# 松弛 ILU 预处理器在油藏数值模拟软件中的移植*

戴 民
(北京大学数学科学学院 100871)
郭艳玲
(中国人民公安大学理科部 100038)
石济民
(香港理工大学应用数学系)
王瑞河
(中国石油勘探开发研究院)

# IMPLEMENTATION OF THE RELAXED ILU PRECONDITIONER TO RESERVOIR SIMULATORS

Dai Min
(*School of Mathematical Science, Peking University, Beijing* 100871)
Guo Yanling
(*Chinese People's Public and Security University* 100038)
Shih Tsimin
(*Dept. of Applied Mathematics, The Hong Kong Polytechnic University*)
Wang Ruihe
(*Institute of Oil Exploration/Development of China*)

**Abstract**

In this paper, the Relaxed-ILU preconditioner is applied to solve the linear equations arising from the black oil models. Numerical experiments demonstrate that the method is superior to the ILU preconditioner which is already extensively used in reservoir simulations. We have implemented the relaxed-ILU preconditioner into some practical reservoir simulators.

**Key words:** ILU MILU relaxed preconditioner, numerical reservoir simulation

* 2003 年 3 月 10 日收到.

## §1. 前 言

油藏数值模拟问题最终归结为求解一组代数方程

$$Ax = b. \tag{1}$$

众所周知, 网格方程的系数矩阵 $A$ 具有大型、稀疏、病态等特点, 其条件数 $\kappa(A) = \mathrm{O}(h^{-2})$, 这里 $h$ 表示网格步长. 考虑到计算机的存储限制及求解速度, 迭代法是求解大型系统的首选方法. 对于迭代求解, 其收敛速度取决于系数矩阵 $A$ 的条件数, 降低 $A$ 的条件数可以大大提高求解速度. 针对这一问题, 自七十年代开始兴起了预处理方法.

所谓预处理方法指将问题 (1) 的求解转化为另一条件数大为降低的等价问题的求解, 即寻找可逆矩阵 $M$ 和 $N$, 满足:

a. $\kappa(N^{-1}AM^{-1}) << \kappa(A)$;

b. $M^{-1}$, $N^{-1}$ 易求.

这样我们可以将问题 (1) 的求解转化为如下等价问题的求解:

$$\overline{A}\overline{x} = \overline{b}, \tag{2}$$

这里 $\overline{A} = N^{-1}AM^{-1}$, $\overline{x} = Mx$, $\overline{b} = N^{-1}b$. 通常我们称 $M$, $N$ 为预处理器.

目前在油藏数值模拟软件中广泛采用的预处理器为 ILU. 该预处理器在油藏数值模拟领域的有效性已经被许多文献和实际应用所证实 [4,5]. 有关 MILU 和松弛 ILU 预处理器在该领域的应用并不多见. 文 [1] 曾将 MILU-CG 方法应用到三维三相黑油模型, 数值算例表明与 ILU 相比, MILU 的速度有明显提高. 理论上 MILU 应比 ILU 有效, 但是已有文章 [10] 指出, MILU 的效果有时并不及 ILU.

文 [1] 所使用的是经典的 MILU 预处理器 (即 MILU(0)), 与之相比较的是 ILU(0), 但是目前油藏数值模拟软件对于规则剖分自然排序产生的七对角 (或二维问题五对角) 矩阵通常采用 ILU(1) 预处理器. 此外, 该文采用通常的 IMPES 离散方式, 所得到的矩阵对称性较强, 故可采用 CG 方法求解. 自八十年代起, 为了提高 IMPES 方法的稳定性, 对消去饱和度关于时间的差分后所得到的压力方程采用牛顿法求解, 且隐式处理井方程, 这就大大加强了矩阵的非对称性, CG 方法已不适合用于这类问题的求解.

本文使用 Orthomin 方法求解上述矩阵方程, 分别以松弛 ILU, ILU 和 MILU 作预处理器. 数值结果表明, 松弛 ILU 比 ILU 和 MILU 预处理器更有效. 我们已将该方法成功地移植到实用的油藏数值模拟软件中. 第二节介绍预处理 Orthomin 方法. 第三节介绍 ILU, MILU 和松弛 ILU 预处理器. 第四节简介软件实现. 第五节给出一些数值算例, 最后给出结论.

