Genetic Programming meets Linear Algebra

How genetic programming can be used to find improved iterative numerical methods

Reza Gholami M.
MAOT, FAU Erlangen-Nürnberg
91052 Erlangen, Germany
reza.gholami.mahmoodabadi@gmail.com

Harald Köstler
Department Computer Science
Friedrich-Alexander-Universität Erlangen-Nürnberg
Cauerstr. 11, 91058 Erlangen, Germany
Harald.koestler@fau.de

Abstract—Iterative schemes play central role in solving large scale simulations in science and engineering. Development of such methods over the past few hundreds of years faces inevitable difficulty of manual design. Herein, we report, for the first time, iterative schemes that are automatically evolved by genetic programming (GP) and outperform the well-known iterative methods. To cope with the diversity of the systems of linear equations, the proposed technique is applied on a sparse system in 1D and 2D domains and on a non-sparse asymmetric system. Our proof-of-principle experiments demonstrate GP evolved schemes that converge up to 4 times faster than the conventional Gauss-Seidel scheme. Our work paves the way towards automatic design of efficient iterative solvers for large scale systems of linear equations.

Keywords—genetic programming, iterative solvers, sparse linear algebra

I. Introduction

The field of computational science and engineering deals with various applications ranging from physics to engineering and life sciences. At its base there are often mathematical models that require efficient numerical methods to compute a reasonable solution. One class of models leads to the solution of partial differential equations. Here, large and typically sparse linear systems have to be solved. This is often done iteratively, i.e. one starts with some initial guess for the solution (a large vector) and then tries to improve the solution in several steps. Classical iterative methods can be understood as a function that takes as input the old solution from one or all former steps and outputs a new, improved approximation to the solution. Unfortunately, this function in general depends on the system matrix and thus is highly problem-dependent. Nevertheless, in practice a small number of such simple functions (also called iteration matrices when the functions are linear) are used for many years [3]. Standard methods are then manually adopted to solve more complicated problems [4].

In the field of Genetic programming, evolutionary algorithms are used to automatically generate programs that solve problems [5][6]. Human competitive results and patentable inventions are amongst the successful discoveries using GP [7].

GECCO '17 Companion, July 15-19, 2017, Berlin, Germany
 ACM ISBN 978-1-4503-4939-0/17/07…$15.00
http://dx.doi.org/10.1145/3067695.3082502.

In this manuscript we will try to find good iteration matrices automatically by a genetic programming approach. We concentrate on linear systems that have a unique solution. Furthermore, we are mainly interested in sparse linear systems.

In section II our MATLAB framework for genetic programming is introduced and the supported tree operations are described.

We start the section III with presenting our first proof-of-principle experiment on enhancement of the conventional iterative solvers. To demonstrate the versatility of our method, we then experiment with non-sparse and sparse systems in 1D and 2D domains. We show that solving these systems using our evolved expressions reduce the computational cost up to 4 times. Our promising results opens a new chapter in designing novel iterative numerical methods.

II. Framework Description

Here, a tree-based genetic programming framework [9][10][10], is developed in MATLAB to search for an improved iterative scheme.

Let's formulate the linear system of interest as $\boldsymbol{Ax} = \boldsymbol{b}$, where $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is the system matrix, $\boldsymbol{x} \in \mathbb{R}^{N}$ is the unknown vector and $\boldsymbol{b} \in \mathbb{R}^{N}$ is the right hand side vector. Note that $\boldsymbol{A}$ can be a non-square system matrix but in the experiments here it is a square matrix. We denote the iteration matrix with $\boldsymbol{M} \in \mathbb{R}^{N\times N}$ and it is represented as a tree in our framework. The approximated solution in the n+1 iteration is obtained using the old solution and the iteration matrix using the relation $\boldsymbol{x}^{n+1} = \boldsymbol{M}\boldsymbol{x}^{n}$.

A tree in any stage of the framework is a valid expression based on the defined matrix operations i.e. the nodes of the tree are selected such that the generated tree is consistent in terms of dimensions and the generated expressions are secured to be computable.

