Pure Mathematics 理论数学, 2025, 15(4), 298-308
Published Online April 2025 in Hans. https://www.hanspub.org/journal/pm
https://doi.org/10.12677/pm.2025.154133

Hans汉斯

带有保护条件的ν加速ADMM算法

张海林

重庆交通大学数学与统计学院，重庆

收稿日期：2025年2月27日；录用日期：2025年4月11日；发布日期：2025年4月23日

摘　要

本文提出了一种结合L_1-L_2正则化和ν加速的交替方向乘子法(ADMM)，用于解决稀疏信号恢复问题。本文基于L_1-L_2正则化的近端算子解析解，提出了一种带有保护机条件的ν加速ADMM算法(νADMMgd)。该算法通过引入ν加速技术，显著提高了收敛速度，并通过保护机制确保了算法的稳定性。数值实验表明，νADMMgd算法在稀疏信号恢复问题上表现出色，能够在较短时间内达到更优的函数值，且在处理大规模数据时具有较高的计算效率。实验还验证了该算法在不同稀疏度和正则化参数下的鲁棒性。总体而言，本文提出的算法在稀疏信号恢复问题中具有显著的优势，尤其是在高维数据和大规模优化问题中表现尤为突出。

关键词

稀疏信号恢复，L_1-L_2正则化，交替方向乘子法，ν加速

ν-Accelerated ADMM Algorithm with Safe Guard Conditions

Hailin Zhang

School of Mathematics and Statistics, Chongqing Jiaotong University, Chongqing

Received: Feb. 27th, 2025; accepted: Apr. 11th, 2025; published: Apr. 23rd, 2025

Abstract

This paper proposes a novel Alternating Direction Method of Multipliers (ADMM) combined with L_1-L_2 regularization and ν-acceleration for solving sparse signal recoveryproblems. Based on the analytical solution of the proximal operator for L_1-L_2 regularization, we introduce a ν-accelerated ADMM algorithm with safeguard conditions(νADMMgd). This algorithm significantly enhances the convergence speed by incorporating ν-acceleration techniques and ensures stability through safeguard mechanisms. Numerical experiments demonstrate that the νADMMgd algorithm performs

文章引用: 张海林. 带有保护条件的 ν 加速 ADMM 算法[J]. 理论数学, 2025, 15(4): 298-308.
DOI: 10.12677/pm.2025.154133

excellently in sparse signal recovery, achieving superior function values in a shorter time and exhibiting high computational efficiency when handling large-scale data. The experiments also validate the robustness of the algorithm under different sparsity levels and regularization parameters. Overall, the proposed algorithm shows significant advantages in sparse signal recovery problems, particularly in high-dimensional data and large-scale optimization scenarios.

Keywords

Sparse Signal Recovery, L_1-L_2 Regularization, Alternating Direction Method of Multipliers, ν-Acceleration

1. 引言

随着科学技术的进步和大数据时代的到来，越来越多的大规模数据集变得可用且重要。在这种背景下，压缩感知作为一种有效的数据处理方法应运而生。压缩感知(Compressive Sensing, CS)的历史可以追溯到 20 世纪初，但它在现代科学和工程中的广泛应用始于 2004 年。压缩感知的核心思想是通过少量的线性测量恢复稀疏信号，这与传统的采样定理有很大的不同。压缩感知的核心包括编码和解码两个过程。编码的过程是通过一组线性测量获取数据，这可以表示为 $b=Ax$，其中 A 是一个 $M\times N$ 的矩阵，b 是测量结果。如果矩阵 A 的行数 M 小于列数 N (即 $M<N$)，那么信号 可以被压缩。解码的过程是从 b 中恢复原始信号 x，但前提是假设信号 x 是稀疏的。解码问题可以形式化为一个优化问题：

$$\min_{x\in\mathbb{R}^n}\|x\|_0$$

$$\text{s.t}\quad Ax=b,$$

其中 $\|x\|_0$ 表示信号中非零元素的数量(即 L_0 范数)。优化目标是找到一个稀疏度最高(即非零元素最少)的信号 x。

压缩感知中最大的挑战之一就是解码问题 L_0 范数最小化是一个 NP-hard 问题[1]，计算复杂度极高，需穷举所有可能解，尤其在高维情况下难以求解。为克服这一难题，研究人员提出用 L_1 范数替代 L_0 范数。L_1 最小化是凸优化问题，计算更高效，通常能找到全局最优解，尽管稀疏度可能略逊于 L_0 最小化，但在许多应用中仍能提供接近最优的稀疏解。近年来，研究者们提出了多种非凸惩罚函数(如 L_1-L_2)来替代传统的 L_1 范数，以提升稀疏性，尤其是在感知矩阵相关性较高的情况下。Lou [2]采用凸差分(DC)方法将 L_1-L_2 问题转化为 L_1 问题，并用 ADMM 求解。Pong [3]提出的 pDCAe 算法结合了 DC 分解和迭代重加权 L_1 方法。由于 DC 算法求解 L_1 子问题的计算开销较大，Lou [4]进一步给出了 $L_1-\alpha L_2$ 范数的解析近端映射，并基于此构建了 FBS 和 ADMM 算法，其中 ADMM 计算效率明显优于现有方法。2024 年，Lei [5]结合 Nesterov 加速[6] [7]和非单调线搜索，提出近端梯度法(PGels)，其实验结果略优于 pDCAe 算法。

