A Appendix: Pseudocode for the AlphaSymbol

Algorithm 1 presents an overview of the AlphaSymbol framework. Prior to selecting a symbol, the $UseConstraint(\tau, s, \pi)$ function is applied to enforce constraints. The $counter$ is updated after each symbol selection, and we check whether the $counter$ is zero. If the $counter$ is zero, the reward value of the obtained expression is computed and backpropagation is performed. If the current symbol is a leaf node but not a terminal node, we expand and evaluate the node.

Algorithm 1: AlphaSymbol

```
Data: $X = [x_1, x_2, ..., x_n]$ ;$y = [y_1, y_2, ..., y_n]$;S=[$+, -, \times, \div, ...$]
Result: Find an expression such that y = f(X)
initialization;
while $Reward \neq 1$ do
    ▷ A threshold can also be set, for example, Reward >= 0.9999.
    repeat
        Self-Search :
        UseConstraint($\tau, s, \pi$)3
        symbol = arg max($\pi$)    ▷ Choosing the symbol with the highest probability
        $\tau$.append(symbol)
        $counter$ = $counter$ + Arity(symbol)4 - 1    ▷ Whether or not the expression is complete
        if counter=0 then
            z = $\frac{1}{1+S_{NRMSE}}$    ▷ Calculating rewards
            if $z > T$ then
                ▷ T represents the termination threshold of the algorithm
                break;    ▷ Terminate the program upon achieving expected rewards
            end
            Backpropagate : $z$    ▷ Backpropagate the final reward
            Storing data : $[s, \pi, z]$
            Train Neural Network : $\mathcal{N}_\theta \longrightarrow \mathcal{N}_{\theta NEW}$    ▷ Further training of LSTM
        end
        MCTS:
        Expand and evaluate:
        $parent||sibling$ = ParentSibling($\tau_t$) 2    ▷ Get the neural network input
        (p,v) =$\mathcal{N}_\theta(parent||sibling)$    ▷ Calculating probability distribution p and evaluat value v
          with LSTM
        for $j \leftarrow 2$ to $n_{evaluate}$ do
            if current node = leafnode & counter $\neq 0$ then
                Expend(p)    ▷ Expanding leaf nodes with probability p
            else
                Select: $a_{t+1}$ = arg max($\mathcal{UCT}(s_t, a_t)$)    ▷ Selecting the symbol with the largest
                  UCT value as the next symbol
            end
            Backpropagate(v)    ▷ Backpropagate the evaluate value v
        end
    until Find the target epxression;
end
```

Algorithm 2 describes the function $ParentSibling(\tau)$ used in Algorithm 1 to find the parent and sibling nodes of the next symbol to the sample. This algorithm uses the following logic: If the last symbol in the partial traversal is a unary or binary operator and has no siblings, it is the parent node and its sibling is set to empty. Otherwise, the algorithm iterates forward in the traversal until it finds a node with unselected children. This node is the parent and the subsequent node is the sibling.

Algorithm 2: ParentSibling(τ) (To retrieve the father and sibling nodes as inputs for an LSTM)

```
Input : Partially sampled traversal $\tau$
Output : Concatenated parent and sibling nodes of the next nodes to be generated
$T \leftarrow len(\tau)$                                   ▷ Length of partial traversal
$counter \leftarrow 0$                 ▷ Initializes a counter with no selected number of nodes
if $Arity(\tau_T) > 0$ then
    $parent \leftarrow \tau_T$
    $sibling \leftarrow empty$
end
for $i \leftarrow T$ to $1$ do
    $counter = counter + Arity(\tau_i) - 1$
                                                   ▷ Update counter of unselected nodes
    if $counter = 0$ then
        $parent \leftarrow \tau_i$
        $sibling \leftarrow \tau_{i+1}$
    end
end
```

Algorithm 3 demonstrates the application of a series of constraints during the symbol generation process. The specific steps are as follows: we first obtain the types of symbols in our symbol library, and then based on the current state, we sequentially determine whether each function in the symbol library should be "restricted". If a symbol is restricted, we set the probability of selecting that symbol to zero and finally normalize the probabilities of all symbols.

Algorithm 3: UseConstraints(τ, S_i, π)

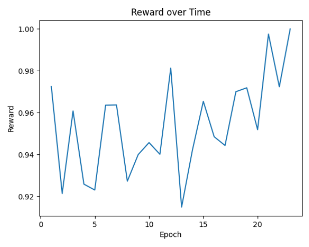
```
Input : The simulated probability $\pi$; partially sampled traversal $\tau$; Used symbol library $S$
Output : The probability distribution $\pi$ adjusted according to the constraints
$L \leftarrow len(S)$                                                  ▷ Length of $S$
for $i \leftarrow 1$ to $L$ do
    if $Constraint(\tau, S_i)$ then
        $\pi_i \leftarrow 0$                    ▷ Sets the restricted symbol probability to zero
    end
    $\pi \leftarrow \frac{\pi}{\sum_{i=0}^{L} \pi_i}$                          ▷ The probability is normalized
end
```

Algorithm 4 describes the Arity(s) function used in Algorithm 1, which obtains the arity of an operator. Specifically, for a variable $[x_1]$ and a constant $[c]$, Arity(s)=0. For a unary operator $[sin, cos, exp, ln, sqrt]$, Arity(s)=1. Similarly, for a binary operator, Arity(s)=2, and so on for other operators.

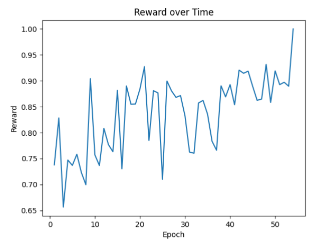
Algorithm 4: Arity(s)

```
Input : Newly selected operator symbol $s$;
Output : The arity of an operator
$s \leftarrow select(S)$                                    ▷ Selecting new symbols.
if $s \in [x_1, x_2, ..., x_n, c]$ then
    return 0            ▷ If the symbol is a variable or a constant, the arity would be 0
end
if $s \in [sin, cos, exp, log, sqrt]$ then
    return 1                        ▷ If the operator is unary, the arity would be 1.
end
if $s \in [+, -, *, /]$ then
    return 2                   ▷ If the operator is a binary operator, it returns 2.
end
```

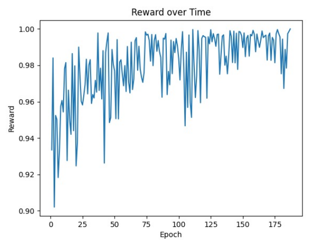
B APPENDIX: REWARD VARIATION CURVE (ON DATASET NGUYEN).



