Computing the Anomalous Dimensions

Evolution of the Nucleon Wave Function

W. SCHROERS

Institut für Theoretische Physik II,

Ruhr-Universität Bochum, D-44780 Bochum, Germany

January 16, 1999

We give a short account of the technical details regarding the implementation of the anomalous dimensions program. The reader is assumed to be acquainted with the literature on this subject.

Contents

1 Introduction

This report gives a short introduction into the technical details of the determination of the anomalous dimensions appearing in the evolution of the nucleon wave function. The reader is assumed to be familiar with the literature on this subject. The technical details are given in section 2, while the details on the program implementation are given in section 3.

2 Diagonalizing the evolution kernel

Following [1] we choose as a basis for the solution of the Brodsky-Lepage evolution kernel the (anti-)symmetrized Appell polynomials which are defined by

$$\begin{aligned}\tilde{F}(x_1,x_3) &= x_1x_3(1-x_1-x_3)\frac{\Gamma(2+m)\Gamma(2+n)}{2\left(\Gamma(2)\right)^2}\\ &\quad\times\frac{\partial^{m+n}}{\partial x_1^m\partial x_3^n \pm \partial x_1^n x_3^m}\left(x_1^{m+1}x_3^{n+1}(1-x_1-x_3)^{m+n+1}\right)\,.\end{aligned}$$

The $+$-sign in the differential operator applies to $m \geq n$, the $-$-sign applies to $m < n$. The solution of the Eigenvalue equation can be greatly simplified, because the operator $\hat{V}$ is block-diagonal in the basis of Appell polynomials. Furthermore, it can be splitted in a symmetric and an antisymmetric part which are found to be almost degenerate [1, 2]. Thus, for the investigation of the large M-limit it is sufficient to consider only either the symmetric or the antisymmetric functions. In the present work we chose to work with the antisymmetric functions.

The basis of polynomials is complete but of course not unique. Due to the remaining degeneracy we can choose a basis of functions which contains $(M+2)/2$ linear independant polynomials. We denote the basis in this space by $\tilde{F}'_q \equiv \langle\vec{x}|q\rangle$. Again, this choice is not unique and one could use this ambiguity to find functions which minimize the numerical effort. This observation, however, won't help simplifying the original problem concerning the diagonalization of the Brodsky-Lepage kernel $\hat{V}$ since the operator is still given in the basis of simple polynomials $x_1^m x_3^n \equiv \langle\vec{x}|mn\rangle$. The matrices for the change of basis would still have to contain the whole information. Hence, if one is interested in an exact determination of the solution the complexity of the problem is not affected.

After some algebra one finds the following expression for the transformation from the $|mn\rangle$-basis to the antisymmetric Appell polynomials $|q\rangle$:

$$\begin{aligned}2\tilde{F}_{m,n}(x_1,x_3) &= \frac{\Gamma(m+2)\Gamma(n+2)}{\left(\Gamma(2)\right)^2}\sum_{i=0}^{m}\sum_{k=0}^{n}\sum_{o=0}^{m+n-k-i}\sum_{p=0}^{o}\binom{n}{k}\binom{m}{i}\binom{o}{p}\\ &\quad\times\binom{m+n-k-i}{o}(-1)^{i+k+o}\frac{(n+1)!}{(k+1)}\frac{(m+n+1)!}{(m+n-k+1)!}\\ &\quad\times\frac{(m+1)!}{(i+1)!}\frac{(m+n-k+1)!}{(m+n-k-i+1)!}x_1^{i+o-p}x_3^{k+p}\,. \qquad (1)\end{aligned}$$

In this expression one has to choose $m, n \geq 0$ and $m + n \leq M$.
We used the following expression for the polynomials $\tilde{\mathcal{F}}'_q(x_1, x_3) \equiv \langle \vec{x} | q \rangle$:

$$\tilde{\mathcal{F}}'_q(x_1, x_3) = \frac{1}{2} \left(\tilde{F}_{q-1,M+1-q}(x_1, x_3) - \tilde{F}_{M+1-q,q-1}(x_1, x_3) \right) .$$

From these expressions one arrives — after applying the Gram-Schmidt orthogonalization scheme — at the polynomials $\tilde{F}'_q(x_1, x_3)$. Using the notation from above these factors can also be written $\tilde{F}'_q(x_1, x_3) = \sum_{m,n} \langle \vec{x} | mn \rangle \langle mn | q \rangle$.
The evolution kernel of Brodsky and Lepage [3] in the $|kl\rangle$-basis is given by

$$\frac{\langle \vec{x} | \hat{V} | kl \rangle}{2x_1 x_3 (1 - x_1 - x_3)} = \frac{1}{2} \sum_{i,j} \langle \vec{x} | ij \rangle \langle ij | \hat{V} | kl \rangle .$$