## §2. 预处理 Orthomin 方法

CG 方法是针对对称问题设计的. 预处理 CG 方法对于对称或近似对称的问题非常有效, 但对于非对称性较强的问题, 预处理 CG 方法的效果不理想. 因此许多研究者针对非对

称问题提出一类广义 CG 方法， Orthomin 方法就是其中一种. 早在 1976 年 Vinsome 已将 Orthomin 方法应用到油藏数值模拟. 如今结合预处理技术的 Orthomin 方法已被广泛地应用到油藏数值模拟问题的代数方程组的求解.

本文采用的是右端预处理 Orthomin 方法. 所谓右端预处理指在 (2) 式中取 $N = I$($I$ 为单位方阵), 即把问题 (1) 的求解转化为如下问题的求解:

$$\overline{A}\overline{x} = b,$$

这里 $\overline{A} = AM^{-1}$, $\overline{x} = Mx$. 类似可定义左端预处理, 但右端预处理比左端预处理计算量小, 且程序代码也较简单. 下面给出右端预处理 Orthomin 的算法.

右端预处理 Orthomin 算法

Choose $x_0$,$k$

Compute $r_0 = f - Ax_0$

Set $p_0 = r_0$

for $i = 0$ until convergence do

$$\alpha_i = \frac{(r_i, Ap_i)}{(Ap_i, Ap_i)}$$

$$x_{i+1} = x_i + \alpha_i p_i$$

$$r_{i+1} = r_i - \alpha_i Ap_i$$

$$b_j^{(i)} = -\frac{(AM^{-1}r_{i+1}, Ap_j)}{(Ap_j, Ap_j)}, \max\{0, i-k+1\} \le j \le i$$

$$p_{i+1} = M^{-1}r_{i+1} + \sum_{j=j_i}^{i} b_j^{(i)} p_j, j_i = \max\{0, i-k+1\}$$

end (i-loop)

现有的油藏数值模拟软件通常选用 ILU 作为上述右端预处理器 $M$, 本文将采用 MILU 和松弛 ILU, 这里的 Orthomin 方法为非重启动型.

## §3 ILU, MILU 和松弛 ILU 算法

### (1) ILU 算法

ILU 的思想最早出现在六十年代初, 该方法在七十年代由于预处理方法的兴起得到发展. 它的基本思想是充分考虑稀疏矩阵的特点, 在 LU 分解过程中仅仅允许对预先给定的某些位置的填充 (fill-ins), 即预先给定允许填充位置的指标集 $S$, 使得

$$l_{i,j} = 0 \text{ if } j > i \text{ or } (i,j) \notin S;\ u_{i,j} = 0 \text{ if } i > j \text{ or } (i,j) \notin S. \tag{3}$$

通常我们选取

$$S = \{(i,j) \mid a_{ij} \neq 0\}, \tag{4}$$

这里 $A=(a_{ij})$, 即算法仅对 $A$ 的非零位置执行, 由此导出的 LU 因子的乘积作为预处理器 $M$, $M=LU$. 因为作为一个好的预处理器, 它必须在某种意义下与 $A$ 近似. 一种典型的策略是要求在集合 $S$ 上 $M$ 的元素与 $A$ 一致, 即

$$m_{i,j}=a_{i,j} \text{ if } (i,j)\in S. \tag{5}$$

具体算法如下 [6]:

ILU 算法

for $r$ := 1step 1 until n-1 do

  $d:=1/a_{r,r}$

  for $i:=(r+1)$ step 1 until n do

    if $(i,r)\in S$ then

      $e:=da_{i,r}; a_{i,r}:=e;$

      for $j:=(r+1)$step 1 until n do

        if $(i,j)\in S$and if $(r,j)\in S$ then

          $a_{i,j}:=a_{i,j}-ea_{r,j}$

        end if

      end (j-loop)

    end if

  end (i-loop)

end (r-loop)