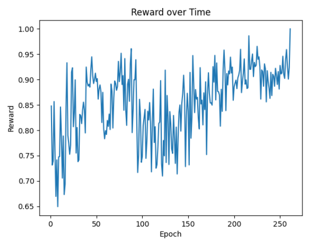
(a) Nguyen-1



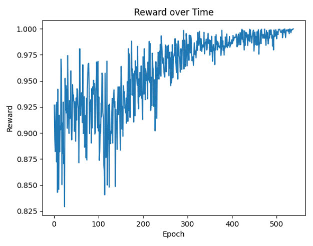
(b) Nguyen-2



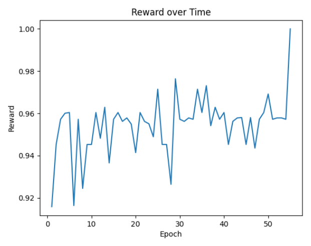
(c) Nguyen-3



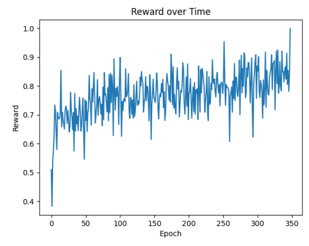
(d) Nguyen-4



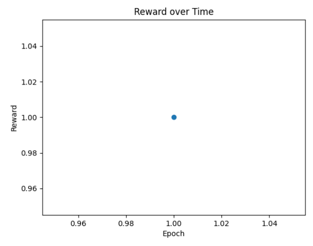
(e) Nguyen-5



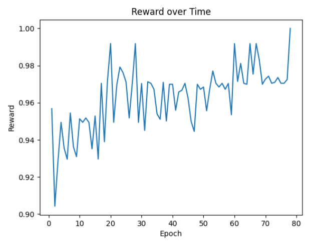
(f) Nguyen-6



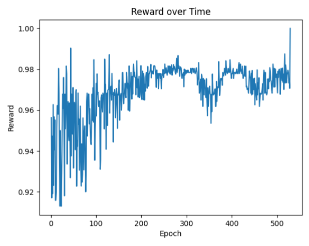
(g) Nguyen-7



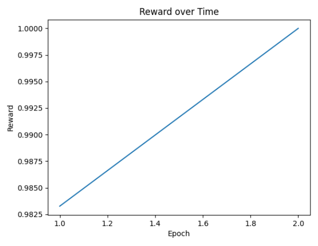
(h) Nguyen-8



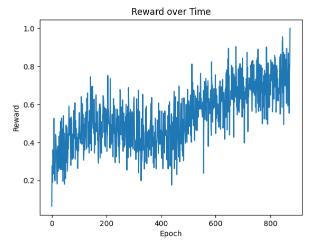
(i) Nguyen-9



(j) Nguyen-10



(k) Nguyen-11



(l) Nguyen-12

Figure B.1: The series of figures above presents line graphs depicting the reward values of AlphaSymbol on the Nguyen dataset over time. As observed from the figures, throughout the search process, the reward values for all expressions demonstrate an oscillatory ascent with the increase in training iterations. Notably, Expression 8 is an exception due to its comparatively simple structure, achieving its best result within just one epoch.

C APPENDIX: EXPERIMENTAL DETAILS FOR EACH EXPRESSION

Table $B.1 - B.3$ shows some specific details of different expressions when tested. The benchmark problem specifications for symbolic regression are as follows:

- Input variables are represented as $[x_1, x_2, ..., x_n]$
- $U(a, b, c)$signifies c random points uniformly sampled between a and b for each input variable. Different random seeds are used for training and testing datasets.
- $E(a, b, c)$ indicates c points evenly spaced between a and b for each input variable.
- To simplify the notation, libraries are defined relative to a base library $[+, -, \times, \div, sin, cos, ln, exp, sqrt, x_1]$.
- Any unassigned operand is represented by , for instance, ^{2} corresponds to the square operator.

Table C.1: Symbol library and value range of the three data sets Nguyen, Korns, and Jin.

