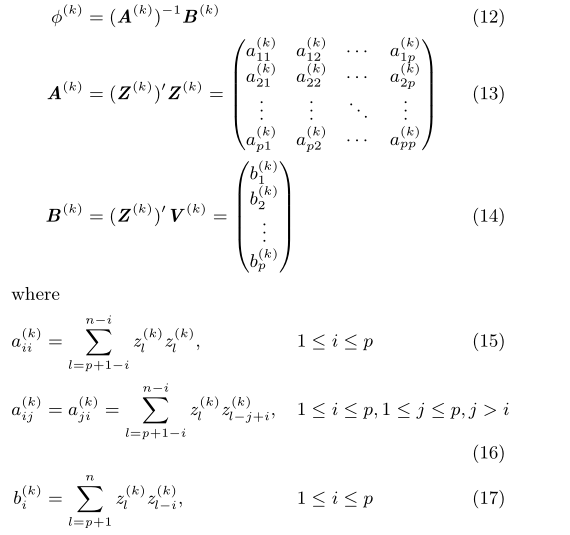
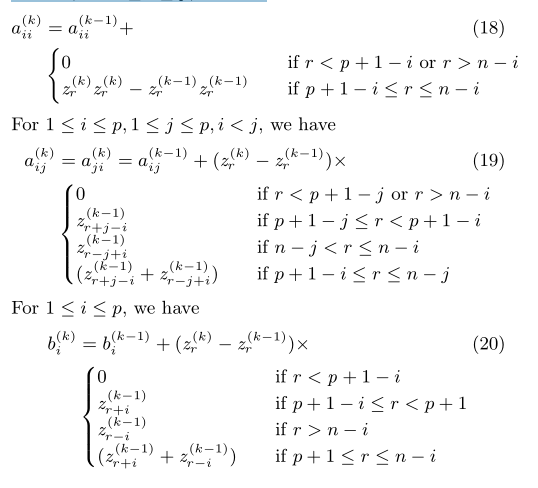
5.2 增量计算直觉。

在算法 1 的每次迭代中，参数 φ(k) 由公式 5 w.r.t 估计。 Z (k) 和 V (k) 在所有 n 个点上。但是，参考第 3.4 节中的最小变化原则，只有一个点，即 r，在每次迭代中都会发生变化，即 y (k)r 6= y (k−1)r 。也就是说，Z(k)和V(k)中的大多数值与Z(k-1)和V(k-1)相同。我们表明（在下面的命题 10 中）可以通过仅考虑更改的值而不是整个 Z (k) 和 V (k) 来增量计算 φ(k)。每次迭代中参数估计的时间复杂度因此从线性时间降低到恒定时间。

5.2.1 递归公式为了实现增量计算，我们将方程 5 重写为参数估计如下，

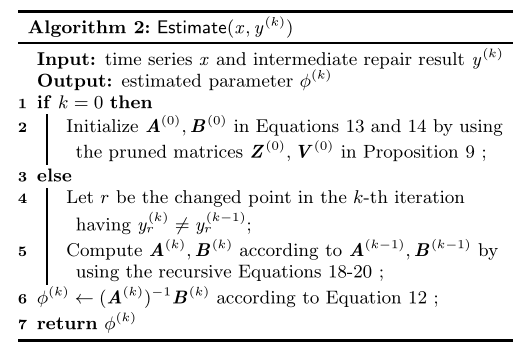


下面的结论说明了 A(k) 和 B (k) 的大小分别为 p × p 和 p × 1，可以从 A(k-1) 和 B (k-1) 递归计算。 AR(p)/ARX(p) 模型中预定义的 p 在算法中是一个固定值，并且有 p ≪ n。等式 12 在 A(k) 和 B(k) 上的计算成本将显着低于等式 5 w.r.t。 Z (k) 和 V (k) 在所有 n 个点上。命题 10。设 r 为当前修复迭代中的变化点，具有 y (k)r 6= y (k-1)r 或等效地 z (k)r 6= z (k-1)r 。 A(k), B (k) 可以从 A(k-1), B (k-1) 递归计算。也就是说，对于 1 ≤ i ≤ p，我们有



