High Order Lifting

AECF MPRI

Béguinot Julien
julien.beguiot@enst.fr
10 mars 2022

Context and Contributions

- Arne Storjohann 2002 on High Order Lifting
- Present applications of High-Order Lifting
- Can be seen as an extension of Dixon 1982 works on p-adic expansion for system solving
- Similarities with Hensel Lifting

Adic Representation

Definition

Let l be a nonnegative integer and $X \in K[x]$ of degree greater than zero. The X-adic expansion of $a \in K[x]$ is the unique expansion of $a = a_0 + a_1 X + a_2 X^2 + \ldots + a_l X^l$ where $\deg a_i < \deg X$. If $A \in K^{n \times m}[x]$ then the definition extends with polynomial matrices coefficients.

Example

```
K = GF(5)
Pol.<x> = PolynomialRing(K)
```

```
A = random_matrix(ring = Pol, nrows = 3 , ncols = 3, degree=6)
print(A)
```

```
[                  4*x^6 + x^5 + 2*x^4 + x^2 + 3         4*x^6 + 2*x^5 + 4*x^4 + x^3 + 3*x^2 + 4          2*x^6 + 3*x
^5 + 4*x^3 + 4*x^2 + 2*x + 4]
[  4*x^6 + 4*x^5 + 2*x^4 + 2*x^3 + 4*x^2 + x + 3 3*x^6 + 2*x^5 + 2*x^4 + 3*x^3 + 4*x^2 + 3*x + 2       4*x^6 + 2*x^5
+ 2*x^4 + x^3 + x^2 + x + 3]
[  3*x^6 + 3*x^5 + x^4 + 4*x^3 + 2*x^2 + 2*x + 3                   3*x^6 + 4*x^4 + 3*x^2 + x + 2          2*x^6 + x
^5 + 3*x^4 + 3*x^3 + x^2 + 3]
```

```
X = Pol.random_element(degree=3)
print(X)
```

```
3*x^3 + x^2 + 2*x + 1
```

```
NaiveLifting(A,X)
```

```
[
[             3       x^2 + 2*x       3*x^2 + 2]
[      2*x^2 + 4   x^2 + 4*x + 3   4*x^2 + x + 1]
[3*x^2 + 2*x + 3 4*x^2 + 3*x + 2             2*x],

[        3*x + 4 2*x^2 + 3*x + 3         2*x + 4]  [1 1 3]
[    x^2 + x + 3         2*x + 2 2*x^2 + 4*x + 1]  [1 2 1]
[  2*x^2 + x + 3   x^2 + 4*x + 3         x^2 + x], [2 2 3]
]
```

Some Operations

Definition

Let $a = a_0 + a_1X + a_2X^2 + \ldots + a_lX^l$ we define the followig operations.

- $Left(a, k) = a_k + a_{k+1}X + \ldots + a_lX^{l-k}$
- $Trunc(a, k) = a_0 + a_1X + a_2X^2 + \ldots + a_{k-1}X^{k-1}$
- If a (or $\det A$) $\perp X$, $Inverse(a, k)$ is the unique b such that $b = Trunc(b, k)$ and $Trunc(ab, k) = Trunc(ba, k) = Id$.

Proposition

Lemma (Linearity)

With fixed integer k, $Left(*, k)$ is linear.

Lemma (Invariance of *Left*)

If $\deg \gamma < k \deg X$ *then,*

$$Left(a + \gamma, k) = Left(a, k).$$

Lemma (Left And Trunc commutes)

If $l \leq k$,

$$Left(Trunc(a, k), l) = Trunc(Left(a, l), k - l).$$

Computations With X-adic polynomials

- Store elements by the list of minimum length of its X adic expension
- Truncature is free as it corresponds to drop the end of the list
- Left is free as it corresponds to shift the begining of the list
- addition/substraction of a, b requires to copy the term of lowest degree (assumed free here) and then to add the term of the lowest degree in $O(1 + \min(\deg a, \deg b))$
- multiplication by X correponds to appending a zero a the begining of the list an is hence free. (Similar for power of X)
- product is done in $O((1 + \deg a)(1 + \deg b)^(1 + \epsilon))$. Indeed you have to muliply the two representation and then eventually spread carries
- It as been shown that $Inverse(a, k)$ can be computed in $O((k \deg X)^{1+\epsilon}$