Name	Expression	Dataset	Library
Nguyen-1	$x_1^3 + x_1^2 + x_1$	$\text{U}(-1, 1, 20)$	
Nguyen-2	$x_1^4 + x_1^3 + x_1^2 + x_1$	$\text{U}(-1, 1, 20)$	
Nguyen-3	$x_1^5 + x_1^4 + x_1^3 + x_1^2 + x_1$	$\text{U}(-1, 1, 20)$	
Nguyen-4	$x_1^6 + x_1^5 + x_1^4 + x_1^3 + x_1^2 + x_1$	$\text{U}(-1, 1, 20)$	
Nguyen-5	$\sin(x_1^2)\cos(x) - 1$	$\text{U}(-1, 1, 20)$	
Nguyen-6	$\sin(x_1) + \sin(x_1 + x_1^2)$	$\text{U}(-1, 1, 20)$	
Nguyen-7	$\log(x_1 + 1) + \log(x_1^2 + 1)$	$\text{U}(0, 2, 20)$	
Nguyen-8	$\sqrt{x}$	$\text{U}(0, 4, 20)$	
Nguyen-9	$\sin(x) + \sin(x_2^2)$	$\text{U}(0, 1, 20)$	
Nguyen-10	$2\sin(x)\cos(x_2)$	$\text{U}(0, 1, 20)$	
Nguyen-11	x_1^{x2}	$\text{U}(0, 1, 20)$	
Nguyen-12	$x_1^4 - x_1^3 + \frac{1}{2}x_2^2 - x_2$	$\text{U}(0, 1, 20)$	
Nguyen-2′	$4x_1^4 + 3x_1^3 + 2x_1^2 + x$	$\text{U}(-1, 1, 20)$	
Nguyen-5′	$\sin(x_1^2)\cos(x) - 2$	$\text{U}(-1, 1, 20)$	
Nguyen-8′	$\sqrt[3]{x}$	$\text{U}(0, 4, 20)$	
Nguyen-8′′	$\sqrt[3]{x_1^2}$	$\text{U}(0, 4, 20)$	
Nguyen-1^c	$3.39x_1^3 + 2.12x_1^2 + 1.78x$	$\text{U}(-1, 1, 20)$	
Nguyen-5^c	$\sin(x_1^2)\cos(x) - 0.75$	$\text{U}(-1, 1, 20)$	
Nguyen-7^c	$\log(x + 1.4) + \log(x_1^2 + 1.3)$	$\text{U}(0, 2, 20)$	
Nguyen-8^c	$\sqrt{1.23x}$	$\text{U}(0, 4, 20)$	
Nguyen-10^c	$\sin(1.5x)\cos(0.5x_2)$	$\text{U}(0, 1, 20)$	
Korns-1	$1.57 + 24.3 * x_1^4$	$\text{U}(-1, 1, 20)$	
Korns-2	$0.23 + 14.2\frac{(x_4+x_1)}{(3x_2)}$	$\text{U}(-1, 1, 20)$	
Korns-3	$4.9\frac{(x_2-x_1+\frac{x_1}{x_3}}{(3x_3))} - 5.41$	$\text{U}(-1, 1, 20)$	
Korns-4	$0.13sin(x_1) - 2.3$	$\text{U}(-1, 1, 20)$	
Korns-5	$3 + 2.13log(\lvert x_5\rvert)$	$\text{U}(-1, 1, 20)$	
Korns-6	$1.3 + 0.13\sqrt{\lvert x_1\rvert}$	$\text{U}(-1, 1, 20)$	
Korns-7	$2.1(1 - e^{-0.55x_1})$	$\text{U}(-1, 1, 20)$	
Korns-8	$6.87 + 11\sqrt{\lvert 7.23x_1x_4x_5\rvert}$	$\text{U}(-1, 1, 20)$	
Korns-9	$12\sqrt{\lvert 4.2x_1x_2x_2\rvert}$	$\text{U}(-1, 1, 20)$	
Korns-10	$0.81 + 24.3\frac{2x_1+3x_2^2}{4x_3^3+5x_4^4}$	$\text{U}(-1, 1, 20)$	
Korns-11	$6.87 + 11cos(7.23x_1^3)$	$\text{U}(-1, 1, 20)$	
Korns-12	$2 - 2.1cos(9.8x_1^3)sin(1.3x_5)$	$\text{U}(-1, 1, 20)$	
Korns-13	$32.0 - 3.0\frac{tan(x_1)}{tan(x_2)}\frac{tan(x_3)}{tan(x_4)}$	$\text{U}(-1, 1, 20)$	
Korns-14	$22.0 - (4.2cos(x_1) - tan(x_2))\frac{tanh(x_3)}{sin(x_4)}$	$\text{U}(-1, 1, 20)$	
Korns-15	$12.0 - \frac{6.0tan(x_1)}{e^{x2}}(log(x_3) - tan(x_4))))$	$\text{U}(-1, 1, 20)$	
Jin-1	$2.5x_1^4 - 1.3x_1^3 + 0.5x_2^2 - 1.7x_2$	$\text{U}(-3, 3, 100)$	
Jin-2	$8.0x_1^2 + 8.0x_2^3 - 15.0$	$\text{U}(-3, 3, 100)$	
Jin-3	$0.2x_1^3 + 0.5x_2^3 - 1.2x_2 - 0.5x_1$	$\text{U}(-3, 3, 100)$	
Jin-4	$1.5\exp x + 5.0cos(x_2)$	$\text{U}(-3, 3, 100)$	
Jin-5	$6.0sin(x_1)cos(x_2)$	$\text{U}(-3, 3, 100)$	
Jin-6	$1.35x_1x_2 + 5.5sin((x_1 - 1.0)(x_2 - 1.0))$	$\text{U}(-3, 3, 100)$	

Table C.2: Symbol library and value range of the three data sets neat, Keijzer and Livermore.