Transforming the operator to our new basis yields

$$\frac{\langle q' | \hat{V} | q \rangle}{2x_1 x_3 (1 - x_1 - x_3)} = \sum_{i,j} \sum_{k,l} \langle \vec{x} | q \rangle \langle q | ij \rangle \langle ij | \hat{V} | kl \rangle \langle kl | q' \rangle .$$

In order to arrive at the desired eigenvalues of $\hat{V}$ one has to diagonalize the matrix

$$\mathcal{G}_{q'q} = \int_0^1 [dx] x_1 x_3 (1 - x_1 - x_3) \langle q' | \hat{V} | q \rangle ,$$

which turns out to be highly unbalanced. Consequently, the diagonalization of the matrix requires a numerical effort of order M^3; this effort, however, is only of minor importance since the determination of $\hat{V}$ in the proposed basis of anti-symmetrized Appell polynomials is a problem of the order of M^5. The complexity results from the fact that the transformation matrix contains 4 nested sums and one needs to calculate the M^2 matrix elements of $\langle q' | \hat{V} | q \rangle$ from it. One sum, however, can be performed explicitely by hand, but the remaining sums must be evaluated in nested loops.
For the actual computation we have chosen to implement the algorithm for computing the transformation matrix and the kernel $\langle ij | \hat{V} | kl \rangle$ on the basis of the GNU Multiple Precision library [4]. The computations can either be performed using floating point arithmetics or using exact rational numbers. For higher orders floating point numbers are faster, but one must have an idea how accurate the numbers must be (since the FP-numbers are not exact but have at least a given accuracy) for the computation to succeed. The program has been written in C and is available at [5].
For the final determination of the basis and for the diagonalization we simply read in the output from the C program into MATHEMATICA and use its routines for equation solving and diagonalization. The numerical effort for computing the spectrum on a standard PC is thus of the order of 1 hour for orders of the magnitude of $M \approx 100$. The computations are even faster if one uses floating point number with intermediate accuracy. This should be far sufficient for extracting the asymptotic behavior. (Please note that since the computation time scales at least

with M^5 the computation time will increase by at least a factor of 32 if you double the order. Taking into account that also the numbers get larger the time increase should be even more dramatic.)

In table 1 we show the highest and lowest eigenvalues for several orders M.

M	λ_{max}	λ_{min}
1	-0.6666	-0.6666
2	-2.3333	-2.3333
3	-3.73574	-2.58092
4	-4.81464	-3.71869
5	-5.75609	-3.87204
6	-6.57395	-4.74452
7	-7.29344	-4.84751
8	-7.93578	-5.5595
9	-8.51551	-5.6317
10	-9.04369	-6.2357
15	-11.1501	-7.34502
20	-12.7037	-8.53412
25	-13.9358	-9.17848
100	-21.9168	-14.5370
150	-24.3105	-16.1187
200	-26.0169	-17.2488
250	-27.344	-18.1288
300	-28.43	-18.8496
400	-30.1461	-19.9895

Table 1: The largest and smallest eigenvalues (λ_{max} and λ_{min}) of the BL-kernel for selected orders M.

3 Program implementation

In the following we use some variables directly in the language of the implementation: `kx` is the order of the Appell polynomials used, `okx` is the number of terms of the form $x_1^i x_3^j$ (clearly we have $\texttt{okx} = (\texttt{kx}+1)(\texttt{kx}+2)/2$). `ca` is the number of eigenvalues; in the case of the antisymmetric spectrum it is `Floor` $[(\texttt{kx}+1)/2]$. Since the dimension of the space of polynomials $x_1^i x_3^j$ is `okx` we need to map the numbers $0 \le i, j \le$ `kx` onto the `okx`. This choice is not unique, but we adopt (following the work of M. Bergmann [1]) the structure $\texttt{index}(i,j) = (i+j+1)(i+j)/2 + j + 1$. This choice, however, is not significant for the following analysis.

