

Computational Economics

Kenneth Judd

Report by Laurens Swinkels*

1 Introduction

For a long time, economists have been able to develop theories by mainly verbal reasoning. Gradually these theories were translated into mathematics and by logical reasoning the assumptions underlying these theories were uncovered. However, these models had to be kept simple in order to yield a closed-form solution. The derivation of such analytical solution is very difficult – or in many cases even impossible – in more realistic, and hence more complex, economic models. The easiest response to the non-existence of such solution is to stop doing research in such field. In recent years, however, a new branch of researchers has focused on finding numerical solutions to these complex problems.¹

The increase in computer power has enabled the development of new optimization schemes which find numerical solutions for these complicated problems within a feasible time span. These newly developed solution methods are nowadays available in standard software packages. It usually takes some time to get familiar with the syntax used by these programs, but this is usually worth the effort for applied researchers. They should not waste their time by trying to write new computer code themselves, since these widely available optimization routines are programmed very efficiently.

The ability to find numerical solutions to difficult optimization problems has proven to be of much help in a wide area of economics, e.g. macro economics, public finance, and game theory. These lines of research typically used to resort to one or two period models in order to be able to find a solution, but nowadays dynamic multiple period models are

*Department of Econometrics, Tilburg University, P.O. Box 90153, 5000 LE Tilburg.

¹In recognition of the importance of this new field in economics, the Society for Computational Economics was founded in 1995 and now has an annual conference to exchange new research ideas. This conference is alternately held in Europe and the United States.

common and applied researchers are required to understand the basics of the computation behind their economic models.

This paper, which is based on the NAKE lecture series of Professor Kenneth Judd at the Free University in Amsterdam, tries to give some intuition and guidelines for researchers who have to solve analytically non-tractable optimization problems. In the next section, we will describe some of the basic numerical methods and their robustness. In Section 3, we focus on approximation methods for unknown functions. In Section 4, projection methods and its application in macro economics are discussed, and Section 5 describes perturbation methods. Finally, Section 6 concludes this report.

2 Numerical and Theoretical Solutions

For many simple problems, it is quite easy to find theoretical solutions. However, to find the numerical answer to the same question can be hard. Take for example an invertible matrix A and a vector b . If we want to solve the set of linear equations

$$Ax = b, \tag{1}$$

we know that this simply boils down to

$$x = A^{-1}b.$$

When we are actually faced with such problem of a sizeable dimension, we usually have to use the computer to solve it to get an numerical answer. Rounding errors in b may have a large impact on the estimated x . To investigate the robustness to rounding errors, we can develop an error model

$$A(x + e) = (b + r),$$

and define the elasticity of error by

$$\xi = \frac{\|e\|}{\|x\|} \div \frac{\|r\|}{\|b\|}, \tag{2}$$

with $\|\cdot\|$ the norm operator. The ξ is the ratio of the output to the input error. We can define the condition number of a matrix in order to approximate the elasticity of error ξ . The condition number is denoted by $cond(A)$ and defined as

$$cond(A) = \|A\| \circ A^{-1} \circ.$$

The condition number is a measure of determining whether matrix A is near-singular. It is more useful than the determinant of matrix A , which is mistakenly used by many researchers for this purpose. For example, a clearly non-singular matrix would be $B = \varepsilon I_n$, where I_n is the unit matrix of dimension n . The determinant of this matrix B would be near zero when ε is small (to be precise, $\det(B) = \varepsilon^n$), but the condition number would be exactly equal to one. If the determinant of a matrix would be truly zero, then the condition number would be infinity.

In order to illustrate the fact that the influence of these round off errors can be large we constructed the following simulation example. First, a random square matrix A of dimension five is drawn, together with a five-dimensional random vector b . Then, we rounded all numbers of b to three decimals, and this vector is called b^* . We calculated both solution x and x^* , belonging to the vectors b and b^* , respectively. We then proceeded by calculating the norm of the differences in both solutions and the condition number of matrix A . This procedure is repeated thousand times. The results are plotted in Figure 1. On the horizontal axis we put the condition number, and on the vertical axis the norm of the rounding error. The results illustrate what was mentioned before, a rounding error may have a large influence on the results, especially when the condition number of a matrix is large.