Name	Expression	Dataset	Library
Neat-1	$x_1^4 + x_1^3 + x_1^2 + x$	U$(-1, 1, 20)$	
Neat-2	$x_1^5 + x_1^4 + x_1^3 + x_1^2 + x$	U$(-1, 1, 20)$	
Neat-3	$\sin(x_1^2)\cos(x) - 1$	U$(-1, 1, 20)$	
Neat-4	$\log(x + 1) + \log(x_1^2 + 1)$	U$(0, 2, 20)$	
Neat-5	$2\sin(x)\cos(x_2)$	U$(-1, 1, 100)$	
Neat-6	$\sum_{k=1}^{x} \frac{1}{k}$	E$(1, 50, 50)$	
Neat-7	$2 - 2.1\cos(9.8x_1)\sin(1.3x_2)$	E$(-50, 50, 10^5)$	
Neat-8	$\frac{e^{-(x_1)^2}}{1.2+(x_2-2.5)^2}$	U$(0.3, 4, 100)$	
Neat-9	$\frac{1}{1+x_1^{-4}} + \frac{1}{1+x_2^{-4}}$	E$(-5, 5, 21)$	
Keijzer-1	$0.3x_1 sin(2\pi x_1)$	U$(-1, 1, 20)$	
Keijzer-2	$2.0x_1 sin(0.5\pi x_1)$	U$(-1, 1, 20)$	
Keijzer-3	$0.92x_1 sin(2.41\pi x_1)$	U$(-1, 1, 20)$	
Keijzer-4	$x_1^3 e^{-x_1} cos(x_1)sin(x_1)sin(x_1)^2 cos(x_1) - 1$	U$(-1, 1, 20)$	
Keijzer-5	$3 + 2.13log(\lvert x_5\rvert)$	U$(-1, 1, 20)$	
Keijzer-6	$\frac{x1(x1+1)}{2}$	U$(-1, 1, 20)$	
Keijzer-7	$log(x_1)$	U$(0, 1, 20)$	
Keijzer-8	$\sqrt{(x_1)}$	U$(0, 1, 20)$	
Keijzer-9	$log(x_1 + \sqrt{x_1^2} + 1)$	U$(-1, 1, 20)$	
Keijzer-10	$x_1^{x_2}$	U$(-1, 1, 20)$	
Keijzer-11	$x_1x_2 + sin((x_1 - 1)(x_2 - 1))$	U$(-1, 1, 20)$	
Keijzer-12	$x_1^4 - x_1^3 + \frac{x_2^2}{2} - x_2$	U$(-1, 1, 20)$	
Keijzer-13	$6sin(x_1)cos(x_2)$	U$(-1, 1, 20)$	
Keijzer-14	$\frac{8}{2+x_1^2+x_2^2}$	U$(-1, 1, 20)$	
Keijzer-15	$\frac{x_1^3}{5} + \frac{x_2^3}{2} - x_2 - x_1$	U$(-1, 1, 20)$	
Livermore-1	$\frac{1}{3} + x_1 + sin(x_1^2))$	U$(-3, 3, 100)$	
Livermore-2	$sin(x_1^2) * cos(x1) - 2$	U$(-3, 3, 100)$	
Livermore-3	$sin(x_1^3) * cos(x_1^2)) - 1$	U$(-3, 3, 100)$	
Livermore-4	$log(x_1 + 1) + log(x_1^2 + 1) + log(x_1)$	U$(-3, 3, 100)$	
Livermore-5	$x_1^4 - x_1^3 + x_2^2 - x_2$	U$(-3, 3, 100)$	
Livermore-6	$4x_1^4 + 3x_1^3 + 2x_1^2 + x_1$	U$(-3, 3, 100)$	
Livermore-7	$\frac{(exp(x1)-exp(-x_1)}{2})$	U$(-1, 1, 100)$	
Livermore-8	$\frac{(exp(x1)+exp(-x1)}{3}$	U$(-3, 3, 100)$	
Livermore-9	$x_1^9 + x_1^8 + x_1^7 + x_1^6 + x_1^5 + x_1^4 + x_1^3 + x_1^2 + x_1$	U$(-1, 1, 100)$	
Livermore-10	$6 * sin(x_1)cos(x_2)$	U$(-3, 3, 100)$	
Livermore-11	$\frac{x_1^2 x_2^2}{(x_1+x_2)}$	U$(-3, 3, 100)$	
Livermore-12	$\frac{x_1^5}{x_2^3}$	U$(-3, 3, 100)$	
Livermore-13	$x_1^{\frac{1}{3}}$	U$(-3, 3, 100)$	
Livermore-14	$x_1^3 + x_1^2 + x_1 + sin(x_1) + sin(x_2^2)$	U$(-1, 1, 100)$	
Livermore-15	$x_1^{\frac{1}{5}}$	U$(-3, 3, 100)$	
Livermore-16	$x_1^{\frac{2}{3}}$	U$(-3, 3, 100)$	
Livermore-17	$4sin(x_1)cos(x_2)$	U$(-3, 3, 100)$	
Livermore-18	$sin(x_1^2) * cos(x_1) - 5$	U$(-3, 3, 100)$	
Livermore-19	$x_1^5 + x_1^4 + x_1^2 + x_1$	U$(-3, 3, 100)$	
Livermore-20	$e^{(-x_1^2)}$	U$(-3, 3, 100)$	
Livermore-21	$x_1^8 + x_1^7 + x_1^6 + x_1^5 + x_1^4 + x_1^3 + x_1^2 + x_1$	U$(-1, 1, 20)$	
Livermore-22	$e^{(-0.5x_1^2)}$	U$(-3, 3, 100)$	

Table C.3: Symbol library and value range of the three data sets Vladislavleva and others.

Name	Expression	Dataset	Library
Vladislavleva-1	$\frac{(e^{-(x1-1)^2})}{(1.2+(x2-2.5)^2))}$	$\mathrm{U}(-1, 1, 20)$	
Vladislavleva-2	$e^{-x_1}x_1^3cos(x_1)sin(x_1)(cos(x_1)sin(x_1)^2-1)$	$\mathrm{U}(-1, 1, 20)$	
Vladislavleva-3	$e^{-x_1}x_1^3cos(x_1)sin(x_1)(cos(x_1)sin(x_1)^2-1)(x_2-5)$	$\mathrm{U}(-1, 1, 20)$	
Vladislavleva-4	$\frac{10}{5+(x1-3)^2+(x_2-3)^2+(x_3-3)^2+(x_4-3)^2+(x_5-3)^2}$	$\mathrm{U}(0, 2, 20)$	
Vladislavleva-5	$30(x_1-1)\frac{x_3-1}{(x_1-10)}x_2^2$	$\mathrm{U}(-1, 1, 100)$	
Vladislavleva-6	$6sin(x_1)cos(x_2)$	$\mathrm{E}(1, 50, 50)$	
Vladislavleva-7	$2-2.1\cos(9.8x)\sin(1.3x_2)$	$\mathrm{E}(-50, 50, 10^5)$	
Vladislavleva-8	$\frac{e^{-(x-1)^2}}{1.2+(x_2-2.5)^2}$	$\mathrm{U}(0.3, 4, 100)$	
Test-2	$3.14 * x1 * x1$	$\mathrm{U}(-1, 1, 20)$	
Const-Test-1	$5 * x1 * x1$	$\mathrm{U}(-1, 1, 20)$	
GrammarVAE-1	$1./3 + x1 + sin(x_1^2))$	$\mathrm{U}(-1, 1, 20)$	
Sine	$sin(x_1) + sin(x_1 + x_1^2))$	$\mathrm{U}(-1, 1, 20)$	
Nonic	$x_1^9 + x_1^8 + x_1^7 + x_1^6 + x_1^5 + x_1^4 + x_1^3 + x_1^2 + x_1$	$\mathrm{U}(-1, 1, 100)$	
Pagie-1	$\frac{1}{1+x_1^{-4}+\frac{1}{1+x2^{-4}}}$	$\mathrm{E}(1, 50, 50)$	
Meier-3	$\frac{x_1^2x_2^2}{(x_1+x_2)}$	$\mathrm{E}(-50, 50, 10^5)$	
Meier-4	$\frac{x_1^5}{x_2^3}$	$\mathrm{U}(0.3, 4, 100)$	
Poly-10	$x_1x_2 + x_3x4 + x_5x_6 + x_1x_7x_9 + x_3x_6x_{10}$	$\mathrm{E}(-1, 1, 100)$	