**(2) MILU 算法**

MILU 的思想在文 [9] 中已经出现, 文 [11] 将它推广到一般矩阵. 它的基本思想是非常简单的, 改变 ILU 所要求的条件 $m_{i,i}=a_{i,i}$, 即以下式代替 (5),

$$\sum_{j=1}^{n} m_{i,j}=\sum_{j=1}^{n} a_{i,j}\ \forall i \text{ and } m_{i,j}=a_{i,j} \text{ if } i\neq j \text{ and } (i,j)\in S. \tag{6}$$

上式与 (3) 可完全决定分解过程. 事实上, 该算法将 ILU 算法中所扔掉的元素加到主对角线上. 具体算法如下 [6]:

MILU 算法

for $r$ := 1step 1 until n-1 do

  $d:=1/a_{r,r}$

  for $i:=(r+1)$ step 1 until n do

    if $(i,r)\in S$ then

      $e:=da_{i,r}; a_{i,r}:=e;$

      for $j:=(r+1)$ step 1 until n do

        if $(r,j)\in S$ then

          if $(i,j)\in S$ then

```
            a_{i,j} := a_{i,j} − e a_{r,j}
          else
            a_{i,i} := a_{i,i} − e a_{r,j}
          end if
        end if
      end (j-loop)
    end if
  end (i-loop)
end (r-loop)
```

**(3) 关于 ILU 和 MILU 的数学结果**

ILU 与 MILU 的理论至今仍不够完善. 下面介绍一些理论结果.

首先考虑 ILU 和 MILU 的存在性, 即条件 (3), (5) 对 ILU 以及条件 (3), (6) 对 MILU 是否足够保证相应的不完全分解的存在? 理论结果表明对某些类型的矩阵可以保证 [8]. 特别地, 矩阵的对角占优性越强, MILU 的存在性越可以得到保证 [12].

除了存在性, 我们关心对系数矩阵的条件数的改进情况. 考虑二阶椭圆型方程, 采用自然排序, 由五点格式导出的系数矩阵 $A$, 可以证明对 ILU 有 $\kappa\left(M^{-1}A\right)=\mathrm{O}\left(h^{-2}\right)$(即不改进 $A$ 的条件数的渐进状态), 而对 MILU 有 $\kappa\left(M^{-1}A\right)=\mathrm{O}\left(h^{-1}\right)$[9]. 此外文 [1] 也对 MILU 的原理给出初步阐述.

**(4) 松弛 ILU 算法**

虽然理论上以 MILU 作预处理器的条件数比 ILU 好, 但 MILU 对实际问题的应用效果并不稳定. 有文章指出 MILU 对舍入误差很敏感, 因此在八十年代提出了在 ILU 和 MILU 之间作插值的松弛 ILU 方法 [13,14], 即在 MILU 的算法中最里层循环关于 $a_{i,i}$ 的修正用下式代替:

$$a_{i,i}:=a_{i,i}-\omega e a_{r,j},$$

这里 $0\leq\omega\leq1$ 为松弛因子. 易见 $\omega=0$ 和 $\omega=1$ 分别对应 ILU 和 MILU 算法. $\omega$ 的取值很关键. 最简单的方法是取 $\omega$ 为某个常数. 本文通过一些算例推荐软件中 $\omega$ 的缺省值.

**(5) 关于允许填充集 $S$ 的选取**

有关 $S$ 的选取是很重要的. 可以看到, 若允许所有的填充, 即选取

$$S=\{(i,j)\mid\forall i,j\},\tag{7}$$

那么由 (3), (5) 和 (7) 决定的算法即为完全的 LU 分解. 我们通常称由 (3), (5) 和 (4) 决定的算法为 ILU(0); (3), (6) 和 (4) 决定的算法为 MILU(0). 文 [1] 采用的即是 MILU(0), 与之比较的 ILU(0). 为保证预处理器较好地近似于 $A$, 允许填充集应包含 (4) 式的 $S$. 一般说来, 允许填充集越大, 预处理效果越好. 若允许填充集太大, 则不仅需占用较多的存储空间, 且增加了预处理过程的计算量, 因此要适当选取 $S$.

