Roman Z. Morawski
phone: (22) 234-7721, e-mail: *roman.morawski@pw.edu.pl*

Numerical Methods (ENUME)

1. INTRODUCTION

Lecture notes for Spring Semester 2022/2023

Lecture classes

Contents:

Analysis of numerical problems and algorithms

Solving linear algebraic equations

Solving nonlinear algebraic equations

Approximation of functions

Solving ordinary differential equations

Lecture classes

Schedule:

Date	Lectures	Tests
February 23	2 h	
March 2	2 h	
March 9	2 h	
March 23	2 h	
March 30	2 h	
April 6	1 h	1 h (Test#1)
April 13	2 h	
April 20	2 h	
April 27	2 h	
May 11	2 h	
May 18	2 h	
May 25	2 h	
June 1	1 h	1 h (Test#2)
June 15		2 h (Tests#1'&2')
Total	**26 h**	**4 h**

Note: Test #1' is fully equivalent to Test #1, and Tests #2' is fully equivalent to Test #2; all the ENUME students are eligible to take all the tests.

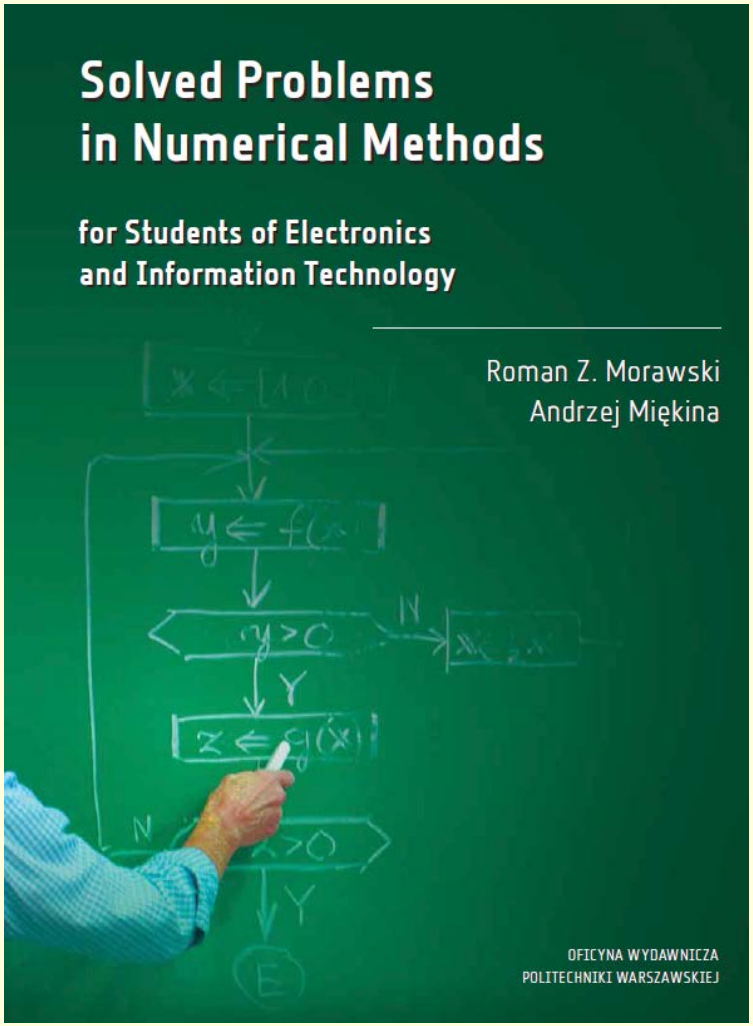
Lecture classes

Teaching materials:

R. Z. Morawski, *Lecture notes for ENUME students*, Spring 2023

R. Z. Morawski, A. Miękina: *Solved problems in numerical methods*, Oficyna Wydawnicza PW, Warszawa 2021

Electronic textbooks on numerical methods available in the Main Library of Warsaw University of Technology (examples on the next slide)



Lecture classes

Addresses of selected e-books on numerical methods, available in the Main Library of WUT:

https://primo-48tuw.hosted.exlibrisgroup.com/permalink/f/1nmcehm/48TUW_ejournals3710000000360322
https://primo-48tuw.hosted.exlibrisgroup.com/permalink/f/1nmcehm/48TUW_ejournals3710000000269608
https://primo-48tuw.hosted.exlibrisgroup.com/permalink/f/1nmcehm/48TUW_ejournals3710000000717968
https://primo-48tuw.hosted.exlibrisgroup.com/permalink/f/1nmcehm/48TUW_ejournals3710000000097667
https://primo-48tuw.hosted.exlibrisgroup.com/permalink/f/1nmcehm/48TUW_ejournals2550000000104910
https://primo-48tuw.hosted.exlibrisgroup.com/permalink/f/1nmcehm/48TUW_ejournals3710000000205415
https://primo-48tuw.hosted.exlibrisgroup.com/permalink/f/1nmcehm/48TUW_ejournals3710000000331863
https://primo-48tuw.hosted.exlibrisgroup.com/permalink/f/1nmcehm/48TUW_WTU01000196082
https://primo-48tuw.hosted.exlibrisgroup.com/permalink/f/1nmcehm/48TUW_ejournals2670000000501196
https://primo-48tuw.hosted.exlibrisgroup.com/permalink/f/1nmcehm/48TUW_ejournals2670000000501196
https://primo-48tuw.hosted.exlibrisgroup.com/permalink/f/1nmcehm/48TUW_ejournals2550000000045851
https://primo-48tuw.hosted.exlibrisgroup.com/permalink/f/1nmcehm/48TUW_ejournals2550000000045851
https://primo-48tuw.hosted.exlibrisgroup.com/permalink/f/1nmcehm/48TUW_ejournals2560000000324791

Project classes

Tutors:

Paweł Mazurek, e-mail: *pawel.mazurek@pw.edu.pl*

Andrzej Miękina, e-mail: *andrzej.miekina@pw.edu.pl*

Jakub Wagner, e-mail: *jakub.wagner@pw.edu.pl*

Teaching materials:

MATLAB introduction by Paweł Mazurek

- part 1: *https://youtu.be/0qMhRGm0ppo*
- part 2: *https://youtu.be/e6WhfLuF4NM*

Assignments:

A. Accuracy of computation	*ca.* 8 h of homework
B. Nonlinear algebraic equations	*ca.* 10 h of homework
C. Ordinary differential equations	*ca.* 12 h of homework

Grading

Partial grading

Project assignment A: accuracy of computation	15 pts
Project assignment B: nonlinear algebraic equations	15 pts
Project assignment C: ordinary differential equations	20 pts
Test #1	20 pts
Test #2	30 pts
	100 pts

Final grading

precondition: $\geq$ 25 pts from project assignments & $\geq$ 25 pts from tests

$0 \leq \sum \text{pts} < 50 \Rightarrow 2$

$50 \leq \sum \text{pts} < 60 \Rightarrow 3$

$60 \leq \sum \text{pts} < 70 \Rightarrow 3.5$

$70 \leq \sum \text{pts} < 80 \Rightarrow 4$

$80 \leq \sum \text{pts} < 90 \Rightarrow 4.5$

$90 \leq \sum \text{pts} \leq 100 \Rightarrow 5$

1.1. Mathematical modelling in engineering practice

1.2. Vectors and matrices – recapitulation of linear algebra

Basic definitions

$$\mathbf{x} \equiv \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \equiv \begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix}^T - N\text{-dimensional vector}$$

$$\mathbf{A} \equiv \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M,1} & a_{M,2} & \cdots & a_{M,N} \end{bmatrix} \equiv \begin{bmatrix} a_{1,1} & a_{2,1} & \cdots & a_{M,1} \\ a_{1,2} & a_{2,2} & \cdots & a_{M,2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1,N} & a_{2,N} & \cdots & a_{M,N} \end{bmatrix}^T - M \times N\text{-dimensional matrix}$$