5.2.2 递归算法算法 2 显示了增量估计参数 φ(k) 的过程。对于第一次迭代中的 k = 0，通过公式 13 和 14 w.r.t 计算 A(0)、B(0)。然而，矩阵 Z (0), V (0) 是不可避免的。尽管如此，可以如算法 2 中的第 2 行所示应用命题 9 中值为 0 的行的有效修剪。

算法2：Estimate(x, y (k)) 输入：时间序列x和中间修复结果y(k) 输出：估计参数φ(k) 1 if k = 0 then 2 初始化A(0), B(0)在等式 13 和 14 中，使用命题 9 中的修剪矩阵 Z (0), V (0) ； 3 else 4 设 r 为第 k 次迭代中的变化点， y (k)r 6= y (k-1)r ； 5 使用递归方程 18-20 根据 A(k-1)、B(k-1) 计算 A(k)、B(k)； 6 φ(k) ← (A(k))-1B (k) 根据公式 12； 7 返回 φ(k)



对于以下迭代 k > 0，执行从 A(k-1)、B(k-1) 对 A(k)、B(k) 的递归计算。如命题 10 所示，A(k)、B(k) 中的所有 p2 + p 值都可以在恒定时间内递归更新。因此，参数估计的复杂性从 O(n)（参见公式 15-17）降低到公式 18-20 中的 O(1)。

示例 10（使用增量计算的参数估计，示例 6 继续）。再次考虑 x = {6, 10, 9.6, 8.3, 7.7, 5.4, 5.6, 5.9, 6.3, 6.8, 7.5, 8.5} 和 y (0) = {6, 5.6, 5.4, 8.3, 7.7, 5.4, 5.6,例 6 中的 5.9, 6.3, 6.8, 7.5, 8.5}。我们有 V (0) = {-4.4, -4.2, 0, 0, 0, 0, 0, 0, 0, 0, 0}′ 和 Z ( 0) = {0, -4.4, -4.2, 0, 0, 0, 0, 0, 0, 0, 0}'。给定 p = 1，矩阵 A(0), B (0) 只有一个元素，其中 a(0) 11 = (−4.4)2 + (−4.2)2 = 37 和 b(0) 1 = (− 4.4) ∗ (−4.2) = 18.48 由算法 2 中的第 2 行初始化。根据例 7 和例 8，修复点为 y (1) 4 = 6.2。我们有 z (1) 4 = 6.2 − 8.3 = −2.1，而 z (0) 4 = 0。算法 2 中的第 5 行从 A(0)、B(0) 增量计算 A(1)、B(1) ，即，a(1) 11 = a(0) 11 + (-2.1)2 = 41.41 通过使用等式 18 中的增量更新，并且 b(1) 1 = b(0) 1 +(-2.1- 0) ∗ (−4.2 + 0) = 27.3 参考公式 20。最后，使用公式 12 根据 A(1)、B(1) 计算参数 φ(1) 1 ，即 φ(1) 1 = 27.3 /41.41 = 0.66。

