \langle \xi^n, \boldsymbol{x} - \boldsymbol{x}_n \rangle \leqslant 0\},$$

$$Q_n = \{\boldsymbol{y} \in R^M \mid q(\boldsymbol{Ax}_n) + \langle \eta^n, \boldsymbol{y} - \boldsymbol{Ax}_n \rangle \leqslant 0\},$$

其中,ξ^n 和 η^n 分别是 $\partial c(\boldsymbol{x}_n)$ 和 $\partial q(\boldsymbol{Ax}_n)$ 中的元素.

在上面的算法中,P_C,P_Q 分别为到 C 和 Q 的正交投影.$P_C\boldsymbol{x}$ 在 $\boldsymbol{x}^* \in C$ 上,极小化 $\|\boldsymbol{x}^* - \boldsymbol{x}\|$.定理 1.1 中所谓的收敛,更一般的讲就是极小化 $\|P_Q\boldsymbol{Ax}^* - \boldsymbol{Ax}^*\|$.因此,定义一个目标函数

$$f(\boldsymbol{x}) = \frac{1}{2}\|P_Q\boldsymbol{Ax} - \boldsymbol{Ax}\|^2, \qquad \forall \boldsymbol{x} \in C, \tag{6}$$

SFP 也等价于 $f(\boldsymbol{x})$ 在 $\boldsymbol{x} \in C$ 上的最小值问题.

1.2 多重集合分裂可行性问题

从 BICQ 算法,两个集合的分裂可行性问题被拓展为多重集合分裂可行性问题[9](MSS-

FP)即寻找一个向量 $\tilde{\boldsymbol{x}}$ 满足：

$$\tilde{\boldsymbol{x}} \in C := \bigcap_{i=1}^{t} C_i,\ \boldsymbol{A}\tilde{\boldsymbol{x}} \in Q := \bigcap_{j=1}^{r} Q_j, \tag{7}$$

其中，$C_i \subseteq R^N, i = 1,2\cdots,n, Q_j \subseteq R^M$，$j = 1,2,\cdots,m$ 分别是 N 维，M 维欧氏空间的闭凸子集，$\boldsymbol{A}$ 是 $M \times N$ 的实矩阵。为了表达方便，我们另外定义一个闭凸集 $\boldsymbol{\Omega} \subseteq R^N$，则约束的多重集合分裂可行性问题是寻找 $\boldsymbol{x}^* \in \boldsymbol{\Omega}$ 解决问题(7)。

定义目标函数如下：

$$p(\boldsymbol{x}) := \frac{1}{2}\sum_{i=1}^{t}\alpha_i \| P_{C_i}(\boldsymbol{x}) - \boldsymbol{x} \|^2 + \frac{1}{2}\sum_{j=1}^{r}\beta_j \| P_{Q_j}(\boldsymbol{A}\boldsymbol{x}) - \boldsymbol{A}\boldsymbol{x} \|^2, \tag{8}$$

其梯度是

$$\nabla p(\boldsymbol{x}) = \sum_{i=1}^{t}\alpha_i(P_{C_i}(\boldsymbol{x}^k) - \boldsymbol{x}^k) + \sum_{j=1}^{r}\beta_j\boldsymbol{A}^{\mathrm{T}}(P_{Q_j}(\boldsymbol{A}\boldsymbol{x}^k) - \boldsymbol{A}\boldsymbol{x}^k), \tag{9}$$

其中，$\alpha_i > 0$ 和 $\beta_j > 0$，并且 $\sum_{i=1}^{t}\alpha_i + \sum_{j=1}^{r}\beta_j = 1$。为了寻找约束的分裂可行性问题的解，我们考虑极小化：

$$\min\{p(\boldsymbol{x}) \mid \boldsymbol{x} \in \boldsymbol{\Omega}\}。 \tag{10}$$

相应的算法是

算法 4 设 $\boldsymbol{x}_0$ 是任意的，且 $n = 0,1,\cdots$，计算

$$\boldsymbol{x}_{n+1} = P_{\Omega}\left\{\boldsymbol{x}_n + s\left(\sum_{i=1}^{t}\alpha_i(P_{C_i}(\boldsymbol{x}_n) - \boldsymbol{x}_n) + \sum_{j=1}^{r}\beta_j\boldsymbol{A}^{\mathrm{T}}(P_{Q_j}(\boldsymbol{A}\boldsymbol{x}_n) - \boldsymbol{A}\boldsymbol{x}_n)\right)\right\}, \tag{11}$$

其中，s 是一个正的标量，$0 < s < 2/L$，L 是梯度 $\nabla p(\boldsymbol{x})$ 的 Lipschitz 常数。我们不加证明地给出下面定理。

定理 1.2 如果 s 是一个正标量，$0 < s < 2/L$，L 是梯度 $\nabla p(\boldsymbol{x})$ 的 Lipschitz 常数，则算法 4 产生的任一序列 $\{\boldsymbol{x}_n\}_{n=0}^{\infty}$ 收敛到 $p(\boldsymbol{x})$ 的不动点，即约束的多重集合可行性问题的解。

1.3 分块逐次混合算法

在上述算法中，算法 2 和 3 的凸集同算法 4 中的一样，都是多重集合。但是在实际应用时，上述 4 种算法都需要在对非空闭凸集中的数据采集完全后，或完成对非空闭凸集的分块后才能进行运算。为了保持算法的稳定性和快速性，笔者在改善上述迭代算法结构的基础上，提出了一种实时性的分块逐次混合算法：

算法 5 初始化：$\forall \boldsymbol{x}_0 \in C$，$C \subseteq R^N$，$Q \subseteq R^M$，$n = 0$，$m > 1$。

步骤 1 $k = 1$，$i = 1$；

步骤 2 $\boldsymbol{x}_n^{\mathrm{I}} = P_C(\boldsymbol{I} - \gamma_i\boldsymbol{A}_i^{\mathrm{T}}(\boldsymbol{I} - P_{Q_i})\boldsymbol{A}_i)\boldsymbol{x}_n$，$Q_i \subseteq R^1$；

