APPM 4650 — Project list

Project 1: Hybrid and alternative root finding techniques

Consider the task of finding the roots of $f(x)$ where f is a scalar function. In this class, we discussed a small subset of the numerical methods that can be used to find the roots. In this project, you will make a table reviewing the methods we developed in class and stating when to apply each one. Once you have this table, you will combine the different methods to help improve the chances of convergence. For example, one could use bisection until the midpoint lies in the basin of convergence for Netwon's method. At that point, the new method would switch to Newton's method. This new method is second order convergent and converges for a larger set of initial guesses than Newton's method alone. For the independent part of this project, you will explore other root finding techniques which were not presented in class. Your report will include derivations, convergence information, and numerical experiments. You will also update the table you made originally to include the new methods.

Project 2: Sensitivity of discrete least squares

In many applications the data is noisy; i.e. the data is not exactly the data you want to fix. In this project, your task is to explore the sensitivity of the discrete least squares approximation to the noise in the data. Does the accuracy depend on the number of data points and/or the size of the noise? Are there other things that come into play? Find theory in the literature to support your findings. Possible options for the independent part include techniques for pre-processing data to make it less noisy and least squares techniques that make it more robust to noise and outliers (e.g. smoothing).

References:

- *Golub, Van Loan*, Matrix Computations, Chapters 5 (Ordinary Least Squares) and 6 (Modified Least Squares).
- *Demmel*, Applied Numerical Linear Algebra, Chapter 3: *Linear Least Squares problems*
- *Hastie, Tibishirani*, The Elements of Statistical Learning, Chapter 3: *Linear Methods for Regression*.

Project 3: Regularization in least squares

Regularized least squares is a family of methods that adds terms to the least squares formulations to constrain its solution. For example, in ridge regression, the modified problem reads:

$$\min \sum_{i=1}^{n} (y_i - \sum_j a_j \Phi_j(x_i))^2 + \lambda^2 \sum_{i=1}^{n} a_j^2$$

or, equivalently,

$$\min ||\vec{y} - M\vec{a}||_2^2 + \lambda^2 ||\vec{a}||_2^2$$

This is useful when the number of variables in the linear system exceeds the number of observations (under-determined case) and also when the least squares problem is ill-posed (doing the normal fit is very unstable). Regularization can also help us incorporate assumptions about the function we are using to fit the data, such as the size of the coefficients, sparsity (how many are non-zero), smoothness of our model, etc.

In this project, you will explore the least squares regularization techniques used in ridge regression and Tikhonov regularization. How do the penalties affect the solution of the LS problem in terms of normal equations and other linear algebra techniques? Find examples of problems from applications where these techniques are useful, and explain what criteria is used to choose the constant λ (ridge regression), or the matrix Γ in (Tikhonov regularization). Possible options for extensions include generalizations of Tikhonov, ℓ_1 regularization (e.g. LASSO), continuous regularized least squares, and smooth spline approximation (Reproducing Kernel Hilbert Spaces).

References:

- *Golub, Van Loan*, Matrix Computations, Chapters 5 (Ordinary Least Squares) and 6 (Modified Least Squares).
- *Golub, Hansen, O'Leary*, Tikhonov Regularization and Total Least Squares, SIAM JMAA, 1999.
- *Demmel*, Applied Numerical Linear Algebra, Chapter 3: *Linear Least Squares problems*
- *Hastie, Tibishirani*, The Elements of Statistical Learning, Chapter 3: *Linear Methods for Regression*.

Project 4: Approximation techniques

For many applications a minimax approximation is desirable because the error is uniformly bounded in the entire interval of interest. In this project, you will explore the minimax approximation deriving it and the associated conditions. Then you will find the link between minimax approximations and the Chebychev polynomials. Additionally you can explore with prepackaged software the different in performance minimax approximations and the least squares approximations. You are free to come up with ideas for the independent part of the project.