本文中，我们考虑 L_1-L_2 正则化的最优化问题：

$$\min_{x\in\mathbb{R}^n}F(x):=f(x)+\lambda\left(\|x\|_1-\|x\|_2\right), \tag{1}$$

其中 f 是一个光滑的凸函数，$\lambda>0$ 是正则化参数。

Alessandro Buccini [8]提出了一种通用的加速 ADMM 框架去解决凸优化问题，其中 ν 加速被成功应

用。虽然 ν 加速只是一种启发式方法，缺乏严格的理论支持，但在实践中往往表现比较良好。基于以上研究，我们将这种带有 ν 加速的 ADMM 框架与 L_1-L_2 的近端映射解析解相结合，并在具体的 L_1-L_2 模型下比较该算法与 ADMM 算法、pDCA$_e$ 算法和 DCA。

2. 带保护条件的 ν 加速 ADMM 算法

在本节中，我们提出了一种结合 L_1-L_2 惩罚函数的近端算子解析解和 ν 加速技术的 ADMM 算法。这里的 ν 加速灵感来源于 ν-方法[9]，这是一种半迭代方法，能够显著加速经典的 Landweber 方法[10]。此外，ν 加速还借鉴了文献[11]中的类似思想，将 ν-方法扩展到了统计方法(例如 Richardson-Lucy 算法)中，以增强这些算法的收敛速度和效果。

尽管 ν-方法拥有坚实的理论基础，尤其是在信号恢复和迭代算法中的应用，但值得注意的是，ν 加速本身是一种启发式的更新公式，其理论证明相对较少，且尚缺乏严格的数学支撑。因此，在实际应用中，虽然该方法表现出显著的加速效果，但如何确保其在所有情形下都具有良好的收敛性仍然是一个挑战。

为了克服这一点，我们通过引入保护条件来确保 ν 加速的有效性。在每一次迭代更新中，保护条件可以动态地调整是否使用加速更新，以保证算法的稳定性和收敛性。具体而言，当满足特定条件时，算法会使用经过 ν 加速处理的点进行更新，从而逐步加速收敛。否则，算法则会采用常规的更新策略。这种机制有效地避免了因不合适的加速操作导致的潜在不稳定性，确保了算法在实际应用中的可靠性。

为了使用 ADMM 去解决问题(1)，我们需要引进辅助变量 y 去将问题(1)转化为如下带约束问题：

$$\begin{aligned}\min_{x,y}\quad & f(y)+\lambda\left(\|x\|_1-\|x\|_2\right)\\ \text{s.t.}\quad & x=y.\end{aligned}\tag{2}$$

问题(2)的增广拉格朗日函数为

$$L_\delta(x,y,u)=f(y)+\lambda\left(\|x\|_1-\|x\|_2\right)+\delta\langle u,x-y\rangle+\frac{\delta}{2}\|x-y\|_2^2,$$

ADMM 的迭代公式为：

$$x_{k+1}\in\arg\min_x L_\delta(x,y_k,u_k)=\arg\min_x\left\{\lambda\left(\|x\|_1-\|x\|_2\right)+\frac{\delta}{2}\|x-y_k+u_k\|_2^2\right\},$$

$$y_{k+1}=\arg\min_y L_\delta(x_{k+1},y,u_k)=\arg\min_y\left\{f(y)+\frac{\delta}{2}\|x_{k+1}-y+u_k\|_2^2\right\},\tag{3}$$

$$u_{k+1}=u_k+x_{k+1}-y_{k+1}.\tag{4}$$

针对带有 L_1-L_2 惩罚函数的近端算子，其定义如下：

$$\operatorname{prox}_{\lambda(L_1-L_2)}(y)=\arg\min_x\left\{\|x\|_1-\|x\|_2+\frac{1}{2\lambda}\|x-y\|_2^2\right\}.\tag{5}$$

在定义问题(5)的封闭形式解时会使用到软阈值函数，$i=1,2\cdots,n-1,n$，其定义如下：

$$S(y_i,\lambda)=\begin{cases}y_i-\lambda, & y_i>\lambda,\\ 0, & |y_i|\le\lambda,\\ y_i+\lambda, & y_i<-\lambda.\end{cases}$$

引理 2.1 给定 $y\in\mathbb{R}^n$，$\lambda>0$，对于优化问题(2)的最优解 x^* 有以下结论[4]：

(a) 当$\|y\|_\infty > \lambda$时， $z = S(y_1,\lambda), S(y_2,\lambda), \cdots, S(y_{n-1},\lambda), S(y_n,\lambda)^T$， $x^* = z\left(\frac{\|z\|_2 + \lambda}{\|z\|_2}\right)$。

(b) 当$\|y\|_\infty = \lambda$时，如果$|y_i| < \lambda$，则$x_i^* = 0$，且$\|x^*\|_2 = \lambda$，对于所有的i，有$x_i^* y_i \geq 0$。当有多个分量的绝对值等于λ时，最优解不唯一。

(c) 当$0 < \|y\|_\infty < \lambda$时，最优解$x^*$是一个1-稀疏向量，如果$|y_i| < \|y\|_\infty$，则$x_i^* = 0$，且$\|x^*\|_2 = \|y\|_\infty$，对于所有的$i$，有$x_i^* y_i \geq 0$。最优解的数量与具有最大绝对值$\|y\|_\infty$的分量的数量相同。