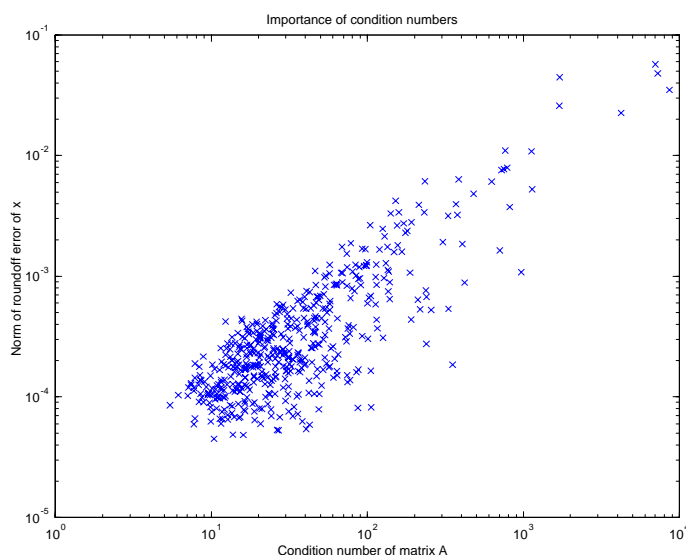


Figure 1: Error of calculation of $x = A^{-1}b$ in rounding off b to three decimals. On the horizontal axis is the condition number of the matrix, on the vertical axis the norm of the error in x after rounding off b .

These approximation errors might not just occur when we manually round numbers to three decimals, but also when machine precision is involved. Especially in sequences of optimization problems, these small errors add up and may become influential for the final result. For example, a sequence of linear problems arises when we use Newton's optimization method. Theoretically, the optimization problem $\min_x f(x)$ has a clear solution, i.e. $x^* = \arg \min_x f(x)$. However, to calculate the x^* that minimizes f can be problematic. A well-known method to solve this problem is Newton's method, which approximates the function f with a quadratic Taylor series expansion. If we denote J to be the gradient and H the Hessian of function f , the iteration scheme to find the optimal solution is

$$x^{k+1} = x^k - H^{-1}(x^k) J'(x^k).$$

The iteration procedure starts by taking any x^0 as an initial guess, but the method converges faster when a sensible point near the optimum is chosen. The iteration procedure stops when a stopping criterion is met. This can either be the distance between two consecutive iterations, or the size of the gradient, which should be close to zero in all directions for the optimal solution. The main disadvantage of this method is that one may end up in local minima or maxima, and thus fails to identify the true global extreme value. If the Hessian matrix has a large condition number, this procedure generally converges slowly to the true value, if it converges at all.

3 Approximation Methods

Many applications require the estimation of a complicated unknown function. In traditional econometrics, we would for example think of a regression line through a large set of data points. The applications we discuss will not involve statistical regression techniques, but the links with regression will be mentioned at various points in this paper. The objective of this section is to construct an easy function $g(x)$ which is a good approximation of a difficult function $f(x)$, from which we have some data available.

Several issues may arise when trying to find this easy function. First, it is not immediately clear which points x are suitable to approximate f . Unlike with regression problems in which the data is given, we can choose which particular x we want to use for the approximation. A second issue is the shape of the function g , which might be depending on our knowledge about f . For example, it seems plausible that for functions with a periodic component other approximating functions are used than for functions without these cycle components. Third, an approximation without knowledge about its quality is

useless, thus we want to know something about the distance between the function f and g . Connected with this is the question how easy an approximation g can be while still being relatively close to the original function f .

The challenge we face is to find algorithms that are reliable as more data becomes available. In this sense, we could speak of consistency of such algorithm, since the approximation should become closer when more information about the original function is available. These algorithms should be well-formulated in the sense that small computational errors should not heavily influence the resulting approximation. These ill-conditioned approximations appear to happen more often than one would like, so one should always be very cautious about this problem.