现有的油藏数值模拟软件所采用的 ILU 预处理器通常为 ILU(1). 我们以七对角矩阵为例, 叙述 ILU(1) 算法中 $S$ 的结构. 对于三维规则网格、采用自然排序差分离散所得到的七

对角矩阵， ILU(0) 算法中 $S$ 即为这七条对角线 (如图 1 实线所示) 的位置的集合. 由定义 (5), ILU(0) 预处理器在这七条对角线上与原始矩阵一致, 但另外还有六条对角线 (如图 1 虚线所示). 在 ILU(1) 的算法中, 以上述十三条对角线的位置的集合作为允许填充集. 对于二维问题所产生的五对角矩阵, 也可类似导出相应的 ILU(1).

MILU(1) 和松弛 ILU(1) 算法中的 $S$ 与 ILU(1) 完全相同.

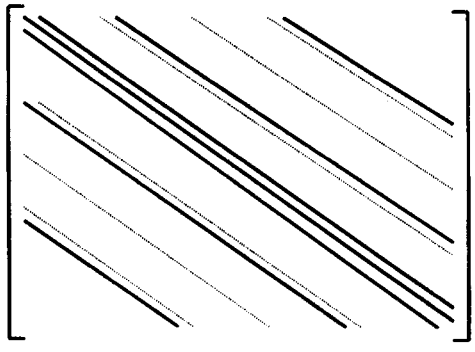

图 1

## §4. 软件实现

本文考虑黑油模型, 采用隐压显饱格式, 并隐式处理井方程. 我们将松弛 ILU 和 MILU 移植到两个模拟器. 下面将分别介绍:

模拟器 A. 该模拟器对消去饱和度关于时间的差分后所得到的压力方程采用牛顿法, 考虑规则区域上的规则剖分、自然排序及五点 (二维) 或七点 (三维) 格式, 用 ILU(1) -Orthomin 求解. 了解 ILU(1)-Orthomin 程序代码的研究人员熟知, 对于这样导出的代数方程组所对应的五对角或七对角矩阵, 在编制实现 ILU(1)-Orthomin 这段程序时不必照搬第二, 三节的算法, 而应充分考虑矩阵的特殊性, 并尽量做到向量化. 松弛 ILU(1), MILU(1) 与 ILU(1) 预处理器的差别仅仅在于不完全分解所得到的矩阵的对角线元素不同, 因此, 我们只要修改 ILU(1)-Orthomin 的程序代码中不完全分解部分的对角线元素的计算即可得到 MILU(1)-Orthomin 和松弛 ILU(1)-Orthomin 的程序代码, 不必增加存储量, 也不降低代码的向量化程度.

模拟器 B. 该模拟器由本文第四作者研制. 它以 ILU(0) 作为预处理器, 采用 Orthomin 求解. 它的一个显著特点是无论存储还是计算都不考虑死节点, 因此代数方程组的系数矩阵形状不规则. 我们分别将 MILU(0) 和松弛 ILU(0) 预处理器移植到该模拟器.

## §5. 数值试验

我们采用国内一些油田的实际数据, 比较了分别以松弛 ILU, MILU, ILU 为预处理器的 Orthomin 方法的效果.

测试 A.

该测试在模拟器 A 上完成. 采用国内油田的一些三维实际数据, 分别有多口在不同层面有不同射开程度且在不同时间加上的生产、注水井.

表格 1 列出了 ILU(1) 和 MILU(1) 预处理器的模拟结果 (均采用自动选取步长). 从表 1 可以看出对这两个算例 MILU(1) 的效果明显比 ILU(1) 好, 不仅所用的 CPU 时间少, 总外迭代的次数也明显少于 ILU(1) 预处理器. 这里的 CPU 时间为整个模拟时间, 如果仅比较代数方程组的求解时间, MILU(1) 的效果将更明显.