步骤 3 如果 $i = mk$，继续步骤 4；否则，$i = i + 1$，$\boldsymbol{x}_n = \boldsymbol{x}_n^{\mathrm{I}}$，返回步骤 2；

步骤 4 $\boldsymbol{x}_n^{\mathrm{II}} = P_{C_k}(\boldsymbol{I} - \gamma_k\boldsymbol{A}_k^{\mathrm{T}}(\boldsymbol{I} - P_{Q_k})\boldsymbol{A}_k)\boldsymbol{x}_n^{\mathrm{I}}$，$C_k \subseteq R^N$，$Q_k \subseteq R^m$；

步骤 5 如果 $k = M/m$，继续步骤 6；否则，$k = k + 1$，$i = i + 1$，$\boldsymbol{x}_n = \boldsymbol{x}_n^{\mathrm{II}}$，返回步骤 2；

步骤 6 $\boldsymbol{x}_{n+1} = P_C\left\{\boldsymbol{x}_n^{\mathrm{II}} + s\left(\sum_{k=1}^{M/m}\alpha_k(P_{C_k}(\boldsymbol{x}_n^{\mathrm{II}}) - \boldsymbol{x}_n^{\mathrm{II}}) + \sum_{k=1}^{M/m}\beta_k\boldsymbol{A}_k^{\mathrm{T}}(P_{Q_k}(\boldsymbol{A}_k\boldsymbol{x}_n^{\mathrm{II}}) - \boldsymbol{A}_k\boldsymbol{x}_n^{\mathrm{II}})\right)\right\}$；

步骤 7 如果满足停止准则，停止；否则，$n = n + 1$，返回步骤 1。

其中，$\gamma_i \in (0,2/L_i)$，$\gamma_k \in (0,2/L_k)$，L_i 和 L_k 分别为矩阵 $\boldsymbol{A}_i^{\mathrm{T}}\boldsymbol{A}_i$ 和 $\boldsymbol{A}_k^{\mathrm{T}}\boldsymbol{A}_k$ 的最大特征值，s 的定义同算法 4。

2　应用对象及问题转化

在医学、地震探测、遥感、断层扫描、航空航天等领域的许多实际问题,都可以归结为分裂可行性问题或多重集合分裂可行性问题.典型的应用实例是医学的强度可调放射疗法[9],本文以 CT 断层扫描图像重建为例,将相关图像重建问题转化为集合的分裂可行性问题进行研究.

2.1　稀疏角度 CT 重建问题模型

本文采用稀疏角度的 CT 图像重建模型为上述算法的应用对象.为了建立分裂的凸集 C 和 Q,忽略射线的折射及衍射效应,首先把图像离散化,把图像用一个个像素拼摆而成,如图 1 所示.像素 $x_j(j=1,2,\cdots)$ 按照从左到右,从上到下的顺序拼接成一个向量 $\boldsymbol{x}$,显然向量 $\boldsymbol{x}$ 很大.其数学模型为[12]

$$\boldsymbol{Ax}=\boldsymbol{b}, \tag{12}$$

式中,$\boldsymbol{x}=(x_1,x_2,\cdots,x_N)^{\mathrm{T}}$ 是图像离散化后的向量形式,$\boldsymbol{b}=(b_1,b_2,\cdots,b_M)^{\mathrm{T}}$ 为射线的投影数据.$\boldsymbol{A}_{M\times N}$ 是方程组的系数矩阵,矩阵 $\boldsymbol{A}$ 的元素 a_{ij} 代表第 j 个像素对第 i 个投影值的贡献,也就是待重建物体的各点吸收系数.在仿真中,这个系数可以近似取投影射线在该像素内所截取线段的长度.另外,我们采用扇形束投影,扫描模型见图 2.定义扇形束的中心距离为 d,每个角度射线的通道数为 m,像素的总数为 N,2π 角度内投影次数为 K,全部通道数 $M=m\times K$.

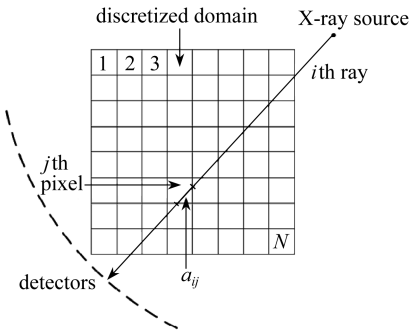

图 1　图像重建的离散模型

Fig. 1　Image reconstruction in the discredited model

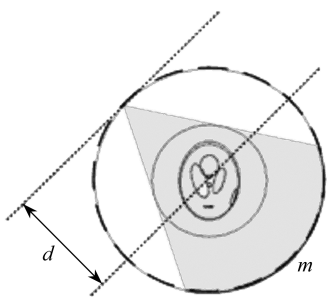

图 2　扇形束稀疏角度扫描模型

Fig. 2　The fan beam CT imaging geometry

2.2　问题转化

利用第 1 节的算法来解决 CT 图像重建问题,一般所采用的思路是先将重建问题转化成分裂可行性问题,然后再用上述算法求解.求解 SFP 等价于寻找属于 Q 和 $\boldsymbol{A}(C)=\{\boldsymbol{Ac}\mid \boldsymbol{c}\in C\}$ 的交集,或 $\boldsymbol{A}^{-1}(Q)$ 和 C 的交集的元素.

本文中,设 $\boldsymbol{x}=(x_1,x_2,\cdots,x_N)\in R^N$ 为待重建图像.$i=1,2,\cdots,M$ 表示在所有投影射线数中的第 i 条射线,并且 $M=m\times K$,是所有的投影数,则 b_i 是所有投影数据中的第 i 个数据.$k=1,2,\cdots,K$ 表示第 k 次的投影,则 $\boldsymbol{b}^{(k)}$ 是一个 $m\times 1$ 向量,为一个投影角度下的投影数据,此时 $i=(k-1)m+1,(k-1)m+2,\cdots,km$,为射线次序的取值范围.$b_i^{(k)}\in R^m$ 表示第 k 次投影下的第 i 条射线的投影数据.