Naive Adic Lifting for System Solving

We want to compute $A^{-1}b$ to order k. In a divide and conquer approach we writte

$$A^{-1}b = S_0 + S_1 X^{k/2}.$$

- We compute S_0 as subproblem of size $k/2$
- We substitute S_0 to compute S_1 as the subproblem of size $k/2$,

$$S_1 = A^{-1}\frac{b - S_0}{X^{k/2}}$$

Limitation and motivation for High Order Lifting

- This is sequential, we need to compute S_0 to compute S_1. So subproblems cannot be stacked together.
- We cannot benefit from the fast multiplications between matrices. We sufer from Winograds theoreme of matrix vector multiplication.
- High Order Lifting will help to avoid this issue.

Adic Lifting

Lemma (Degree Bound)

$$\deg \det A \leq n \deg A$$

$$\deg((\det A)A^{-1}B) \leq \deg B + (n-1)\deg A$$

Idea, if $\det A \perp X$ then we can express $A^{-}1$ as an infinite X adic expansion!
We will exploit that for fast algorithms.

Theorem

Let $A = C + O(X^l)$ *and*
$A^{-1}B = \underbrace{* + *X + \ldots + *X^{k-1}}_{D} + \underbrace{*X^k + \ldots + *X^{k+l-1}}_{EX^k} + O(X^{k+l})$ *then*
$E = Trunc(CLeft(B - AD, k), l)$.

Démonstration.

$$B - AD = B - A(A^{-1}B - EX^k + O(X^{k+l})) = AEX^k + AO(X^{k+l})$$

$$Left(B - AD, k) = AE + AO(X^l)$$

$$CLeft(B-AD, k) = CA(E+O(X^l)) = (A^{-1}+O(X^l))A(E+O(X^l)) = E+O(X^l)$$

$$Trunc(CLeft(B - AD, k), l) = Trunc(E + O(X^l), l) = E$$

High Order Components

The objective is to compute the $E^{(i)} = Left(Z^{(i)}, 2^i - 2)$ where $Z^{(i)} = Inverse(A, 2^i)$ and $\deg X = d$. We writte $A^{-1} = C_0 + C_1 X + C_2 X^2 + \ldots$

- With successive quadratic lifting this would cost $O(2^k n^\theta d^{1+\epsilon})$
- The HighOrderComp algorithm will compute these specifically with cost reduced to $O(kn^\theta d^{1+\epsilon})$

High Order Components

Algorithm `HighOrderComp`$[X](A, k)$
Input: $A \in \mathsf{K}[x]^{n \times n}$ and $k \geq 2$
Output: $(E^{(1)}, E^{(2)}, \ldots, E^{(k)})$ as shown above
Condition: $X \perp \det A$ and $d = \deg X \geq \deg A$

1. $L := \text{Inverse}(A, 1)$;
   $H := \text{Trunc}(L\,\text{Left}(I - AL, 1), 1)$;
   $E^{(1)} := L + XH$;

2. **for** i **from** 2 **to** k **do**
   $L := \text{Trunc}(\text{Left}(E^{(i-1)}\,\text{Left}(-AL, 1), 1), 1)$;
   $H := \text{Trunc}(\text{Left}(E^{(i-1)}\,\text{Left}(-AH, 1), 1), 1)$;
   $E^{(i)} := L + XH$
   od;
   return $(E^{(1)}, E^{(2)}, \ldots, E^{(k)})$

FIGURE – High Order Components

Algorith flows

For $i = 1$,

$$L, H \leftarrow C_0, C_1$$

.

