Numerical Methods for CSE

Cumulative Sums and Reusing Intermediate Results

(Explanation of Exercise 1.7)

Giuseppe Accaputo
g@accaputo.ch

October 3, 2016

In this document I will guide you through the efficient implementation of the matrix-vector product $\mathbf{y} = \mathbf{A}\mathbf{x}$ as shown in exercise 1.7.b [1], where $(\mathbf{A})_{i,j} = a_{i,j} = \min\{i, j\}$ for $i, j = 1, \ldots, n$.

Notation

Throughout this document I will be using the following notation in mathematical formulas:

$\mathbf{x}$:	Column vector (small letter, bold)
$\mathbf{x}^T$:	Row vector (small letter, bold, transposed)
$\mathbf{A}$:	Matrix (large letter, bold)

Step 1: Visualize the Matrix A

From the given definition $(\mathbf{A})_{i,j} = a_{i,j} = \min\{i, j\}$ for $i, j = 1, \ldots, n$ it follows that $\mathbf{A}$ has the form

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 & 1 \\ 1 & 2 & 2 & \dots & 2 & 2 \\ 1 & 2 & 3 & \dots & 3 & 3 \\ \vdots & & & & & \vdots \\ 1 & 2 & 3 & \dots & n-1 & n-1 \\ 1 & 2 & 3 & \dots & n-1 & n \end{pmatrix} \tag{1}$$

Step 2: Visualize the Matrix-Vector Product $\mathbf{y} = \mathbf{A}\mathbf{x}$

Using $\mathbf{A}$ from Eq. 1, the matrix-vector product $\mathbf{y} = \mathbf{A}\mathbf{x}$ looks as follows:

$$\mathbf{A}\mathbf{x} = \mathbf{y} = \begin{pmatrix} 1x_1 + 1x_2 + 1x_3 + \dots + 1x_{n-2} + 1x_{n-1} + 1x_n \\ 1x_1 + 2x_2 + 2x_3 + \dots + 2x_{n-2} + 2x_{n-1} + 2x_n \\ 1x_1 + 2x_2 + 3x_3 + \dots + 3x_{n-2} + 3x_{n-1} + 3x_n \\ \vdots \\ 1x_1 + 2x_2 + 3x_3 + \dots + (n-2)x_{n-2} + (n-2)x_{n-1} + (n-2)x_n \\ 1x_1 + 2x_2 + 3x_3 + \dots + (n-2)x_{n-2} + (n-1)x_{n-1} + (n-1)x_n \\ 1x_1 + 2x_2 + 3x_3 + \dots + (n-2)x_{n-2} + (n-1)x_{n-1} + nx_n \end{pmatrix} \tag{2}$$

Cumulative Sums and Reusing Intermediate Results

If we further analyze the resulting vector $\mathbf{y}$ in Eq. 2, we can see that each entry consists of partial sums [2] (colored in black in Eq. 3) of the sequence $\{i\,x_i\}_{i=1}^n = x_1,\ 2\,x_2,\ 3\,x_3,\ \dots,\ n\,x_n$:

$$\mathbf{A}\mathbf{x} = \mathbf{y} = \begin{pmatrix} 1x_1 + 1x_2 + 1x_3 + \dots + 1x_{n-2} + 1x_{n-1} + 1x_n \\ 1x_1 + 2x_2 + 2x_3 + \dots + 2x_{n-2} + 2x_{n-1} + 2x_n \\ 1x_1 + 2x_2 + 3x_3 + \dots + 3x_{n-2} + 3x_{n-1} + 3x_n \\ \vdots \\ 1x_1 + 2x_2 + 3x_3 + \dots + (n-2)x_{n-2} + (n-2)x_{n-1} + (n-2)x_n \\ 1x_1 + 2x_2 + 3x_3 + \dots + (n-2)x_{n-2} + (n-1)x_{n-1} + (n-1)x_n \\ 1x_1 + 2x_2 + 3x_3 + \dots + (n-2)x_{n-2} + (n-1)x_{n-1} + nx_n \end{pmatrix} \tag{3}$$

The sums in the blackened part of $\mathbf{y}$ in Eq. 3 are actually part of the *cumulative sum* [3] of the sequence $\{i\,x_i\}_{i=1}^n$. A cumulative sum is a sequence of partial sums of a sequence. In the case of the sequence $\{i\,x_i\}_{i=1}^n$ the cumulative sum is defined as

$$x_1,\ x_1 + 2x_2,\ \underbrace{x_1 + 2x_2 + 3x_3}_{\text{partial sum}},\ \dots,\ x_1 + 2x_2 + \dots + (n-1)x_{n-1} + nx_n \tag{4}$$