于是我们可以将 CT 图像重建问题转化成分裂可行性问题表述如下:

定义 $\boldsymbol{A}=[a_{ij}]_{\substack{1\le i\le M\\ 1\le j\le N}}$ 为非零稀疏 $M\times N$ 阶矩阵,包含 K 个角度所有射线经过每个像素的线段长度值;

定义 $C=\{\boldsymbol{x}\in R^N\mid 0\leqslant x_i\leqslant 1,\ 1\leqslant i\leqslant N\}$，为吸收系数空间，或图像向量空间；

定义 $Q=\{\boldsymbol{b}\}$，为投影数据空间.

多重集合分裂可行性问题的凸集定义如下：

定义 $C_k=\{\boldsymbol{x}\in R^N\mid \boldsymbol{A}^{(k)}\boldsymbol{x}=\boldsymbol{b}^{(k)},\ k\geqslant 1\}$，$C:=\cap_{k=1}C_k$ 为图像向量空间；

定义 $Q_k=\{\boldsymbol{b}^{(k)},\ k\geqslant 1\}$，$Q:=\cap_{k=1}Q_k$ 为投影数据空间.

若考虑噪声，则 $\boldsymbol{Ax}\in R(\boldsymbol{A})\oplus R(\boldsymbol{A})^{\perp}$，当 $P_{\overline{\boldsymbol{x}(C)}}\boldsymbol{Ax}\in \boldsymbol{A}(C)$ 时，$\boldsymbol{Ax}\in Q$，式(6)或(8)有解. 重建问题即扫描后通过系数矩阵 $\boldsymbol{A}$ 的变换，从投影数据空间 Q，寻找满足图像向量空间 C 的图像 $\boldsymbol{x}$.

3 算法的实现方案

本节分别给出算法 1～4 的几种不同的实现方案，目的是在第 4 节通过比较各种方案的试验结果以确定算法 5 的最佳实现方案.

3.1 步长及投影的计算

首先，我们不加证明地给出 L 的计算方法[6]：

定理 3.1 设 $\boldsymbol{A}$ 为 $M\times N$ 矩阵，对 $i=1,2,\cdots,m$，计算 $\boldsymbol{v}_i=\sum_{j=1}^{N}a_{ij}^2>0$，对 $j=1,2,\cdots,N$，如果 $a_{ij}\neq 0$，计算 $\sigma_j=\sum_{i=1}^{M}v_i$，$\sigma=\max(\sigma_j)$，则矩阵 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 的最大特征值不超过 $L=\sigma$.

其次针对多重集合分裂可行性问题，为计算 s，我们不加证明地给出 L 的计算方法：

定理 3.2[6] 设 $\boldsymbol{A}$ 为 $M\times N$ 矩阵，对 $i=1,2,\cdots,M$，计算 $\boldsymbol{v}_i=\sum_{j=1}^{N}a_{ij}^2>0$，对 $j=1,2,\cdots,n$，如果 $a_{ij}\neq 0$，计算 $\sigma_j=\sum_{i=1}^{M}v_i$，$\sigma=\max(\sigma_j)$，则矩阵 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 的谱半径不超过 σ.

定理 3.3[9] 设 $\boldsymbol{A}$ 为 $M\times N$ 矩阵，目标函数(8)的梯度是 Lipschitz 连续的，并且

$$L=\sum_{i=1}^{t}\alpha_i+\rho(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})\sum_{j=1}^{r}\beta_j \tag{13}$$

是它的 Lipschitz 常数，其中 $\rho(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})$ 是 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 的谱半径.

其次，假设算法中到闭凸集上的正交投影是容易计算的. 到集合 C 或 Q 的投影，采用下面方法实现. 当 $\Pi=\{\boldsymbol{y}\mid\langle\boldsymbol{\alpha},\boldsymbol{y}\rangle=\boldsymbol{\beta}\}$ 在 R^N 或 R^M 中时，有

$$P_{\Pi}\boldsymbol{y}=\begin{cases}\boldsymbol{y}, & \boldsymbol{y}\in\Pi,\\ \boldsymbol{y}-\dfrac{\langle\boldsymbol{\alpha},\boldsymbol{y}\rangle-\boldsymbol{\beta}}{\|\boldsymbol{\alpha}\|^2}\boldsymbol{\alpha}, & \boldsymbol{y}\notin\Pi.\end{cases} \tag{14}$$

3.2 算法的实现

首先，对算法 1～4 进行分析. 我们发现在应用其解决 CT 图像重建问题的时候，对非空闭凸集 C 和 Q 的不同的定义，会产生不同的试验结果，因此为了寻求最佳的应用方案，进而给出算法 5 的最佳实现方案，有必要对 C 和 Q 可能所代表的不同的物理意义进行分析. 在实现方案中，我们没有采用直接极小化目标函数(6)和(8)的方法，因为若要使其小于一个很小的正数 ε，哪怕是取 10^{-1} 级也会用一万多步的迭代，因此我们对上述算法采用相同的收敛准则，即相邻两次迭代的相对误差形式，分别为

$$\frac{f(\boldsymbol{x}_{n+1})-f(\boldsymbol{x}_n)}{f(\boldsymbol{x}_n)}<\varepsilon \text{ 和 } \frac{p(\boldsymbol{x}_{n+1})-p(\boldsymbol{x}_n)}{p(\boldsymbol{x}_n)}<\varepsilon.$$

3.2.1 算法1的实现方案

方案1 设 $\boldsymbol{x}=(x_1,x_2,\cdots,x_N)\in R^N$ 为图像向量,$\boldsymbol{b}=(b_1,b_2,\cdots,b_M)^{\mathrm{T}}\in R^M$ 为所有投影数据,b_i 为每条射线的投影数据. $\boldsymbol{A}=(a_{ij})_{(m\times K)\times N}$ 为投影系数矩阵,则 $C=\{\boldsymbol{x}\in R^N\mid \boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}\}$ 为所有图像向量的集合,其中 $\boldsymbol{A}\boldsymbol{x}$ 表示在所有角度下的射线对图像的投影,$Q=\{\boldsymbol{b}\mid \boldsymbol{b}\in R^M\}$ 为所有投影数据集合. 执行关键步骤如下:

步骤1 利用定理3.1计算 γ;

步骤2 $1\leqslant j\leqslant N$,利用式(14),进行第 $n+1$ 次迭代

$$x_j^{(n+1)}=P_C\left(x_j^{(n)}+\gamma a_{ij}\left(b_i-\sum_{l=1}^{N}a_{il}x_l^{(n)}\right)\right);$$

步骤3 $f(\boldsymbol{x}^{(n+1)})=0.5\parallel\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{(n+1)}\parallel^2$.