The most important norms of vectors:

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{n=1}^{N} |x_n|^2} \qquad \text{– the Euclidean norm}$$

$$\|\mathbf{x}\|_\infty = \sup\left\{ |x_n| \;\middle|\; n = 1, ..., N \right\} \qquad \text{– the maximum norm}$$

The most important norms of matrices:

$$\|\mathbf{A}\|_2 = \sqrt{\mathrm{sr}\left(\mathbf{A}^T \mathbf{A}\right)} \qquad \text{– the norm induced by } \|\mathbf{x}\|_2$$

where $\mathrm{sr}(\circ)$ – the operator of spectra radius determination

$$\|\mathbf{A}\|_\infty = \sup\left\{ \sum_{\nu=1}^{N} |a_{n,\nu}| \;\middle|\; n = 1, ..., N \right\} \text{– the norm induced by } \|\mathbf{x}\|_\infty$$

A useful inequality:

$$\|\mathbf{A} \cdot \mathbf{x}\|_p \le \|\mathbf{A}\|_p \cdot \|\mathbf{x}\|_p \quad \text{for} \quad p = 2, \infty$$

Spectral characteristics of matrices

Definitions:

- the eigenvalue (λ_n) and eigenvector ($\mathbf{v}_n$) of a square ($N \times N$) non-singular matrix $\mathbf{A}$:

$$\mathbf{A} \cdot \mathbf{v}_n = \lambda_n \cdot \mathbf{v}_n \text{ and } \left\| \mathbf{v}_n \right\|_2 = 1 \text{ for } n = 1, ..., N$$

- the spectral radius of $\mathbf{A}$:

$$\mathrm{sr}(\mathbf{A}) = \sup\left\{ \left|\lambda_1\right|, ..., \left|\lambda_N\right| \right\}$$

Properties:

- For any matrix $\mathbf{A}$, the eigenvectors ($\mathbf{v}_n$) are linearly independent, and:

$$\mathbf{A} \cdot \mathbf{V} = \mathbf{V} \cdot \mathbf{\Lambda} \quad \Rightarrow \quad \mathbf{A} = \mathbf{V} \cdot \mathbf{\Lambda} \cdot \mathbf{V}^{-1}$$

  where: $\mathbf{V} \equiv \left[\mathbf{v}_1 \ ... \ \mathbf{v}_N \right]$ and $\mathbf{\Lambda} \equiv diag\left\{ \lambda_1, ..., \lambda_N \right\}$

- For a symmetrical real-valued matrix $\mathbf{A}$, the eigenvectors ($\mathbf{v}_n$) are orthonormal, and – consequently – the matrix $\mathbf{V}$ is unitary, *i*.e.:

$$\mathbf{V}^{-1} \equiv \left[\mathbf{v}_1 \ ... \ \mathbf{v}_N \right]^{-1} = \mathbf{V}^T$$

  which implies: $\mathbf{A} = \mathbf{V} \cdot \mathbf{\Lambda} \cdot \mathbf{V}^T$.

1.3. Numerical algorithm (NA) and its description

Numerical problem: $\mathbb{D} \xrightarrow{\mathscr{R}} \mathbb{W}$:

$\mathbb{D} \equiv \{\text{vectors of data}: \ \mathbf{d}_1, \mathbf{d}_2, \ldots \}$

$\mathbb{W} \equiv \{\text{vectors of results: } \mathbf{w}_1, \mathbf{w}_2, \ldots \}$

$\mathscr{R}$ – requirements

Example: $y = P_N(x; \mathbf{a}) = \sum_{n=0}^{N} a_n x^n$

vectors of data: $\mathbf{d} = [\hat{x} \ a_0 \ a_1 \ \ldots a_N]^T$

vectors of results: $\mathbf{w} = [\hat{y}]$

requirements: $\mathscr{R}: \hat{y} = P_N(\hat{x}; \mathbf{a})$.

Numerical algorithm = an ordered sequence of operations, transforming an element of $\mathbb{D}$ in an element of $\mathbb{W}$ satisfying the requirements $\mathscr{R}$.

Forms of NA description:

- (traditional) mathematical notation
- programming languages
- flow diagrams
- sequential notation

Example: The Horner's algorithm for computing the value of a polynomial:

$$y_N = a_N$$

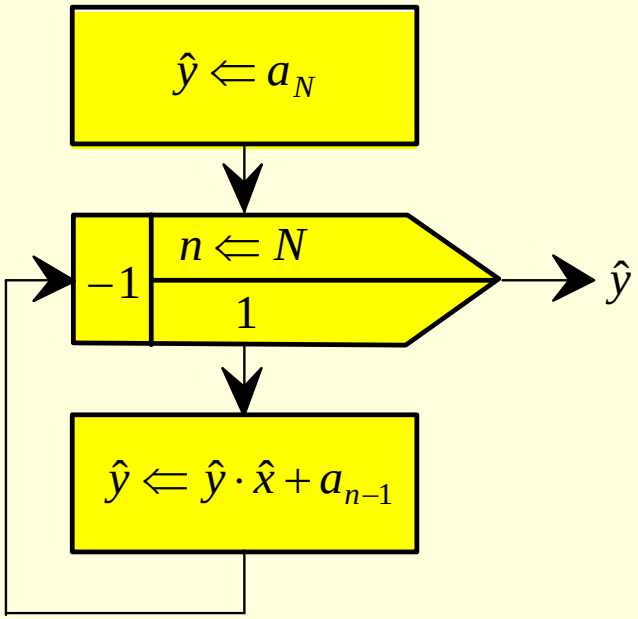
$$y_{n-1} = y_n \hat{x} + a_{n-1} \qquad \text{for} \quad n = N, N-1, \ldots, 1$$

$$\hat{y} = y_0$$

- the C language:

$$y = a[N];\ \text{for}\ (n = N; n > 0; n--)\ \{y = y * x + a[n-1];\}$$

- the flow diagram:



- the sequential notation:

$$\begin{bmatrix} \hat{x} \\ a_0 \\ a_1 \\ \vdots \\ a_{N-1} \\ a_N \end{bmatrix} \in \mathbb{D} \rightarrow \begin{bmatrix} \hat{x} \\ a_0 \\ a_1 \\ \vdots \\ a_{N-1} \\ \hat{y} \end{bmatrix} \rightarrow \begin{bmatrix} \hat{x} \\ a_0 \\ a_1 \\ \vdots \\ \hat{y} \end{bmatrix} \rightarrow \cdots \rightarrow \begin{bmatrix} \hat{x} \\ a_0 \\ \hat{y} \end{bmatrix} \rightarrow [\hat{y}] \in \mathbb{W}$$

Roman Z. Morawski
phone: (22) 234-7721, e-mail: *roman.morawski@pw.edu.pl*

Numerical Methods (ENUME)

2. ACCURACY AND COMPLEXITY OF COMPUTING

Lecture notes for Spring Semester 2022/2023

2.1. Effects of the finite representation of numbers in computers

Example: Subtracting numbers:

$$\begin{array}{rl} 369.711\,(\pm 0.0005) & \text{– relative error } \sim 1.3\cdot 10^{-4}\ \% \\ -369.702\,(\pm 0.0005) & \text{– relative error } \sim 1.3\cdot 10^{-4}\ \% \\ \hline 0.009\,(\pm 0.0010) & \text{– relative error } \sim 11\,\% \end{array}$$

Example: Computing zeros of the polynomial:

$$y = x^{20} + a_{19}x^{19} + \ldots + a_1 x + a_0 = \prod_{n=1}^{20}(x-n)$$

with the coefficient a_{19} disturbed at the level $6\cdot 10^{-6}$ % $\Rightarrow$ complex zeros, *e.g.* $13.99 \pm j2.51$.