A straightforward way to approximate functions is to find a certain family of functions and fit some points from the original function f to this new function. The example we are going to elaborate upon here is the Lagrange polynomial interpolation. We have available data (x_i, y_i) , $i = 1, 2, \dots, n$. The objective is to find a polynomial of degree $n - 1$ that agrees with all data points we have available. This polynomial of order $n - 1$ is denoted by $p_n(x)$. When the points x_i from our data set are distinct, there is exactly one interpolating polynomial. In other words, there are no degrees of freedom left anymore. This is different from statistical regression problems, in which case we usually have many degrees of freedom and consequently the data points are close to, but not exactly on the regression line.

The advantage of using interpolation methods is that only a limited amount of data is necessary to calculate the approximating polynomial. The disadvantage of this technique is that the data has to be computed with very small error, because there are no degrees of freedom left after interpolation. One would expect that the approximating polynomial will get very close to the original function if the order of the polynomial is increased. In other words, we would like to have convergence between $p_n(x)$ and $f(x)$. An insightful example of the unreliability of this method can be found when investigating the function

$$f(x) = \frac{1}{1 + x^2}$$

on the interval $[-5, 5]$. In the lecture notes the example is presented with 11-point interpolation; the figures presented here are for 5- and 13-point interpolation. It is clear from Figures 2 and 3 that the increase in the order of the polynomials does not increase the precision of the approximation.

These figures suggest that simple interpolation techniques are not very reliable and it seems in general not a very good idea to use these polynomials to approximate an unknown function. One may wonder what the reason is that this approximation is poor.

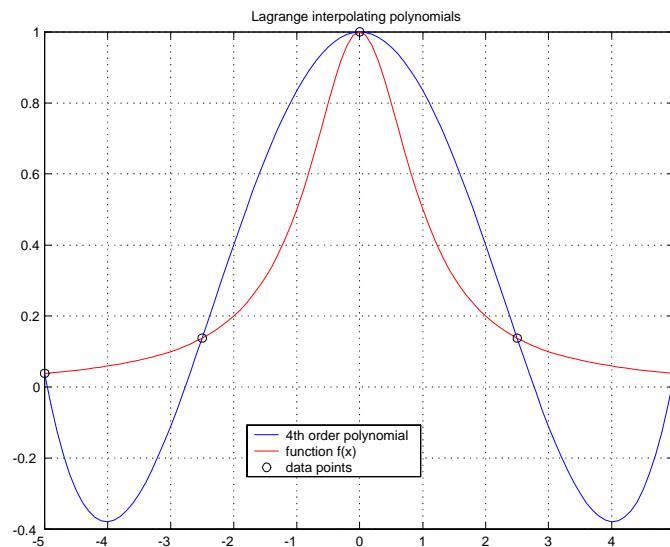


Figure 2: Approximating the function $f(x) = \frac{1}{1+x^2}$ with Lagrange interpolating polynomials. The set of interpolation points (denoted by little circles in the figure) equals one plus the degree of the polynomial and are equally spaced over the domain $[-5, 5]$.

Intuitively, it can be argued that the choice of the polynomials $1, x, x^2, \dots, x^{n-1}$ is not particularly well-chosen. These vectors are highly correlated and therefore are not suitable to serve as a basis for the vector space of all polynomials. More appropriate bases would contain vectors that are orthogonal to each other. We observe the same phenomenon with classical regression. When the regressors are highly correlated, we observe that the point estimates are highly unreliable and we refer to this as the near-multicollinearity problem. The use of orthogonal regressors is to be preferred in classical regression to circumvent this problem, but there we can often not choose our observations ourselves. Since this is possible for the approximation methods, we prefer the use of orthogonal polynomials, since they generate reliable interpolation formulas. There are several types of polynomials that can be used, each one having its own merits. We decide to show the difference with the Lagrange interpolation of figures 2 and 3 and the Chebyshev interpolation, which are depicted in Figures 4 and 3. The approximation precision of a low order (Figures 2 and 4) are comparable. As we observed before, the use of higher-order polynomials with the Lagrange interpolation does not assure that the approximation gets closer to the true function. In the case of Chebyshev interpolation, we see that the approximation gets fairly accurate when the order of the polynomial is increased. The approximating function is