(d) 当$\|y\|_\infty \leq 0$时， $x^* = 0$。

证明：易得关于x^*分量的符号和绝对值大小顺序的关系，即：

$$x_i^* \begin{cases} \geq 0, & y_i > 0, \\ \leq 0, & y_i < 0. \end{cases}$$

以及

$$|y_i| > |y_j| \to |x_i^*| \geq |x_j^*|, \tag{6}$$

否则，我们总可以改变x_i^*的符号或交换x_i^*和x_j^*的绝对值，使目标函数的值更小。因此，我们可以不失一般性地假设y是一个非负、非递增的向量，即：

$$y_1 \geq y_2 \geq \cdots \geq y_n \geq 0.$$

定义目标函数：

$$H(x) = \|x\|_1 - \|x\|_2 + \frac{1}{2\lambda}\|x - y\|_2^2.$$

当$x \neq 0$， $H(x)$进行最小化的一阶最优性条件为

$$\left(1 - \frac{\lambda}{\|x\|_2}\right)x = y - \lambda p, \tag{7}$$

其中$p \in \partial\|x\|_1$是L_1范数的次梯度。当$x = 0$时，一阶最优性条件为：

$$\|y - \lambda p\|_2 = \lambda.$$

简单计算可得，对于满足(7)的任意$x \neq 0$，我们有

$$\begin{aligned} H(x) &= \|x\|_1 - \|x\|_2 + \frac{1}{2\lambda}\|x\|_2^2 - \langle x, p + (\frac{1}{\lambda} - \frac{1}{\|x\|_2})\rangle + \frac{1}{2\lambda}\|y\|_2^2 \\ &= -\|x\|_2 + \frac{1}{2\lambda}\|x\|_2^2 - \left(\frac{1}{\lambda} - \frac{1}{\|x\|_2}\right)\|x\|_2^2 + \frac{1}{2\lambda}\|y\|_2^2 \\ &= -\frac{1}{2\lambda}\|x\|_2^2 + \frac{1}{2\lambda}\|y\|_2^2 < H(0). \end{aligned}$$

(a) 如果$y_1 > \lambda$，则$y_1 - \lambda p_1 > 0$。对于$x^* \neq 0$的情况，我们有$x_1^* > 0$且$1 - \frac{\lambda}{\|x^*\|_2} > 0$。对于任何满足$y_i \leq \lambda$的$i$，我们有$x_i = 0$；否则，对于这样的$i$，(7)式的左侧为正，而右侧为非正。对于任何满足$y_i > \lambda$的$i$，我们有$p_i = 1$。因此，

$$y - \lambda p = \left(S(y_1,\lambda), S(y_2,\lambda), \cdots, S(y_{n-1},\lambda), S(y_n,\lambda)\right)^T.$$

令 $z=y-\lambda p$，我们有 $x^*=z\left(\frac{\|z\|_2+\lambda}{\|z\|_2}\right)$。因此 $x^*=0$ 是最优解。

(b) 如果 $y_1=\lambda$，则 $y_1-\lambda p_1\geq 0$。令 $j=\min\{i:y_i<\lambda\}$，则对于 $i\geq j$，$x_i^*=0$；否则，对于这样的 i，(7)式的右侧为负，因此 $1-\frac{\lambda}{\|x^*\|_2}<0$。这意味着 $x_1^*=0$，且由于(6)式，x^* 不是全局最优解。对于 $x^*=0$ 的情况，我们有 $1-\frac{\lambda}{\|x^*\|_2}=0$。因此，任何最优解 x^* 都满足 $x_i^*=0$ 对于 $i\geq j$，$\|x^*\|_2=\lambda$，且对于所有 i，$x_i^*y_i\geq 0$。当 y 的多个分量具有相同的绝对值 λ 时，存在无限多个解。

(c) 假设 $0<y_1<\lambda$。令 $j=\min\{i:y_i<\|y\|_\infty\}$，则对于 $i\geq j$，有 $x_i^*=0$；否则，对于这样的 i，(7)式的右侧为负，因此 $1-\frac{\lambda}{\|x^*\|_2}<0$ 且

$$y_1-\lambda p_1=\left(1-\frac{\|x^*\|_2}{\lambda}\right)x_1^*\leq\left(1-\frac{\|x^*\|_2}{\lambda}\right)x_i^*=y_i-\lambda p_i,$$

这与 $y_1>y_i$ 矛盾。对于 $x^*=0$ 的情况，我们有 $1-\frac{\lambda}{\|x^*\|_2}<0$。从(7)式可知，$\lambda-\|x^*\|_2=\|y-\lambda p\|_2$。找到具有最大范数的 x^* 等价于找到 $p\in\partial\|x^*\|_1$，使得 $\|y-\lambda p\|_2$ 最小且 $x^*=0$。因此，我们选择 x^* 为 1-稀疏向量，且 $\|x^*\|_2=\lambda-\|y-\lambda p\|_2=\lambda-(\lambda-y_1)=y_1$。