A. Nodes

1) Terminal set

The terminal set is composed of arbitrary symbolic matrices. In the search for an iteration matrix, the following elements are common and used in the experiments done in this work: system matrix **A**, right hand side vector **b**, the diagonal of **A**, inverse of the diagonal of **A**, **A** minus its diagonal, lower triangular part of **A**, inverse of the lower triangular part of **A** and upper triangular (1st diagonal) part of **A**. Note that the set of terminals can be extended.

2) Operation set

The operation set consists of addition, subtraction and multiplication.

B. Growing a tree

A tree in our framework represents an iteration matrix that will be applied on the approximated solution vector $\boldsymbol{x}$ iteratively. Dimensions of $\boldsymbol{x}$ and $\boldsymbol{b}$ set constraints on the dimension of the iteration matrix and the root node of the tree that represents this matrix. Initially the dimension of the root is taken from the dimension of the desired iteration matrix. Starting from the root, a node in any depth less than the tree depth can be a terminal or an operator. This node can be a terminal with the probability of 1/3 or an operator with the probability of 2/3. Note that the nodes at the tree depth can only be a terminal.

When the tree is growing deeper and deeper, the dimension of the nodes has to be compatible with the operations and the terminals in the upper layer. Therefore, a node is selected to be a terminal it can only take a specific dimension based on its position in the tree. Therefore, this node will be a random terminal out of all terminals that have this specific dimension.

C. Fitness Evaluation

The symbolic matrix corresponding to the genetic expression of an individual is simplified using the symbolic Math Toolbox of MATLAB and then evaluated to get the numerical version of the iteration matrix.

An iterative scheme converges to the solution of a linear system if and only if the spectral radius of the iteration matrix is less than one. The spectral radius of a matrix is the maximum absolute value of Eigen values of the matrix. As the spectral radius decreases from 1 to 0, the iterative scheme converges faster to the final solution.

Let **r** be the spectral radius of the numerical iteration matrix of an individual then, the fitness function **f** for r < 1 is defined as $\boldsymbol{f}(r) = \frac{1}{(1-r^2)}$ and for $r \geq 1$ is defined as $\boldsymbol{f}(r) = a_0 + a_1 r$ where a0 and a1 are arbitrary penalty parameters larger than 1. The fitness function here is designed to non-linearly favor the iteration matrices with lower spectral radius.

D. Selection

Candidate individuals are selected using the fitness proportionate selection.

E. R*ecombination*

Crossover is used to exchange genes between a pair of selected individuals to produce an offspring. Initially dimensions of all the nodes in the selected trees (parents) are inspected and a pool of possible crossover points are formed.

A random crossover point is take out of the pool and the genes of one of the parents are substituted with the genes of the other parent.

F. Gene variation

In this step, a pool of all the genes of all the individuals in the population is formed. The probability of the mutation of a gene of an individual in the pool is equal to the mutation rate in the experiment.

The gene selected to undergo mutation is removed along with its sub nodes from the tree. The last step of the gene variation step is to complete the tree (from where it is cut in the previous step) using the growing tree procedure.

III. EXPERIMENTAL RESULTS

The first experiment is the Poisson equation (PE) in 1D. PE is broadly used in science and engineering to describe various phenomena such as heat transfer and electrostatics. This simple case allows us to print the GP-evolved symbolic expressions of interest. The next two experiments are the Poisson equation in 2D for two different number of inner points. The last experiment demonstrates the applicability of the framework on a non-sparse asymmetric system.

For an expression $\boldsymbol{M}$, the relaxation error in the nth iteration is defined as $||\boldsymbol{M}\boldsymbol{x}^{n-1}||_1$. Early results have shown that the starting value does not affect the results, therefore a vector of ones is taken as the initial value of $\boldsymbol{x^0}$. The relaxation error of the iteration schemes are then plotted versus the iteration number in each experiment. The stopping limit of the iteration is when the l_1-norm of the error is less than 1e-9.

