

## Condition and Scaling

Floating-point numbers have a fraction and an exponent. By scaling we find a more suitable coordinate system to represent and compute solutions of polynomial systems. We follow mainly [4], motivated by chemical equilibria. For more on condition numbers we refer to [1].

### 1 Error Analysis and Condition Numbers

Let  $\mathbf{z} \in \mathbb{C}^n$  be a solution of the system  $f(\mathbf{x}) = \mathbf{0}$ . Suppose we computed and found  $\bar{\mathbf{z}}$  with  $f(\bar{\mathbf{z}}) \approx \mathbf{0}$ . Consider  $\Delta f: \bar{f} = f + \Delta f$ , with  $\bar{f}(\bar{\mathbf{z}}) = \mathbf{0}$ . This implies  $f(\bar{\mathbf{z}}) + \Delta f(\bar{\mathbf{z}}) = \mathbf{0}$ , or  $\|f(\bar{\mathbf{z}})\| = \|\Delta f(\bar{\mathbf{z}})\|$ . The residual  $\|f(\bar{\mathbf{z}})\|$  measures the change in the system  $f$  so that the computed solution  $\bar{\mathbf{z}}$  is an exact root. Therefore, we say that the residual measures the backward error of the solution.

When we start out in the solution space and look back to the problem space to justify the error we obtain the backward error. The forward error  $\Delta \mathbf{z}$  is what we have to add to the computed solution  $\bar{\mathbf{z}}$  to obtain the exact solution  $\mathbf{z}$ . With multivariate Taylor expansions of  $f$  at  $\mathbf{z}$  we obtain

$$f(\mathbf{z}) = f(\bar{\mathbf{z}} + \Delta \mathbf{z}) = f(\bar{\mathbf{z}}) + \Delta \mathbf{z} J_f(\bar{\mathbf{z}}) + O((\Delta \mathbf{z})^2), \quad (1)$$

where  $J_f$  is the Jacobian matrix of  $f$ . Using  $f(\mathbf{z}) = \mathbf{0}$  and ignoring the higher order terms  $O((\Delta \mathbf{z})^2)$  leads us to find  $J_f(\bar{\mathbf{z}})\Delta \mathbf{z} = -f(\bar{\mathbf{z}})$ , the formula we use to compute the update vector  $\Delta \mathbf{z}$  in Newton's method. The accuracy of the forward error depends on the solution of a linear system.

Given a linear system  $A\mathbf{x} = \mathbf{b}$ , we solve  $\bar{A}\bar{\mathbf{x}} = \bar{\mathbf{b}}$  with

$$\bar{A} = A + \Delta A, \quad \bar{\mathbf{x}} = \mathbf{x} + \Delta \mathbf{x}, \quad \bar{\mathbf{b}} = \mathbf{b} + \Delta \mathbf{b}. \quad (2)$$

To find a bound for the error  $\Delta \mathbf{x}$  on  $\mathbf{x}$ , we do

$$\begin{array}{r} (A + \Delta A)(\mathbf{x} + \Delta \mathbf{x}) = \mathbf{b} + \Delta \mathbf{b} \\ - [ \quad \quad \quad A\mathbf{x} = \mathbf{b} \quad \quad \quad ] \\ \hline A\Delta \mathbf{x} + \Delta A(\mathbf{x} + \Delta \mathbf{x}) = \Delta \mathbf{b} \end{array} \quad (3)$$

Assuming  $\det(A) \neq 0$ :  $\Delta \mathbf{x} = A^{-1}(-\Delta A\bar{\mathbf{x}} + \Delta \mathbf{b})$ . For some  $\|\cdot\|$ , we have  $\|\Delta \mathbf{x}\| \leq \|A^{-1}\| (\|\Delta A\| \cdot \|\bar{\mathbf{x}}\| + \|\Delta \mathbf{b}\|)$  and with reshuffling we find

$$\frac{\|\Delta \mathbf{x}\|}{\|\bar{\mathbf{x}}\|} \leq \|A^{-1}\| \cdot \|A\| \left( \frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta \mathbf{b}\|}{\|A\| \cdot \|\bar{\mathbf{x}}\|} \right) \quad (4)$$

Ignoring  $\Delta \mathbf{b}$ , the magnitude of the relative error  $\|\Delta \mathbf{x}\|/\|\mathbf{x}\|$  of the solution of a linear system  $A\mathbf{x} = \mathbf{b}$  is bounded by

$$\frac{\|\Delta \mathbf{x}\|}{\|\mathbf{x}\|} \leq \|A\| \cdot \|A^{-1}\| \frac{\|\Delta A\|}{\|A\|}. \quad (5)$$