For $i = 2$;

$$L \leftarrow Trunc(Left((C_0 + C_1 X)Left(-AC_0, 1); 1), 1)$$

.

$$-AC_0 = -I + AC_1X + \ldots = o(1) + AC_1X + O(X^2)$$
$$Left(-AC_0, 1) = AC_1 + AC_2X + O(X^2)$$

$$\begin{aligned}(C_0 + C_1X)Left(-AC_0, 1) &= (C_0 + C_1X)A(C_1 + C_2X + O(X^2)) \\ &= (A^{-1} + O(X^2))A(C_1 + C_2X + O(X^2)) \\ &= C_1 + C_2X + O(X^2)\end{aligned}$$

$$\begin{aligned}L &\leftarrow Trunc(Left((C_0 + C_1X)Left(-AC_0, 1); 1), 1) \\ &\leftarrow Trunc(Left(C_1 + C_2X + O(X^2), 1), 1) \\ &\leftarrow C_2\end{aligned}$$

Similarly $H \leftarrow C_3$.

i = 3

$$L \leftarrow Trunc(Left(C_2 + C_3X)Left(-AC_2, 1), 1), 1)$$

$$-AC_2X^2 = -I + AC_0 + AC_1X + AC_3X^3 + AC_4X_4 + \dots$$
$$-AC_2 = o(1) + A(C_3X + C_4X^2 + \dots)$$
$$Left(-AC_2X^2) = A(C_3 + C_4X + \dots)$$

$$C_3 + C_4X = \frac{A^{-1} - C_0 - C_1X - C_2X^2 - C_5X^5 - \dots}{X^3}$$

i = 3

$$
\begin{aligned}
(C_3 + C_4X)A(C_3 + C_4X + \ldots) &= \frac{I - (C_0 + C_1X + C_2X^2 + C_5X^5 + \ldots)A}{X^3}(C_3 + C_4X \\
&= \frac{I - (C_0 + C_1X + O(X^5))A}{X^3}(C_3 + C_4X + \ldots) \\
&= \frac{I - (C_0 + C_1X + O(X^5))A(A^{-1} - o(X^2))}{X^6} \\
&= A^{-1}X^{-6} + \textit{negligibile} \\
&= C_6 + C_7X + \textit{negligible}
\end{aligned}
$$

And so on to compute all the terms.

Proof Sketch

- The previous theorem give the quaratic lift

$$Left(Z^{(i)}, 2^{i-1}) = Trunc(Z^{(i-1)} Left(I - AZ^{(i-1)}, 2^{i-1}), 2^{i-1})$$

- The algorithms first compute C_0, C_1
- Then by induction $L^{(j)} = C_{2^j-2}$ and H^{2^j-1}
-

$$H^{(i)} = Trunc(Left(E^{(i-1)} Left(-AH^{(i-1)}, 1), 1), 1)$$

- Using invariance of $Left$ and $Trunc/Left$ commutativity we show :
-

$$R^{(i-1)} = Left(-AH^{(i-1)}, 1) = (I - AZ^{(i-1)})/X^{2^i-1}.$$

-

$$S^{(i-1)} = Left(E^{(i-1)} Left(-AH^{(i-1)}, 1), 1) = Left(Z^{(i-1)} R^{(i-1)}, 2^{i-1} - 1)$$

Sage Math Implementation

```
def HighOrderComp(A,k):

    E = []

    # X addic lifting of A
    C = NaiveLifting(A,X)

    #For convenience
    (col,row)=C[0].dimensions()
    I = [zero_matrix(Pol,ncols =col,nrows=row)]

    #First Step
    L = Invert(C)
    H = Trunc( Multiply(L,Left(Add(I,Opposite(Multiply(C,L,X))),1),X),1)
    E.append(Add(L,mul_power_X(H,1)))

    #Second Step
    for _ in range(k-1):
        L = Trunc(Left(Multiply(E[-1],Left(Opposite(Multiply(C,L,X)),1),X),1),1)
        H = Trunc(Left(Multiply(E[-1],Left(Opposite(Multiply(C,H,X)),1),X),1),1)
        E.append(Add(L,mul_power_X(H,1)))
    return E
```