**表 1　ILU 和 MILU 的模拟结果比较**

| | 算例 1 | | 算例 2 | |
|---|---|---|---|---|
| | CPU(S) | 总外迭代数 | CPU(S) | 总外迭代数 |
| ILU(1) | 3294 | 231 | 6601 | 464 |
| MILU(1) | 2378 | 212 | 5797 | 409 |

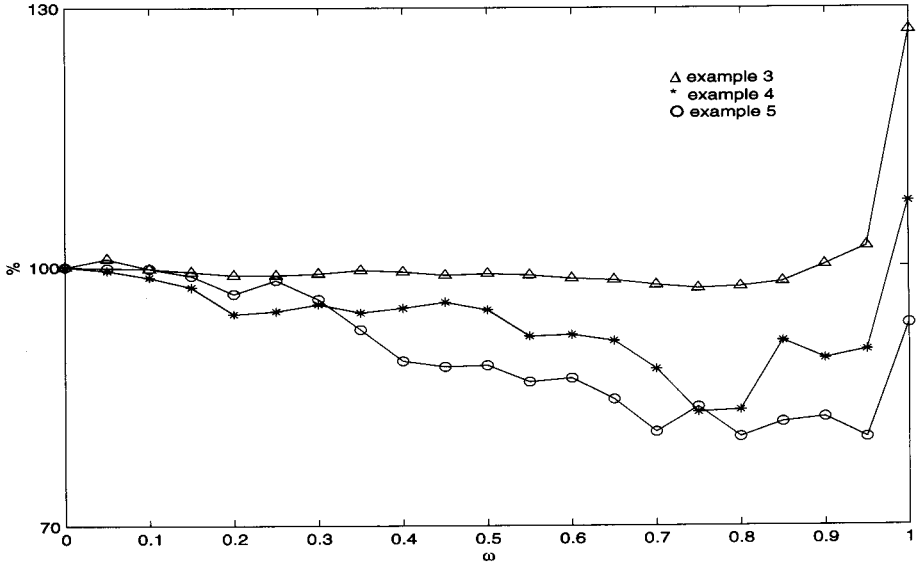


图 2　不同松弛因子的松弛 ILU 算法与 ILU 算法所用 CPU 时间之比

不幸的是, MILU(1) 的效果并不稳定, 对某些算例它所用的 CPU 时间超出 ILU(1) 很多, 因而我们不得不考虑采用松弛 ILU(1) 预处理器. 这里我们给出三个代表性的算例 (算例 3-5), 结果由图 2 显示. 横坐标表示松弛因子的值, 纵坐标表示松弛 ILU(1) 所用 CPU 时间与 ILU(1) 所用 CPU 时间之比. 注意到 $\omega = 0, 1$ 时分别对应 ILU(1) 和 MILU(1) 情形. 算例 3 和算例 4 显示 MILU(1) 效果不好, 而松弛 ILU(1) ($\omega = 0.7, 0.75, 0.8$) 仍能取得相对较好的效果. 而在算例 5 中 MILU(1) 明显优于 ILU(1) 时, 松弛 ILU(1) ($\omega = 0.7, 0.8, 0.95$) 的表现比 MILU(1) 更好. 我们还对另外一些算例作了测试, 结果表明多数情形松弛因子选

在 0.7 到 0.8 之间较为合适. 因此, 如果要在软件中设置一个缺省的松弛因子, 我们建议取为 0.8.

测试 B.

该测试在模拟器 B 上完成. 比较松弛 ILU(0) 与 ILU(0) 的效果. 下面算例中松弛因子始终取 0.8.

我们熟知, 对有底水的油藏进行模拟比没有底水的要困难, 垂向连通情形的模拟比垂向渗透率为零时困难. 我们做了两组算例 (算例 6 和算例 7), 每组分别模拟了有底水、无底水和无底水且垂相渗透率为零三种情形. 松弛 ILU 和 ILU 对各种情形所用的 CPU 时间由表 2, 表 3 列出. 可以看出, 以上结果在无底水且渗透率为零情形松弛 ILU 结果不理想 (算例 6 中松弛 ILU 比 ILU 差, 算例 7 中两者一样). 部分原因在于无底水且垂相渗透率为零情形容易求解, 松弛 ILU 和 ILU 都很快收敛, 而松弛 ILU 在不完全分解时对角线元的计算量比 ILU 大. 对其余情形, 松弛 ILU 都明显较 ILU 优越. 对有底水且垂向渗透率不为零情形效果更明显. 这说明, 对越困难的问题, 松弛 ILU 越能显示它的优越性. 我们在对其它算例的试验中 (如提高求解精度) 也得到相同的结论, 这里不再一一列出.