D Appendix: Average Coefficient of Determination (R^2) on Various Datasets

To assess the goodness of fit of AlphaSymbol on the datasets, we also recorded the average R2 of AlphaSymbol on various testing datasets. From the table, it can be observed that AlphaSymbol achieved an average R2 exceeding 0.99 on the majority of the datasets. This suggests that while AlphaSymbol may not be able to fully recover the original formula of certain expressions, it can still find an equivalent expression that fits the observed data well.

Table D.1: Average Coefficient of Determination (R^2) on Various Datasets

Benchmark	R^2
Nguyen	0.9999
Keijzer	0.9991
Korns	0.9982
Constant	0.9991
Livermore	0.9998
Vladislavlev	0.9831
R	0.9702
Jin	0.9888
Neat	0.9763
AI Feynman	0.9960
Others	0.9982
Average	**0.9917**

E Appendix: R^2 of AlphaSymbol on the AI Feynman dataset.

We tested the performance of our proposed symbol regression algorithm, AlphaSymbol, on the AI Feynman dataset. This dataset contains problems from physics and mathematics across multiple subfields, such as mechanics, thermodynamics, and electromagnetism. The authors provided 100,000 sampled data points in the AI Feynman dataset, however, to better test the performance of AlphaSymbol, we randomly selected only 100 data points from the 100,000 provided as our experimental data. We applied AlphaSymbol to perform symbol regression on each data in the dataset. and recorded the R^2 between the predicted results and the correct answers. The experimental results indicate that

AlphaSymbol can accurately fit the corresponding expressions from a small number of sample points. For the majority of the formulas, the R^2 exceeds 0.99. This indicates that the model performs well on problems in fields such as physics and mathematics, and has great potential for wide application. The experimental results are shown in Table E.1 and Table E.2.

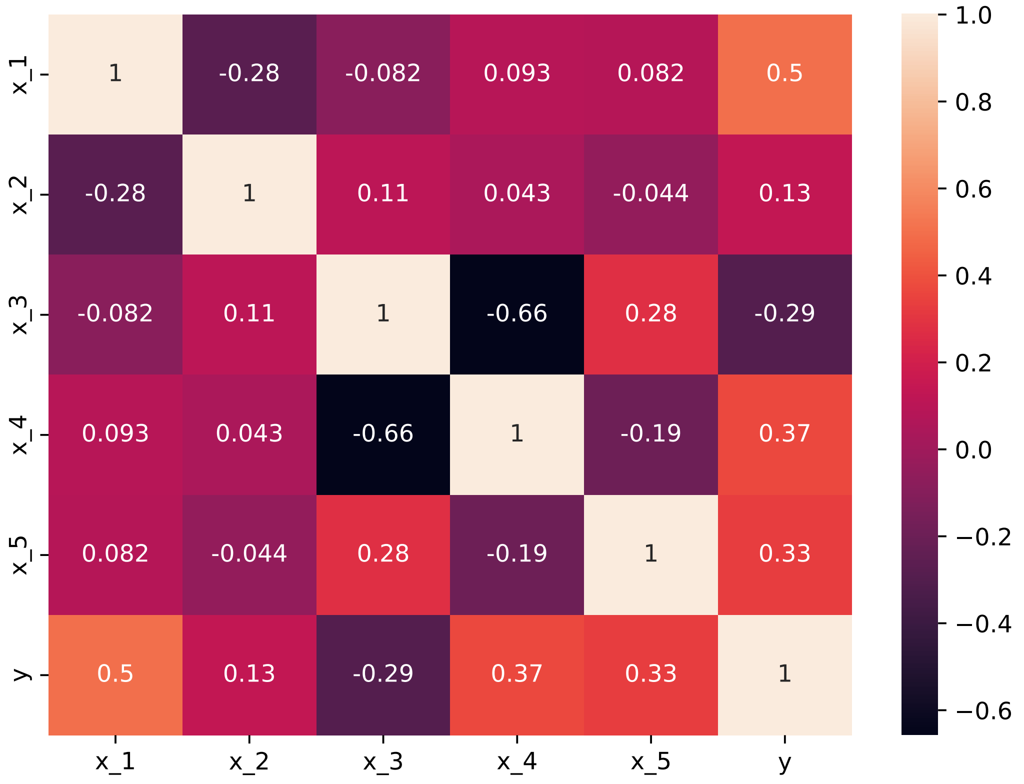


Figure E.1: This is a heatmap of the correlation coefficients between each variable x and y. From the image, we can see that y has a positive correlation with x_1, x_2, x_4, and x_5 to varying degrees, and has a negative correlation with x_3.