2.2. Floating-point representation of numbers in computers

The general form of the floating-point representation:

$$\dot{x} = \pm \dot{m} \cdot 10^{c}$$

$$\dot{m} = 0.m_1 m_2 ... m_{L-1} m_L m_{L+1} ..., \text{ where } m_i \in \{0, ..., 9\},\ m_1 \neq 0$$

The rounding principle:

$$\tilde{x} = \pm \tilde{m} \cdot 10^{c}$$

$$\tilde{m} = \begin{cases} 0.m_1 m_2 ... m_L & \text{if } 0 \le m_{L+1} < 5 \\ 0.m_1 m_2 ... m_L + 10^{-L} & \text{if } 5 \le m_{L+1} \le 9 \end{cases}$$

The limit relative error:

$$\left|\delta\left[\tilde{x}\right]\right| = \left|\frac{\tilde{m} \cdot 10^{c} - \dot{m} \cdot 10^{c}}{\dot{m} \cdot 10^{c}}\right| \le \frac{\sup\left|\tilde{m} - \dot{m}\right|}{\inf\left|\dot{m}\right|} = \frac{5 \cdot 10^{-(L+1)}}{10^{-1}} = 5 \cdot 10^{-L} \equiv eps$$

The convenient representation of disturbed data:

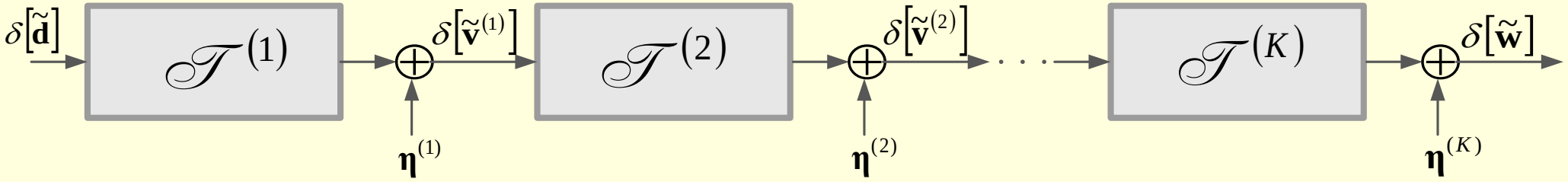
$$\delta\left[\tilde{x}\right] = \varepsilon = \frac{\tilde{x} - \dot{x}}{\dot{x}} \quad \Rightarrow \quad \tilde{x} = \dot{x}\left(1 + \varepsilon\right), \text{ where } \varepsilon \in \left[-eps, +eps\right]$$

2.3. The model of error propagation

The numerical algorithm for solving the numerical problem, $\phi : \mathbb{D} \to \mathbb{W}$, has the form:

$$\mathbf{d} \equiv \mathbf{v}^{(0)} \xrightarrow{\phi^{(1)}} \mathbf{v}^{(1)} \xrightarrow{\phi^{(2)}} \ldots \to \mathbf{v}^{(K-1)} \xrightarrow{\phi^{(K)}} \mathbf{v}^{(K)} \equiv \mathbf{w}$$

where: $\phi = \phi^{(K)} \circ \phi^{(K-1)} \circ \ldots \circ \phi^{(2)} \circ \phi^{(1)}$



where: $\delta\left[\tilde{\mathbf{v}}^{(k)}\right] = \mathscr{T}^{(k)}\left\{\delta\left[\tilde{\mathbf{v}}^{(k-1)}\right]\right\} + \boldsymbol{\eta}^{(k)}$ for $k = 1, \ldots, K$

After linearisation:

$$\delta\left[\tilde{\mathbf{v}}^{(k)}\right] = \mathbf{T}^{(k)} \cdot \delta\left[\tilde{\mathbf{v}}^{(k-1)}\right] + \boldsymbol{\eta}^{(k)} \quad \text{for } k = 1, \ldots, K$$

with $\mathbf{T}^{(k)}$ being a matrix of the coefficients of relative errors propagation

Propagation of error across a scalar function of a scalar variable

$$y = \phi(x)$$

$$\tilde{y} = \phi(\tilde{x}) \;\Rightarrow\; \dot{y} + \Delta\tilde{y} \equiv \phi(\dot{x} + \Delta\tilde{x})$$

where $\Delta\tilde{y}, \Delta\tilde{x}$ – absolute errors corrupting $\tilde{y}$ and $\tilde{x}$, respectively

$$\tilde{y} = \phi(\dot{x}) + \phi'(\dot{x})\Delta\tilde{x} + \tfrac{1}{2}\phi''(\dot{x})(\Delta\tilde{x})^2 + \ldots = \dot{y} + \dot{x}\phi'(\dot{x})\frac{\Delta\tilde{x}}{\dot{x}} + \tfrac{1}{2}\dot{x}^2\phi''(\dot{x})\left(\frac{\Delta\tilde{x}}{\dot{x}}\right)^2 + \ldots$$

$$\Delta\tilde{y} = \tilde{y} - \dot{y} = \dot{x}\phi'(\dot{x})\varepsilon + \tfrac{1}{2}\dot{x}^2\phi''(\dot{x})\varepsilon^2 + \ldots, \text{ where } \varepsilon = \Delta\tilde{x}/\dot{x}$$

$$\delta[\tilde{y}] \equiv \frac{\Delta\tilde{y}}{\dot{y}} = \frac{\dot{x}}{\dot{y}}\phi'(\dot{x})\varepsilon + \tfrac{1}{2}\frac{\dot{x}^2}{\dot{y}}\phi''(\dot{x})\varepsilon^2 + \ldots$$

$$|\varepsilon| << 1 \;\Rightarrow\; |\varepsilon|^2 << |\varepsilon| \quad\Rightarrow\quad \delta[\tilde{y}] \cong T\varepsilon, \text{ where } T = \left.\frac{\frac{dy}{y}}{\frac{dx}{x}}\right|_{x=\dot{x}} = \left.\frac{x}{y}\frac{dy}{dx}\right|_{x=\dot{x}} = \left.\frac{d\ \ln(y)}{d\ \ln(x)}\right|_{x=\dot{x}}$$

Propagation of error across a scalar function of a vector variable

$$\phi:\ y=\phi\left(x_1,x_2,...,x_N\right)\ \Rightarrow\ \ \delta\left[\tilde{y}\right]\cong\sum_{n=1}^{N}T_n\varepsilon_n$$

where: $\varepsilon_n\equiv\delta\left[\tilde{x}_n\right]$ and $T_n=\left.\frac{x_n}{y}\frac{\partial y}{\partial x_n}\right|_{x_n=\dot{x}_n}=\left.\frac{\partial\ \ln\left(y\right)}{\partial\ \ln\left(x_n\right)}\right|_{x_n=\dot{x}_n}$