In each experiment a fine tuning on the experimental parameters is done to achieve an observable improvement in the fitness value of the individuals.

A. Experiment 1: Poisson equation in 1D

In this experiment the Poisson equation in 1D is taken for 1 inner point with operators discretized using finite differences. The system matrix and vectors are given in the following:

$$\boldsymbol{A} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} 4 \\ 0 \\ -2 \end{bmatrix} \text{ and } \boldsymbol{x} = \begin{bmatrix} 2.5 \\ 1.0 \\ -0.5 \end{bmatrix}.$$

The iteration matrix of the Gauss Seidel (GS) method has the spectral radius of 0.5.

The proposed GP framework is used on this system with the following setup: population size of $s = 30$, tree depth of $d = 3$, GP iteration number $n = 5$, crossover rate of $c = .1$ and mutation rate $m = .1$.

Among the population evolved in the GP search, individuals were found with expressions of spectral radii 0.3125 and 0.1250. The evolved expressions are symbolic and they are then simplified to obtain the final iteration matrices.

The row vectors of the evolved iteration matrix $\boldsymbol{M}$ (with $r = 0.3125$) are

$$\boldsymbol{M_1} = [\frac{a_{11}^2 + 1}{a_{11}^4}, 0, 0],$$

$$\boldsymbol{M_2} = [-\frac{a_{21}(a_{11}^2 a_{22}^2 + a_{11}^2 + a_{11}a_{22}^3 + a_{22}^2)}{a_{11}^4 a_{22}^3}, \frac{a_{22}^2 + 1}{a_{22}^4}, 0],$$

$$\boldsymbol{M_3} = [\frac{(a_{33}^2 + 1)(a_{21}a_{32} - a_{22}a_{31})}{a_{11}^2 a_{22}a_{33}^3} + \frac{(a_{11}a_{33} + 1)(a_{21}a_{32} - a_{22}a_{31})}{a_{11}^4 a_{22}a_{33}} + \frac{a_{21}a_{32}(a_{22}a_{33}+1)}{a_{11}^2 a_{22}^3 a_{33}}, -\frac{a_{32}(a_{22}^2 a_{33}^2 + a_{22}^2 + a_{22}a_{33}^3 + a_{33}^2)}{a_{22}^4 a_{33}^3}, \frac{a_{33}^2 + 1}{a_{33}^4}]$$

, where a_{ij} is the (i, j) element of the matrix $\boldsymbol{A}$.

The row vectors of the evolved iteration matrix $\boldsymbol{M}$ (with $r = 0.1250$) are

$$\boldsymbol{M_1} = [\frac{a_{22}a_{33} - a_{11}a_{12}a_{21}a_{33} + a_{11}a_{13}a_{21}a_{32} - a_{11}a_{13}a_{22}a_{31}}{a_{11}^3 a_{22}a_{33}}, \frac{a_{12}a_{33} - a_{13}a_{32}}{a_{11}a_{22}a_{33}}, a_{13}/(a_{11}a_{33})],$$

$$\boldsymbol{M_2} = [-\frac{a_{23}a_{31}a_{22}^2 - a_{21}a_{23}a_{32}a_{22} + a_{21}a_{33}}{a_{11}a_{22}^3 a_{33}}, \frac{a_{33} - a_{22}a_{23}a_{32}}{a_{22}^3 a_{33}}, \frac{a_{23}}{a_{22}a_{33}},$$

$\boldsymbol{M_3} = [\frac{a_{21}a_{32} - a_{22}a_{31}}{a_{11}a_{22}a_{33}^3}, -\frac{a_{32}}{a_{22}a_{33}^3}, \frac{1}{a_{33}^3}]$.