F Appendix: Hyperparameter Settings

- **min_length (int):** minimum number of operators to allow in expression
- **max_length (int):** maximum number of operators to allow in expression
- **type (‘rnn’, ‘lstm’, or ‘gru’):** type of architecture to use
- **num_layers (int):** number of layers in RNN architecture
- **dropout (float):** dropout (if any) for RNN architecture
- **lr (float):** learning rate for RNN
- **optimizer (‘adam’ or ‘rmsprop’):** optimizer for RNN
- **inner_optimizer (‘lbfgs’, ‘adam’, or ‘rmsprop’):** optimizer for expressions
- **inner_lr (float):** learning rate for constant optimization
- **inner_num_epochs (int):** number of epochs for constant optimization
-**batch_size (int):** batch size for training the RNN
- **λ (float):** The weight assigned to the X part of the loss, ranges from 0 to 1.

when training the RNN.
- **batch_size (int):** batch size for training the RNN
- **num_batches (int):** number of batches (will stop early if found)
- **hidden_size (int):** hidden dimension size for RNN

Table E.1: Tested Feynman Equations, part 1.

Feynman	Equation	R^2
I.6.20a	$f = e^{-\theta^2/2}/\sqrt{2\pi}$	0.9992
I.6.20	$f = e^{-\frac{\theta^2}{2\sigma^2}}/\sqrt{2\pi\sigma^2}$	0.9988
I.6.20b	$f = e^{-\frac{(\theta-\theta_1)^2}{2\sigma^2}}/\sqrt{2\pi\sigma^2}$	0.9923
I.8.14	$d = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$	0.8929
I.9.18	$F = \frac{Gm_1m_2}{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}$	0.9944
I.10.7	$F = \frac{Gm_1m_2}{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}$	0.9906
I.11.19	$A = x_1y_1 + x_2y_2 + x_3y_3$	1.0
I.12.1	$F = \mu N_n$	1.0
I.12.2	$F = \frac{q_1q_2}{4\pi\epsilon r^2}$	1.0
I.12.4	$E_f = \frac{q_1}{4\pi\epsilon r^2}$	0.9994
I.12.5	$F = q_2E_f$	1.0
I.12.11	$F = \mathcal{Q}(E_f + Bv\sin\theta)$	0.9999
I.13.4	$K = \frac{1}{2}m(v^2+u^2+w^2)$	0.9969
I.13.12	$U = Gm_1m_2(\frac{1}{r_2}-\frac{1}{r_1})$	1.0
I.14.3	$U = mgz$	1.0
I.14.4	$U = \frac{k_{spring}x^2}{2}$	0.9999
I.15.3x	$x_1 = \frac{x-ut}{\sqrt{1-u^2/c^2}}$	0.9993
I.15.3t	$t_1 = \frac{t-ux/c^2}{\sqrt{1-u^2/c^2}}$	0.9844
I.15.10	$p = \frac{m_0v}{\sqrt{1-v^2/c^2}}$	0.9978
I.16.6	$v_1 = \frac{u+v}{1+uv/c^2}$	0.9873
I.18.4	$r = \frac{m_1r_1+m_2r_2}{m_1+m_2}$	0.9894
I.18.12	$\tau = rF\sin\theta$	0.9999
I.18.16	$L = mrv\sin\theta$	0.9999
I.24.6	$E = \frac{1}{4}m(\omega^2+\omega_0^2)x^2$	0.9986
I.25.13	$V_e = \frac{q}{C}$	1.0
I.26.2	$\theta_1 = \arcsin(n\sin\theta_2)$	0.9991
I.27.6	$f_f = \frac{1}{\frac{1}{d_1}+\frac{n}{d_2}}$	0.9995
I.29.4	$k = \frac{\omega}{c}$	1.0
I.29.16	$x = \sqrt{x_1^2+x_2^2-2x_1x_2\cos(\theta_1-\theta_2)}$	0.9942
I.30.3	$I_* = I_{*0}\frac{\sin^2(n\theta/2)}{\sin^2(\theta/2)}$	0.9912
I.30.5	$\theta = \arcsin(\frac{\lambda}{nd})$	0.9994
I.32.5	$P = \frac{q^2a^2}{6\pi\epsilon c^3}$	0.9857
I.32.17	$P = (\frac{1}{2}\epsilon cE_f^2)(8\pi r^2/3)(\omega^4/(\omega^2-\omega_0^2)^2)$	0.9788
I.34.8	$\omega = \frac{qvB}{p}$	1.0
I.34.10	$\omega = \frac{\omega_0}{1-v/c}$	0.9928
I.34.14	$\omega = \frac{1+v/c}{\sqrt{1-v^2/c^2}}\omega_0$	0.9992
I.34.27	$E = \hbar\omega$	1.0
I.37.4	$I_* = I_1 + I_2 + 2\sqrt{I_1I_2}\cos\delta$	0.9927
I.38.12	$r = \frac{4\pi\epsilon\hbar^2}{mq^2}$	0.9999
I.39.10	$E = \frac{3}{2}p_FV$	1.0
I.39.11	$E = \frac{1}{\gamma-1}p_FV$	0.9998
I.39.22	$P_F = \frac{nk_bT}{V}$	0.9999
I.40.1	$n = n_0e^{-\frac{mgx}{k_bT}}$	0.9947
I.41.16	$L_{rad} = \frac{\hbar\omega^3}{\pi^2c^2(e^{\frac{\hbar\omega}{k_bT}}-1)}$	0.8462
I.43.16	$v = \frac{\mu_{drift}qV_e}{d}$	1.0
I.43.31	$D = \mu_ek_bT$	1.0
I.43.43	$\kappa = \frac{1}{\gamma-1}\frac{k_bv}{A}$	0.9428
I.44.4	$E = nk_bT\ln(\frac{V_2}{V_1})$	0.8322
I.47.23	$c = \sqrt{\frac{\gamma pr}{\rho}}$	0.9926
I.48.20	$E = \frac{mc^2}{\sqrt{1-v^2/c^2}}$	0.8859
I.50.26	$x = x_1[\cos(\omega t) + \alpha\, cos(\omega t)^2]$	0.9999

Table E.2: Tested Feynman Equations, part 2.