For the calculation of the eigenvalue spectrum one needs two ingredients: the operator

$$\langle k,l|\hat{V}|m,n\rangle \equiv rv\,[1\ldots\texttt{okx},1\ldots\texttt{okx}]\ ,$$

which, according to what we defined above, is an `okx`$\times$`okx` sized matrix array. Hence, its size is of the order `kx`^{4}. Furthermore, we need the coefficient matrix of the Appell polynomials, the matrix $\langle q|m,n\rangle \equiv \texttt{rm}\,[1\ldots\texttt{ca},1\ldots\texttt{okx}]$ which has a size of the order `kx`^{3}.
After applying a symmetrization on the eq. (1) we end up with the expression for $\tilde{\mathcal{F}}'_q(x_1,x_3) = \langle m,n|q\rangle$:

$$\tilde{\mathcal{F}}'_q(x_1,x_3) = \frac{1}{2}\left(\tilde{F}_{q-1,M+1-q}(x_1,x_3) - \tilde{F}_{M+1-q,q-1}(x_1,x_3)\right)\ .$$

This expression yields the matrix `rm`. The practical approach to implement this procedure is of course not to loop over m and n and then evaluate the sums but rather to loop over the first (Appell polynomial) index of `rm` and then to add the corresponding sum elements to the correct place in the array `rm`:

```
rm (:,:) = 0.0
do qp = 1, ca
  m = qp-1 ; n = kx+1-qp
  do b = 0, m ; do a = 0, n ; do c = 0, kx-a-b ; do d = 0, c
    temp = factorial(qp-1)*factorial(kx+1-qp)*factorial(kx-qp+2)* &
   &       factorial(kx+1)*factorial(qp)/(factorial(b))**2/       &
   &       (factorial(a))**2/(a+1)/(b+1)/factorial(qp-b-1)/       &
   &       factorial(kx+1-qp-a)/factorial(kx-a-b-c)/factorial(d)/ &
   &       factorial(c-d)/(kx-a-b+1)*(-1)**(a+b+c)
    rm(qp, index(b+c-d,a+d)) += temp
    rm(qp, index(a+d,b+c-d)) -= temp
    ! Please note that the polynomial is antisymmetric!
  enddo
enddo
rm *= 1/2/gamma(1+qp)/gamma(3+kx-qp)
```

From this implementation it is obvious that the evaluation of the polynomial coefficients requires a numerical effort which is of the order `kx`^{5}. It may be possible that one can exploit recursion relations to further simplify this expression (although we don't know how), but one clearly cannot go below `kx`^{3} since this is the size of the coefficient matrix. It may furthermore turn out to be possible that a different choice of polynomials allows to make this more efficient, but again we don't know how. Nonetheless, the `kx`^{3} seems to be a lower bound on the effort which is by far not reached by our implementation (leaving room for different kinds of optimizations).
The implementation of the computation of `rv` is also straightforward; however one has to keep in mind some special conditions for evaluation:

$$\texttt{rv}\,[k,l,m,n] \;=\; \delta_{mk}\delta_{nl}\left(\frac{1}{(k+1)(k+2)} + \frac{1}{(l+1)(l+2)} - 3\left(\sum_{j=2}^{k+1}\frac{1}{j} + \sum_{j=2}^{l+1}\frac{1}{j}\right)\right)$$

$$
\begin{aligned}
&+\frac{m+2}{(k+2)(k-m)}\binom{k-m}{n-l}(-1)^{n-l} \quad \text{for (i)}\\
&+\frac{n+2}{(l+2)(l-n)}\binom{l-n}{m-k}(-1)^{m-k} \quad \text{for (ii)}\\
&-\delta_{m-k,l-n}\cdot\{\\
&\quad \sum_{j=0}^{l-n}\binom{l}{j}\binom{l-j}{n}(-1)^j\sum_{z=2}^{j+k+1}\frac{1}{z} \quad \text{for (iii)}\\
&\quad +\sum_{j=0}^{n-l}\binom{k}{j}\binom{k-j}{m}(-1)^j\sum_{z=2}^{j+l+1}\frac{1}{z}\Big\} \quad \text{for (iv)}\,.
\end{aligned}
$$

The conditions are defined as follows:

$$
\begin{aligned}
&\text{(i)} && k>0, m<k, n\le l+k\,,\\
&\text{(ii)} && l>0, n<l, m\le k+l\,,\\
&\text{(iii)} && l>0, n<l, m\le k+l\,,\\
&\text{(iv)} && l>0, n<l, m\le k+l\,.
\end{aligned}
$$

One could in principle cache this array since for the matrix multiplication (see below) it will be needed more than once; however this multiplication is not leading in CPU time consumption, but since the memory consumption is of the order $\mathtt{kx}^4$ it would clearly be leading in memory consumption. Thus, it turns out to be more efficient *not* to cache `rv` but to evaluate an element each time it is needed.
With the matrices `rm` and `rv` at hand we can in principle determine the eigenvalues by setting

$$
\begin{aligned}
\mathtt{rb} &= (\mathtt{rm}\cdot\mathtt{rv})^T\\
\mathtt{rr} &= (\mathtt{rm})^T\,.
\end{aligned}
$$