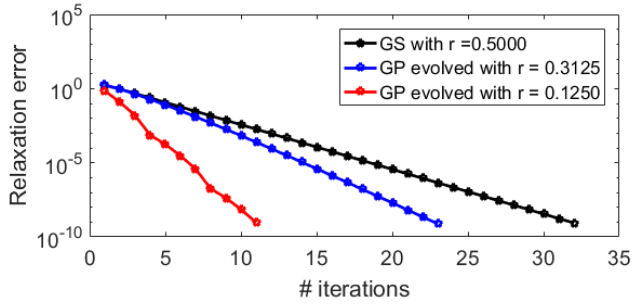

Fig. 1. The GS method hits the stopping limit in 32 iterations while the GP evolved expressions hit the limit in 23 and 11 iterations respectively.

The GP evolved expression with the spectral radius of about the half of that of the GS method hits the stopping limit with a 3rd number of iterations that GS expression does.

B. Experiment 2: Poisson equation in 2D with 3 inner points

Here, a 2D domain with 3 inner points is solved for the Poisson equation. The boundary value at a boundary point (x, y) is $g(x, y) = \cos(\pi x) + \cos(\pi y)$. The system matrix is

$$\boldsymbol{A} = \begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & -1 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix}$$ and the column vectors are $\boldsymbol{b}$ = [1.7, 0.0, -1.4, 0.7, 0, 0.0, -0.7,0.0, 1.4] and $\boldsymbol{x}$ = [53, 9, -31, 33, 15, 5, -6, 12, 39] e-1.

The solution x in 2D is plotted in the following figure.

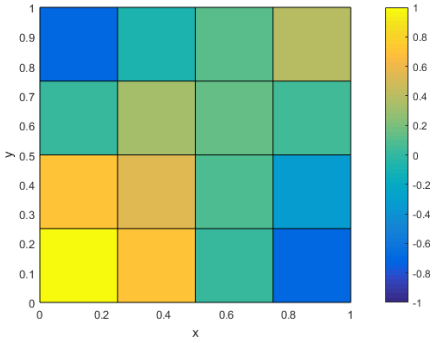

Fig. 2. The solution of the Poisson equation in 2D with total points of 5 and 3 inner points in each dimension. The colorbar shows the value of the solution of the Poisson equation at a specific grid point. The boundary values are calculated from the function g. Note that how column vectors $\boldsymbol{x}$, $\boldsymbol{b}$ and the system matrix $\boldsymbol{A}$ are filled to represent the 2D problem.

The iteration matrix of the Gauss Seidel method has the spectral radius of 0.5. The proposed GP framework is used on this system with the following setup: s=30, d=3, n=5, c= .1 and m= .1. An evolved expression for the iteration matrix with spectral radius of 0.06 is found. The symbolic expressions of the matrix elements are very long and therefore the raw vectors of its numerical version are given in the following:

$\boldsymbol{M_1}$ = [-6, 0, 0, 0, 0, 0, 0, 0, 0]e-2 , $\boldsymbol{M_2}$ = [-9, -6, 0, 0, 0, 0, 0, 0, 0]e-2 , $\boldsymbol{M_3}$ = [-3, -9, -6, 0, 0, 0, 0, 0, 0]e-2 , $\boldsymbol{M_4}$ = [-9, 0, 0, -6, 0, 0, 0, 0, 0]e-2 , $\boldsymbol{M_5}$ = [-5, -9, 0, -9, -6, 0, 0, 0, 0]e-2 , $\boldsymbol{M_6}$ = [-2, -5, -9, -2, -9, -6, 0, 0, 0]e-2 , $\boldsymbol{M_7}$ = [-3, 0, 0, -10, 0, 0, -6, 0, 0]e-2 , $\boldsymbol{M_8}$ = [-2, -3, 0, -5, -10, 0, -10, -6, 0]e-2 , $\boldsymbol{M_9}$ = [-1, -2, -3, -2, -5, -10, -3, -10, -6]e-2.

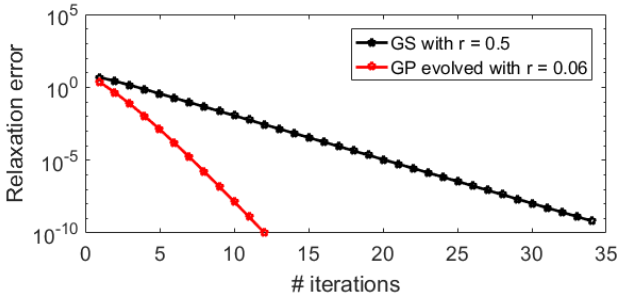

Fig. 3. The GS method hits the stopping limit in 34 iterations while the GP evolved expression hits it in 12 iterations.