(d) 假设 $y_1\leq 0$。如果存在 $x^*\neq 0$，则 $\|y-\lambda p\|_2\geq|y_1-\lambda|\geq\lambda$，而(7)式意味着 $\|y-\lambda p\|_2=\lambda-\|x^*\|_2<\lambda$。因此，我们无法找到 $x^*\neq 0$。然而，我们可以找到 $p\in\partial\|0\|_1$，使得 $\|y-\lambda p\|_2=\lambda$。因此，$x^*=0$ 是最优解。

证毕。

ν 加速基于以下公式：

$$\hat{v}_k=\mu_k\overline{v}_k+(1-\mu_k)\overline{v}_{k-1}+\rho_k(\overline{v}_k-\overline{v}_{k-1}),\tag{8}$$

其中 ν 是自定义的常数，而 μ_k 和 ρ_k 是两个依赖于 k 和 ν 的参数：

$$\mu_k=\frac{1+(k-1)(2k-3)(2k+2\nu-1)}{(k+2\nu-1)(2k+4\nu-1)(2k+2\nu-3)},$$

$$\rho_k=\frac{4(2k+2\nu-1)(k+\nu-1)}{(k+2\nu-1)(2k+4\nu-1)}.$$

我们强调，虽然 ν-方法有着强大的理论背景，但 ν 加速只是一个更新公式，缺乏理论基础，所以我们在这里并不会给出收敛性分析。如文献[11]中所指出的，选择参数 ν 通常是一个很重要的任务，在第三章我们会给出具体的参数值。

为引入我们的保护条件，首先我们假设 $(\overline{y}_{k+1})$ 序列和 $(\overline{u}_{k+1})$ 序列分别由 ADMM 算法中的(3)，(4)生成。由(6)可知，$\overline{y}_{k+1}$ 和 $\overline{u}_{k+1}$ 经过 ν 加速计算过后的点如下：

$$\hat{y}_k=\mu_k\overline{y}_k+(1-\mu_k)\overline{y}_{k-1}+\rho_k(\overline{y}_k-\overline{y}_{k-1}),$$

$$\hat{u}_k=\mu_k\overline{u}_k+(1-\mu_k)\overline{u}_{k-1}+\rho_k(\overline{u}_k-\overline{u}_{k-1}).$$

其次，我们将定义混合残差(combined residual)：

$$\gamma_{k+1}=\delta^{-1}\|\hat{u}_{k+1}-u_k\|_2^2+\delta\|\hat{y}_{k+1}-y_k\|_2^2,$$

其中 y_k， u_k 为上一步的迭代点。同时我们将定义 γ_0，我们令 $\gamma_0 = \chi\gamma$，一般 χ 为大于 1 的常数，这里取 $\chi = 2$，其中 γ 为第一次标准 ADMM 迭代后的组合残差，由于在第一次迭代中 $u_0 = y_0 = 0$，所以 $\gamma_0 = \delta^{-1}\|\overline{u}_1\|_2^2 + \delta\|\overline{y}_1\|_2^2$。

基于上述定义，我们现在能够引入保护条件，以进一步提升算法的稳定性和效率。这个保护条件的核心思想是在迭代过程中动态地决定是否采用经过 ν 加速计算的点作为更新点。具体来说，如果条件 $\gamma_{k+1} < \gamma_0\eta^{k+1}$ 得到满足，这表明当前迭代点满足一定的优化标准，此时我们更新 $y_{k+1} = \hat{y}_{k+1}$ 和 $u_{k+1} = \hat{u}_{k+1}$。这样的更新策略有助于加速算法的收敛。相反，如果不满足该条件，我们则保留未经 ν 加速计算的原始更新值 $y_{k+1} = \overline{y}_{k+1}$ 和 $u_{k+1} = \overline{u}_{k+1}$，以确保算法的稳健性。参数 η 在这一过程中扮演着关键角色，其值通常选取在 0 到 1 之间。这个参数控制着 ν 加速技术的应用程度，允许算法逐步适应并利用 ν 加速的优势。通过精细调节 η 的值，我们可以平衡算法的收敛速度和稳定性，避免因过度加速而导致的潜在不稳定。该保护条件的主要目的是确保算法在每次迭代中都能选择最优的更新点，从而避免算法性能的退化。在最不利的情况下，如果算法总是选择未经 ν 加速计算的点作为更新点，那么 νADMMgd 算法将退化为传统的 ADMM 算法，失去加速优势。通过引入这一保护条件，我们为算法提供了一种稳健的迭代更新机制，确保了算法的收敛性和效率，同时也提高了算法对不同问题条件的适应能力。此外，这种保护条件还有助于防止算法在迭代过程中出现发散现象，特别是在处理非凸优化问题时。通过动态调整更新策略，算法能够更加灵活地应对不同的优化挑战，从而在广泛的应用场景中展现出更好的性能。总之，这一保护条件不仅增强了算法的鲁棒性，也为算法的进一步优化和改进提供了坚实的基础。

结合以上内容，我们给出算法 1：

算法 1：带保护条件的 ν 加速 ADMM 算法(ν ADMMgd)。

步骤 0：初始化 $x_0 = y_0 = u_0 = 0$， $\overline{y}_0 = \overline{u}_0 = 0$，设置 $k = 0$。