1. 实验

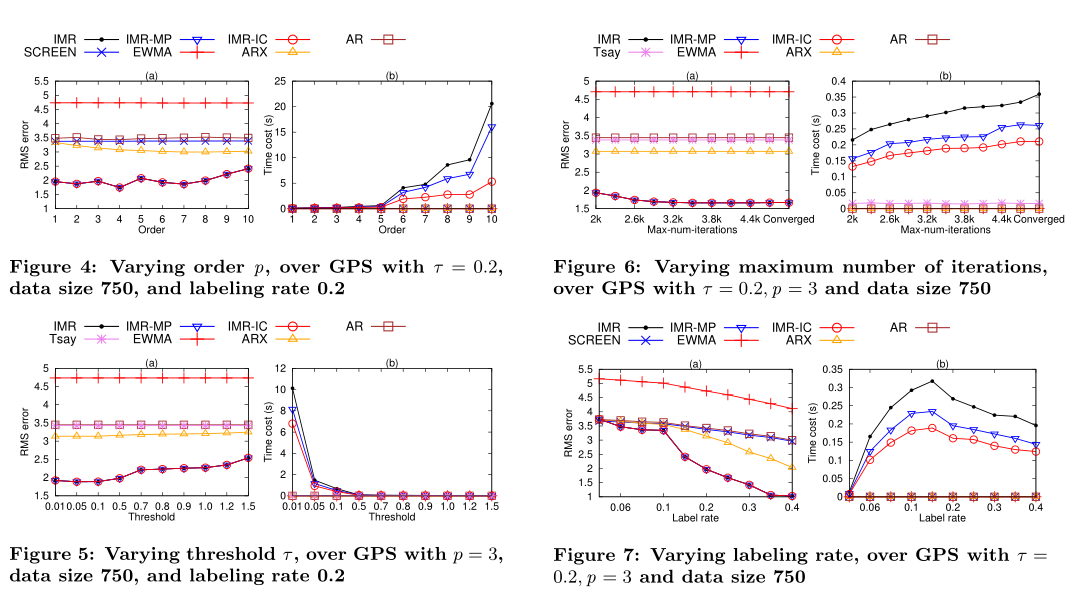
在本节中，我们通过实验将我们提出的方法 IMR 与最先进的方法进行比较，包括使用 (1) AR [3]、(2) ARX [3]、(3) 的异常检测方法ARIMA [18, 3]，(4) Tsay [23] 作为模型，(5) 基于平滑的方法 EWMA [13]，和 (6) 基于约束的方法 SCREEN [22]。 GPS 数据有真正的错误。在 GPS 数据集中，真正的错误是自然嵌入的，相应的基本事实是手动标记的。它通过一个收集 GPS 读数带着智能手机在校园里走来走去的人。由于我们确切地知道行走路径，因此在轨迹的 742 个点中，有 186 个脏点是人工识别的。脏点的真实位置也被手动标记为ground truth。 （参见第 6.1 节中的主要结果。）

具有综合错误的 ILD 数据。英特尔实验室数据（http://db.csail.mit.edu/labdata/labdata.html，ILD）包括在大约 38 天内每 31 秒从 54 个传感器进行的大量测量。以 31 秒为一个 epoch 并忽略缺失数据，从 2 月 29 日到 3 月 1 日在传感器 1 中获得了 4912 个点的数据集。我们通过在高斯分布下以方差 0.1 移动 3 的值来将误差综合注入数据中（参见图 8 中的一些示例）。这种“移位”错误在实践中很常见，例如传感器卡住了一小会儿，或者在一段时间内采集单元错误。 （参见第 6.2 节中的主要结果。） 标准。 RMS 误差 [15] 用于评估修复。令 x truth 是 clean 序列的 ground truth， x dirty 是嵌入了故障的观察序列， x repair 是修复序列。 RMS 误差 [15] 由下式给出：Δ(x truth, x repair) = vuut 1 n nX i=1 (x truthi − x repair i )2。该措施评估地面实况与其修复结果之间的距离。低 RMS 误差是首选。

* 1. 真实误差实验

GPS 数据真实误差的实验考虑了各种算法设置，包括（1）阶 p、（2）收敛阈值 τ、（3）最大迭代次数和（4）标记率。在 ILD 中也观察到类似的结果并省略。

* + 1. 示例结果 图 3 显示了 GPS 数据集的示例部分（在纬度上，转换后），包括收集到的有错误的观测值、标记的真值以及不同方法的修复结果。由于建筑物等各种影响，GPS 读数可能会与真实情况有所偏差。例如，时间点 400 和 500 之间的数据是从高层建筑附近的一个地方收集的，在该地方观察到明显的错误。如图所示，与其他方法相比，建议的 IMR 显示出最接近事实的修复。