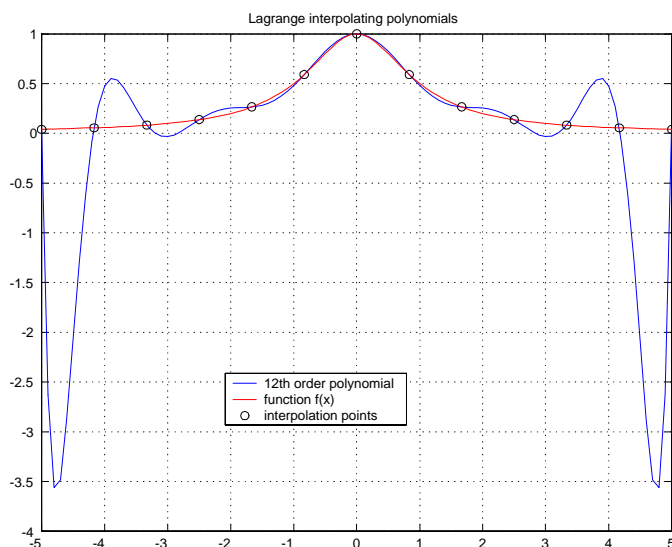


Figure 3: Approximating the function $f(x) = \frac{1}{1+x^2}$ with Lagrange interpolating polynomials. The set of interpolation points (denoted by little circles in the figure) equals one plus the degree of the polynomial and are equally spaced over the domain $[-5, 5]$.

rather wiggly around the true function, which indicates that the first derivative of both functions may be very far from each other. The dot near the point -4 has a positive slope for the true function, while the Chebyshev approximation has a negative slope. The quality of the slope estimate can be increased by doing a Chebyshev regression, which basically means that some degrees of freedom are needed by using some extra points in order to estimate the unknown coefficients.

Next to the Chebyshev polynomials, a variety of other functions can be used. Which one to use depends on the type of application one is working on. Other types are trigonometric, Legendre, or Hermite polynomials. Periodic functions may benefit from the use of trigonometric polynomials, while Hermite polynomials are related to problems which involve near normal distribution functions.

Other ways of approximation are for example rational functions, which are polynomials divided by other polynomials, or neural networks. One could also use splines, which basically means that the domain is split up in several parts for which a polynomial is fitted separately. At the borders of the domains it is often required that both function values and first derivatives are equal, in order to keep a smooth function.

All the functions thus far suffer from the possibility that the shape of the data is not

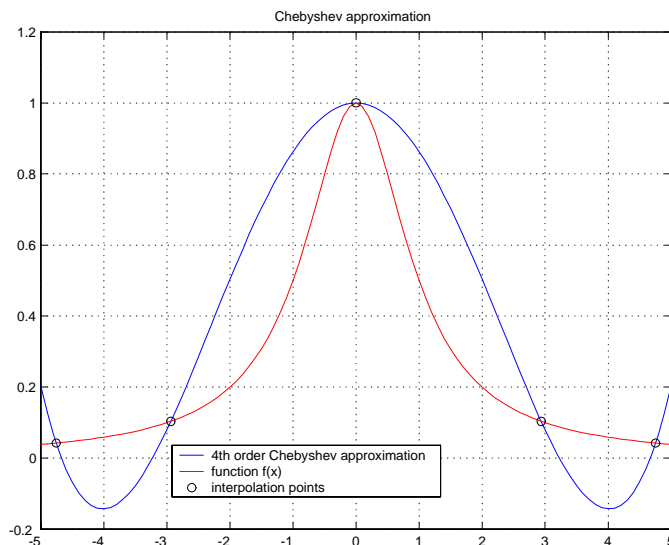


Figure 4: Approximating the function $f(x) = \frac{1}{1+x^2}$ with Chebyshev interpolating polynomials. The set of interpolation points (denoted by little circles in the figure) equals one plus the degree of the polynomial. The following recursion is used, $T_0(x) = 1$, $T_1(x) = x$, $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$.

preserved by the function estimate. When for example the points $f(x_i)$ are increasing for the x_i -s drawn, we might expect that the approximated function is also increasing. In general, this does not have to be the case, and one must be aware of this fact, especially for dynamic optimization problems. Some procedures can be followed in order to preserve shape, e.g. the Schumaker procedure for one-dimensional problems. For problems in a higher dimension things become more difficult, and even current state-of-the-art research on this topic has not yet revealed answers to all questions.