High Order Components

```
Entrée [483]: HighOrderComp(A,4)
```

```
Out[483]: [[
           [      x^2 + 3*x   x^2 + 3*x + 3   3*x^2 + x + 1]
           [    2*x^2 + 3*x   x^2 + 2*x + 2 4*x^2 + 3*x + 2]
           [  2*x^2 + x + 4             2*x         2*x + 2],

           [  3*x^2 + x + 4   x^2 + 3*x + 4             4*x]
           [2*x^2 + 2*x + 4 3*x^2 + 2*x + 1         3*x + 1]
           [2*x^2 + 2*x + 1   3*x^2 + x + 1   4*x^2 + x + 1]
           ],
            [
           [       2*x + 4         2*x + 2           x + 2]
           [     2*x^2 + 1 2*x^2 + 2*x + 1         x^2 + 4]
           [2*x^2 + 3*x + 2   x^2 + 2*x + 2   2*x^2 + x + 1],

           [      x^2 + 4*x         2*x + 4       x^2 + 3*x]
           [3*x^2 + 3*x + 2 2*x^2 + 4*x + 1     x^2 + x + 3]
           [      4*x^2 + x 4*x^2 + 2*x + 2   x^2 + 4*x + 4]
           ],
            [
           [           2 2*x^2 + 3*x + 2         x^2 + x]
           [         x + 2       3*x^2 + 4         2*x + 1]
           [3*x^2 + 2*x + 1 3*x^2 + 2*x + 1     3*x^2 + 4*x],

           [  x^2 + 3*x + 2   4*x^2 + x + 2 3*x^2 + 4*x + 3]
           [3*x^2 + 4*x + 2   3*x^2 + x + 1   3*x^2 + x + 1]
           [  3*x^2 + x + 2 4*x^2 + 4*x + 4   x^2 + 2*x + 3]
           ],
            [
           [2*x^2 + 4*x + 4 3*x^2 + 4*x + 4 2*x^2 + 4*x + 3]
           [3*x^2 + 3*x + 4 3*x^2 + 2*x + 4         2*x + 4]
           [4*x^2 + 2*x + 3   x^2 + 2*x + 3 4*x^2 + 2*x + 3],

           [3*x^2 + 2*x + 2   x^2 + 3*x + 4 2*x^2 + 2*x + 3]
           [     2*x^2 + x 4*x^2 + 3*x + 3   4*x^2 + x + 1]
           [   3*x^2 + 4*x 4*x^2 + 4*x + 2       x^2 + 4*x]
           ]]
```

Series Solution with Small Right Hand Side

1. $E^{(1)}, E^{(2)}, \ldots, E^{(k-1)} := \texttt{HighOrderComp}[X](A, k-1);$

2. $B := \left[\ b \mid O\ \right]$ where 0 is the $n \times (2^k - 1)$ zero matrix;
   for i **from** $k-1$ **by** -1 **to** 1 **do**
   $\bar{B} :=$ the first $2^k - 2^i$ columns of B;
   $\bar{B} := \mathrm{Left}(-A\,\mathrm{Trunc}(\mathrm{Left}(E^{(i)}\bar{B}, 1), 1), 1);$
   $\bar{B} := \left[\ O \mid \bar{B}\ \right]$ where O is the $n \times 2^i$ zero matrix;
   $B := B + \bar{B};$
   od;
   $B := \mathrm{Trunc}(E^{(1)}B, 2);$

3. # Let $B = \left[\ d_0 \mid 0 \mid d_2 \mid 0 \mid \cdots \mid d_{2^k-2} \mid 0\ \right].$
   $B := d_0 + d_2\,X^2 + \cdots + d_{2^k-2}\,X^{2^k-2};$
   return B