3.2.2 算法2的实现方案

首先是按每条射线进行分块:

方案2 设 $\boldsymbol{x}=(x_1,x_2,\cdots,x_N)\in R^N$ 为图像向量,$b_i\in R^1$ 为每条射线的投影数据. 当 $i=1,2,\cdots,M$ 时,$\boldsymbol{A}=[\boldsymbol{A}_{(1)},\boldsymbol{A}_{(2)},\cdots,\boldsymbol{A}_{(M)}]^{\mathrm{T}}$,$\boldsymbol{b}=(b_1,b_2,\cdots,b_M)^{\mathrm{T}}\in R^M$,其中 $\boldsymbol{A}_{(i)}=(a_{ij})_{1\times N}$ 为每一射线投影的加权矩阵. $C=\{\boldsymbol{x}\in R^N\mid \boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}\}$ 为所有图像向量的集合,其中 $\boldsymbol{A}\boldsymbol{x}$ 表示在所有角度下的射线对图像的投影,$Q_i=\{b_i\mid b_i\in R^1\}$ 为每一射线投影数据集合. 执行关键步骤如下:

步骤1 利用定理3.1计算每条射线下的 γ_i;

步骤2 $1\leqslant j\leqslant N$,利用式(14),对于第 n 次迭代的第 $i+1$ 条射线有

$$x_{j_{(i+1)}}^{(n)}=P_C\left(x_{j_{(i)}}^{(n)}+\gamma_i(a_{ij})\left(b_i-\sum_{l=1}^{N}a_{il}x_{l_{(i)}}^{(n)}\right)\right);$$

步骤3 $f(\boldsymbol{x}^{(n)})=0.5\parallel\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{(n)}\parallel^2$.

其次是按投影角度进行分块:

方案3 设 $\boldsymbol{x}=(x_1,x_2,\cdots,x_N)\in R^N$ 为图像向量,$b^{(k)}\in R^m$ 为每一角度投影数据,$k=1,2,\cdots,K$. $\boldsymbol{A}=[\boldsymbol{A}^{(1)},\boldsymbol{A}^{(2)},\cdots,\boldsymbol{A}^{(K)}]^{\mathrm{T}}$,$\boldsymbol{b}=[\boldsymbol{b}^{(1)},\boldsymbol{b}^{(2)},\cdots,\boldsymbol{b}^{(K)}]^{\mathrm{T}}\in R^M$,其中 $\boldsymbol{A}^{(k)}=(a_{ij}^{(k)})_{m\times N}$ 为每一角度投影的加权矩阵. $C=\{\boldsymbol{x}\in R^N\mid \boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}\}$ 为所有图像向量集合,其中 $\boldsymbol{A}\boldsymbol{x}$ 表示在所有角度下的射线对图像的投影,$Q_k=\{\boldsymbol{b}^{(k)}\mid \boldsymbol{b}^{(k)}\in R^m\}$ 为每一角度投影数据集合. $b_i^{(k)}$ 为第 k 个角度下的第 i 条射线的投影数据. 执行关键步骤如下:

步骤1 利用定理3.1计算每个角度下的 γ_k;

步骤2 $1\leqslant j\leqslant N$,利用式(14),对于第 n 次迭代的第 $k+1$ 个角度有

$$x_{j_{(k+1)}}^{(n)}=P_C\left(x_{j_{(k)}}^{(n)}+\gamma_k(a_{ij}^{(k)})\left(b_i^{(k)}-\sum_{l=1}^{N}a_{il}x_{l_{(k)}}^{(n)}\right)\right);$$

步骤3 $f(\boldsymbol{x}^{(n)})=0.5\parallel\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{(n)}\parallel^2$.

3.2.3 算法3的实现方案

方案4 设 $\boldsymbol{x}=(x_1,x_2,\cdots,x_N)\in R^N$ 为图像向量,$b_i\in R^1$ 为每条射线的投影数据. 当 $i=1,2,\cdots,M$ 时,$\boldsymbol{A}=[\boldsymbol{A}_{(1)},\boldsymbol{A}_{(2)},\cdots,\boldsymbol{A}_{(M)}]^{\mathrm{T}}$,$\boldsymbol{b}=(b_1,b_2,\cdots,b_M)^{\mathrm{T}}\in R^M$,其中 $\boldsymbol{A}_{(i)}=(a_{ij})_{1\times N}$ 为每一射线投影的加权矩阵. $C_i=\{\boldsymbol{x}\in R^N\mid\langle\boldsymbol{x},\boldsymbol{A}_{(i)}\rangle=b_i\}$ 为每一射线图像向量集合,其中 $\langle\boldsymbol{x},\boldsymbol{A}_{(i)}\rangle$ 表示每条射线对图像的投影,$Q_i=\{b_i\mid b_i\in R^1\}$ 为每一射线投影数据集合. 执行关键步骤如下:

步骤1 利用定理3.1计算每条射线下的 γ_i;

步骤 2　$1 \leqslant j \leqslant N$,利用式(14),对于第 n 次迭代的第 $i+1$ 条射线有

$$x_{j_{(i+1)}}^{(n)} = P_{C_i}\left(x_{j_{(i)}}^{(n)} + \gamma_i(a_{ij})\left(b_i - \sum_{l=1}^{N} a_{il}x_{l_{(i)}}^{(n)}\right)\right);$$

步骤 3　$f(\boldsymbol{x}^{(n)}) = 0.5 \| \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(n)} \|^2$.