As we know from the lecture [4], complexity can sometimes be reduced by reusing intermediate results. By having a closer look at Eq. 3, we can see that some partial sums reappear again in multiple components of the result vector $\mathbf{y}$, e.g. $x_1 + 2x_2$ reappears in the partial sum $x_1 + 2x_2 + 3x_3$, and so on. Thus, we can try to reuse each partial sum in the calculation of the next partial sum, which can be accomplished by defining the cumulative sum of $\{i\, x_i\}_{i=1}^{n}$ recursively as follows:

$$\boxed{w_1 = x_1, \quad w_j = w_{j-1} + j x_j \quad \text{for} \quad j = 2, \ldots, n} \tag{5}$$

In a next step we try to reuse intermediate results for the gray part of Eq. 3. For one, we define the vector $\mathbf{w}$ as follows:

$$\mathbf{w} = \begin{pmatrix} x_1 \\ w_1 + 2x_2 \\ w_2 + 3x_3 \\ \vdots \\ w_{n-1} + n x_n \end{pmatrix} \tag{6}$$

If we rewrite Eq. 3 as follows

$$\mathbf{y} = \mathbf{w} + \underbrace{\begin{pmatrix} 1x_2 + 1x_3 + \cdots + 1x_{n-1} + 1x_n \\ 2x_3 + \cdots + 2x_{n-1} + 2x_n \\ \vdots \\ (n-2)x_{n-1} + (n-2)x_n \\ (n-1)x_n \\ 0 \end{pmatrix}}_{:=\mathbf{u}} \tag{7}$$

we can see that the vector $\mathbf{u}$ on the right can be rewritten as

$$\mathbf{u} = \begin{pmatrix} x_2 + x_3 + \cdots + x_{n-1} + x_n \\ 2(x_3 + \cdots + x_{n-1} + x_n) \\ \vdots \\ (n-2)(x_{n-1} + x_n) \\ (n-1)x_n \\ 0 \end{pmatrix} \tag{8}$$

and thus we observe that the components of $\mathbf{u}$ resemble a cumulative sum of the *backward* sequence $\{x_i\}_{i=n}^{2} = x_n, x_{n-1}, \ldots, x_2$ (each partial sum is multiplied by a constant factor):

$$\begin{aligned}
u_1 &= 1(x_2 + x_3 + \cdots + x_{n-1} + x_n) \\
u_2 &= 2(x_3 + \cdots + x_{n-1} + x_n) \\
&\vdots \\
u_{n-3} &= (n-3)(x_{n-2} + x_{n-1} + x_n) \\
u_{n-2} &= (n-2)(x_{n-1} + x_n) \\
u_{n-1} &= (n-1)x_n
\end{aligned} \tag{9}$$

As we can see, the factor that multiplies the partial sums of the cumulative sum of $\{x_i\}_{i=n}^{2}$ in each component is just the component index $j = 1, \ldots, n-1$. Further, let v_j be the the j-th partial sum of the cumulative sum of $\{x_i\}_{i=n}^{2}$ (without the multiplying factor), i.e.

$$v_j = \sum_{k=j+1}^{n} x_k \tag{10}$$

giving

$$\mathbf{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_{n-1} \end{pmatrix} \tag{11}$$

with components

$$\begin{aligned}
v_1 &= x_2 + x_3 + \cdots + x_{n-1} + x_n \\
v_2 &= x_3 + \cdots + x_{n-1} + x_n \\
&\vdots \\
v_{n-3} &= x_{n-2} + x_{n-1} + x_n \\
v_{n-2} &= x_{n-1} + x_n \\
v_{n-1} &= x_n
\end{aligned}$$

Since the partial sum v_j contains the complete precedent partial sum v_{j+1} (e.g. $v_{n-3} = v_{n-2} + x_{n-2}$), v_j can be defined recursively with

$$v_{n-1} = x_n, \quad v_j = v_{j+1} + x_{j+1} \quad \text{for} \quad j = n-2, \ldots, 1 \tag{12}$$