FIGURE – RHS

Correction Sketch

$$Trunc(Z^{(i)}B, 2^i) = Trunc(Z^{(i-1)}[B|\bar{B}], 2^{i-1})[1X^{2^{i-1}}]^{\perp}$$

$$\bar{B} = Left(B - ATrunc(Z^{(i-1)}B, 2^{(i-1)}), 2^{i-1})$$

$$\bar{B} = Left(-ATrunc(Left(E^{(i-1)}B, 1), 1), 1)$$

Algorith Flow

- We first compute efficiently tyhe term of high order in $O(kn^{\theta}d^{1+\epsilon})$
- Then we transform recursively the matrix to halve the size of the problem at the cost of doubling the number of them.
- $r_j = (b - ATrunc(Inverse(A, j)b, j))/X^j$

We give the example for $k = 4$. B start as $[r_0 0 \dots 0]$ then $[r_0 0 \dots r_8 0 \dots 0]$ then $[r_0 0 0 r_4 0 0 r_8 0 0 r_{12} 0 0]$ finaly $[r_0 0 r_2 0 r_4 0 r_6 0 r_8 0 r_{10} 0 r_{12} 0 r_{14} 0]$. The final step compute the solution at order 2 for all those systems with $E^{(1)}$. Finally we obtain the wanted solution ! We could do this in one time benefiting from paralelism !

Complexity

The complexity is

$$O((k + 2^k/n)n^\theta d^{1+\epsilon}).$$

Indeed at each time the numbers of systems is doubled and their size halved. So the last step of the computation dominates in the loop and involve the multiplication E^1 of size $n \times n$ with B of size $n \times 2^k - 1$. So we group the columns of B in $\frac{2^k-1}{n}$ subblock of size n. Then we have a fast multiplication in $n^\theta d^{1+\epsilon}$. So the cost of the loop is $O(2^k n^{\theta-1} d^{1+})$. We just have to add the term relative to high order components which concludes the proof.

applications

minimal kernel bases and linear systems

linear system solving:
given $\mathbf{A} \in \mathbb{K}[X]^{m \times m}$ nonsingular and $\mathbf{v} \in \mathbb{K}[X]^{1 \times m}$
find $\mathbf{u} \in \mathbb{K}[X]^{1 \times m}$ and $g \in \mathbb{K}[X]$ such that

$$\mathbf{u}\mathbf{A} = g\mathbf{v} \quad \text{and} \quad g \text{ has minimal degree.}$$

. the equation has a solution: $\mathbf{u} = g\mathbf{v}\mathbf{A}^{-1}$ with $g = \det(\mathbf{A})$
. but there is often no polynomial solution with $g = 1$
. target complexity? (recall that $\det(\mathbf{A})\mathbf{A}^{-1}$ can have degree $\approx m \deg(\mathbf{A})$)
. propose an algorithm based on a kernel computation

compute $[\mathbf{u} \quad g] \in \mathbb{K}[X]^{1 \times (m+1)}$ kernel basis of $\mathbf{F} = \begin{bmatrix} \mathbf{A} \\ -\mathbf{v} \end{bmatrix} \in \mathbb{K}[X]^{(m+1) \times m}$

- using the shift $\mathbf{s} = (\text{rdeg}(\mathbf{A}), \deg(\mathbf{v}))$
- complexity $\tilde{O}(m^\omega \max(\deg(\mathbf{A}), \deg(\mathbf{v})))$ → in fact: $\max(\deg(\mathbf{A}), \frac{\deg(\mathbf{v})}{m})$
- $\mathbf{u}, g$ is a solution to the equation $\mathbf{u}\mathbf{A} = g\mathbf{v}$
- minimality of $\deg(g)$ follows from *basis* of $\mathcal{K}(\mathbf{F})$

46

FIGURE – A Slide of V.Neiger Lecture

Series Solution

What if the right hand side is not small ? Then encode the "fat" entry in a matrix with more columns. Then apply the same approach as before !

How to Use this in Practice ?

- Previous algorithms are deterministic.
- Choose a random X of small degrees such that $\det A \perp X$.
- Use the algorihm with the resulting X adic expansion.
- Convert back to polynomial matrices expression.

The algorithm presented are hence probabilitic in practice due to the choice of the random X.