方案 5　设 $\boldsymbol{x} = (x_1, x_2, \cdots, x_N) \in R^N$ 为图像向量,$b^{(k)} \in R^m$ 为每一角度投影数据,$k = 1, 2, \cdots, K$. $\boldsymbol{A} = [\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \cdots, \boldsymbol{A}^{(K)}]^{\mathrm{T}}$,$\boldsymbol{b} = [\boldsymbol{b}^{(1)}, \boldsymbol{b}^{(2)}, \cdots, \boldsymbol{b}^{(K)}]^{\mathrm{T}} \in R^M$,其中 $\boldsymbol{A}^{(k)} = (a_{ij}^{(k)})_{m \times N}$ 为每一角度投影的加权矩阵. $C_k = \{\boldsymbol{x} \in R^N \mid \boldsymbol{A}^{(k)}\boldsymbol{x} = \boldsymbol{b}^{(k)}\}$ 为每一角度图像向量集合,其中 $\boldsymbol{A}^{(k)}\boldsymbol{x}$ 表示每个角度对图像的投影,$Q_k = \{\boldsymbol{b}^{(k)} \mid \boldsymbol{b}^{(k)} \in R^m\}$ 为每一角度投影数据集合. $b_i^{(k)}$ 为第 k 个角度下的第 i 条射线的投影数据. 执行关键步骤如下:

步骤 1　利用定理 3.1 计算每个角度下的 γ_k;

步骤 2　$1 \leqslant j \leqslant N$,利用式(14),对于第 n 次迭代的第 $k+1$ 个角度有

$$x_{j_{(k+1)}}^{(n)} = P_{C_k}\left(x_{j_{(k)}}^{(n)} + \gamma_k(a_{ij}^{(k)})\left(b_i^{(k)} - \sum_{l=1}^{N} a_{il}x_{l_{(k)}}^{(n)}\right)\right);$$

步骤 3　$f(\boldsymbol{x}^{(n)}) = 0.5 \| \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(n)} \|^2$.

3.2.4　算法 4 的实现方案

方案 6　凸集定义同方案 4. 执行关键步骤如下:

步骤 1　任取 $\sum_{i=1}^{t}\alpha_i = \mu$, $\sum_{j=1}^{r}\beta_j = \tau$;

步骤 2　利用定理 3.1 和 3.2 计算 $s = 2/(\mu + \tau\sigma)$;

步骤 3　对于第 n 次迭代的第 $i+1$ 条射线,将 C_i 看做一个超平面,利用式(14)计算投影 $x_{j_{(i+1)}}^{(n)} = P_{C_i}(x_{j_{(i)}}^{(n)})$;

步骤 4　计算第 $n+1$ 次迭代

$$x_j^{(n+1)} = x_j^{(n)} + s\left(\mu\sum_{i=1}^{M}(x_{j_{(i+1)}}^{(n)} - x_{j_{(i)}}^{(n)}) + \tau\sum_{i=1}^{M}\left((a_{ij})\left(b_i - \sum_{l=1}^{N} a_{il}x_{l_{(i)}}^{(n)}\right)\right)\right);$$

步骤 5　计算范数 $N_1 = \mathrm{norm}(x_{j_{(i+1)}}^{(n)} - x_{j_{(i)}}^{(n)})$ 和 $N_2 = \mathrm{norm}(b_i - \sum_{l=1}^{N} a_{il}x_{l_{(i)}}^{(n)})$;

步骤 6　$p(x^{(n+1)}) = 0.5\mu\sum_{i=1}^{M} N_1^2 + 0.5\tau\sum_{i=1}^{M} N_2^2$.

方案 7　凸集定义同方案 5. 执行关键步骤如下:

步骤 1　任取 $\sum_{i=1}^{t}\alpha_i = \mu$, $\sum_{j=1}^{r}\beta_j = \tau$;

步骤 2　利用定理 3.1 和 3.2 计算 $s = 2/(\mu + \tau\sigma)$;

步骤 3　对于第 n 次迭代的第 k 个角度,将 C_k 看做 m 个超平面的交,利用式(14)计算投影 $x_{j_{(k+1)}}^{(n)} = P_{C_k}(x_{j_{(k)}}^{(n)})$;

步骤 4　计算第 $n+1$ 次迭代

$$x_j^{(n+1)} = x_j^{(n)} + s\left(\mu\sum_{k=1}^{K}(x_{j_{(k+1)}}^{(n)} - x_{j_{(k)}}^{(n)}) + \tau\sum_{k=1}^{K}\sum_{i=1}^{m}\left((a_{ij})\left(b_i - \sum_{l=1}^{N} a_{il}x_{l_{(k)}}^{(n)}\right)\right)\right);$$

步骤 5　计算范数 $N_1 = \mathrm{norm}(x_{j_{(k+1)}}^{(n)} - x_{j_{(k)}}^{(n)})$ 和 $N_2 = \mathrm{norm}(b_i - \sum_{l=1}^{N} a_{il}x_{l_{(k)}}^{(n)})$;

步骤 6　$p(x^{(n+1)}) = 0.5\mu\sum_{k=1}^{K} N_1^2 + 0.5\tau\sum_{k=1}^{K}\sum_{i=1}^{m} N_2^2$.

4　验证结果及方案确定

4.1　方案的验证与对比分析

上述方案采用 Matlab R2011b (Windows)进行仿真. 程序运行的 PC 主机配置是:1.98 G 内存,英特尔奔腾 G630 双核处理器,频率为 2.69 GHz. 我们选择 Shepp-Logan 模型(见图 3(a))为试验对象,其像素值为 0 ~ 1,图像大小为 64×64. 在仿真过程中,设定已知数据 $K = 36$, $m = 64\sqrt{2} \approx 95$, $d = 60$, $N = 64 \times 64$, $\mu = 0.6$, $\tau = 0.4$, $\varepsilon = 0.2\%$. 初始数据 $\boldsymbol{x}_0 = [0.2]_{1\times N}$ 为先验值. 方案 1 ~ 7 的重建结果分别如图 3(b) ~ (h)所示. 图中(e)与(f)即算法 3 的重建结果较清晰.