Feynman	Equation	R^2
II.2.42	$\mathrm{P} = \frac{\kappa(T_2-T_1)A}{d}$	0.7842
II.3.24	$F_E = \frac{P}{4\pi r^2}$	0.9976
II.4.23	$V_e = \frac{q}{4\pi\epsilon r}$	0.9997
II.6.11	$V_e = \frac{1}{4\pi\epsilon}\frac{p_d \cos\theta}{r^2}$	1.0
II.6.15a	$E_f = \frac{3}{4\pi\epsilon}\frac{p_d z}{r^5}\sqrt{x^2+y^2}$	0.9466
II.6.15b	$E_f = \frac{3}{4\pi\epsilon}\frac{p_d}{r^3}\cos\theta\sin\theta$	0.9943
II.8.7	$E = \frac{3}{5}\frac{q^2}{4\pi\epsilon d}$	0.9955
II.8.31	$E_{den} = \frac{\epsilon E_f^2}{2}$	1.0
II.10.9	$E_f = \frac{\sigma_{den}}{\epsilon}\frac{1}{1+\chi}$	0.9999
II.11.3	$x = \frac{qE_f}{m(\omega_0^2-\omega^2)}$	0.9901
II.11.7	$n = n_0(1+\frac{p_d E_f \cos\theta}{k_b T})$	0.8826
II.11.20	$P_* = \frac{n_\rho p_d^2 E_f}{3k_b T}$	0.7783
II.11.27	$P_* = \frac{n\alpha}{1-n\alpha/3}\epsilon E_f$	0.9859
II.11.28	$\theta = 1+\frac{n\alpha}{1-(n\alpha/3)}$	0.9947
II.13.17	$B = \frac{1}{4\pi\epsilon c^2}\frac{2I}{r}$	0.9997
II.13.23	$\rho_c = \frac{\rho_{c_0}}{\sqrt{1-v^2/c^2}}$	0.9807
II.13.34	$j = \frac{\rho_{c_0} v}{\sqrt{1-v^2/c^2}}$	0.9938
II.15.4	$E = -\mu_M B\cos\theta$	1.0
II.15.5	$E = -p_d E_f \cos\theta$	1.0
II.21.32	$V_e = \frac{q}{4\pi\epsilon r(1-v/c)}$	0.9954
II.24.17	$k = \sqrt{\frac{\omega^2}{c^2}-\frac{\pi^2}{d^2}}$	0.9872
II.27.16	$F_E = \epsilon c E_f^2$	1.0
II.27.18	$E_{den} = \epsilon E_f^2$	1.0
II.34.2a	$I = \frac{qv}{2\pi r}$	0.9982
II.34.2	$\mu_M = \frac{qvr}{2}$	0.9918
II.34.11	$\omega = \frac{g\_qB}{2m}$	0.9937
II.34.29a	$\mu_M = \frac{qh}{4\pi m}$	1.0
II.34.29b	$E = \frac{g\_\mu_M B J_z}{\hbar}$	0.8882
II.35.18	$n = \frac{n_0}{\exp(\mu_m B/(k_b T))+\exp(-\mu_m B/(k_b T))}$	0.9466
II.35.21	$M = n_\rho \mu_M \tanh(\frac{\mu_M B}{k_b T})$	0.8722
II.36.38	$f = \frac{\mu_m B}{k_b T}+\frac{\mu_m \alpha M}{\epsilon c^2 k_b T}$	0.9244
II.37.1	$E = \mu_M(1+\chi)B$	0.9999
II.38.3	$F = \frac{YAx}{d}$	1.0
II.38.14	$\mu_S = \frac{Y}{2(1+\sigma)}$	0.9999
III.4.32	$n = \frac{1}{e^{\frac{\hbar\omega}{k_b T}}-1}$	0.9877
III.4.33	$E = \frac{\hbar\omega}{e^{\frac{\hbar\omega}{k_b T}}-1}$	0.9998
III.7.38	$\omega = \frac{2\mu_M B}{\hbar}$	0.9914
III.8.54	$p_\gamma = \sin(\frac{Et}{\hbar})^2$	0.9943
III.9.52	$p_\gamma = \frac{p_d E_f t}{\hbar}\frac{\sin((\omega-\omega_0)t/2)^2}{((\omega-\omega_0)t/2)^2}$	0.7266
III.10.19	$E = \mu_M\sqrt{B_x^2+B_y^2+B_z^2}$	0.9928
III.12.43	$L = n\hbar$	1.0
III.13.18	$v = \frac{2Ed^2 k}{\hbar}$	0.9999
III.14.14	$I = I_0(e^{\frac{qV_e}{k_b T}}-1)$	0.9982
III.15.12	$E = 2U(1-\cos(kd))$	0.9999
III.15.14	$m = \frac{\hbar^2}{2Ed^2}$	0.9983
III.15.27	$k = \frac{2\pi\alpha}{nd}$	0.9998
III.17.37	$f = \beta(1+\alpha\cos\theta)$	1.0
III.19.51	$E = \frac{-mq^4}{2(4\pi\epsilon)^2\hbar^2}\frac{1}{n^2}$	0.9894
III.21.20	$j = \frac{-\rho_{c_0} q A_{vec}}{m}$	0.7489

- use_gpu (bool): whether or not to train with GPU

Table F.1: Tuned hyperparameters for AlphaSymbol.

Parameter	Value
min_length	2
max_length	-
type	LSTM
Num_layers for LSTM	2
hidden_size	250
dropout	0.0
optimizer for LSTM	adam
Learn rate	0.0005
batch_size	1000
Use_gpu	False
inner_optimizer	lbfgs
inner_lr	0.1
inner_num_epochs	5
num_batches	10000
λ	0.1

