Supplementary Material: Program synthesis of sparse algorithms for wavefunction and energy prediction in grid-based quantum simulations

Scott Habershon*

Department of Chemistry, University of Warwick, Coventry, CV4 7AL, United Kingdom

E-mail: S.Habershon@warwick.ac.uk

This Supplementary Material document contains:

- Tables of input function definitions (Table 1) and internal function definitions (Table 2) used in PS simulations;

- Results and working equations for algorithms optimized for specific grid-sizes.
1 PS function definitions

Table 1: Definition of input vectors $y$ and matrices $M$ defined for the $n_i = 11$ input terminals used in PS simulations. Here, $V$ is the PES evaluated on the input coordinate grid. Explicit elements of the matrix are written as $M_{ij}$, with $M_{ii}$ representing the matrix diagonal elements.

| Input index | Input vector | Input matrix |
|-------------|--------------|--------------|
| 1           | $y = 1$      | $M = I$,     |
| 2           | $y = 1$      | $M_{ij} = 1$, for all $i, j$ |
| 3           | $y = 1$      | $M_{ii} = 0$, $M_{ij} = 1$, for $j \neq i$ |
| 4           | $y = 0$      | $M = I$,     |
| 5           | $y = 0$      | $M_{ij} = 1$, for all $i, j$ |
| 6           | $y = 0$      | $M_{ii} = 0$, $M_{ij} = 1$, for $j \neq i$ |
| 7           | $y = V$      | $M = I$,     |
| 8           | $y = V$      | $M_{ij} = 1$, for all $i, j$ |
| 9           | $y = V$      | $M_{ii} = 0$, $M_{ij} = 1$, for $j \neq i$ |
| 10          | $y = 1$      | $M_{ij} = 1$, if $j = i$ or $j = i \pm 1$, $M_{ij} = 0$ otherwise. |
| 11          | $y = V$      | $M_{ij} = 1$, if $j = i$ or $j = i \pm 1$, $M_{ij} = 0$ otherwise. |
Table 2: Internal functions set used in PS simulations. Here, \( y \) indicates the current workspace vector, and \( \mathbf{M} \) indicates the workspace matrix. Explicit elements of the vector and matrix are written \( y_i \) and \( M_{ij} \) respectively, with \( M_{ii} \) representing the matrix diagonal elements. The elements \( x_i \) represents the position of the \( i \)-th grid-point in the uniform grid. Except where indicated, all operations act in an element-wise manner on all entries in the workspace vector or matrix. We note that \( I_0(x) \) is the Modified Bessel Function (first kind, zero order), as implemented in \texttt{numpy}. Considering all combinations of constants, \( c = [m, 2, 3, \pi, 4, L] \), and functions, we have a total of \( n_r = 134 \) possible operations at each internal code line. We note that this list of functions is clearly not exhaustive and is somewhat arbitrarily chosen based on trial-and-error investigations; however, the results of the main article demonstrate that this set is sufficient to generate new codes which perform very well in eigenfunction prediction when compared to standard DVR schemes.