References:

- *Phillips, George*, Interpolation and approximation by polynomials, Chapter 2.4
- NIST, Digital Library of Mathematical Functions, Section 3.11 Approximation Techniques.
- *Yeates, Anthony*, Lectures on Approximation theory (and references therein)

Project 5: Orthogonal polynomials

Orthogonal polynomials have lots of fun properties. One of the lesser used properties is that that roots are nested. Specifically, let $P_n(x)$ denote an orthogonal polynomial on $[a, b]$ and let $x_{n,i}$

denote the roots of $P_n(x)$. Then the roots of $P_{n+1}(x)$ satisfy $x_{n,i-1} < x_{n+1,i} < x_{n,i}$. For the points near the end they are bounded by the endpoints of the intervals. In this project, you will prove the property about the nested roots and derive an efficient and robust algorithms for evaluating the roots of arbitrary orthogonal polynomials which makes use of the three term recursion and other properties of orthogonal polynomials. One possible option for the independent part of the project include developing robust and efficient techniques for evaluating the associated quadrature weights and building from your algorithm for general weight functions to get an efficient least squares approximation routine. Another possible option is to develop a different way of evaluating the quadrature nodes based on the eigenvalues of the tridiagonal matrix that is associated with the three term recursion formula for orthogonal polynomials.

References:

- *Szego*, Orthogonal Polynomials, AMS.
- *Narla*, Orthogonal polynomials, an illustrated guide.
- NIST, Digital Library of Mathematical Functions, Section 18 Orthogonal Polynomials.

Project 6: Nonlinear systems and variations of Newton's method

We discussed various quasi-Newton methods in class. These always use an approximate of the Jacobian at a step. In this project, explore the option of introducing updated Jacobians. When is a good time to introduce an updated Jacobian? How can you adapt this so that it saves you from the iteration not converging? For the independent study, explore other variations of Newtons method that guarantee convergence and/or are more robust variants of quasi-Newtons method.

References:

- *Nocedal, Wright* Numerical Optimization, Chapters 6 (Practical Newton methods) and 8 (Quasi-Newton methods).
- *Martinez*, Practical Quasi-Newton methods for solving nonlinear systems, JCAM 2000.
- *Dembo, Eisenstat, Steinhaug*, Inexact Newton Methods, SIAM Journal on Numerical Analysis, 1982.

Project 7: Steepest Descent and the Conjugate Gradient method

In class we talked about the steepest descent algorithm and it's connection to minimizing a quadratic. Another option (when the corresponding matrix is symmetric) is the conjugate gradient method. In this project, you will derive the conjugate gradient method and compare the performance of the two methods for a collection of problems. For the independent portion of the project, you will explore alternatives. This can include for non-symmetric matrices.

References:

- *Nocedal, Wright* Numerical Optimization, Chapter 5 (Conjugate Gradient method)

- *Saad*, Iterative methods for sparse linear systems, SIAM.
- *Gutknecht*, A Brief Introduction to Krylov Space Methods for Solving Linear Systems, Frontiers of Computational Science.

Project 8: Robust methods for rootfinding and optimization

In class, we discussed the steepest descent method applied to the rootfinding problem $F(\vec{x}) = 0$. After re-writing it in terms of finding a minimum for the quadratic $g(\vec{x}) = \sum_{j=1}^{n} F_j(\vec{x})^2$, we derived the steepest descent method, with an iteration of the form:

$$x_{n+1} = x_n - \alpha \nabla g(x_n)$$

There are various methods to choose this α: in our sections, we discussed finding it as the minimizer of a 1D quadratic, or using a back-tracking line-search method (with *Armijo* or *Wolfe* conditions). In this project, you will explore and compare methods to choose α. For one or more problems, discuss how applying these makes the steepest descent method more robust. Which method seems to perform best in terms of number of iterations and number of function evaluations?

For the independent study, you can explore how these methods may be applied to make other rootfinding or optimization methods more robust (e.g. Newton, Broyden and other Quasi-Newton), variations of these (e.g. derivative-free line search methods) or alternatives such as trust-region methods.

References:

- *Nocedal, Wright* Numerical Optimization, Chapters 3 (line-search methods) and 4 (trust region methods)
- *Boyd, Vandenberghe*, Convex Optimization (Book and lecture slides are free online).
- *Griewank* The "global" convergence of Broyden-like methods with suitable line search, 1986.