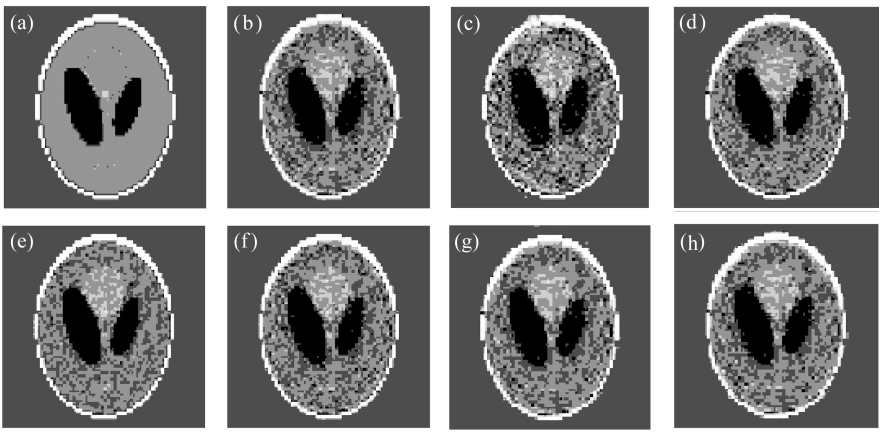

图 3　Shepp-Logan 模型及相同收敛条件下的重建结果

Fig. 3　Shepp-Logan model and the reconstructions on the same convergence condition

表 1　相同收敛条件下的迭代次数及误差评价

Table 1　Iterations and errors on under the same convergence condition

	case1	case 2	case 3	case 4	case 5	case 6	case 7
$n_{\text{iterations}}$	811	89	787	725	835	816	812
e_{MSE}	0.004 4	0.004 2	0.003 4	0.000 721 5	0.002 4	0.004 3	0.004 4
P_{PSNR}	71.715 5	71.921 2	72.862 2	79.548 4	74.246 9	71.804 6	71.711 9

每种方案的迭代次数、均方误差(e_{MSE})和峰值信噪比(P_{PSNR})如表 1 所示. 我们可以看出在相同的收敛条件下,方案 2 和方案 4 对每条射线的松弛因子 γ_i 都是不同的,因此方案 2 可以使目标函数的变化率得到较快的下降,收敛速度最快,但其重建精度最差. 方案 4 计算了向 C 按行分块子集上的投影,虽然迭代次数增多,重建精度最好. 方案 3 和方案 5 的松弛因子只是随投影角度的不同而改变,需要更多的迭代次数,由于方案 5 计算了向 C 按角度分块子集上的投影,重建精度稍好于方案 3. 方案 1、方案 6 和 7 的松弛因子保持不变,重建结果相差不大.

下面,针对各种不同的分块方案,进一步展开分析:

首先,分析面向按行分块的凸集的方案,即算法 2 的方案 2 和算法 3 的方案 4. 图 4 为两种方案在 200 次迭代内的均方误差曲线. 很明显方案 4 的精度高,但是其收敛速度较慢,在 200 次迭代后,曲线仍保持下降的趋势,而方案 2 大概在迭代 60 次以后曲线基本保持水平. 因此,在按行分块算法中,算法 2 能够取得较快的收敛速度.

其次,分析面向按角度分块的凸集的方案,即算法 2 的方案 3 和算法 3 的方案 5. 图 5 为两

种方案在 200 次迭代内的均方误差曲线.两种方案的收敛速度都比较慢,200 次迭代后曲线仍呈下降趋势.在迭代 10 次以后,方案 5 明显比方案 3 能取得更小的误差.因此,在按角度分块的算法中,算法 3 能取得较高的重建精度.

再次,分析面向全体凸集的方案,即算法 1 的方案 1,算法 4 的方案 6 和 7.图 6 为 3 种方案在 200 次迭代内的均方误差曲线.3 种方案在 200 次迭代内均未有收敛的趋势,其中算法 4 的方案 6 和 7 明显要优于算法 1.对于方案 6 和 7,图 7 给出了二者重建结果的第 51 条横截线与原图像的对比.在相同的迭代次数和约束权因子不变的情况下,方案 6 和 7 无论是收敛速度还是重建精度都相差不大,方案 6 略优于方案 7.但在迭代次数继续加大的情况下,两种方案的结果几乎是一致的.与分裂可行性问题的 CQ 算法相比,多重分裂可行性问题的算法在实现方案的选取上与给定不同的非空闭凸集 C_i 和 Q_j 关系不大.