4 Projection Methods for Functional Problems

In the previous section, we analyzed how to approximate an unknown function. In the presented example, we actually knew the function which we wanted to analyze and observed how close the approximation was to the theoretical function. While such comparison is a very insightful exercise, for many realistic models the function we are after is really unknown and comparison with reality is impossible. Examples of the functions that economists want to estimate are consumption functions or pricing functions. The projection

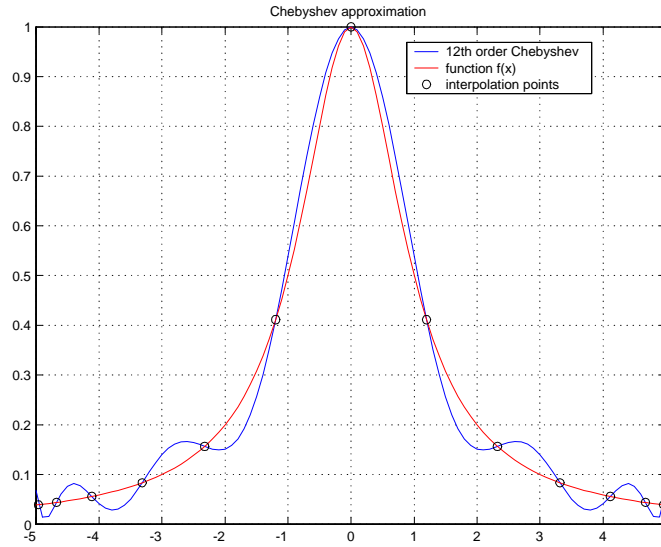


Figure 5: Approximating the function $f(x) = \frac{1}{1+x^2}$ with Chebyshev interpolating polynomials. The set of interpolation points (denoted by little circles in the figure) equals one plus the degree of the polynomial. The following recursion is used, $T_0(x) = 1$, $T_1(x) = x$, $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$.

method is a robust method for solving these type of dynamic optimization problems.

Consider the following example in which an agent wants to maximize her utility function by choosing her stream of consumption, and hence determines also the level of capital in the economy. The optimization problem is

$$\max_{c_t} \sum_{t=1}^{\infty} \rho^t U(c_t) \quad (3a)$$

$$k_{t+1} = f(k_t) - c_t, \quad (3b)$$

with U the utility function, c_t and k_t the level of consumption and capital in period t , f the production function, and ρ the time preference rate. If we derive the first-order conditions of this problem, it turns out that for optimality it must hold that

$$U'(c_t) = \rho U'(c_{t+1}) f'(k_{t+1}), \quad (4)$$

which implies that the ratio of marginal utilities in two consecutive periods equals the time preference factor times the marginal production rate. The problem is that we have an infinitely large number of unknown elements c_t , which form the solution. In order to find the optimal consumption path one can proceed by the following recipe. First, express

the solution in terms of an unknown function. The consumption function can be written as $c_t \equiv C(k_t) = f(k_t) - k_{t+1}$ by observing the problem described in (3). The function $C(k)$ should satisfy equation (4)

$$0 = U'(C(k)) - \rho U'(C(f(k) - C(k))) f'(f(k) - C(k)),$$

and now define the non-trivial function on the right-hand side as $[N(C)](k)$, which is a function of k . In equilibrium, $N(C)$ should be equal to zero. The next step is to come up with an approximation for the function C , which we call \mathcal{C} . This approximation might be a polynomial in k ,

$$\mathcal{C} = \sum_{i=1}^{\infty} a_i k^i, \quad (5)$$

which is according to some criterion close to the solution, i.e. $N(\mathcal{C}) \approx 0$. We have now reduced an infinite dimensional problem to an n -dimensional problem without discretizing the state space form of the problem. Next, calculate the error that was made by the initial guess in equation (5), which we denote by R , and is a function of both the capital k as well as the coefficients of the approximation a ,

$$R(k; a) = U'(\mathcal{C}(k)) - \rho U'(\mathcal{C}(f(k) - \mathcal{C}(k))) f'(f(k) - \mathcal{C}(k)).$$

The final step of solving for $C(k)$ is to minimize the Euler equation errors R relative to a certain measure. The optimal solution can be obtained for example by minimization with respect to a of the squared residuals over all possible values of k . Another example is to use Galerkin, which uses n weighting functions to determine a . The method called collocation requires that the Euler errors are exactly zero for some prespecified values of k .

