MATHEMATICAL SOFTWARE PROGRAMMING

Least-Squares Solver

Alexandros Arnogiannakis

20/11/2023

1 High-Level Functionality and Features

This project focuses on creating a command-line tool that efficiently solves the least-squares problem by utilizing the LAPACK library's DGELS routine. The tool is designed to read a matrix A and a vector b from text files and processes them to find an optimal solution. Upon successful execution, it outputs two key metrics - the residual norm and the coefficient of determination (r^2) to the standard output (stdout), providing users with valuable insights into the accuracy and effectiveness of the solution.

1.1 Implementation Overview

The tool is structured around two main components:

1. **lssolve.c**: This serves as the primary interface of the program. It is responsible for validating user inputs and managing the integration of external libraries. Its key role is to ensure that the input data is in the correct format and that all necessary prerequisites are met before proceeding with the computations.

2. **call_dgels.c**: This component is at the heart of the computational process. It leverages the DGELS routine from the LAPACK library to perform the intricate calculations required for solving the least-squares problem. This function not only handles the mathematical computations but also returns crucial information such as the residual norm and the coefficient of determination (r^2) through pointers.

Throughout this project, we employed advanced data structures and routines for dynamic and multidimensional arrays from MSP Tools (msptools), developed by Martin S. Andersen. This library was crucial for effective array handling and numerical computations, enhancing our tool's functionality and performance. For more details on msptools, visit Martin S. Andersen's GitLab (Copyright 2021 Martin S. Andersen).

1.2 Technical Challenges and Implementation Strategies

A primary challenge encountered during the project was implementing the DGELS routine from the LAPACK library, which by default assumes column-major ordering for matrix elements. The task required adapting this routine to accommodate matrices in RowMajor format. To address this, we implemented logic to detect the storage order of the input matrix and adjust our approach accordingly. The crucial changes involved:

- Swapping rows and columns dimensions when processing a row-major matrix.
- Adjusting the leading dimension parameter to reflect the matrix's actual layout in memory.

Another significant hurdle was the calculation of the coefficient of determination (r^2). The DGELS routine modifies the input vector b by overwriting it with the solution and residuals of the least squares calculation. This modification necessitated creating a copy of the original vector b to preserve its initial values. We achieved this through dynamic memory allocation and introduced a function, `dcemv`, to center the vector and compute the squared L2 norm. This approach was inspired by an exercise from Week 4 on Centering Matrices.

A critical aspect of the implementation was the robust calculation of r^2. Initially, we encountered difficulties when the residual norm pointer was NULL, leading to incorrect calculations. To resolve this, we computed the residuals separately and only assigned them to the pointer if it was non-NULL, ensuring accurate computation of r^2.

Finally, the project demanded meticulous memory management. We made sure to free all dynamically allocated memory at the end of the program to prevent memory leaks, thus ensuring the efficiency and reliability of our solution.

1.3 Testing Strategy

To ensure the robustness and correctness of our implementation, we adopted a comprehensive testing strategy. This encompassed a variety of checks and validations, both in local environments and on DTU Compute's Autolab platform.

1. **Error Handling:** We implemented specific return values to communicate distinct error conditions. For instance, a return value of -13 indicates an unacceptable scenario where the number of rows (m) is less than the number of columns (n), adhering to our assumption of an over-determined system. Additionally, we ensured that the residual norm for square matrices was correctly set to 0.

2. **Local Testing:** This involved validating the program against a range of matrices and vectors, ensuring alignment with reference values. Key checks included dimensionality validation, input error handling, and consistent behavior across row-major and column-major matrix formats.

3. **Function-Specific Testing:** In the `call_dgels.c` function, we rigorously tested for null pointers, compatibility of matrix and vector dimensions, and correct handling of the workspace storage required by the DGELS routine.

4. **Rank Deficiency:** The DGELS routine's `info` pointer plays a crucial role in identifying rank deficiencies. Although a non-zero return value from `info` suggests rank deficiency, this is not an exhaustive method. Ideally, rank deficiency should be confirmed through additional techniques such as singular value decomposition (SVD). This area remains open for more in-depth testing to further ascertain the reliability of our implementation.

Our approach to testing was meticulous, striving to cover all possible scenarios and edge cases. This diligence paid off, as our code successfully navigated through the comprehensive test suite on Autolab, validating its effectiveness and reliability.

1.4 Modified/Overwritten Input Arguments

In our implementation of the DGELS routine, special attention is paid to the handling of input arguments, particularly since they are pointers and thus susceptible to being modified. Key aspects of this handling are outlined below:

- **Vector b:** The input vector b undergoes modification during the computation. Post-execution, it contains the optimal solution x^*. It's important to note that while b initially holds the input vector for the least-squares problem, it is overwritten by the solution vector by the end of the routine. This overwriting is a crucial aspect, as the latter part of the vector b may contain residuals that are instrumental in computing the least-squares solution.
- **Matrix A:** Our version of the DGELS routine also modifies the input matrix A, replacing it with its QR factorization. This modification of A is an inherent part of the DGELS algorithm but is not explicitly utilized or captured in our current implementation. However, if required, the functionality to retain or utilize the modified state of A can be seamlessly integrated.
- **Residual and r^2 Values:** The function is designed to update the residual norm and coefficient of determination (r^2) values. These updates occur only if the corresponding pointers (resnorm and rsquared) are provided by the user. If these pointers are NULL, the function omits the calculation and updating of these values, ensuring flexibility based on user requirements.

Overall, our implementation ensures that any modifications to input arguments are deliberate and aligned with the functional requirements of the least-squares problem-solving process.

1.5 Conclusions

In summary, this project successfully demonstrates the effective application of the LAPACK DGELS routine in solving least-squares problems with meticulous handling of input arguments and memory. The implementation accommodates both row-major and column-major matrix formats, ensuring versatility and robustness. Rigorous testing and thoughtful error handling further reinforce the reliability and precision of the tool, making it a valuable asset for mathematical computations in various contexts.