Project 9: Rank revealing QR factorization

In class, we talked about the QR factorization. In many applications, it useful to create a low rank factorization which approximates a matrix to a user prescribed accuracy. One of the most popular techniques for doing this is the rank revealing QR factorization. In fact, it is becoming a driving force in machine learning and reduced order modeling. In this project, you will learn about QR factorizations with pivoting and investigate the different rank revealing QR algorithms available. You will determine which methods are the easiest to implement, are most stable, most likely to succeed at their given task, etc.

References:

- *Gu, Eisenstat* Efficient algorithms for computing a strong, rank-revealing QR factorization, SIAM J. Sci. Comput. 1996.
- *Chandrasekaran, Ipsen*, On Rank-Revealing Factorizations, SIAM JMAA, 1994.

- *Higham*, What is a Rank-Revealing factorization?

Project 10: Fast Fourier transform

In class, we talked about trigonometric interpolation. It turns out that when this interpolation involves uniformly distributed points, the cost of doing the interpolation can be reduced from $O(n^2)$ to $O(n\log(n))$. The technique for doing this is called the fast Fourier transform (FFT) and it exploits the fact that you can write the evaluation as beautifully nested series that all look the same but are scaled by a constant. In this project you will actually derive it and investigate its asymptotic behavior. Next you will learn the non-uniform FFT. What are the challenges of this method? How does it perform relative to the uniform FFT? What are recent developments on these methods?

References

- *Van Loan*, Computational frameworks for the Fast Fourier Transform, Frontiers in Applied Mathematics, 1992.
- *Kong*, Python Programming and Numerical Methods, Chapter 24 (Fourier Transform)
- *Alok, Rokhlin* "Fast Fourier Transforms for Nonequispaced Data". SIAM Journal on Scientific Computing, 1993.
- *Lee, Greengard,* "The type 3 nonuniform FFT and its applications". Journal of Computational Physics, 2005.

Project 11: Quadrature rules for weakly singular integrals

Consider the problem of finding the definite integral of a function $f(x)$ in the interval $[a, b]$ with an integrable singularity at $x = a$ (the improper integral converges). Assume you know what kind of singularity it is, e.g. you can write $f(x) = \frac{q(x)}{(x-a)^\alpha} + r(x)$ for $\alpha \in (0, 1)$, and $q(x), r(x)$ analytic. In this project you will derive an algorithm building quadrature to approximate the integrals of these weakly singular functions.

In class, we went over the Gaussian quadrature rule, and how it is optimal in that, using n points, it is exact for polynomials of degree $2n-1$. We built this quadrature by choosing the zeroes of a family of orthogonal polynomials (Legendre). The techniques you will explore in this project are a generalization of Gaussian quadrature tailored to evaluate weakly singular integrals. It builds the Gaussian quadrature points and weights to evaluate integrals of the form

$$\int_a^b w(x)p(x)s(x)dx$$

exactly for polynomials $p(x)$ of degree $\leq 2n-1$ and weight function $w(x)$ which contains the weak singularity. For at least one type of singularity (choice of $w(x)$), you will explore how the rules which need to be enforced in order to build the quadrature and develop algorithms for computing the nodes and weights. What integrals does this rule help with, and how is this useful in applications? For the independent part, you can explore extensions of this idea to other singular (or hypersingular) integrals, to higher dimensions, or alternative techniques for singular integrals (e.g. corrected trapezoidal rules, singularity subtraction).

References:

- *Kolm, Rokhlin*, Numerical Quadratures for Singular and Hypersingular Integrals, Research Report, Yale, 2000.
- *Kaneko, Xu*, Gauss-Type Quadratures for Weakly Singular Integrals and their Application to Fredholm Integral Equations of the Second Kind, AMS Mathematics of Computation, 1994.
- *Szego*, Orthogonal Polynomials, AMS.
- *Narla*, Orthogonal polynomials, an illustrated guide.