Example: $\varepsilon_n\equiv\delta\tilde{x}_n$ for $n=1,2,...$

$$\delta\left[\tilde{x}_1\pm\tilde{x}_2\right]\cong\frac{\dot{x}_1}{\dot{x}_1\pm\dot{x}_2}\varepsilon_1\pm\frac{\dot{x}_2}{\dot{x}_1\pm\dot{x}_2}\varepsilon_2\quad\Rightarrow\quad\delta\left[\tilde{x}_1-\tilde{x}_2\right]=\frac{\dot{x}_1}{\dot{x}_1-\dot{x}_2}\varepsilon_1-\frac{\dot{x}_2}{\dot{x}_1-\dot{x}_2}\varepsilon_2\xrightarrow[\substack{\dot{x}_1\to\dot{x}_2\\ \varepsilon_1\neq\varepsilon_2\neq0}]{}\infty$$

$$\delta\left[\tilde{x}_1^{a}\tilde{x}_2^{b}\right]\cong a\varepsilon_1+b\varepsilon_2\ \text{ for }\left|a\right|,\left|b\right|<<eps^{-1}$$

$$\delta\left[\tilde{x}_1^{\tilde{x}_2}\right]\cong\dot{x}_2\varepsilon_1+\dot{x}_2\ln\left(\dot{x}_1\right)\varepsilon_2$$

$$\delta\left[\ln\left(\tilde{x}\right)\right]\cong\frac{1}{\ln\left(\dot{x}\right)}\varepsilon$$

Basic rules of "epsilon" calculus

$$(1+\varepsilon_1)(1+\varepsilon_2) \cong 1+\varepsilon_1+\varepsilon_2$$

$$(1+\varepsilon)^a \cong 1+a\varepsilon \quad \text{for} \quad |a| << eps^{-1}$$

$$\ln(1+\varepsilon) \cong \varepsilon$$

$$e^{1+\varepsilon} \cong (1+\varepsilon)e$$

Example: Two methods for computing $y = x_1^2 - x_2^2$:

$$\mathscr{A}_1: \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} v_1 = x_1^2 \\ v_2 = x_2^2 \end{bmatrix} \rightarrow [y = v_1 - v_2]$$

$$\mathscr{A}_2: \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} v_1 = x_1 + x_2 \\ v_2 = x_1 - x_2 \end{bmatrix} \rightarrow [y = v_1 v_2]$$

$$\mathcal{A}_1: \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} v_1 = x_1^2 \\ v_2 = x_2^2 \end{bmatrix} \rightarrow \left[y = v_1 - v_2 \right]$$

$$\tilde{y}_1 = \left\{ \left[\dot{x}_1 \left(1+\varepsilon_1\right) \right]^2 \left(1+\eta_1\right) - \left[\dot{x}_2 \left(1+\varepsilon_2\right) \right]^2 \left(1+\eta_2\right) \right\} \left(1+\eta_3\right)$$

$$\tilde{y}_1 \cong \left\{ \dot{x}_1^2 \left(1+2\varepsilon_1\right)\left(1+\eta_1\right) - \dot{x}_2^2 \left(1+2\varepsilon_2\right)\left(1+\eta_2\right) \right\} \left(1+\eta_3\right)$$

$$\tilde{y}_1 \cong \left\{ \dot{x}_1^2 \left(1+2\varepsilon_1+\eta_1\right) - \dot{x}_2^2 \left(1+2\varepsilon_2+\eta_2\right) \right\} \left(1+\eta_3\right)$$

$$\tilde{y}_1 \cong \left\{ \left(\dot{x}_1^2 - \dot{x}_2^2 \right) + \left[\dot{x}_1^2 \left(2\varepsilon_1+\eta_1\right) - \dot{x}_2^2 \left(2\varepsilon_2+\eta_2\right) \right] \right\} \left(1+\eta_3\right)$$

$$\tilde{y}_1 \cong \dot{y} \left\{ 1 + \left[\frac{\dot{x}_1^2}{\dot{y}} \left(2\varepsilon_1+\eta_1\right) - \frac{\dot{x}_2^2}{\dot{y}} \left(2\varepsilon_2+\eta_2\right) \right] \right\} \left(1+\eta_3\right) = \dot{y} \left\{ 1 + \left[\frac{\dot{x}_1^2}{\dot{y}} \left(2\varepsilon_1+\eta_1\right) - \frac{\dot{x}_2^2}{\dot{y}} \left(2\varepsilon_2+\eta_2\right) \right] + \eta_3 \right\}$$

$$\delta\left[\tilde{y}_1\right] \cong T_1\varepsilon_1 + T_2\varepsilon_2 + K_1\eta_1 + K_2\eta_2 + K_3\eta_3, \text{ where: } T_1 = 2\frac{\dot{x}_1^2}{\dot{y}},\ T_2 = -2\frac{\dot{x}_2^2}{\dot{y}},\ K_1 = \frac{\dot{x}_1^2}{\dot{y}},\ K_2 = -\frac{\dot{x}_2^2}{\dot{y}},\ K_3 = 1$$

$$\left|\delta\left[\tilde{y}_1\right]\right| \le \left|T_1\right| eps + \left|T_2\right| eps + \left|K_1\right| eps + \left|K_2\right| eps + \left|K_3\right| eps$$

$$= \left(\left|T_1\right| + \left|T_2\right|\right) eps + \left(\left|K_1\right| + \left|K_2\right| + \left|K_3\right|\right) eps \le 2\frac{\dot{x}_1^2 + \dot{x}_2^2}{\left|\dot{y}\right|} eps + \left(\frac{\dot{x}_1^2 + \dot{x}_2^2}{\left|\dot{y}\right|} + 1 \right) eps$$

$$\mathscr{A}_2: \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} v_1 = x_1 + x_2 \\ v_2 = x_1 - x_2 \end{bmatrix} \rightarrow \left[y = v_1 v_2 \right]$$

$$\ldots$$

$$\left|\delta\left[\tilde{y}_2\right]\right| \le \left|T_1\right| eps + \left|T_2\right| eps + \left|K_s\right| eps + \left|K_o\right| eps + \left|K_m\right| eps = \left(\left|T_1\right| + \left|T_2\right|\right) eps + \left(\left|K_s\right| + \left|K_o\right| + \left|K_m\right|\right) eps$$

$$\ldots$$

$$\left|\delta\left[\tilde{y}_2\right]\right| \le \underbrace{2\frac{\dot{x}_1^2 + \dot{x}_2^2}{\left|\dot{y}\right|} eps}_{\equiv \delta_{GR}^{O}} + \underbrace{3 eps}_{\equiv \delta_{GR2}^{G}}$$

The above assessment differs from that derived for $\mathscr{A}_1$:

$$\left|\delta\left[\tilde{y}_1\right]\right| \le \underbrace{2\frac{\dot{x}_1^2 + \dot{x}_2^2}{\left|\dot{y}\right|} eps}_{\equiv \delta_{GR}^{O}} + \underbrace{\left(\frac{\dot{x}_1^2 + \dot{x}_2^2}{\left|\dot{y}\right|} + 1\right) eps}_{\equiv \delta_{GR1}^{G}}$$

only by the component generated by rounding errors:

$$\frac{1}{\sqrt{3}} \le \left|\frac{\dot{x}_1}{\dot{x}_2}\right| \le \sqrt{3} \quad \Rightarrow \quad \delta_{GR1}^{G} \ge \delta_{GR2}^{G}$$

Notice: The dots indicating the exact values of variables (e.g. $\dot{x}$, $\dot{y}$, ...) are usually omitted, if their absence does not imply ambiguity.