**表 2 松弛 ILU 与 ILU 效果比较 (算例 6)**

| | $K_z \neq 0$, 有底水 | $K_z \neq 0$, 无底水 | $K_z = 0$, 无底水 |
|---|---|---|---|
| 松弛 ILU | 620 | 544 | 524 |
| ILU | 742 | 601 | 463 |
| % | 83.5% | 90.5% | 113% |

**表 3 松弛 ILU 与 ILU 效果比较 (算例 7)**

| | $K_z \neq 0$, 有底水 | $K_z \neq 0$, 无底水 | $K_z = 0$, 无底水 |
|---|---|---|---|
| 松弛 ILU | 663 | 641 | 631.7 |
| ILU | 848 | 772 | 631.2 |
| % | 78% | 83% | 100% |

## §6. 结 论

从以上实际算例的测试可以得出下述结论:

(1) 松弛 ILU 的预处理效果比 ILU 好.

(2) 缺省的松弛因子取为 0.8.

(3) 问题越困难 (如有底水情形), 松弛 ILU 的效果越明显.

(4) MILU 的效果不稳定, 这与一些文献所指出的一致.

因此, 松弛 ILU 预处理器在不增加存储量、不降低算法的向量化程度的情况下取得比 ILU 更好的效果, 且很容易嵌入到现有的以 ILU 为预处理器的油藏数值模拟软件中. 事实上, 我们已将该方法移植到一些实用的油藏数值模拟软件中.

致谢: 本文得到北京石油勘探开发科学研究院的协助, 作者在此表示感谢.

## 参 考 文 献


[1] 石济民，林振宝，岳兴业，三维三相黑油模型的 MILU-CG 方法及数值结果比较，石油勘探与开发，21:6 (1994), 52-58.

[2] 吕涛，石济民，林振宝，区域分解算法，科学出版社，1992.

[3] 韩大匡，陈钦雷，闫存章，油藏数值模拟基础，石油工业出版社，1993.

[4] 周维四，二维二相模型的不完全乔里斯基－共轭斜量法，石油学报，17:2 (1986), 83-90,

[5] P.K.W. Vinsome, Orthomin, An Iterative Method for Solving Sparse Sets of Simultaneous Linear Equations, Paper SPE 5729, Society of Petroleum Engineers of AIME, 1976.

[6] T.F. Chan, H.A. Van Der Vorst, Approximate and Incomplete Factorizations, 1994. (preprint)

[7] J.A. Meijerink, H.A. Van Der Vorst, Guidelines for the Usage of Incomplete Decompositions in Solving Sets of Linear Equations as They Occur in Practical Problems, J. Comp. Phys., 44:1 (1981), 134-155.

[8] W. Hackbusch, Iterative Solution of Large Sparse Linear Systems of Equations. Applied Math. Sci. Series, No. 95, Springer Verlag, 1994.

[9] T. Dupont, R. Kendall, and H. Rachford. An Approximate Factorization Procedure for Solving Self-adjoint Elliptic Difference Equations. SIAM J. Numer. Anal., 5 (1968), 559-573.

[10] H.A. Van der Vorst, The convergence behaviour of preconditioned CG and CG-S in the presence of rounding errors. Preconditioned Conjugate Gradient Methods, edited by O. Axelsson and L.Yu. Kolotilina, Springer-Verlag, 1990.

[11] I. Gustafsson, A Class of First-order Factorization Methods. BIT, 18 (1978), 142-156.

[12] I. Gustafsson, Modified incomplete cholesky (MIC) methods. Preconditioning Methods-Theory and Applications, edited by D. Evans, 265-293. Gordon and Breach, new York, 1983.

[13] O. Axelsson and G. Lindskog. On the eigenvalue distribution of a class of preconditioning matrices. Numer. Math., 48 (1986), 479-498.

[14] C. Ashcraft and R. Grimes. On vectorizing incomplete factorizations and SSOR preconditioners. SIAM J. Sci. Stat. Comput., 9 (1988), 122-151.