This inequality bounds the relative error on the solution by the relative error on the coefficient matrix  $A$  and the number  $\|A\| \cdot \|A^{-1}\| = \text{cond}(A)$ , the condition number of  $A$ . If  $\text{cond}(A)$  is large, then the linear system is ill conditioned and no matter what algorithm we use to solve it, small errors in the calculations will amplify and cause large errors in the answer. An indicate for large  $\text{cond}(A)$  is often that  $A$  contains numbers of vastly different magnitudes, but matrices with nice numbers can be ill conditioned as well. A standard example is the upper triangular matrix with ones on its diagonal and  $-1$  above the diagonal. This condition number for this matrix grows exponentially in the dimension.

If for a computed vector  $\bar{\mathbf{z}}$  we find that the residual  $\|f(\bar{\mathbf{z}})\| \approx \epsilon$ , that moreover, the forward error  $\|\Delta \mathbf{z}\| \approx \epsilon$ , and  $\text{cond}(J_f(\bar{\mathbf{z}})) \approx 1$ , then  $\bar{\mathbf{z}}$  is a valid approximate root for the system  $f(\mathbf{x}) = \mathbf{0}$  since one step with Newton's method will reduce  $\|f(\bar{\mathbf{z}})\|$  and  $\|\Delta \mathbf{z}\|$  to be of size  $\epsilon^2$ .

However, the computation of the residual is an inherent numerically instable operation as its result is so close to zero, with a great potential for loss of accuracy when subtractions of numbers of equal magnitude occur. To evaluate the residual, multiprecision arithmetic is recommended. For the computation of the condition number of the Jacobian matrix, the singular value decomposition is a standard tool, although estimates computed as a byproduct of LU decomposition also mostly give already reliable indicators.

Let  $f(\mathbf{x}) = \mathbf{0}$  be a system of  $n$  equations in  $n$  unknowns. Denote the Jacobian matrix of  $f$  by  $J_f$  and let  $\mathbf{z} \in \mathbb{C}^n$  be an isolated solution of  $f(\mathbf{x}) = \mathbf{0}$ . Then, *the relative condition number of the zero  $\mathbf{z}$  as a solution of  $f(\mathbf{x}) = \mathbf{0}$*  is

$$\kappa(f, \mathbf{z}) = \|J_f(\mathbf{z})\|_2 \|J_f^{-1}(\mathbf{z})\|_2, \quad (6)$$

i.e.:  $\kappa(f, \mathbf{z})$  is the condition number of the Jacobian matrix of the polynomials in the system evaluated at  $\mathbf{z}$ . Three numbers measure the quality of an approximate solution  $\mathbf{z}$ : the residual  $\|f(\mathbf{z})\|$ , the correction term  $\|\Delta\mathbf{z}\|$  (computed via Newton's method) and an estimate for  $\kappa(f, \mathbf{z})$ .

The singular value decomposition (SVD) of  $A \in \mathbb{C}^{n \times n}$  is

$$A = U\Sigma V^H, \quad U^H U = I, V^H V = I, \quad (7)$$

and with singular values  $\sigma_i$ ,  $i = 1, 2, \dots, n$ :

$$\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n), \quad \sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n. \quad (8)$$

Two important applications of the SVD are the numerical rank of a matrix and the determination of the nearest singular problem:

1.  $\|A\|_2 = \sigma_1$ , for  $\det(A) \neq 0$ :  $\|A^{-1}\|_2 = \sigma_n^{-1}$

$$\Rightarrow \text{cond}(A) = \|A\|_2 \cdot \|A^{-1}\|_2 = \frac{\sigma_1}{\sigma_n}. \quad (9)$$

2. if  $\sigma_{R+1} < \epsilon$ , then  $R = \text{Rank}(A, \epsilon)$  and

$$\widehat{\Sigma} = \text{diag}(\sigma_1, \dots, \sigma_R, 0, \dots, 0), \quad \widehat{A} = U\widehat{\Sigma}V^H \quad (10)$$

We have that the matrix  $\widehat{A}$  is the projection of  $A$  onto the space of rank  $R$  matrices.