2.4. Propagation of errors in the data

The model of the propagation of error across an elementary operation:

$$\delta\left[\tilde{\mathbf{v}}^{(k)}\right] = \mathbf{T}^{(k)} \cdot \delta\left[\tilde{\mathbf{v}}^{(k-1)}\right] + \boldsymbol{\eta}^{(k)}$$

under an assumption that $\boldsymbol{\eta}^{(k)} = \mathbf{0}$ for $k = 1, ..., K$, enables one to evaluate the component of the error in the final result of computation, caused by the errors in the data:

$$\delta^{O}\left[\tilde{\mathbf{y}}\right] = \mathbf{T}^{(K)} \cdot \mathbf{T}^{(K-1)} \cdot \ldots \cdot \mathbf{T}^{(1)} \cdot \boldsymbol{\varepsilon} = \mathbf{T} \cdot \boldsymbol{\varepsilon}, \text{ where } \mathbf{T} \equiv \mathbf{T}^{(K)} \cdot \mathbf{T}^{(K-1)} \cdot \ldots \cdot \mathbf{T}^{(1)}$$

If the small relative changes of the data $\mathbf{d}$ cause large relative changes of the result $\mathbf{w}$, then the numerical problem is called *ill-conditioned*.

The conditioning of the numerical problem (and consequently $\delta^{O}\left[\tilde{\mathbf{y}}\right]$) is characterised by the matrix $\mathbf{T}$ which does not depend on the numerical algorithm used for solving this problem.

Example: Solving quadratic equation:

$$y^2 - 2x_1 y + x_2 = 0 \text{ for } x_1 > 0,\ x_2 < 0$$

$$y_1 = x_1 + \sqrt{x_1^2 - x_2},\ y_2 = x_1 - \sqrt{x_1^2 - x_2}$$

$$T_{1,1} = \frac{x_1}{y_1} \cdot \frac{\partial y_1}{\partial x_1} = \frac{x_1}{y_1} \cdot \frac{y_1}{y_1 - x_1} = \frac{x_1}{\sqrt{x_1^2 - x_2}} \in (0,1)$$

$$T_{1,2} = \frac{x_2}{y_1} \cdot \frac{\partial y_1}{\partial x_2} = \frac{x_2}{y_1} \cdot \frac{1}{2(x_1 - y_1)} = \frac{1}{2}\left(1 - \frac{x_1}{\sqrt{x_1^2 - x_2}}\right) \in \left(0, \frac{1}{2}\right)$$

$$T_{2,1} = \frac{x_1}{y_2} \cdot \frac{\partial y_2}{\partial x_1} = \frac{x_1}{y_2} \cdot \frac{y_2}{y_2 - x_1} = \frac{-x_1}{\sqrt{x_1^2 - x_2}} \in (-1,0)$$

$$T_{2,2} = \frac{x_2}{y_2} \cdot \frac{\partial y_2}{\partial x_2} = \frac{x_2}{y_2} \frac{1}{2(x_1 - y_2)} = \frac{1}{2}\left(1 + \frac{x_1}{\sqrt{x_1^2 - x_2}}\right) \in \left(\frac{1}{2}, 1\right)$$

$|T_{i,j}| \le 1$ for $i, j = 1,2 \quad \Rightarrow$ very good numerical conditioning.

The conditioning of the numerical problem may be characterised by the so-called *condition number*, *i*.e. the worst-case amplification of the aggregated relative error in the data $\frac{\|\Delta\tilde{\mathbf{d}}\|}{\|\dot{\mathbf{d}}\|}$:

$$\mathrm{cond}(\mathbf{d}) \equiv \sup\left\{ \frac{\frac{\|\Delta\tilde{\mathbf{w}}\|}{\|\dot{\mathbf{w}}\|}}{\frac{\|\Delta\tilde{\mathbf{d}}\|}{\|\dot{\mathbf{d}}\|}} \;\middle|\; \forall \Delta\tilde{\mathbf{d}} \right\} \Rightarrow \frac{\|\Delta\tilde{\mathbf{w}}\|}{\|\dot{\mathbf{w}}\|} \le \mathrm{cond}\left(\mathbf{d}\right) \cdot \frac{\|\Delta\tilde{\mathbf{d}}\|}{\|\dot{\mathbf{d}}\|}$$

where: $\Delta\tilde{\mathbf{d}} = \tilde{\mathbf{d}} - \dot{\mathbf{d}}$ and $\Delta\tilde{\mathbf{w}} = \tilde{\mathbf{w}} - \dot{\mathbf{w}}$

2.5. Propagation of rounding errors

The model of the propagation of error across an elementary operation:

$$\delta\left[\tilde{\mathbf{v}}^{(k)}\right] = \mathbf{T}^{(k)} \cdot \delta\left[\tilde{\mathbf{v}}^{(k-1)}\right] + \boldsymbol{\eta}^{(k)}$$

under an assumption that $\delta\left[\tilde{\mathbf{d}}\right] = \mathbf{0}$, enables one to evaluate the component of the error in the result of computation $\tilde{\mathbf{w}}$, caused by the rounding errors:

$$\delta^G\left[\tilde{\mathbf{w}}\right] = \sum_{k=1}^{K} \mathbf{K}^{(k)} \boldsymbol{\eta}^{(k)}$$

where $\mathbf{K}^{(k)} = \mathbf{T}^{(K)} \cdot \mathbf{T}^{(K-1)} \cdot \ldots \cdot \mathbf{T}^{(k+1)}$ for $k = 1,\ldots,K-1$; $\mathbf{K}^{(K)} = \mathbf{I}$

Example: Two algorithms for computing $y = \frac{1}{x} - \frac{1}{1+x} \equiv \frac{1}{x \cdot (1+x)}$ for $x > 0$:

$$\mathscr{A}_1: \quad [x] \rightarrow \begin{bmatrix} v_1 = 1/x \\ v_2 = 1/(1+x) \end{bmatrix} \rightarrow [y = v_1 - v_2]$$

$$\mathscr{A}_2: \quad [x] \rightarrow \begin{bmatrix} v_1 = x \\ v_2 = 1+x \end{bmatrix} \rightarrow \left[y = \frac{1}{v_1 v_2} \right]$$

$$\mathscr{A}_1:[x]\to\begin{bmatrix} v_1=1/x \\ v_2=1/(1+x)\end{bmatrix}\to[y=v_1-v_2]$$

$$\tilde{y}_1=\left[\frac{1}{x}(1+\eta_1)-\frac{1}{(1+x)(1+\eta_s)}(1+\eta_2)\right](1+\eta_o)$$

$$\tilde{y}_1=\left[\frac{1}{x}(1+\eta_1)-\frac{1}{1+x}(1+\eta_2-\eta_s)\right](1+\eta_o)$$

$$\tilde{y}_1=\left\{\left(\frac{1}{x}-\frac{1}{1+x}\right)+\left[\frac{1}{x}\eta_1-\frac{1}{1+x}(\eta_2-\eta_s)\right]\right\}(1+\eta_o)$$

$$\tilde{y}_1=y\left\{1+\frac{1}{xy}\eta_1-\frac{1}{(1+x)y}(\eta_2-\eta_s)\right\}(1+\eta_o)$$

$$\delta^G[\tilde{y}_1]=(1+x)\eta_1-x\eta_2+x\eta_s+\eta_o\Rightarrow\left|\delta^G[\tilde{y}_1]\right|\le|1+x|eps+|x|eps+|x|eps+eps=(3x+2)eps$$

$$\mathscr{A}_2:[x]\to\begin{bmatrix} v_1=x \\ v_2=1+x\end{bmatrix}\to\left[y=\frac{1}{v_1v_2}\right]$$

$$\tilde{y}_2=\frac{1+\eta_d}{x(1+x)(1+\eta_s)(1+\eta_m)}\cong y(1+\eta_d-\eta_s-\eta_m)\quad\Rightarrow\quad\left|\delta^G[\tilde{y}_2]\right|\le 3eps$$

$$x<\tfrac{1}{3}\quad\Rightarrow\quad\sup\left|\delta^G[\tilde{y}_1]\right|<\sup\left|\delta^G[\tilde{y}_2]\right|.$$