* + 1. 变阶 p 图 4 显示了变阶 p 的结果，用于 AR(p)、ARX(p) 和 IMR(p)。首先，如图 4(a) 所示，基于 AR 的方法随着阶数 p 的增加表现出更好的性能，其中更多的历史值在每个值的预测中生效。然而，过大的 p 并没有显示出进一步的改进，因为这个简单的模型可能无法在大窗口中捕获复杂的语义。出于同样的原因，在 ARX 中也观察到了类似的结果。值得注意的是，由于每次迭代中最小修复的迭代策略，我们提出的 IMR 方法即使在 p = 1 时也已经实现了低 RMS 修复误差。结果验证了分析 p = 1 的 IMR(1) 特殊情况的必要性在第 4 节中。在图 4(b) 中，迭代 IMR 需要比其他仅通过数据一次的现有方法需要更高的时间成本也就不足为奇了。除了算法 1 中的原始 IMR 之外，我们还在第 5.1 节中展示了 IMR 与矩阵剪枝 (IMR-MP) 和算法 2 中用于有效参数估计的增量计算 (IMR-IC) 的结果。 IMR、IMR-MP 和 IMR-IC 在图 4(a) 中显示完全相同的精度结果。两种有效的估计方法都提高了图 4(b) 中的时间成本。特别是，用于恒定时间增量参数估计的 IMR-IC 显着降低了时间成本。总之，如图 4 和图 10 所示，分别在 GPS 和 ILD 数据集上，我们提出的 IMR 在修复精度方面没有明确的 p 阶偏好，而较大的 p 阶会导致更高的时间成本。
    2. 改变收敛阈值 τ 图 5 通过改变阈值 τ（p = 3）报告了结果。通过设置较小的 τ，IMR 需要更多的迭代才能收敛。图 5(b) 中相应的时间成本更高。具有较小 τ 的 IMR 可实现更好的修复性能，如图 5(a) 所示。然而，通过进一步降低阈值 τ，例如从 0.1 到 0.01，修复精度几乎无法进一步提高，而相应的迭代次数和时间成本显着增加。另一方面，通过增加阈值τ，时间成本降低。实际上，阈值 τ 提供了 IMR 的修复精度和时间成本之间的权衡。总之，较低的阈值确实会带来更好的结果（更低的 RMS 误差）并且需要更多的迭代（更高的时间成本）。有关阈值的更明确影响，请参见 ILD 的图 11。
    3. 指定最大迭代次数 在第 4 节中，我们分析了几种特殊情况，其中修复在某些条件下理论上可以保证收敛。对于不满足此类条件的一般情况，（尽管所有实验在第 6 节中在各种设置下收敛，有/没有理论收敛保证），可以指定最大迭代次数，作为实践中的一种补救措施，以避免等待收敛。也就是说，算法 1 在迭代次数达到 max-num-iterations 时终止，即使第 5 行中的收敛条件不满足。图 6 评估了最大迭代次数的各种设置（通过重复每个测试 10 次来报告平均时间成本）。如图所示，适度大量的迭代已经达到了良好的修复结果，即接近（最右边的）收敛结果。
    4. 可变标记率 图 7 说明了各种标记率的结果。标记率 0.1 表示 10% 的数据点在数据集中被标记为真实。毫不奇怪，标记率越高，IMR 和 ARX 的修复性能就越好，它们利用标记的真值，如图 7(a) 所示。一个有趣的结果是，随着标注率的增加，相应的时间成本在

图 7(b)，先增加后下降。原因是对于点几乎没有修改的小标记率（例如 0.06），迭代修复可以快速收敛，同时保持大多数脏数据不变。在这种情况下，相应的 RMS 误差很高，如图 7(a) 所示。随着输入中标记的数据越多，算法将识别和修复更多的脏点，从而导致更高的计算成本。当标注率较大时，如0.25，可能会标注大量脏点。因此，迭代修复可以再次快速收敛。作为异常检测方法，Tsay 和 ARIMA 的结果与 AR 和 ARX 的结果大致相似。 SCREEN 和 EWMA 方法不受阶数 p、阈值 τ 和标记率的影响。 SCREEN 表现不佳并不奇怪，这验证了我们在引言中的动机和分析。类似地，由于 EWMA 没有使用标记的真值，因此其性能很弱。