步骤 1：

$$
\begin{aligned}
&x_{k+1} \in \arg\min_x \left\{ \lambda\left(\|x\|_1 - \|x\|_2\right) + \frac{\delta}{2}\|x - y_k + u_k\|_2^2 \right\},\\
&\overline{y}_{k+1} = \arg\min_y \left\{ f(y) + \frac{\delta}{2}\|x_{k+1} - y + u_k\|_2^2 \right\},\\
&\overline{u}_{k+1} = u_k + x_{k+1} - \overline{y}_{k+1},\\
&\hat{y}_k = \mu_k\overline{y}_k + (1-\mu_k)\overline{y}_{k-1} + \rho_k\left(\overline{y}_k - \overline{y}_{k-1}\right),\\
&\hat{u}_k = \mu_k\overline{u}_k + (1-\mu_k)\overline{u}_{k-1} + \rho_k\left(\overline{u}_k - \overline{u}_{k-1}\right),\\
&\gamma_{k+1} = \delta^{-1}\|\hat{u}_{k+1} - u_k\|_2^2 + \delta\|\hat{y}_{k+1} - y_k\|_2^2.
\end{aligned}
$$

步骤 2：设置保护条件

If $\gamma_{k+1} < \gamma_0\eta^{k+1}$

$y_{k+1} = \hat{y}_{k+1}$;

$u_{k+1} = \hat{u}_{k+1}$;

Else:

$y_{k+1} = \overline{y}_{k+1}$;

$u_{k+1} = \overline{u}_{k+1}$;

End.

步骤 3：如果不满足终止迭代条件，则令 $k = k+1$ 并转至步骤 1。

3. 数值实验

本节，我们将进行数值实验去测试带有保护机制的加速 ADMM 算法解决 L_1-L_2 正则化最小二乘问题的效果。所有实验都在 Matlab2022a 上进行，电脑 CPU 为 Ryzen 5 6600H (3.30 GHz)，RAM 大小为 16 GB，操作系统为 Windows 11。

其中 L_1-L_2 正则化最小二乘问题为：

$$\min_{x\in\mathbb{R}^n} G(x) := \frac{1}{2}\|Ax-b\|_2^2 + \lambda\left(\|x\|_1 - \|x\|_2\right). \tag{9}$$

在我们的数值实验中，我们分别选择两种不同的正则化参数 $\lambda = 1\times10^{-3}$ 和 $\lambda = 5\times10^{-4}$ 来比较 vADMMgd 算法与现有算法的性能。针对问题(9)，我们设置了一个 A 为 $720j\times2560j$ 维的随机高斯矩阵，其中的元素是独立同分布的标准高斯随机变量。接着，我们对矩阵 A 进行了归一化处理，使得其每一列的范数均为单位范数。同时，生成一个 $(2560j)$ 维的稀疏向量 z (z 只有 $(80j)$ 个非零元素，且这些非零元素服从标准正态分布)，z 中非零元素的位置和数值是随机生成的，其中 $j = 1,2,3,\cdots,10$ 表示实验的不同规模。接下来，我们构造了观测向量 $b = Az + 0.01\cdot\omega$，其中 $\omega\in\mathbb{R}^m$ 是一个具有独立同分布标准高斯分布的随机向量，且 $m = 2560j$。为了评估算法的表现，我们在 10 种不同尺寸的矩阵 A 进行实验。每种尺寸的矩阵，我们分别进行了 10 次独立实验，并计算每次实验的 CPU 时间和停止迭代时的函数值。

对于 vADMMgd 算法，我们通过经验调参确定了加速因子 $\nu = 0.5$ 和衰减因 $\eta = 0.8$。此外，算法的终止条件设置为：

$$\frac{\|x_{k+1}-x_k\|_2}{\max\left(\|x_k\|_2,\epsilon\right)} < 10^{-5},$$

其中 $\epsilon = 1\times10^{-16}$，用于避免终止条件过于严格而导致无意义的迭代停止。

这里，我们通过实验展示了我们如何选择参数 ν 的取值。这里我们只展示部分参数下的迭代次数和停止迭代时的损失函数值，具体是实验中将会更加细致。通过表 1 我们可以看到，ν 加速中的参数 ν 只要在合适的范围内选取算法便不会对停止迭代时的函数值有影响，所以只要参数 ν 的选取在合理范围内，便不会对算法的鲁棒性产生影响，而只是在迭代次数和 CPU 时间上有影响。但当 A 规模较大时，算法的 CPU 时间便会有较为明显的差距。

Table 1. The influence of parameter v in vADMMgd algorithm on the algorithm
表 1. vADMMgd 算法中 v 参数对算法的影响

A	v	Iters	Time (s)	Favl
720 * 2560	0.1	348	0.7	3.10588e-2
	0.3	406	0.7	3.10588e-2
	0.5	277	0.5	3.10588e-2
	0.7	413	0.7	3.10588e-2
	0.9	382	0.6	3.10588e-2
	1.0	2	0.0	3.83000e+01
3600 * 12800	0.1	3.8	12.2	1.57694e-01
	0.3	365	14.9	1.57694e-01
	0.5	302	11.9	1.57694e-01
	0.7	346	13.6	1.57694e-01