Note that  $\widehat{A}$  offers an approach to solve ill-posed problems. For a rank deficient matrix  $A$ , solving  $A\mathbf{x} = \mathbf{b}$  could mean the computation of a basis for the null space of  $\widehat{A}$ .

We close this section with two cautionary examples. The Chebyshev polynomial  $T_{d+1}$  degree  $d+1$  has all its roots in  $[-1, +1]$  with leading coefficient  $2^d$ . Since  $T_{d+1}$  is defined as a cosine,  $|T_{d+1}(x)| \leq 1$  over  $[-1, +1]$ . For  $p(x) = T_{d+1}(x)/2^d$  we then have  $|p(x)| \leq 1/2^d$  for all  $x \in [-1, +1]$ , or  $\epsilon = 1/2^d$  for large  $d$ .

Even a simple system of quadratic equations can cause difficulties. The telescope example is

$$f(\mathbf{x}) = \begin{cases} x_1 - c = 0 \\ x_2 - x_1^2 = 0 \\ x_3 - x_2^2 = 0 \\ x_4 - x_3^2 = 0 \end{cases} \quad c \in \mathbb{C}. \quad (11)$$

As the equations lead to repeated squaring, the solution is obviously  $\mathbf{z} = (c, c^2, c^4, c^8)$ . For  $c \approx 100$ ,  $z_4 \approx 10^{16}$ . As the granularity of the floating-point number system is finer closer to zero, changing coordinates  $x_i = 1/y_i$  may be most appropriate to represent the solution of the system for  $c > 1$ .

## 2 Chemical Equilibria

Polynomial systems whose coefficients take extreme values arise in the computation of equilibria of chemical reactions [2, 3]. As in [5], we also follow the treatment in [4].

While the equations which govern the behavior of chemical reactions are differential equations, the equilibria are solutions of polynomial systems. The variables in the system are the molar concentrations of the species participating to the chemical reaction. There are two types of equations: the chemical reaction equations and the conservation equations which express that the species do not leave the closed space in which the reaction occurs.

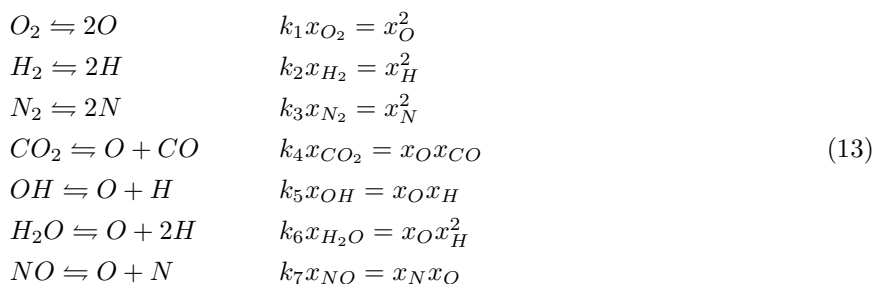
For the equilibrium balance between water  $H_2O$ , hydrogen  $H$  and oxygen  $O$ , we derive:

$$\begin{array}{l} H_2O \rightleftharpoons 2H + O \\ \text{total amount } T_H \text{ of } H \text{ is conserved} \\ \text{total amount } T_O \text{ of } O \text{ is conserved} \end{array} \quad f(x_H, x_O, x_{H_2O}) = \begin{cases} kx_{H_2O} = x_H^2x_O \\ 2x_{H_2O} + x_H = T_H \\ x_{H_2O} + x_O = T_O \end{cases} \quad (12)$$

where the constant  $k$  is a dimensionless stoichiometric coefficient. Translating the reaction equations into the algebraic relations is identical to applying an exponential function to the reaction equation.

The chemical reaction considered involves 11 species, 4 components:  $O$  atomic oxygen,  $H$  atomic hydrogen,  $CO$  carbon monoxide,  $N$  atomic nitrogen; and 7 compounds:  $O_2$  molecular oxygen,  $H_2$  molecular hydrogen,  $N_2$  molecular nitrogen,  $CO_2$  carbon dioxide,  $OH$  hydroxyl radical,  $H_2O$  water vapor,  $NO$  nitric oxide.