This experiment is designed to demostrate the applicability of the framework to a real 2D application where the GP evolved expression needs only a 3rd number of iterations of what the GS expression needs to hit the same error.

C. Experiment 3: Poisson equation in 2D with 4 inner points

In this experiment the Poisson equation is solved in 2D for a domain of 4 inner points and in total 16 points including the boundary points. The system matrix has the same pattern as the one in the last experiment, and the same boundary condition is used to fill the column vector $\boldsymbol{b}$.

The iteration matrix of the Gauss Seidel method has the spectral radius of 0.6545. The proposed GP framework is used on this system with the following setup: s=40, d=4, n=10, c= .1 and m= .1. An evolved expression for the iteration matrix with spectral radius of 0.1636 is found and plotted in the following figure to compare with that of GS method.

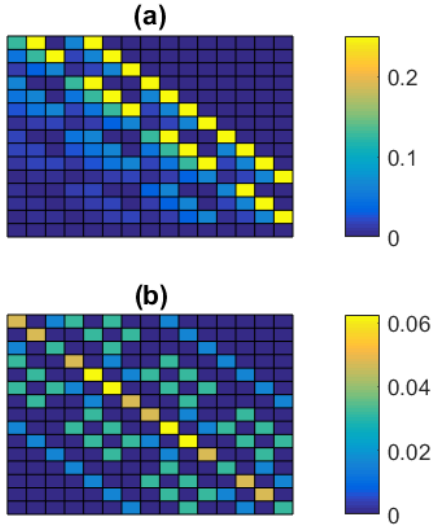

Fig. 4. The iteration matrices, (a) for GS method with the spectral radius of 0.6545 and (b) for the GP evolved expression for the iteration matrix with the spectral radius of 0.1636. The colorbars show the absolute values of the entries of the matrices and are scaled individually to the sub figures.

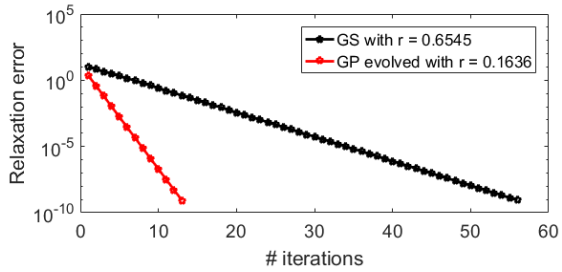

Fig. 5. The GS method hits the stopping limit in 56 iterations while the GP evolved expression hits it in 13 iterations.

Only one inner point is added to the domain in this experiment compare to the previous experiment and the GS method now requires double the iteration number to hit the stopping limit while the GP evolved expression reaches the stopping limit in the same number of iterations as before.

D. Experiment 4 A non-sparse assymetric system

In this example the proposed GP framework is applied on the following non-sparse asymmetric system with raw vectors

$\boldsymbol{A_1}$ = [8, 2, 7, 2, 2], $\boldsymbol{A_2}$ = [2, 7, 5, 3, 2], $\boldsymbol{A_3}$ = [3, 4, 8, 3, 3], $\boldsymbol{A_4}$ = [2, 1, 3, 8, 3], $\boldsymbol{A_5}$ = [2, 3, 5, 1, 8] and the column vector $\boldsymbol{b}$ = [105, 108, 100, 56, 82].