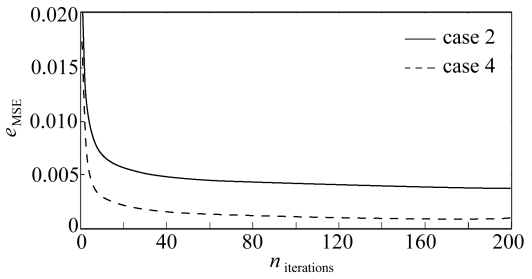

图 4 方案 2 和 4 的均方误差

Fig. 4 The MSE of cases 2 and 4

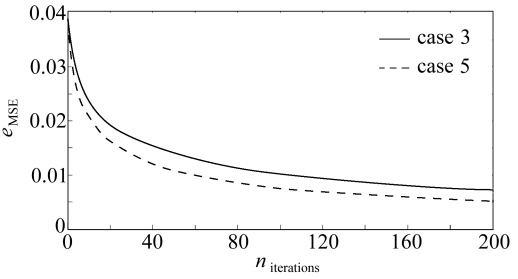

图 5 方案 3 和 5 的均方误差

Fig. 5 The MSE of cases 3 and 5

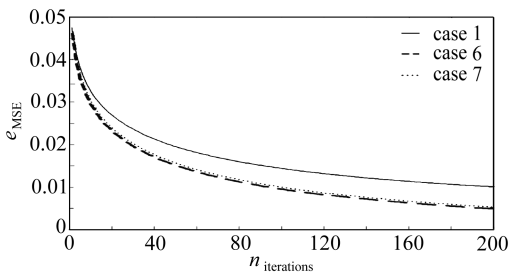

图 6 方案 1、方案 6 和 7 的均方误差

Fig. 6 The MSE of cases 1,6 and 7

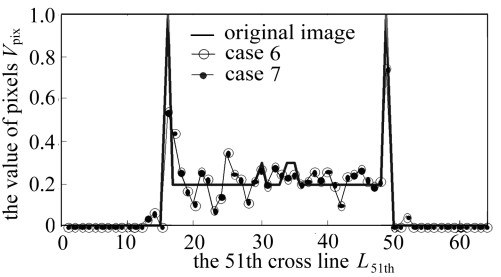

图 7 方案 6 和方案 7 的第 51 条横截线对比图

Fig. 7 The 51th cross lines of cases 6 and 7

最后,讨论方案 6 和 7 中约束权因子改变的情况.取 τ 从 0.01 到 0.9,采用方案 7 的凸集定义方式,迭代次数取 200 次,其 e_{MSE} 如图 8 所示.图中右下角曲面最低,即 e_{MSE} 的值最小.因此,可以得出当 $\tau = 0.01$ 的情况 e_{MSE} 最小.

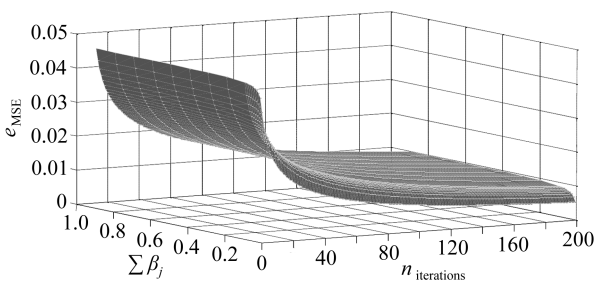

图 8 约束因子可变时的均方误差

Fig. 8 Mean square errors while constraint weights are mutative

从上述结果中我们发现,算法 4 的后半部分对于投影数据取偏差来修正图像所取得的效果,远比对图像数据取偏差带来的效果的权重要小得多.因此,在选取约束权因子时, $\sum_{j=1}^{r}\beta_j$ 应尽量小于 $\sum_{i=1}^{t}\alpha_i$.

通过对上述各种算法和方案的对比分析,得到的结论是:在不改变算法原有结构的情况下,按行分块方案中算法 2 收敛速度最快,按角度分块算法中算法 3 能够取得最好的重建精度,面向所有凸集的方案中算法 4 能取得最好的重建精度.

4.2 分块逐次混合算法实现方案及结果

应用 4.1 节的结论,并采用与 3.2 节相同的收敛准则,可以得出算法 5 的最佳实现方案:

方案 8 凸集定义同方案 2 和方案 5,执行关键步骤如下:

步骤 1 $k=1$, $i=1$;

步骤 2 利用定理 3.1 计算第 i 条射线下的 γ_i;

步骤 3 $1\leqslant j\leqslant N$,利用式(14) 对于第 n 次迭代的第 $i+1$ 条射线有

$$x_{j_{(i+1)}}^{(n)}=P_C\left(x_{j_{(i)}}^{(n)}+\gamma_i(a_{ij})\left(b_i-\sum_{l=1}^{N}a_{il}x_{l_{(i)}}^{(n)}\right)\right);$$

步骤 4 如果 $i=m\times k$,利用定理 3.1 计算第 k 个角度下的 γ_k 并继续,否则,$i=i+1$ 返回步骤 2;

步骤 5 $1\leqslant j\leqslant N$,利用式(14),对于第 n 次迭代的第 $k+1$ 个角度有

$$x_{j_{(k+1)}}^{(n)}=P_{C_k}\left(x_{j_{(k)}}^{(n)}+\gamma_k(a_{ij}^{(k)})\left(b_i^{(k)}-\sum_{l=1}^{N}a_{il}x_{l_{(k)}}^{(n)}\right)\right);$$

步骤 6 如果 $i=M$,取 $\mu=0.99$,$\tau=0.01$,利用定理 3.1 和 3.2 计算 $s=2/(\tau\sigma)$ 并继续,否则,$k=k+1$, $i=i+1$, 返回步骤 2;

步骤 7 对于第 n 次迭代的第 k 个角度,将 C_k 看做 m 个超平面的交,利用式(14) 计算投影 $x_{j_{(k+1)}}^{(n)}=P_{C_k}(x_{j_{(k)}}^{(n)})$;

步骤 8 计算第 $n+1$ 次迭代

$$x_j^{(n+1)}=x_j^{(n)}+s\left(\mu\sum_{k=1}^{K}(x_{j_{(k+1)}}^{(n)}-x_{j_{(k)}}^{(n)})+\tau\sum_{k=1}^{K}\sum_{i=1}^{m}\left((a_{ij})\left(b_i-\sum_{l=1}^{N}a_{il}x_{l_{(k)}}^{(n)}\right)\right)\right);$$

步骤 9 计算 $N_1=\mathrm{norm}(x_{j_{(k+1)}}^{(n)}-x_{j_{(k)}}^{(n)})$ 和 $N_2=\mathrm{norm}(b_i-\sum_{l=1}^{N}a_{il}x_{l_{(k)}}^{(n)})$,

$$p(x^{(n+1)})=0.5\mu\sum_{k=1}^{K}N_1^2+0.5\tau\sum_{k=1}^{K}\sum_{i=1}^{m}N_2^2.$$

该方案在本文中的物理意义是:在完成一条射线的扫描后立即进行重建,完成一个角度的扫描后,又进行一次重建,完成所有扫描后对所有数据再进行重建.因此,在执行时可以一边扫描,一边重建,为一种实时性算法的实现方案.

方案 8 的重建结果见图 9,与图 3 中重建精度最高的方案 4 的结果相比较,具备较高的分辨率.其与方案 4 第 51 条横截线对比见图 10,方案 8 的灰度值与原图像较契合.