Using the notation of [5], we have the seven reaction equations



and the four equations conserving the total amounts of the four components:

$$\begin{array}{l} T_H = x_H + 2x_{H_2} + x_{OH} + 2x_{H_2O} \\ T_C = x_{CO} + x_{CO_2} \\ T_O = x_O + x_{CO} + 2x_{O_2} + 2x_{CO_2} + x_{OH} + x_{H_2O} + x_{NO} \\ T_N = x_N + 2x_{N_2} + x_{NO}. \end{array} \quad (14)$$

So we end up with a total of 11 equations in 11 unknowns. The constants are given in the table, for various temperatures:

constants	$T = 1000^\circ$	$T = 2000^\circ$	$T = 3000^\circ$	totals
$\log_{10}(1/k_1)$	24.528	7.289	3.108	$T_O = 5.0E-5$
$\log_{10}(1/k_2)$	22.206	6.997	3.270	$T_H = 3.0E-5$
$\log_{10}(1/k_3)$	47.970	15.107	6.942	$T_C = 1.0E-5$
$\log_{10}(1/k_4)$	24.942	6.825	2.559	$T_N = 1.0E-5$
$\log_{10}(1/k_5)$	22.120	7.208	3.541	
$\log_{10}(1/k_6)$	46.989	14.680	6.791	
$\log_{10}(1/k_7)$	32.187	10.285	4.878	

### 3 Equation and Variable Scaling

Equation and variable scaling are presented in [4, Chapter 5]. In equation scaling we divide every coefficient in the same equation by the same constant. This scaling constant is obvious in an equation like

$$f(x) = 10^{20}x^2 + 4 \cdot 10^{20}x + 2 \cdot 10^{20} = 0. \quad (15)$$

Variable scaling is needed to deal with

$$f(x) = 10^{-20}x^2 + 4 \cdot x + 2 \cdot 10^{20} = 0. \quad (16)$$

The change of variables  $x = 10^{20}z$  turns (16) into (15).

A combined equation and variable scaling method aims to center the coefficients around unity and at the same time reduce the variability of the coefficients. For the equation given in (16), we need to determine two constants  $c_1$  and  $c_2$  used as

$$10^{c_2}f(z = 10^{c_1}x) = 10^{e_1}z^2 + 10^{e_2}z + 10^{e_3}. \quad (17)$$

The two objectives are met by minimizing  $r(c_1, c_2) = r_1(c_1, c_2) + r_2(c_1, c_2)$  where

$$r_1(c_1, c_2) = e_1^2 + e_2^2 + e_3^2 \quad \text{and} \quad r_2(c_1, c_2) = (e_1 - e_2)^2 + (e_1 - e_3)^2 + (e_2 - e_3)^2. \quad (18)$$

Since  $r(c_1, c_2)$  is a quadratic and has no maximum, the minimum must occur at the solution of the linear system defined by  $\frac{\partial r}{\partial c_1} = 0$  and  $\frac{\partial r}{\partial c_2} = 0$ . We derive a general formulation for the optimization problem below.

Consider a polynomial  $f$  in  $n$  variables  $\mathbf{x} = (x_1, x_2, \dots, x_n)$  with complex coefficients supported on  $A$

$$f(\mathbf{x}) = \sum_{\mathbf{a} \in A} c_{\mathbf{a}} \mathbf{x}^{\mathbf{a}}, \quad c_{\mathbf{a}} \in \mathbb{C}, \quad \mathbf{a} = (a_1, a_2, \dots, a_n), \quad \mathbf{x}^{\mathbf{a}} = x_1^{a_1} x_2^{a_2} \dots x_n^{a_n}. \quad (19)$$

For some base  $b$  of the number system ( $b$  is 10 or some power of 2), we scale the equation  $f(\mathbf{x}) = 0$  by multiplication with  $b^{\gamma_0}$  and substitute the variables  $x_i$  by  $b^{\gamma_i} y_i$ , for  $i = 1, 2, \dots, n$ . We denote  $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \dots, \gamma_n)$  and abbreviate  $\gamma_1 a_1 + \gamma_2 a_2 + \dots + \gamma_n a_n$  by  $\langle \boldsymbol{\gamma}, \mathbf{a} \rangle$ . The scaled polynomial is

$$b^{\gamma_0} f(x_1 = b^{\gamma_1} y_1, x_2 = b^{\gamma_2} y_2, \dots, x_n = b^{\gamma_n} y_n) = \sum_{\mathbf{a} \in A} c_{\mathbf{a}} b^{\gamma_0 + \gamma_1 a_1 + \gamma_2 a_2 + \dots + \gamma_n a_n} y^{\mathbf{a}} \quad (20)$$

$$= \sum_{\mathbf{a} \in A} b^{\log_b(c_{\mathbf{a}}) + \gamma_0 + \langle \boldsymbol{\gamma}, \mathbf{a} \rangle} y^{\mathbf{a}}. \quad (21)$$

Although formally we write  $b^{\log_b(c_{\mathbf{a}})} = c_{\mathbf{a}}$ , for complex coefficients, the logarithm is multivalued. In practice, since we are interested mainly in the magnitude of the coefficients we compute  $\log_b |c_{\mathbf{a}}|$ .