Thus, for a component u_j of the vector $\mathbf{u}$ we get

$$u_j = j\,v_j \quad \text{for} \quad j = 1, \ldots, n-1 \tag{13}$$

and finally for our result vector $\mathbf{y}$ we would have

$$y_n = w_n, \quad ,y_j = w_j + u_j = w_j + j\,v_j \quad \text{for} \quad j = 1, \ldots, n-1 \tag{14}$$

If $\odot$ denotes the componentwise multiplication of two column-vectors, i.e.

$$\mathbf{a} \odot \mathbf{b} = \begin{pmatrix} a_1 \cdot b_1 \\ a_2 \cdot b_2 \\ \vdots \\ a_n \cdot b_n \end{pmatrix} \tag{15}$$

then $\mathbf{u}$ can be defined as

$$\mathbf{u} = \begin{pmatrix} 1 \\ 2 \\ \vdots \\ n-1 \end{pmatrix} \odot \mathbf{v} \quad . \tag{16}$$

If we would leave it at Eq. 14, we would need three separate `for`-loops in our code; one `for`-loop to initialize $\mathbf{w}$ (Eq. 5), another one to initialize $\mathbf{v}$ (Eq. 16) and a final one to calculate $\mathbf{y}$ (Eq. 14). This would be fine in regard to the algorithm's complexity, since we managed to move from $O(n^2)$ to $O(n)$, but we would still need three `for`-loops in our implementation.

A more efficient implementation of the algorithm can be achieved by initializing both $\mathbf{w}$ and $\mathbf{v}$ within the same `for`-loop and thus removing one `for`-loop. This can be accomplished by reversing the order on how we define $\mathbf{v}$ recursively; instead of going from $j = n-2, \ldots, 1$ we now define a $\tilde{v}_j$ recursively for $j = 2, \ldots, n-1$, starting with $\tilde{v}_1 = x_n$:

$$\boxed{\tilde{v}_1 = x_n, \quad \tilde{v}_j = \tilde{v}_{j-1} + x_{n-j+1} \quad \text{for} \quad j = 2, \ldots, n-1} \tag{17}$$

resulting in

$$\begin{aligned} \tilde{v}_1 &= x_n \\ \tilde{v}_2 &= x_{n-1} + x_n \\ &\vdots \\ \tilde{v}_{n-2} &= x_3 + \cdots + x_{n-1} + x_n \\ \tilde{v}_{n-1} &= x_2 + x_3 + \cdots + x_{n-1} + x_n \end{aligned}$$

which is the reversed version of $\mathbf{v}$ in Eq. 12.

We now have to update the calculation of $\mathbf{y}$ shown in Eq. 14 since we are now working with $\tilde{v}_j$ from Eq. 17, thus resulting in

$$\boxed{y_n = w_n, \quad y_j = w_j + j\,\tilde{v}_{n-j} \quad \text{for} \quad j = 1, \ldots, n-1} \tag{18}$$

The initialization of both $\mathbf{w}$ (Eq. 5) and $\tilde{\mathbf{v}}$ (Eq. 17), and the final calculation of $\mathbf{y}$ (Eq. 18) can now be done with two `for`-loops:

```
v(0) = x(n-1);
w(0) = x(0);

for(unsigned int j = 1; j < n; ++j) {
    v(j) = v(j-1) + x(n-j-1);
    w(j) = w(j-1) + (j+1)*x(j);
}
for(unsigned int j = 0; j < n-1; ++j) {
    y(j) = w(j) + v(n-j-2)*(j+1);
}
y(n-1) = w(n-1);
```

It is important to note that x_{n-j+1} in Eq. 17 is accessed with `x(n-j-1)` ↪ in the code and $\tilde{v}_{n-j}$ in Eq. 18 with `v(n-j-2)`, both depending on the different definitions of the variable `j` used in both `for`-loops respectively.

References

[1] R. Hiptmair, "Numerical methods for computational science and engineering, homework problems." https://www.sam.math.ethz.ch/~grsam/HS16/NumCSE/NCSEProblems.pdf, 2016.

[2] E. W. Weisstein, "Partial sum. From MathWorld—A Wolfram Web Resource." http://mathworld.wolfram.com/PartialSum.html.

[3] E. W. Weisstein, "Cumulative sum. From MathWorld—A Wolfram Web Resource." http://mathworld.wolfram.com/Projection.html.

[4] R. Hiptmair, "Numerical methods for computational science and engineering." https://www.sam.math.ethz.ch/~grsam/HS16/NumCSE/NumCSE16.pdf, 2016.