Project 12: Sparse solvers

In many applications (including constructing cubic splines, the finite difference and finite element discretization of a differential equation) involving inverting a sparse linear system. A matrix A is called sparse if the number of nonzero entries is significantly smaller than the number of zero entries in the matrix. For example, the matrix that results from building cubic splines is tridiagonal. In this project, you will derive the Thomas algorithm which is able to solve a linear system involving a tridiagonal matrix with linear cost. Then you will investigate the sparse solver called nested dissection (multifrontal method). These methods are designed to minimize fill in while creating the LU factorization of the sparse matrix. You will explore some of the different pivoting techniques and investigate where these methods are used in practice and their stability.

References:

- *Golub, Van Loan*, Matrix Computations, Chapter 11, Large Sparse Linear System Problems.
- The Thomas algorithm: an overview
- Sparse Matrix Matlab tutorial and Sparse matrix Scipy tutorial
- Lecture: Sparse-Direct Solvers

Project 13: Conditioning of linear systems

The condition number of a square matrix A is defined as $\kappa(\mathsf{A}) = \|\mathsf{A}\|_2\|\mathsf{A}^{-1}\|_2$. This condition number has an impact on how well we can construct solutions to linear systems that require the inverse of A. In this project, you will derive the (relative) condition number of solving a linear system. This will involve $\kappa(\mathsf{A})$. Then you will explore how sensitive solving a linear system is relative to the condition number. The lowest the condition number $\kappa(\mathsf{A})$ can be is 1. In practice, people spend a lot of effort in trying to construct preconditioners which result in having to solve a better conditioned linear system. For example, suppose our goal is to solve $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ but $\kappa(\mathsf{A})$ is large. Then we would want to construct a matrix M such that the problem $\mathsf{MA}\boldsymbol{x} = \mathsf{M}\boldsymbol{b}$ has a small condition number. In the extension of this project, you will explore different techniques for building preconditioners. Are there preconditioners that work well for all systems or are they all problem specific? Pick a few an explore.

References

- *Bradie*, A friendly introduction to numerical analysis, Chapter 3.4.
- *Atkinson*, An introduction to numerical analysis, Chapter 8.
- *Benzi* Preconditioning Techniques for Large Linear Systems: A Survey, JCP 2002.
- *Wathen* Preconditioning, Acta Numerica.

Project 14: Integral equations

Many boundary value problems (differential equations) can be recast as integral equations. In this project, you will explore the recasting of a 1D boundary value problem as an integral equation and a two dimensional Laplace boundary value problem with Dirichlet boundary data as an integral equation. Once you have the integral equation, you need to approximate the integral in order to obtain an approximate solution to the boundary value problem. You will utilize the appropriate numerical technique from class to do this and solve a collection of boundary value problems. In the extension of the project, you can explore many things including what happens if you change the type of boundary data you have, and fast techniques for inverting the dense matrix that results from the discretization of the integral equation.

References

- *Kress*, Linear integral equations, Springer.
- Flatiron Institute, Resources on Boundary Integral Equations.

Project 15: Interpolation/Approximation with other bases

In class, we talked about interpolating and approximating functions with polynomials, Pade approximations, and trigonmetric approximations. In this project, you will explore approximating functions with different bases. Some options include sums of Gaussians and sums of rational functions. For the extension of the project, you can explore different approximation techniques or apply the techniques to different data sets and determine when you should use certain methods to create an approximation.

References:

- *Belykin*, Approximation by exponential sums revisited, SIAM Journal on Scientific Computing.
- *Stefano Costa, Lloyd N. Trefethe,*, AAA-least squares rational approximation and solution of Laplace problems.
- *Anil Damle, Gregory Beylkin, Terry Haut, Lucas Monzon*, Near optimal rational approximations of large data sets.
- *Brandon A Jones, Gregory Beylkin, George H Born, Robert S Provence*, A multiresolution model for small-body gravity estimation.

Some other interesting papers

LN Trefethen, Exactness of quadrature formulas.

Lloyd N Trefethen, Yuji Nakatsukasa, JAC Weideman, Exponential node clustering at singularities for rational approximation, quadrature, and PDEs.