| Operation 1 | Operation 2 | Operation 3 |
|-------------|-------------|-------------|
| \( y \rightarrow y \times c \) | \( y \rightarrow y + c \) | \( M_{ij} \rightarrow M_{ij} - c \) |
| \( y \rightarrow y/c \) | \( y \rightarrow -y \) | \( M_{ij} \rightarrow \frac{M_{ij}}{(y_i - j)} \), \( i \neq j \) |
| \( M_{ij} \rightarrow M_{ij} \times (i - j), i \neq j \) | \( M_{ij} \rightarrow M_{ij} \times (-1)^i, i \neq j \) | \( M_{ij} \rightarrow M_{ij} \times e^{(x_i-x_j)^2} \) |
| \( M_{ij} \rightarrow \sin(M_{ij}) \) | \( M_{ij} \rightarrow \cos(M_{ij}) \) | \( M_{ij} \rightarrow M_{ij} \times e^{M_{ij}} \) |
| \( M_{ij} \rightarrow M_{ij} \times e^{-(x_i-x_j)^2} \) | \( M_{ij} \rightarrow e^{M_{ij}} \) | \( M_{ij} \rightarrow M_{ij} \times M_{ij} \) |
| \( \mathbf{M} \rightarrow M \times c \) | \( \mathbf{M} \rightarrow \mathbf{M} + c \) | \( \mathbf{M} \rightarrow -\mathbf{M} \) |
| \( M_{ii} \rightarrow M_{ii} + c \) | \( M_{ii} \rightarrow M_{ii} - c \) | \( M_{ii} \rightarrow M_{ii} \times c \) |
| \( \mathbf{M} \rightarrow \ln(\mathbf{M}) \) | \( \mathbf{M} \rightarrow \ln(M_{ii}) \) | \( M_{ii} \rightarrow M_{ii} \times y_i \) |
| \( M_{ij} \rightarrow M_{ij} + y_i \) | \( M_{ij} \rightarrow M_{ij} - y_i \) | \( M_{ii} \rightarrow M_{ii} + \bar{M}_{ii} \) |
| \( M_{ij} \rightarrow e^{M_{ij}} \) | \( M_{ij} \rightarrow e^{M_{ij}} \times e^{(x_i-x_j)^2} \) | \( M_{ij} \rightarrow M_{ij} \times e^{M_{ij}} \) |
| \( M_{ij} \rightarrow M_{ij} \times e^{M_{ij}} \times e^{-(x_i-x_j)^2} \) | \( M_{ij} \rightarrow \frac{M_{ij}}{(x_i-x_j)^3}, i \neq j \) | \( M_{ij} \rightarrow M_{ij} \times (e^{M_{ij}})^2 \) |
| \( M_{ij} \rightarrow \tanh(M_{ij}) \) | \( M_{ij} \rightarrow \tanh(M_{ij}) \times \bar{x}^4 \) | \( M_{ij} \rightarrow \tanh(M_{ij}) \times (x_i-x_j)^2 \) |
| \( M_{ij} \rightarrow e^{M_{ij}} \) | \( M_{ij} \rightarrow e^{M_{ij}} \times e^{M_{ij}} \) | \( M_{ij} \rightarrow e^{M_{ij}} \times e^{M_{ij}} \times e^{M_{ij}} \) |
| \( y_i \rightarrow \cosh(y_i) \) | \( y_i \rightarrow \sinh(y_i) \) | \( y_i \rightarrow \sinh(y_i) \) |
| \( y_i \rightarrow 2y_i \) | \( y_i \rightarrow \tanh(y_i) \) | \( y_i \rightarrow e^{y_i} - 1 \) |
| \( y_i \rightarrow I_0(y_i) \) | \( y_i \rightarrow I_0(y_i) \) | \( y_i \rightarrow \ln(1 + y_i) \) |
| \( y_i \rightarrow y_i^4 \) | \( y_i \rightarrow \log_{10}(y_i) \) | \( y_i \rightarrow y_i^2 \) |
| \( M_{ii} \rightarrow M_{ii} \times V_i \) | \( M_{ii} \rightarrow M_{ii} + V_i \) | \( M_{ii} \rightarrow M_{ii} - V_i \) |
2 Grid-optimized algorithms

It is interesting to ask if the PS-generated algorithms from Table 1 in the main text (which work well across a range of grid-sizes and broadly demonstrate similar convergence to CM-DVR) can be improved upon by seeking separate optimal algorithms for each different grid-size. By removing the constraint that a PS-generated code has to work well for a range of grid-sizes, we might anticipate that generating different algorithms tuned for different grid-sizes might lead to further improvements in accuracy. As such, we performed further PS optimizations but, instead of using a range of grid-sizes in the randomly-generated PES target data, we used single specific grid-sizes. Here, for code size \( N = 20 \), we performed 100 PS optimizations for fixed grid-sizes \( n_g = [13, 15, 19, 21, 31] \) using method E1; beyond fixing the grid-size, the remaining calculation details were the same as noted in sections 2 and 3. At each grid-size, we then selected the best algorithm from each of the 100 PS optimization runs, and further evaluated the performance at the targeted grid-size by calculating \( E_f \) for 500 randomly-generated PESs.