The measurement of the errors over the values of k requires an integration procedure. In general, exact integration is not possible and one has to resort to numerical integration. It is generally wise to use quadrature formulas (e.g. Chebyshev) to evaluate the integral numerically at sensible points. An alternative that is frequently used in practice is Monte Carlo simulation, but for high-dimensional problems this takes a tremendous amount of time.² Several large dimensional problems require the use of number theoretic methods to be solved. In order to solve for the unknown parameter vector a one can use Newton's method as long as the Jacobian is well-conditioned. If this is not the case one can use

²The disadvantages of using Monte Carlo techniques to solve these type of problems were emphasized repeatedly by Professor Judd, suggesting that many applied economists waste much time by refusing to use more advanced optimizations techniques discussed in this lecture series.

functional or time iteration methods, which are more complicated. The use of homotopy methods is advised only when nothing else seems to work, since its advantage of being almost surely globally convergent is offset by its complicated calculation.

In equation (5), the basis for the solution is taken to be the set $\{1, k, k^2, \dots\}$ which is in general not a very sensible choice for the basis of the polynomial space. The choice for Chebyshev polynomials might be more attractive in many applications that do not involve periodicities, for which Fourier functions are more suitable. In equation (5) the specification is chosen to be linear, which suffices for many applications. The use of neural networks, wavelets, and rational functions leads to incorporation of nonlinearities. Very little is known about the performance of these methods. Future research on this topic is necessary to provide sound recipes for problem solving.

5 Perturbation Methods

In this section, we will shortly describe the main idea of regular perturbation methods. The method boils down to approximating an unknown function by a Taylor series expansion around a trivial point of which the answer is known. By evaluation of the approximation over the area close to this known point the solution for the interesting cases are obtained.

Suppose we have a real-valued function h , which can be approximated by a Taylor series expansion at x_0

$$h(x) = h(x_0) + h'(x_0)(x - x_0) + \frac{1}{2}h''(x_0)(x - x_0)^2 + \dots$$

which can be computed by repeated differentiation of the function h . This approximation can be extended to vector-valued functions, but for expositional purposes this is not done here. The zeroth, second, and fourth order approximations of the function $h(x) = \frac{1}{1+x^2}$ at zero are

$$\begin{aligned} h^0(x) &= 1, \\ h^2(x) &= 1 - x^2, \\ h^4(x) &= 1 - x^2 + x^4, \end{aligned}$$

which are graphically shown in Figure 6. Note that the odd order approximations are equal to the even order approximations. This picture shows that the approximation close to the point in which the Taylor series is evaluated gets better when the order gets higher. This does not mean that the approximation is closer uniformly over the entire interval.

It can be seen in Figure 7 that close to zero the fourth order approximation is closest, but more than one unit from zero the zeroth order approximation is actually closest. We should be aware of the fact that approximations can only lead to sensible results if we stay close to the point around which the expansion took place.

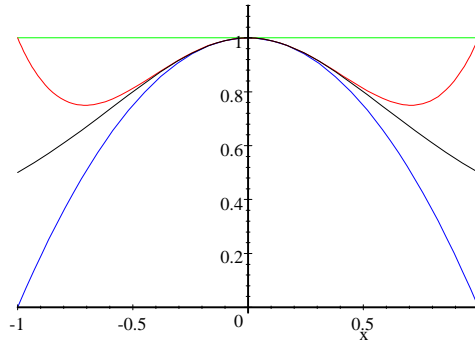


Figure 6: Taylor series approximations of the function $f(x) = \frac{1}{1+x^2}$. The function itself is shown in black, the zeroth, second, and fourth order approximations are colored green, blue, and red, respectively.