We determine conditions on the scaling constants  $\gamma_0$  and  $\boldsymbol{\gamma}$  to center the coefficients around one and to reduce the variability of the coefficients. The distance of the coefficients from one is expressed by  $r_1$ :

$$r_1(\gamma_0, \boldsymbol{\gamma}) = \sum_{\mathbf{a} \in A} (\log_b |c_{\mathbf{a}}| + \gamma_0 + \langle \boldsymbol{\gamma}, \mathbf{a} \rangle)^2. \quad (22)$$

The variability between the coefficients is expressed by  $r_2$ :

$$r_2(\gamma_0, \boldsymbol{\gamma}) = \sum_{\mathbf{a}_1 \in A} \sum_{\mathbf{a}_2 \in A \setminus \{\mathbf{a}_1\}} (\log_b |c_{\mathbf{a}_1}| + \langle \boldsymbol{\gamma}, \mathbf{a}_1 \rangle - \log_b |c_{\mathbf{a}_2}| - \langle \boldsymbol{\gamma}, \mathbf{a}_2 \rangle)^2. \quad (23)$$

Minimizing  $r_1(\gamma_0, \boldsymbol{\gamma}) + r_2(\gamma_0, \boldsymbol{\gamma})$  is a least squares problem. Instead of taking derivatives, we can take a direct look at the overconstrained linear system and apply QR decomposition on its matrix representation.

We formulate the problem for a system  $f(\mathbf{x}) = \mathbf{0}$  of  $n$  equations  $f = (f_1, f_2, \dots, f_n)$  supported on  $(A_1, A_2, \dots, A_n)$ . Instead of one  $\gamma_0$ , we have  $n$  constants  $\gamma_{i0}$ ,  $i = 1, 2, \dots, n$ .

$$\begin{cases} \langle \boldsymbol{\gamma}, \mathbf{a} \rangle + \gamma_{i0} &= -\log_b |c_{\mathbf{a}}|, & \mathbf{a} \in A_i, i = 1, 2, \dots, n \\ \langle \boldsymbol{\gamma}, \mathbf{a}_1 \rangle - \langle \boldsymbol{\gamma}, \mathbf{a}_2 \rangle &= -\log_b |c_{\mathbf{a}_1}| + \log_b |c_{\mathbf{a}_2}|, & \mathbf{a}_1 \in A_i, \mathbf{a}_2 \in A_i \setminus \{\mathbf{a}_1\}, i = 1, 2, \dots, n \end{cases} \quad (24)$$

Observe we have  $2n$  unknowns, while the number of equations is determined by the number of monomials. The fewer monomials a system has, the better scaling will work.

## 4 Preconditioning and Postconditioning

The scaling methods of [4] are numerical preconditioning methods. By preconditioning we mean the reformulation of a problem which leads to a better numerical conditioning, where the answers are less sensitive to small changes in the input coefficients.

If there is a good place for multiprecision arithmetic, then the scaling operations might be executed using arithmetic beyond the standard double precision in order to avoid roundoff. By postconditioning we mean the interpretation of the solutions to the scaled problem in the original coordinate system. If the original coefficients of the problem were of extreme magnitudes, then we may expect to find also solutions of large magnitudes and extra precision is likely to be needed to transform the solutions into the original coordinate system and to evaluate them.