Numerical correctness of a numerical algorithm

An algorithm $\mathscr{A}(\mathbf{d})$ for $\mathbf{d} \in \mathbb{D}$ is numerically correct if:

– Interpretation #1: when applied to the exact data, it provides the results being the exact solutions of the numerical problem for slightly disturbed data.

– Interpretation #2: the contribution of rounding errors to the total error in the result of computation $\tilde{\mathbf{w}}$ remains in a "practically acceptable" relationship to the contribution of the errors in the data $\tilde{\mathbf{d}}$.

Example: An algorithm for computing $y = x_1^2 + x_2^2$ is numerically correct because:

$$\tilde{y} = \left[x_1^2 (1+\eta_1) + x_2^2 (1+\eta_2) \right](1+\eta_s)$$

$$\tilde{y} \cong \left[x_1^2 (1+\eta_1+\eta_s) + x_2^2 (1+\eta_2+\eta_s) \right]$$

$$\tilde{y} \cong \left[x_1 (1+\tfrac{1}{2}\eta_1+\tfrac{1}{2}\eta_s) \right]^2 + \left[x_2 (1+\tfrac{1}{2}\eta_2+\tfrac{1}{2}\eta_s) \right]^2$$

Example: The "classical" algorithm for solving quadratic equations:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} x_1 \\ v = \sqrt{x_1^2 - x_2} \end{bmatrix} \rightarrow \begin{bmatrix} y_1 = x_1 + v \\ y_2 = x_1 - v \end{bmatrix} \quad \text{for} \quad x_1 > 0, \quad x_2 < 0$$

is numerically unstable because of the subtracting close numbers when $x_2 \rightarrow 0$:

$$K_{2,\sqrt{\ }} = \frac{\sqrt{x_1^2 - x_2}}{y_2} \cdot \frac{\partial y_2}{\partial\left(\sqrt{x_1^2 - x_2}\right)} = -\frac{\sqrt{x_1^2 - x_2}}{x_1 - \sqrt{x_1^2 - x_2}} \xrightarrow[x_2 \rightarrow 0]{} \infty$$

2.6. Using computers for the analysis of computational accuracy

The most important techniques for evaluation of the coefficients of error propagation:

- "epsilon" calculus
- symbolic differentiation
- numerical differentiation
- statistical simulation

2.7. Non-numerical applications of the methods for accuracy analysis

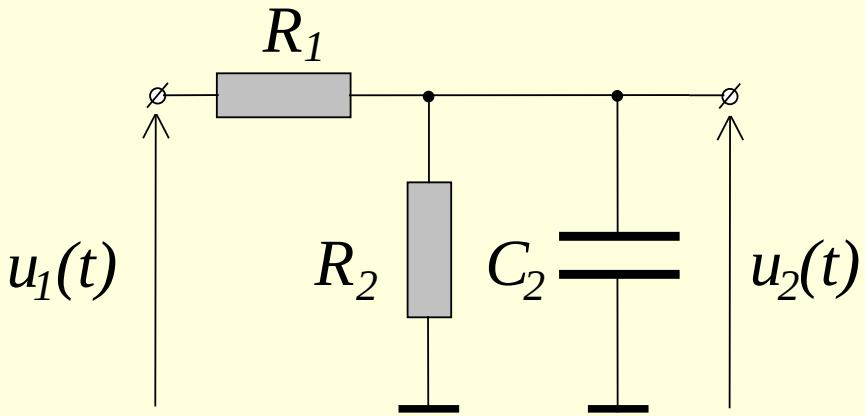
Implementation of numerical algorithms in floating-point processors:

- estimation of limit errors
- estimation of the minimum number of digits of mantissa, sufficient for providing required accuracy
- estimation of admissible level of measurement errors to guarantee the required accuracy of computation using measurement data
- selection of algorithms providing most accurate results of computation
- correction of numerically unstable algorithms

Other engineering tasks:

- accuracy analysis of measuring devices and systems
- accuracy analysis of electronic circuits such as filters, bridges, converters
- sensitivity analysis of electronic circuits
- "small-signal" analysis of electronic circuits

Example: Estimate the sensitivity of the cut-off frequency ω_{3dB} of the filter:



to the scattering of the parameters of its elements:

$$\tilde{R}_1 = R_1(1+\rho_1),\ \ \tilde{R}_2 = R_2(1+\rho_2),\ \ \tilde{C}_2 = C_2(1+\zeta_2),\ \text{where } |\rho_1|,|\rho_2|,|\zeta_2| \le 1\%$$

$$K(s) = \frac{U_2(s)}{U_1(s)} = \frac{1}{(1+R_1/R_2)\cdot\left(1+s\dfrac{R_1C_2}{1+R_1/R_2}\right)}$$

$$\omega_{3dB} = \frac{1+R_1/R_2}{R_1C_2} \Rightarrow \tilde{\omega}_{3dB} = \frac{1+\tilde{R}_1/\tilde{R}_2}{\tilde{R}_1\tilde{C}_2} = \frac{1+\dfrac{R_1(1+\rho_1)}{R_2(1+\rho_2)}}{R_1(1+\rho_1)C_2(1+\zeta_2)}$$

$$\tilde{\omega}_{3dB} = \omega_{3dB} \cdot \left[1 + \frac{R_1}{R_1 + R_2}\left(\rho_1 - \rho_2\right) - \rho_1 - \zeta_2 \right]$$

$$\delta\left[\tilde{\omega}_{3dB}\right] = \frac{R_1}{R_1 + R_2}\left(\rho_1 - \rho_2\right) - \rho_1 - \zeta_2 = -\frac{R_2}{R_1 + R_2}\rho_1 - \frac{R_1}{R_1 + R_2}\rho_2 - \zeta_2$$

$$\left|\delta\left[\tilde{\omega}_{3dB}\right]\right| \le \left|\frac{R_2}{R_1 + R_2}\right|\left|\rho_1\right| + \left|\frac{R_1}{R_1 + R_2}\right|\left|\rho_2\right| + \left|\zeta_2\right| \le \left(\frac{2R_2}{R_1 + R_2} + 1\right) \cdot 1\% = \frac{R_1 + 3R_2}{R_1 + R_2} \cdot 1\%$$

2.8. Numerical complexity

$f_{NA}(N)$ – the dependence of the number of operations on the dimension of the problem N