Figure 1 shows the results of these simulations as a function of grid-size, compared to the CM-DVR method. (working equations given below). It is found that the grid-targeted algorithms generally improve on the codes generated for grid-ranges (Fig. 4), as might be expected; for example, whereas the average RMS fractional errors obtained by the algorithms in Fig. 4 are around 0.5-1.0% smaller than the corresponding CM-DVR results, in the case grid-optimized algorithms, it is found that the errors are decreased further still, reducing the RMS fractional errors relative to converged CM-DVR by up to \( \sim 9\% \). Of course, the price paid for this small improvement is the requirement of using different algorithms for different grid-sizes, which is not particularly convenient if one is interested in using PS-generated codes in general analysis of quantum molecular vibrational properties.
Figure 1: Performance of different grid-targeted algorithms with code size $N = 20$. Here, unique algorithms were generated by PS which were specifically optimized to work for a single grid-size; the PS results here (blue circles) show the RMS fractional errors for five different algorithms targeted to work for different grid-sizes. For comparison, the RMS fractional errors for the Colbert-Miller DVR scheme are also shown (red dashed line). Fractional errors were calculated as in main manuscript for the first $n_e = 3$ eigenstates of 500 randomly-generated PESs; error bars are typically much smaller than the symbol sizes and are not shown for clarity.

2.1 Working equations for grid-optimized algorithms

The following working equations were derived from PS simulations optimized for different grid-sizes:

- $n_g = 13$

$$M_{ij} = (-1) \left[ \left( 2\tilde{M}\tilde{M} - V_i\delta_{ij} + m \right)^3 + 3 + V_i\delta_{ij} \right] e^{-(x_i - x_j)^2} - y_i\delta_{ij}$$

(1)

where
\[ y_i = 6(V_i - \pi) + 3 \]  \tag{2} \\

and

\[ \tilde{M}_{ij} = e^{-(m+1)-4\delta_{ij}} \]  \tag{3} \\

- \( n_g = 15 \)

\[ M_{ij} = \begin{cases} 
[\cos(V_i + \pi) - V_i \left(\frac{y_i}{2}\right)] - \pi, & \text{if } j = i \\
\cos(1)(-1)^{(i-j)} \frac{1}{2(i-j)(x_i-x_j)^2}, & \text{if } j \neq i 
\end{cases} \]  \tag{4} \\

where

\[ y_i = \sinh(\tanh[-m]) \]  \tag{5} \\

- \( n_g = 19 \)

\[ M_{ij} = \begin{cases} 
4 \left( \ln \left[ \sin \left( 2 - \frac{2}{L} \right) \right] \times 4 + V_i \right), & \text{if } j = i \\
\frac{4}{(x_i-x_j)^2} \left( \ln \left[ \sin \left( 2 - \left( \frac{1}{m(x_i-x_j)^2} + e^{-(x_i-x_j)} \left( \frac{2}{L(x_i-x_j)} \right) \right) \right] \right), & \text{if } j \neq i 
\end{cases} \]  \tag{6} \\

- \( n_g = 21 \)

\[ M_{ij} = \begin{cases} 
2m(10V_i + 2), & \text{if } j = i \\
\frac{2(-1)^{(i+j)}}{4(x_i-x_j)^2} \left( 4 + \frac{1}{(i-j)} \right), & \text{if } j \neq i 
\end{cases} \]  \tag{7} \\

- \( n_g = 31 \)
\[ M_{ij} = \begin{cases} \frac{\pi v_i}{y_i} \left( \sin \left[ \frac{mL + m}{\pi} \right] - 4 \right), & \text{if } j = i \\ \frac{\pi \sin(L^j)}{(i-j)}, & \text{if } j = i \pm 1 \end{cases} \] (8)

where

\[ y_i = 3 + L \left( \tanh(V_i) - L \right) \] (9)