续表

	0.9	325	13.2	1.57694e-01
	1.0	2	1.4	1.65341e+02

根据我们的实验结果，表 3 和表 4 显示了各算法在不同的 λ 取值下的停止迭代时间和损失函数值，其中加黑字体为最优结果。从结果中可以看出，*v*ADMMgd 算法始终能够以最少的时间达到最佳的优化效果，表现出明显的优势。从表 3 和表 4 中较大规模的 A 的实验结果看，还证明了 *v*ADMMgd 算法在高效求解稀疏优化问题中的潜力，尤其是在大规模问题中，能够显著提高计算效率，同时保持较高的恢复精度。尽管 *v*ADMMgd 算法在处理大规模数据时表现出较高的计算效率，但在需要实时处理的场景中，*v*ADMMgd 算法的性能依赖于多个参数的选择，如 ν 加速因子和衰减因子 η。这些参数的选择通常需要通过经验验证，可能在不同问题中需要不同的调参策略，增加了算法的使用难度。此外，我们还探讨了算法在稀疏重建任务中的效果。我们提供了一个仿真实验来展示在噪声情况下的稀疏恢复性能，实验设置参考了文献[12]。这里选取一个长度为 $N=512$ 的真实信号 x，其中有 $K=130$ 个非零元素。我们从 M 个测量值 b 中恢复该信号，测量值 b 由一个正态分布的矩阵 A 生成(矩阵的每一列被归一化为零均值和单位范数)，并添加了标准差为 $\sigma=0.01$ 的高斯白噪声。为了补偿噪声的影响，我们使用均方误差(MSE)来量化恢复性能。如果已知真实解 x 的支持集，记为 $\Lambda=\mathrm{supp}(\mathbf{x})$，我们可以通过公式

$$\mathrm{MSE}_{\mathrm{oracle}}=\sigma^2\cdot\mathrm{tr}\left(\left(\mathbf{A}_\Lambda^\top\mathbf{A}_\Lambda\right)^{-1}\right)$$

计算 Oracle 解的 MSE，作为基准。实验结果如表 2，vADMMgd 算法在 CPU 时间上整体表现最优，仅略逊于 pDCAe，但从 MSE 可知 vADMMgd 算法总体上为最稳定的算法。

Table 2. Restoration results of noisy signals (average and standard deviation of 100 experiments)
表 2. 噪声信号的恢复结果(100 次实验的平均值和标准差)

方法	M	MSE	Time (s)
Oracle		1.20e-02	
ADMM		4.008e-01 (7.15e-02)	5.44e-02 (1.12e-02)
DCA	238	4.503e-01 (7.36e-02)	8.84e-02 (1.36e-02)
pDCAe		4.223e-01 (8.78e-02)	4.34e-02 (7.4e-03)
*v*ADMMgd		4.006e-01 (7.15e-02)	4.47e-02 (1.20e-02)
Oracle		1.17e-02	
ADMM		3.918e-01 (1.185e-01)	3.88e-02 (1.54e-02)
DCA	250	3.948e-01 (1.235e-01)	7.60e-02 (1.21e-02)
pDCAe		4.024e-01 (1.399e-01)	3.86e-02 (6.6e-03)
*v*ADMMgd		3.918e-01 (1.185e-01)	3.88e-02 (1.54e-02)

为直观的去观察算法的迭代情况，我们在规模为 7200×25600 的 A 上将算法在每一次迭代的真实信号与重建信号之差的范数记录并作图。当我们分别设置 $\lambda=1e-3$ 和 $\lambda=5e-4$ 时，如图 1(a)，(b)所示，*v*ADMMgd 算法在解决问题(9)的前 100 次迭代中，虽然前 10 次迭代时函数值会产生大幅度的震荡，但 ν ADMMgd 算法却是迭代次数最少的算法。紧接着，我们还探究了各算法在稀疏信号恢复上的表现效果，我们将测试矩阵设置为 64×256 的高斯矩阵。当 $\lambda=1e-3$ 和 $\lambda=5e-4$ 时，如图 1(c)，图 1(d)所示，*v*ADMMgd 算法与其他算法在稀疏恢复上表现与其他算法几乎一致。