The iteration matrix of the Gauss Seidel method has the spectral radius of 0.4574. The proposed GP framework is used on this system with the following setup: s=40, d=4, n=10, c= .1 and m= .1. The row vectors of the evolved iteration matrix $\boldsymbol{M}$ (with $r = 0.1385$) are

$$\boldsymbol{M_1} = \left[\frac{1}{a_{11}}, 0, 0, 0, 0\right], \boldsymbol{M_2} = \left[0, \frac{1}{a_{22}}, 0, 0, \quad 0\right], \boldsymbol{M_3} = \left[0, 0, \frac{1}{a_{33}}, 0, 0\right], \boldsymbol{M_4} = \left[0, 0, 0, \frac{1}{a_{44}}, 0\right], \boldsymbol{M_5} = \left[0, 0, 0, 0, \frac{1}{a_{55}}\right].$$

The evolved expression is a common and very simple pre-conditioner. Note that $a_{i,j}$ (for $i = j$) are diagonal elements of the system matrix and the GP evolved expression is just the inverse of the diagonal of the system matrix.

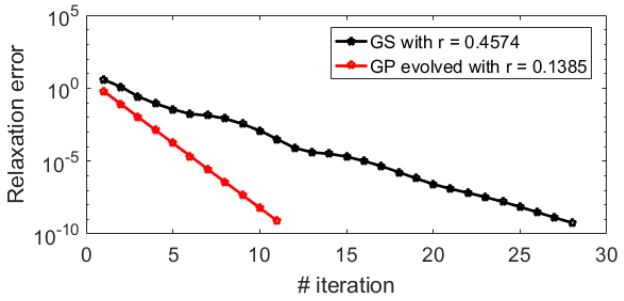

Fig. 6. The GS method hits the stopping limit in 28 iterations while the GP evolved expression hits it in 11 iterations.

The GP evolved expression converges three times faster than the GS scheme. This experiment for a non-sparse assymetric system matrix shows that not only the GP evolved expression is a very well-known preconditioner but also it is very sparse and computationally efficient.

IV. Discussion and Future Work

The experiments in this work confirmed that the GP evolved solvers can outperform the well-known methods and converge up to 4 times faster comparatively. However, although current results are promising we are just at the beginning of the research in this area.

What we plan next is to investigate larger system matrices and to incorporate the GP framework into the ExaStencils solver framework (www.exastencils.org), where we develop a domain specific language for the numerical solution of partial differential equations[1][2].

Acknowledgment

The authors would like to thank Jonas Schmitt for the valuable comments. The project ExaStencils is part of the DFG Priority Research Initiative SPPEXA, grant no. LE 912/15-1.

References

[1] Lengauer, Christian, et al. "ExaStencils: advanced stencil-code engineering."European Conference on Parallel Processing. Springer International Publishing, 2014.

[2] Harald Köstler, Christian Schmitt, Sebastian Kuckuk, Stefan Kronawitter, Frank Hannig, Jürgen Teich, Ulrich Rüde, and Christian Lengauer, “A Scala Prototype to Generate Multigrid Solver Implementations for Different Problems and Target Multi-Core Platforms,” International Journal of Computational Science and Engineering (IJCSE), vol. 14(2), pp. 150–163, 2017.

[3] Saad, Yousef, “Iterative methods for sparse linear systems,” Society for Industrial and Applied Mathematics, 2003.

[4] Barrett, Richard, et al., “Templates for the solution of linear systems: building blocks for iterative methods,” Society for Industrial and Applied Mathematics, 1994.

[5] J.R. Koza, “Gentetic Programming: On the Programming of Computers by Means of Natural Selection,” MIT Press, Cambridge, 1992.

[6] J.R. Koza, ‘Genetic Programming II: Automatic Discovery of Reusable Programs,” MIT Press, Cambridge, Massachusetts, 1994.

[7] R. Poli, W. B. Langdon, N. F. McPhee, “A Field Guide to Genetic Programming,” Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk, 2008. (With contributions by J.R. Koza)

[8] D. E. Goldberg, “Genetic Algorithms in Search, Optimization,and Machine Learning,” Addison-We. 1989.

[9] L. Davis, ‘Handbook of Genetic Algorithms,’’ 1991.

[10] R. L. Haupt and S. E. Haupt, ”Practical Genetic Algorithms,” 2004.