G APPENDIX: RELATED WORK SUPPLEMENT

Self-Learning_Gene_Expression_Programming (SL-GEP)(Zhong et al., 2015), The SL-GEP method utilizes Gene Expression Programming (GEP) to represent each chromosome, which consists of a main program and a set of Automatically Defined Functions (ADFs). Each ADF is a sub-function used to solve sub-problems and is combined with the main program to address the target problem of interest. In the initialization step, all encoded ADFs in each chromosome are randomly generated. Then, during the evolutionary search process, SL-GEP employs a self-learning mechanism to improve the search outcomes. Specifically, SL-GEP utilizes an adaptive learning algorithm to dynamically evolve the ADFs online and integrate them with the main program to construct more complex and higher-order sub-functions, thereby enhancing search accuracy and efficiency.
semantic genetic programming (SGD)(Huang et al., 2022), Traditional genetic programming approaches often rely on random search to find optimal solutions, but this method is inefficient and prone to getting stuck in local optima. Therefore, SGD utilizes program behavior to guide the search, aiming to improve the efficiency and accuracy of symbolic regression problems. Specifically, this method starts by transforming input data into vector form and uses it as a constraint in a linear programming model. Then, semantic information is employed to evaluate each program and classify them based on their behavioral characteristics. Subsequently, the best programs are selected within each category and used to generate a new generation of programs. This approach effectively reduces the search space and accelerates convergence speed.
shape-constrained symbolic regression (SCSR) (Haider et al., 2023), The main idea of SCSR is a shape-constrained symbolic regression algorithm. This method leverages prior knowledge about the shape of the regression function to improve the accuracy of the regression model. Specifically, the article introduces both single-objective and multi-objective algorithms. The single-objective algorithm utilizes genetic programming techniques to generate the best-fitting curve. On the other hand, the multi-objective algorithm considers multiple optimization objectives and employs Pareto front techniques to search for a set of non-dominated solutions.

H APPENDIX: COMPUTING RESOURCES

The server we use is equipped with an Intel(R) Xeon(R) Gold 5218R CPU, which has a base frequency of 2.10 GHz. It has a total of 20 CPU cores, allowing for parallel processing and improved computational performance. The high core count and efficient architecture of the Intel Xeon Gold 5218R make it suitable for handling demanding computational tasks and workloads.

I Appendix: MCTS

In order to clearly show the MCTS search process, we assume that there are only two basic symbols [sin,x]. The target expression is y = sin(x). The search process is as follows.

Initialization: Initially there is a root node S_0, and each node in the tree has two values, the reward Q of the node and the number of visits to that node N.

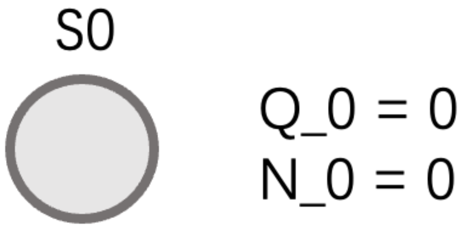


Figure I.1

First iteration: Node S_0 is the root and leaf node, and is not the terminating node, so it is extended. Assume that S_0 has two actions (the basic symbol [sin,x]) after it , which are transferred to S_1 and S_2 respectively.

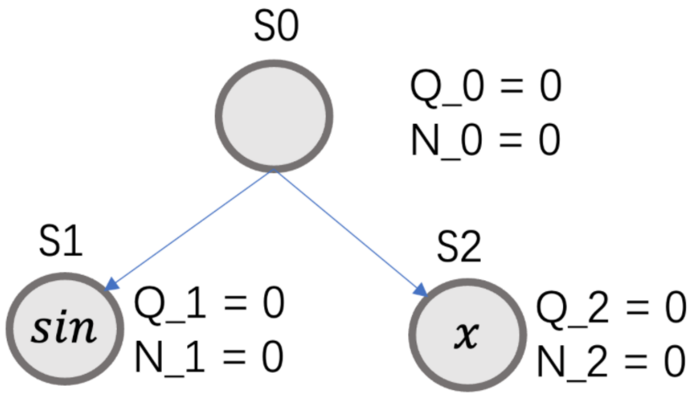


Figure I.2

You can then use the UCT formula to choose whether to extend S_1or S_2. Here $N_1$and $N_2$are both 0, so the UCT value of both nodes is infinite, so any node can be selected, here $S_1$is selected for extension and simulation (random selection of symbols). After simulation, it was found that the final reward value was 0.2, so it was updated retrospectively. $Q_1 = 0.2$,N_1= 1,Q_0=0.2,N_0= 1.

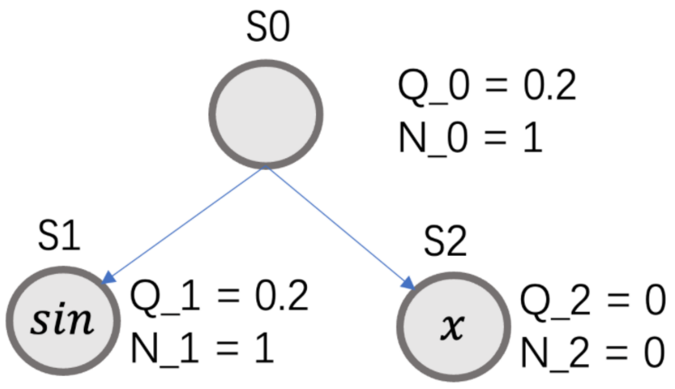


Figure I.3

The second iteration: Starting from S_0, calculate the UCT values of S_1 and S_2, and select the larger one for expansion. (assuming $S_1 > S_2$ after calculation)
Then according to the UCT value, S_1 is selected for expansion. After reaching S_1, it is found that it is a leaf node and has been explored, then enumerate all possible states of the current node (each action corresponds to a state), and add them to the tree.

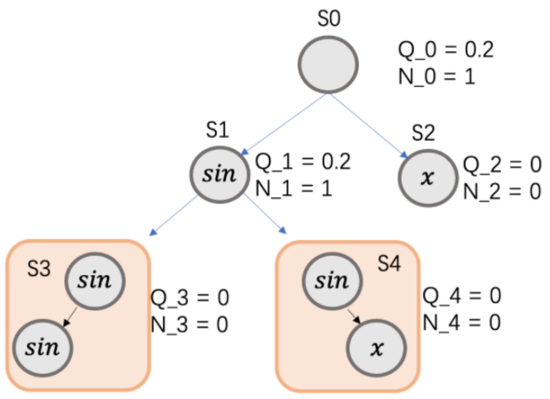


Figure I.4

Then we can select either S_3 or S_4 at random as before. Keep iterating. (In this example, S4 has successfully found the target expression)