CLASS OF ALGORITHMS	CLASS OF $f_{NA}(N)$	EXEMPLARY COMPUTING TIME	
		$N=10$	$N=60$
polynomial (effective) algorithms $\exists K, \{\alpha_k\}\ \forall N : f_{NA}(N) \le \sum_{k=0}^{K} \alpha_k N^k$	$O(N)$	$0.0001\,s$	$0.0006\,s$
	$O(N^3)$	$0.001\,s$	$0.216\,s$
	$O(N^5)$	$0.1\,s$	$780\,s$
exponential (ineffective) algorithms $\forall K, \{\alpha_k\}\ \exists N_0\ \forall N > N_0 : f_{NA}(N) > \sum_{k=0}^{K} \alpha_k N^k$	$O(2^N)$	$0.001\,s$	336 600 years
	$O(3^N)$	$0.059\,s$	$1.3 \cdot 10^{15}$ years

Engineering measures of numerical complexity:

– computing time measured on a reference computer

– size of memory occupied in a reference computer

Roman Z. Morawski
phone: (22) 234-7721, e-mail: *roman.morawski@pw.edu.pl*

Numerical Methods (ENUME)

3. SOLVING LINEAR ALGEBRAIC EQUATIONS

Lecture notes for Spring Semester 2022/2023

3.1. Formulation and numerical conditioning of the problem

The system of linear algebraic equations to be solved has the form:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$$

where:

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,N} \\ a_{2,1} & a_{2,2} & \dots & a_{2,N} \\ a_{3,1} & a_{3,2} & .. & a_{3,N} \\ \vdots & \vdots & \vdots & \vdots \\ a_{N,1} & a_{N,2} & \dots & a_{N,N} \end{bmatrix}, \ \det(\mathbf{A}) \neq 0, \text{ and } \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_N \end{bmatrix}, \text{ with } a_{n,m}, b_n \in \mathbb{R}$$

The propagation of errors in the data $\mathbf{b}$:

$$\mathbf{A}\cdot(\mathbf{x}+\Delta\mathbf{x})=\mathbf{b}+\Delta\mathbf{b} \quad\Rightarrow\quad \Delta\mathbf{x}=\mathbf{A}^{-1}\cdot\Delta\mathbf{b} \quad\Rightarrow\quad \|\Delta\mathbf{x}\|\le\|\mathbf{A}^{-1}\|\cdot\|\Delta\mathbf{b}\|$$

$$\mathbf{b}=\mathbf{A}\cdot\mathbf{x} \quad\Rightarrow\quad \|\mathbf{b}\|\le\|\mathbf{A}\|\cdot\|\mathbf{x}\| \quad\Rightarrow\quad \frac{1}{\|\mathbf{x}\|}\le\|\mathbf{A}\|\cdot\frac{1}{\|\mathbf{b}\|}$$

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|}\le\|\mathbf{A}^{-1}\|\cdot\|\mathbf{A}\|\cdot\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|}=\mathrm{cond}(\mathbf{A})\cdot\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|}$$

where: $\mathrm{cond}(\mathbf{A})$ is a condition number of the matrix $\mathbf{A}$ defined as follows:

$$\mathrm{cond}(\mathbf{A})=\mathrm{cond}_{\mathrm{p}}(\mathbf{A})=\|\mathbf{A}\|_p\cdot\|\mathbf{A}^{-1}\|_p \text{ for } p=2 \text{ or } p=\infty$$

The propagation of errors in the data $\mathbf{b}$ and $\mathbf{A}$:

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|}\le\frac{\mathrm{cond}(\mathbf{A})\cdot\dfrac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|}}{1-\mathrm{cond}(\mathbf{A})\cdot\dfrac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|}}+\mathrm{cond}(\mathbf{A})\cdot\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|}$$

Example: The condition number of the Hilbert's matrix:

$$a_{m,n} = \frac{1}{m+n-1} \qquad m, n = 1, \ldots, N$$

N	2	5	10	50
$cond(\mathbf{A})$	19.28	$4.77 \cdot 10^{5}$	$1.60 \cdot 10^{13}$	$8.51 \cdot 10^{19}$

General classification of the methods for solving $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$:

- finite (non-iterative) methods
- iterative methods

3.2. Finite methods for solving $\mathbf{A}\cdot\mathbf{x}=\mathbf{b}$

Solving a system with an upper-triangular matrix

$$\begin{aligned}
a_{1,1}x_1 &+ a_{1,2}x_2 &+ \cdots &+ a_{1,N-1}x_{N-1} &+ a_{1,N}x_N &= b_1\\
&\quad a_{2,2}x_2 &+ \cdots &+ a_{2,N-1}x_{N-1} &+ a_{2,N}x_N &= b_2\\
&&&\quad\vdots &\quad\vdots &\quad\vdots\\
&&&\quad a_{N-1,N-1}x_{N-1} &+ a_{N-1,N}x_N &= b_{N-1}\\
&&&&\quad a_{N,N}x_N &= b_N
\end{aligned}$$

The algorithm:

$$x_N=\frac{b_N}{a_{N,N}} \text{ and } x_n=\frac{\left(b_n-\sum_{\nu=n+1}^{N}a_{n,\nu}x_\nu\right)}{a_{n,n}} \text{ for } n=N-1,\ldots,1$$

The number of operations:

$$L(+,-)=\tfrac{1}{2}N^2-\tfrac{1}{2}N\propto O\left(\tfrac{1}{2}N^2\right) \quad \text{and} \quad L(*,/)=\tfrac{1}{2}N^2+\tfrac{1}{2}N\propto O\left(\tfrac{1}{2}N^2\right)$$

Method of Gauss elimination

The two-phase algorithm:

- the phase of elimination: a sequence of linear transformations leading to the conversion of $\mathbf{A}\cdot\mathbf{x}=\mathbf{b}$ into a system with an upper-triangular matrix having the same solution
- the phase of solving the system with an upper-triangular matrix

$$\left.\begin{array}{ccccccccc} a_{1,1}^{(1)}x_1 & + & a_{1,2}^{(1)}x_2 & + & \cdots & + & a_{1,N}^{(1)}x_N & = & b_1^{(1)} \\ a_{2,1}^{(1)}x_1 & + & a_{2,2}^{(1)}x_2 & + & \cdots & + & a_{2,N}^{(1)}x_N & = & b_2^{(1)} \\ \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ a_{N,1}^{(1)}x_1 & + & a_{N,2}^{(1)}x_2 & + & \cdots & + & a_{N,N}^{(1)}x_N & = & b_N^{(1)} \end{array}\right\} \quad \Leftrightarrow \quad \mathbf{A}^{(1)}\cdot\mathbf{x}=\mathbf{b}^{(1)}$$

where: $\mathbf{A}^{(1)}\equiv\mathbf{A}$ and $\mathbf{b}^{(1)}\equiv\mathbf{b}$

Elimination, step #1: zeroing of the elements number $2, ..., N$ in the first column of $\mathbf{A}^{(1)}$ by elimination of x_1 from the equations number $2, ..., N$, under an assumption that $a_{1,1}^{(1)} \neq 0$:

$$l_{n,1} \overset{\text{df}}{=} \frac{a_{n,1}^{(1)}}{a_{1,1}^{(1)}} \text{ for } n = 2, \ldots, N$$

$$\left.\begin{aligned} a_{n,\nu}^{(2)} &= a_{n,\nu}^{(1)} - l_{n,1} a_{1,\nu}^{(1)} \quad \text{for} \quad \nu = 1, \ldots, N \\ b_n^{(2)} &= b_n^{(1)} - l_{n,1} b_1^{(1)} \end{aligned}\right\} \text{ for } n = 2, \ldots, N$$

$$\Downarrow$$

$$\begin{array}{ccccccccc} a_{1,1}^{(1)} x_1 & + & a_{1,2}^{(1)} x_2 & + & \cdots & + & a_{1,N}^{(1)} x_N & = & b_1^{(1)} \\ 0 & + & a_{2,2}^{(2)} x_2 & + & \cdots & + & a_{2,N}^{(2)} x_N & = & b_2^{(2)} \\ \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & + & a_{N,2}^{(2)} x_2 & + & \cdots & + & a_{N,N}^{(2)} x_N & = & b_N^{(2)} \end{array} \quad \Leftrightarrow \quad \mathbf{A}^{(2)} \cdot \mathbf{x} = \mathbf{b}^{(2)}$$