表 2 3 种方案的迭代次数及误差评价

Table 2 Iterations and errors of the three cases

	case 2	case 4	case 8
$n_{\text{iterations}}$	89	725	32
e_{MSE}	0.004 2	0.000 721 5	0.000 350 08
P_{PSNR}	71.921 2	79.548 4	82.689 2

图 9 分块逐次混合算法重建结果

Fig. 9 The reconstruction of block successive mixed algorithm

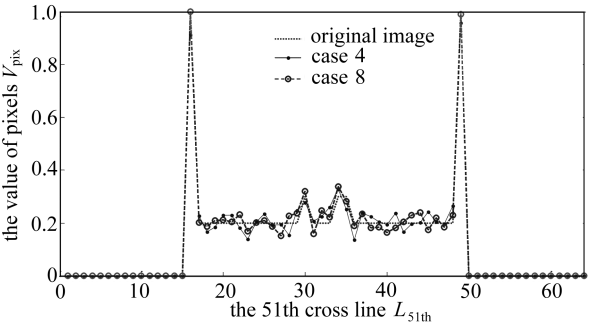

图 10 方案 4 和方案 8 的第 51 条横截线对比图

Fig. 10 The 51th cross lines of cases 4 and 8

表 2 给出了方案 8 与方案 2、4 的迭代次数和误差评价,相同条件下,方案 8 能够取得更少的迭代次数和更高的重建精度,具备明显的优势.需要注意的是,虽然方案 8 的迭代次数较少,但是在处理大型数据时,若使用串行的计算方法未必能取得较好的时间效率,尤其是在处理图像重建的问题时.例如本文中的实例,仅在不完全采样数据的情况下,系数矩阵 $\boldsymbol{A}$ 的维数就高达 3 420×4 096,若使用串行计算方法实现是非常耗时的.因此可以使用多线程访问共享内存的方式进行并行运算,这样可以很好地提高算法的运算效率和运算速度.

5 结 论

本文针对不同集合的分裂可行性问题,探讨了其相关算法在解决稀疏角度 CT 图像投影重建时的实现方案,并在此基础上提出了一种新的算法.文中通过试验表明,对于按行分块的方案,采用算法 2 可以取得较好的时间效率,对于按角度分块的方案,算法 3 可以取得较好的重建精度;而对于面向全部凸集的算法,算法 4 可以取得较好的重建精度.因此,我们通过综合算法 2、3 和 4,及对应的快速收敛,精度较高的重建方案,提出了算法 5 和方案 8.试验表明实时性算法 5 能够取得比其他算法和方案更好的结果.另外,我们还发现算法 4 对于不同凸集的定义并不敏感,只是在约束权因子的选取上能对算法的效果有所影响.因此,与分裂可行性问题的算法的实现相比,多重集合分裂可行性问题的算法要稳定得多.我们以本文 CQ 算法的实现方案为基础,继续结合其改进类算法,更有效地解决 CT 不完全投影数据图像重建问题,是下一步研究的方向.

致谢 作者对审稿人为改进本文所提出的的宝贵意见表示衷心感谢.

参考文献(References):

[1] Bauschke H H, Borwein J M. On projection algorithms for solving convex feasibility problems [J]. *SIAM Rev*,1996,**38**(3): 367-426.

[2] Byrne C L. A unified treatment of some iterative algorithms in signal processing and image reconstruction[J]. *Inverse Problems*, 2004 ,**20**(1): 103-120.

[3] Censor Y, Elfving T. A multiprojection algorithm using Bregman projections in a product space[J]. *J Numer Math*, 1994, **8**(2): 221-239.

[4] Byrne C L. Iterative projection onto convex sets using multiple Bregman distances[J]. *Inverse Problems*, 1999, **15**(5): 1295-1313.

[5] Byrne C L. Bregman-Legendre multidistance projection algorithms for convex feasibility and

optimization[C]//*Inherently Parallel Algorithms in Feasibility and Optimization and Their Applications*. Amsterdam, The Netherlands: Elsevier Science Publishers, 2001: 87-99.

[6] Byrne C L. Iterative oblique projection onto convex sets and the split feasibility problem [J]. *Inverse Problems*,2002, **18**(2): 441-453.

[7] Fukushima M. A relaxed projection method for variational inequalities[J]. *Math Program*, 1986 ,**35**(1): 58-70.

[8] Yang Q. The relaxed CQ algorithm solving the split feasibility problem[J]. *Inverse Problems*, 2004, **20**(4): 1261-1266.

[9] Censor Y, Elfving T, Kopf N, Bortfeld T. The multiple-sets split feasibility problem and its applications for inverse problems[J]. *Inverse Problems*, 2005, **21**(6): 2071-2084.

[10] Censor Y, Bortfeld T, Martin B, Trofimov A. A unified approach for inversion problems in intensity-modulated radiation therapy[J]. *Phys Med Biol*, 2006, **51**(10): 2353-2365.

[11] López G, Martín-Márquez V, WANG Feng-hui, XU Hong-kun. Solving the split feasibility problem without prior knowledge of matrix norms[J]. *Inverse Problems*, 2012, **28**(8): 1-18.

[12] Zeng G S. *Medical Image Reconstruction: a Conceptual Tutorial*[M]. Beijing: Higher Education Press; Berlin, Heidelberg: Springer-Verlag, 2010.

Apply the Multiple-Sets Split Feasibility Problem to CT Image Reconstruction

WANG Pei-yuan, ZHOU Hai-yun

(*Department of Mathematics*, *Ordnance Engineering College*, *Shijiazhuang* 050003, *P. R. China*)

Abstract: To apply the CQ algorithm on the sparse angular CT image reconstruction better, a new real-time block successive mixed algorithm was proposed. Firstly, the problem of image reconstruction was transformed into the split feasibility problem. Secondly, through analyzing the different defineitions of nonempty closed convex sets C and Q, 7 different implementation cases in N dimension real space were proposed. Through simulations the convergence rate and reconstruction precision to different cases were analyzed, and how to select the constraint weights in algorithm and the output was studied. Then it obtains the best cases of CQ algorithm applyied on sparse angular CT image reconstruction were obtained. Therefore, the best case of proposed algorithm is obtained. The results show that the proposed algorithm has faster convergence rate and better reconstruction precision. New ideas for applying the split feasibility problem and its extending norms to the CT incomplete projection data image reconstruction were proposed.

Key words: CQ algorithm; multiple-sets split feasibility problem; nonempty convex set; image reconstruction