Multiprecision arithmetic to return to the original coordinates may not be enough, due to the limited accuracy of the zeroes computed in the scaled coordinates. The coordinate transformation

$$x_i = y_i b^{\gamma_i}, \quad i = 1, 2, \dots, n \quad (25)$$

could be interpreted as a scaled floating-point representation for the zeroes of  $f(\mathbf{x}) = \mathbf{0}$ , where  $y_i$  is the fraction and  $\gamma_i$  the exponent. Denote by  $\tilde{y}_i$  the computed  $i$ th component of a solution of the scaled system  $f(\mathbf{y}) = \mathbf{0}$ . If the solution is regular and the scaling worked well, then we may assume that the error on  $\tilde{y}_i$  is of the same magnitude as the machine precision  $\epsilon$ . The approximation  $\tilde{z}_i$  for the  $i$ th component  $z_i$  of the corresponding solution in the original coordinate  $x_i$  is then

$$\tilde{z}_i = (\tilde{y}_i + \delta_i) b^{\gamma_i}, \quad \delta_i \text{ is } O(\epsilon) \quad \Rightarrow \quad z_i - \tilde{z}_i = \delta_i b^{\gamma_i}. \quad (26)$$

We see that the error in the original coordinates  $\delta_i b^{\gamma_i}$  erases all meaningful information from the fraction as soon as  $b^{\gamma_i} > \delta_i^{-1}$ . There are two different iterative approaches to postcondition a scaled solution:

1. Execute Newton's method in multiprecision arithmetic to refine the solution in the scaled  $y$ -coordinates. The working precision should equal to  $b^{-m + \log(\epsilon)}$ , where  $m$  is the maximum over all  $\gamma_i$ 's.
2. Turn to the original coordinates gradually increasing the exponent of  $b$  till the values of the  $\gamma_i$ 's are met. After each increase of the exponent of  $b$ , a couple of Newton steps are needed to refine the solution in the partially rescaled coordinate system.

The two iterative approaches can be combined. If the original system is given with coefficients of limited precision, then the results of the multiprecision calculations should be interpreted with caution.

While badly scaled problems are problematic for algorithms working in limited precision and are thus by definition ill conditioned, it is important to realize that this type of bad conditioning is different from singularities or solutions at infinity. To deal with solutions at infinity, we use projective coordinates and we compute singularities via derivatives.

## 5 Exercises

1. Consider the matrix

$$A = \begin{bmatrix} +1 & -1 & -1 & -1 \\ 0 & +1 & -1 & -1 \\ 0 & 0 & +1 & -1 \\ 0 & 0 & 0 & +1 \end{bmatrix}. \quad (27)$$

Compute the condition number with MATLAB or Octave and make a table for larger versions of  $A$ , up to  $n = 10$  at least. Choose a random right hand side vector  $\mathbf{b}$  and solve  $A\mathbf{x} = \mathbf{b}$ . Do the solutions  $\mathbf{x}$  grow in size as the dimension  $n$  grows? Choose a random solution vector  $\mathbf{x}$  and compute  $\mathbf{b} = A\mathbf{x}$ . Solve  $A\mathbf{x} = \mathbf{b}$ . Compare your finding with the previous experiments.

2. Consider  $f(x) = 10^{-20}x^2 + 4 \cdot x + 2 \cdot 10^{20} = 0$ . For a zero  $z$ , compute  $\gamma(f, z)$  (see Lecture 5) and the corresponding bound on the radius of the disc of guaranteed quadratic convergence of Newton's method. Scale the coefficients of  $f$  and do the same computations of  $\gamma$  and the radius on a zero of the scaled polynomial. Compare and interpret the results.
3. Use Gröbner bases to solve the system (12) symbolically.
4. Make a Maple worksheet or Sage notebook to model the problem given in (13) and (14), using the constants in the table. With this worksheet or notebook you can then generate three different instances of the same problem. Use `phc` to solve these instances. Scaling is available via `phc -s`.
5. Consider the intersection of two circles:

$$f(x_1, x_2, c) = \begin{cases} x^2 + y^2 - 1 = 0 \\ (x - c)^2 + y^2 - 1 = 0. \end{cases} \quad (28)$$

Examine the condition number of one of the solutions of  $f(x_1, x_2, c) = \mathbf{0}$  as  $c$  goes from 0 to 2.

Is the condition number a continuous function of  $c$ ?

## References

- [1] J.W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.
- [2] K. Meintjes and A.P. Morgan. Performance of algorithms for calculating the equilibrium composition of a mixture of gases. *Journal of Computational Physics*, 60:219–234, 1985.
- [3] K. Meintjes and A.P. Morgan. Chemical equilibrium systems as numerical test problems. *ACM Transactions on Mathematical Software*, 16(2):143–151, 1990.
- [4] A. Morgan. *Solving polynomial systems using continuation for engineering and scientific problems*. Prentice-Hall, 1987.
- [5] A.J. Sommese and C.W. Wampler. *The Numerical solution of systems of polynomials arising in engineering and science*. World Scientific Press, Singapore, 2005.