Here the dots denote matrix multiplications and T means transposition - now `rb` is a $\mathtt{ca}\times\mathtt{okx}$ matrix (size of order $\mathtt{kx}^3$) and `rr` is a matrix of the same size) and then solve the $k=1\ldots\mathtt{ca}$ systems of linear matrix equations given by

$$\mathtt{rr}\cdot\mathtt{res}_k=\mathtt{rb}_k\,, \tag{2}$$

where $\mathtt{rb}_k$ means the kth column of `rb` (which is a column vector of size `okx`). The vector $\mathtt{res}_k$ is a column vector of size`ca` and after solving all `ca` systems in eq. (2) we can write all column vectors `res` as the columns of a matrix and then diagonalize it; this yields the eigenvalues we have been looking after:

$$\mathtt{Spectrum}=\mathtt{Eigenvalues}\,[\mathtt{res}]\,. \tag{3}$$

Focus again system (2): It is a system of `okx` equations but only `ca` unknowns. It is clearly overdetermined and can (in general) only be solved if the matrices have been evaluated exactly and we still keep infinite precision.

The first thing one can do to render the system (2) easier to evaluate is to consider instead

$$\begin{aligned} \mathtt{rb} &= \mathtt{rm} \cdot (\mathtt{rm} \cdot \mathtt{rv})^T \\ \mathtt{rr} &= \mathtt{rm} \cdot (\mathtt{rm})^T , \end{aligned} \tag{4}$$

and then solving again for the equation

$$\mathtt{rr} \cdot \mathtt{res}_k = \mathtt{rb}_k , \tag{5}$$

which is now a system of `ca` equations and `ca` unknowns. This is far better suited for calculation since we now evade the requirement of infinite precision.
However, we still need to calculate the whole matrix `rv`. A better method would be not to calculate the whole matrix `rb` but only a subset of rows (since the system is heavily overdetermined). Only in this way both the memory requirements and the computation time can be kept at a minimum. Sadly, we still need to calculate the whole matrix `rm` for the inner multiplication with `rv`. For the matrix `rr` we don't need the whole matrix `rm` anymore. But we are still left with the decision which rows we want to choose (and we should better choose those that are really linearly independant). The choice we made was to take elements of the form $|k, k+1\rangle$.
After making this choice we are able to calculate the matrix product $\mathtt{rm} \cdot \mathtt{rv}$ directly inside the interior loop: Since now we only need the matrix elements $\mathtt{rv}\,[1 \ldots \mathtt{okx}, 1 \ldots \mathtt{ca}] \equiv \langle i, j|\hat{V}|k+1, k\rangle$ and we contract the first index with the corresponding index of the matrix `rm`, the object we only need to *store* is the matrix `rb`, which now has the size

$$\mathtt{rb}\,[1 \ldots \mathtt{ca}, 1 \ldots \mathtt{ca}] \ .$$

Furthermore the size of the matrix `rr` is also only $\mathtt{rr}\,[1 \ldots \mathtt{ca}, 1 \ldots \mathtt{ca}]$, so the whole storage requirement of the program is just of order $\mathtt{kx}^2$. However, although we got rid of the need to evaluate the whole matrix `rv` (which requires an effort of order $\mathtt{okx}^2 = \mathtt{kx}^4$) we still have to compute the whole matrix $\mathtt{rm}\,[1 \ldots \mathtt{ca}, 1 \ldots \mathtt{okx}]$ (albeit we don't have to store it, but we can calculate the required contraction inside the inner loop of the above program instead).
One further comment: the matrices `rr` and `rb` are highly unbalanced (due to the factorials which sum up to pretty high numbers at some places). So for large orders ($\mathtt{kx} > 30$) a high accuracy is still required for the linear system (2) due to the need to add and subtract factors differing by huge orders of magnitude. For orders of about $\mathtt{kx} \simeq 400$ several hundreds of digits are required. Although the calculations could in principle be done using exact (infinite precision) rational numbers this is less efficient than using approximate floating point numbers. But even in this case the required accuracy increases with order resulting in an increase in required computing time which is even worse than $\mathtt{kx}^5$. Here is again much room for improvement, because there could be polynomial basises which don't exhibit this factorial growth of their coefficients. One should always keep in mind that efficiency in computation was *not* the reason for the usage of Appell polynomials.

References

[1] M. Bergmann, Dissertation, Ruhr-Universität Bochum, 1994. RX-1518.

[2] M. Bergmann and N. G. Stefanis, RUB-TPII-11-94 *Invited talk at International Conference on Quark Confinement and the Hadron Spectrum, Como, Italy, 20-24 Jun 1994*.

[3] G. P. Lepage and S. J. Brodsky, Phys. Rev. D **22** (1980) 2157.

[4] Homepage of the GMP-library: `http://www.swox.com/gmp/`.

[5] Wolfi's Physics Page: `www.feldtheorie.de`.