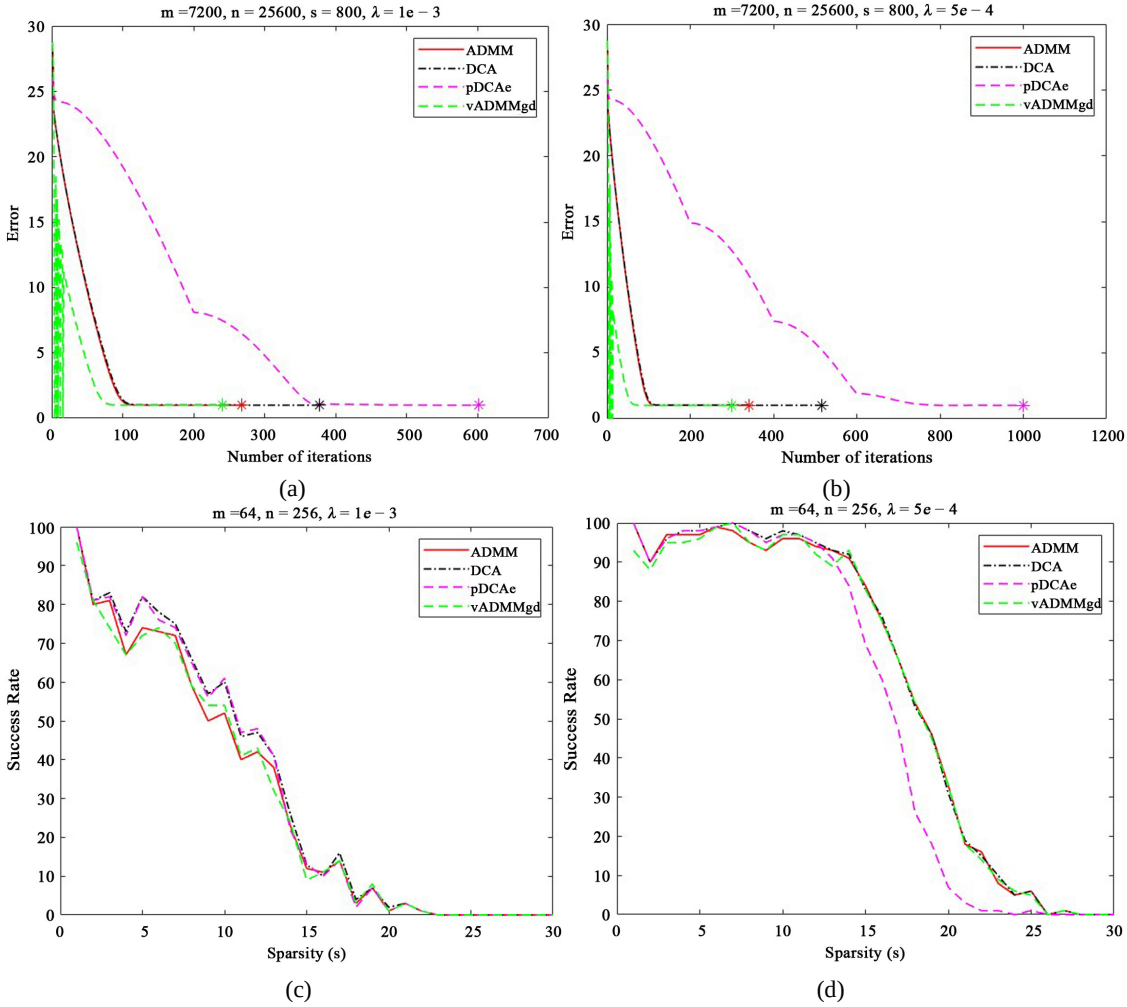

Figure 1. (a) and (b) show the norm of the difference between the recovered values and the true values for each algo rithm as the iterations progress when the test matrix is a 7200 × 25600 Gaussian matrix, with λ = 1e-3 and λ = 5e-4, respectively.(c) and (d) illustrate the success rate of sparse reconstruction for true signals with sparsity ranging from 1 to 30, using different algorithms when the test matrix is a 64 × 256 Gaussian matrix, with λ = 1e-3 and λ = 5e-4, respectively.

图 1. (a)和(b)为各算法在测试矩阵选取为 7200×25600 高斯矩阵，$\lambda = 1e-3$ 和 $\lambda = 5e-4$ 的情况下，恢复值与真实值之差的范数随算法迭代的变化。c 和 d 为各算法在测试矩阵选取为 64×256 高斯矩阵，$\lambda = 1e-3$ 和 $\lambda = 5e-4$ 的情况下，对稀疏度为 1 至 30 的真实信号进行稀疏重建的成功率

Table 3. $\lambda = 5e-4$

表 3. $\lambda = 5e-4$

Problem Size		Time (s)				Favl			
m	n	ADMM	DCA	pDCAe	*v*ADMMgd	ADMM	DCA	pDCAe	*v*ADMMgd
720	2560	0.62	1.10	0.95	0.56	2.9418e-02	2.9418e-02	2.9421e-02	2.9418e-02
1440	5120	2.90	4.35	4.38	2.47	6.1645e-02	6.1645e-02	6.1650e-02	6.1645e-02
2160	7680	7.18	11.39	11.93	6.34	9.3376e-02	9.3376e-02	9.3382e-02	9.3376e-02
2880	10240	12.88	19.45	21.27	11.23	1.25885e-01	1.25885e-01	1.25896e-01	1.25885e-01
3600	12800	19.68	30.18	34.89	17.51	1.62678e-01	1.62681e-01	1.62678e-01	1.62678e-01
4320	15360	21.42	31.86	36.25	18.97	1.93117e-01	1.93126e-01	1.93117e-01	1.93117e-01

续表

5040	17920	26.81	39.23	46.64	24.03	2.29211e-01	2.29211e-01	2.29215e-01	2.29211e-01
5760	20480	35.51	51.67	59.88	31.81	2.54939e-01	2.54947e-01	2.54939e-01	2.54939e-01
6480	23040	44.84	65.02	73.98	35.49	2.90211e-01	2.90224e-01	2.90211e-01	2.90211e-01
7200	25600	56.25	80.41	95.31	48.48	3.21470e-01	3.21476e-01	3.21470e-01	3.21470e-01