The Taylor series are a numerical approximation technique, which is useful when h and its derivatives can be computed with ease at a point x_0 . This is essential for the success of this technique. Note that for technical reasons the function h should be analytic in order to let this method work, but luckily this is frequently the case.

Suppose now that the function h is defined implicitly, which is the case in many applications, so that e.g.

$$H(x, h(x)) = 0. \quad (6)$$

If we know a special point (x_0, y_0) for which this implicit solution is true, i.e. $H(x_0, y_0) = 0$, the function h is known in a special point, i.e. $y_0 = h(x_0)$. If we derive implicitly equation (6) we obtain

$$H_1(x, h(x)) + H_2(x, h(x)) h'(x) = 0$$

which implies

$$h'(x_0) = -H_2(x_0, y_0)^{-1} H_1(x_0, y_0)$$

for invertible $H_2(x, h(x))$.

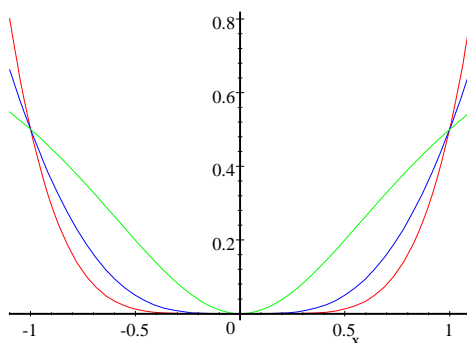


Figure 7: Errors of the Taylor series approximations of the function $f(x) = \frac{1}{1+x^2}$. The differences with the zeroth, second, and fourth order approximations are colored green, blue, and red, respectively.

Now the function h can be linearly approximated by a Taylor series expansion around x_0 , which yields

$$h^1(x) = h(x_0) + h'(x_0)(x - x_0).$$

In order to evaluate how close the linear approximation is to the solution we can calculate $H(x^*, h^1(x^*))$ and check its distance from zero, which it should be in equilibrium. We can impose a rule which states that the approximation is sufficient when this difference is smaller than a certain threshold value ε . A better approximation might be obtained when instead of the linear approximation the quadratic approximation is used.

The big advantage of using these methods is that researchers do not need to get bored by endless calculations with the possibility of human errors, but instead they can get reliable results very quickly by using widely available software packages.

6 Conclusions

During this workshop, we have become aware of the difficulties of actually calculating the numerical solution to a theoretical problem. In this report, we compare these numerical solutions to their theoretical counterparts to see how well they do. However, when the problems become more interesting, and hence more complicated the theoretical solutions cannot be computed anymore and we have to rely on the quality of the numerical approximation.

Numerical errors are important for calculation of the solutions, and sadly they cannot be avoided. All computers have some truncation point after which numbers are rounded, and these occur even in the simplest operations. When an algorithm to find a solution involves many computations, these individually small errors accumulate and may turn out to be a huge error in the final result. We should be aware of these problems and use methods that minimize the buildup of these errors as much as possible. For example, we could use a stopping criterion that allow iterative schemes to proceed until the round off error is relatively small. As illustrated, a matrix with a high condition number may lead to high errors in the solutions. Therefore, these ill-conditioned matrices should be avoided when solving linear problems. This is also observed when approximating a function with a polynomial. If we use a bad basis for the polynomial space, we are in fact using an ill-conditioned matrix and taking its inverse may result in numerical difficulties.

The methods used to solve problems that arise in economics are generally highly mathematical and economic intuition does not play a role to find the answers. We argue that this is good rather than bad. Most algorithms that arise by economic reasoning have very poor convergence properties, if they converge at all. We stress that in the problem formulation phase economic reasoning should be top priority, and motivations of the possibility to calculate the solution should stay at the background. However, when the formulation phase is finished, we should not try to find a solution that is motivated by economics, but resort to methods of numerical analysis who are especially designed to find a good approximation as fast as possible.

Nowadays, numerous commercial and non-commercial software packages are available to fully take advantage of the increased computer power to determine solutions to optimization problems. Since these sophisticated algorithms are programmed in fast languages, there is no need for the economist to program optimization routines. For them, it would be wiser to spend some time getting used to existing software (like e.g. CONOPT, NPSOL, GAUSS, or NAG) instead.