Table 4. $\lambda = 1e-3$
表 4. $\lambda = 1e-3$

Problem Size		Time (s)				Favl			
m	n	ADMM	DCA	pDCAe	*v*ADMMgd	ADMM	DCA	pDCAe	*v*ADMMgd
720	2560	0.45	0.71	0.55	0.36	5.8701e-02	5.8701e-02	5.8703ee-02	5.8701e-02
1440	5120	1.78	2.65	2.43	1.49	1.23018e-01	1.23018e-01	1.23019e-01	1.23018e-01
2160	7680	3.95	5.69	5.48	3.22	1.86358e-01	1.86358e-01	1.86359e-01	1.86358e-01
2880	10240	7.07	9.93	9.93	5.94	2.51239e-01	2.51239e-01	2.51241e-01	2.51239e-01
3600	12800	10.98	15.36	14.79	9.21	3.24686e-01	3.24686e-01	3.24689e-01	3.24686e-01
4320	15360	21.42	15.98	21.91	13.68	3.85427e-01	3.85427e-01	3.85431e-01	3.85427e-01
5040	17920	26.81	22.05	28.47	19.16	4.57481e-01	4.57481e-01	4.57485e-01	4.57481e-01
5760	20480	29.20	39.56	37.27	26.63	5.08803e-01	5.08803e-01	5.08808e-01	5.08803e-01
6480	23040	36.63	48.94	46.87	33.51	5.79212e-01	5.79212e-01	5.79216e-01	5.79212e-01
7200	25600	46.42	61.67	58.02	39.72	6.38260e-01	6.38260e-01	6.38265e-01	6.38260e-01

4. 总结

本文提出了一种结合 $L_1 - L_2$ 正则化近端算子解析解和 v 加速 ADMM 算法，用于求解稀疏信号恢复问题。通过仿真实验验证了 *v*ADMMgd 算法在稀疏信号恢复任务中的恢复性能和计算效率优势。数值实验表明，*v*ADMMgd 算法在不同规模的测试矩阵上均表现出优异的性能，尤其是在大规模问题上，其计算时间和迭代次数明显优于传统的 ADMM 算法、DCA 算法和 pDCA e 算法。尽管 *v*ADMMgd 算法在稀疏信号恢复问题中表现出色，但仍存在一些局限性，如理论基础不足和不同问题中需要不同的调参策略，增加了算法的使用难度。未来的研究可以通过推广到其他类型正则化问题、在分布式计算中应用、加强理论基础和开发自适应参数选择策略等方向，进一步提升算法的性能和适用范围。

基金项目

本文在重庆市自然科学基金(CSTB2022NSCQ-MSX0409, CSTB2022NSCQMSX0406)和重庆市教育委员会科技项目(KJZD-M202201303, sKJQN202201349)资助下完成。

参考文献

[1] Natarajan, B.K. (1995) Sparse Approximate Solutions to Linear Systems. *SIAM Journal on Computing*, **24**, 227-234. https://doi.org/10.1137/s0097539792240406

[2] Yin, P., Lou, Y., He, Q. and Xin, J. (2015) Minimization of L_1 - L_2 for Compressed Sensing. *SIAM Journal on Scientific Computing*, **37**, A536-A563. https://doi.org/10.1137/140952363

[3] Wen, B., Chen, X. and Pong, T.K. (2017) A Proximal Difference-Of-Convex Algorithm with Extrapolation. *Computational Optimization and Applications*, **69**, 297-324. https://doi.org/10.1007/s10589-017-9954-1

[4] Lou, Y. and Yan, M. (2017) Fast L_1 - L_2 Minimization via a Proximal Operator. *Journal of Scientific Computing*, **74**, 767-785. https://doi.org/10.1007/s10915-017-0463-2

[5] Yang, L. (2023) Proximal Gradient Method with Extrapolation and Line Search for a Class of Non-Convex and Non-

Smooth Problems. *Journal of Optimization Theory and Applications*, **200**, 68-103. https://doi.org/10.1007/s10957-023-02348-4

[6] Nesterov, Y. (2983) A Method for Solving the Convex Programming Problem with Convergence Rate o($1/k^2$). *Doklady Akademii Nauk SSSR*, **269**, 543-547.

[7] Nesterov, Y. (2013) Introductory Lectures on Convex Optimization: A Basic Course, Volume 87. Springer Science & Business Media.

[8] Buccini, A., Dell'Acqua, P. and Donatelli, M. (2019) A General Framework for ADMM Acceleration. *Numerical Algorithms*, **85**, 829-848. https://doi.org/10.1007/s11075-019-00839-y

[9] Hanke, M. (1991) Accelerated Landweber Iterations for the Solution of Ill-Posed Equations. *Numerische Mathematik*, **60**, 341-373. https://doi.org/10.1007/bf01385727

[10] Landweber, L. (1951) An Iteration Formula for Fredholm Integral Equations of the First Kind. *American Journal of Mathematics*, **73**, 615. https://doi.org/10.2307/2372313

[11] Dell'Acqua, P. (2015) *v* Acceleration of Statistical Iterative Methods for Image Restoration. *Signal, Image and Video Processing*, **10**, 927-934. https://doi.org/10.1007/s11760-015-0842-9

[12] Xu, Z.B., Chang, X.Y., Xu, F.M. and Zhang, H. (2012) $L_{1/2}$ Regularization: A Thresholding Representation Theory and a Fast Solver. *IEEE Transactions on Neural Networks and Learning Systems*, **23**, 1013-1027. https://doi.org/10.1109/tnnls.2012.2197412