Elimination, step #2: zeroing of the elements number $3, ..., N$ in the second column of $\mathbf{A}^{(2)}$ by elimination of x_2 from the equations number $3, ..., N$, under an assumption that $a_{2,2}^{(2)} \neq 0$:

$$l_{n,2} \overset{\text{df}}{=} \frac{a_{n,2}^{(2)}}{a_{2,2}^{(2)}} \text{ for } n = 3, \ldots, N$$

$$\left.\begin{aligned} a_{n,\nu}^{(3)} &= a_{n,\nu}^{(2)} - l_{n,2} a_{2,\nu}^{(2)} \quad \text{for} \quad \nu = 2, \ldots, N \\ b_n^{(3)} &= b_n^{(2)} - l_{n,2} b_2^{(2)} \end{aligned}\right\} \text{ for } n = 3, \ldots, N$$

$$\Downarrow$$

$$\begin{array}{ccccccccc} a_{1,1}^{(1)} x_1 & + & a_{1,2}^{(1)} x_2 & + & \cdots & + & a_{1,N}^{(1)} x_N & = & b_1^{(1)} \\ 0 & + & a_{2,2}^{(2)} x_2 & + & \cdots & + & a_{2,N}^{(2)} x_N & = & b_2^{(2)} \\ \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & + & 0 & + & \cdots & + & a_{N,N}^{(3)} x_N & = & b_N^{(3)} \end{array} \quad \Leftrightarrow \quad \mathbf{A}^{(3)} \cdot \mathbf{x} = \mathbf{b}^{(3)}$$

$$\vdots$$

After N – steps: $\mathbf{A}^{(N)} \cdot \mathbf{x} = \mathbf{b}^{(N)}$, where $\mathbf{A}^{(N)}$ is an upper-triangular matrix.

LU factorisation

The Gauss method of elimination is equivalent to the following decomposition of $\mathbf{A}$:

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{U}$$

where $\mathbf{U} = \mathbf{A}^{(N)}$ and $\mathbf{L} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{2,1} & 1 & \cdots & 0 \\ l_{3,1} & l_{3,2} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ l_{N,1} & l_{N,2} & \cdots & 1 \end{bmatrix}$

The system of equations:

$$\mathbf{L} \cdot \mathbf{U} \cdot \mathbf{x} = \mathbf{b}$$

may be solved in two steps – by solving two simpler systems of equations:

$$\mathbf{L} \cdot \mathbf{y} = \mathbf{b} \quad \text{and} \quad \mathbf{U} \cdot \mathbf{x} = \mathbf{y}$$

with the triangular matrices $\mathbf{L}$ and $\mathbf{U}$.

Example: The *in-place* processing of the matrix $\mathbf{A}$ leading to its LU factorisation:

$$\begin{bmatrix} 3 & 1 & 6 \\ 2 & 1 & 3 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 7 \\ 4 \end{bmatrix}$$

$$\left[\begin{array}{ccc|c} 3 & 1 & 6 & 2 \\ 2 & 1 & 3 & 7 \\ 1 & 1 & 1 & 4 \end{array}\right] \Rightarrow \begin{array}{l} l_{2,1} = \dfrac{a_{2,1}^{(1)}}{a_{1,1}^{(1)}} = \dfrac{2}{3} \\ l_{3,1} = \dfrac{a_{3,1}^{(1)}}{a_{1,1}^{(1)}} = \dfrac{1}{3} \end{array} \Rightarrow \left[\begin{array}{ccc|c} 3 & 1 & 6 & 2 \\ \frac{2}{3} & \frac{1}{3} & -1 & \frac{17}{3} \\ \frac{1}{3} & \frac{2}{3} & -1 & \frac{10}{3} \end{array}\right] \Rightarrow l_{3,2} = \frac{a_{3,2}^{(2)}}{a_{2,2}^{(2)}} = 2 \Rightarrow \left[\begin{array}{ccc|c} 3 & 1 & 6 & 2 \\ \frac{2}{3} & \frac{1}{3} & -1 & \frac{17}{3} \\ \frac{1}{3} & 2 & 1 & -8 \end{array}\right]$$

Hence:

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{2}{3} & 1 & 0 \\ \frac{1}{3} & 2 & 1 \end{bmatrix} \text{ and } \mathbf{U} = \begin{bmatrix} 3 & 1 & 6 \\ 0 & \frac{1}{3} & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

The number of operations:

- the LU factorisation: $L(+,-) \propto O\left(\frac{1}{3}N^3\right)$ and $L(*,/) \propto= O\left(\frac{1}{3}N^3\right)$
- the solution of two systems with triangular matrices: $L(+,-) \propto O\left(N^2\right)$ and $L(*,/) \propto O\left(N^2\right)$

Some practical comments:

- The LU factorisation is especially convenient and efficient if the system of linear equations is solved several times, for the same matrix $\mathbf{A}$ and various vectors $\mathbf{b}$.
- To minimize numerical errors, the rows and columns of the matrix $\mathbf{A}$ are re-ordered at each step of the LU factorisation as to maximize $a_{n,n}^{(n)} \neq 0$. This operation is equivalent to the multiplication of the system by a zero-one pivoting matrix $\mathbf{P}$:

  $$\mathbf{P} \cdot \mathbf{A} \cdot \mathbf{x} = \mathbf{P} \cdot \mathbf{b} \quad \Rightarrow \quad \mathbf{L} \cdot \mathbf{U} \cdot \mathbf{x} = \mathbf{P} \cdot \mathbf{b}$$

  Then, two triangular systems of linear equations ale solved to find the solution:

  $$\mathbf{L} \cdot \mathbf{y} = \mathbf{P} \cdot \mathbf{b} \text{ and } \mathbf{U} \cdot \mathbf{x} = \mathbf{y}$$

Cholesky-Banachiewicz factorisation

A symmetric matrix $\mathbf{A}$ is positive definite if $\forall \mathbf{x} \neq \mathbf{0}\colon\ \mathbf{x}^T \cdot \mathbf{A} \cdot \mathbf{x} > 0$.

Any positive definite matrix $\mathbf{A}$ may be factorised in the following way:

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{L}^T$$

where $\mathbf{L} = \begin{bmatrix} l_{1,1} & 0 & \cdots & 0 \\ l_{2,1} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ l_{N,1} & l_{N,2} & \cdots & l_{N,N} \end{bmatrix}$ is a unique matrix with positive diagonal elements

The elements of the matrix $\mathbf{L}$ may be found by comparison of the LHS and RHS of:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & \cdots & a_{N,N} \end{bmatrix} = \begin{bmatrix} l_{1,1} & 0 & \cdots & 0 \\ l_{2,1} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{N,1} & l_{N,2} & \cdots & l_{N,N} \end{bmatrix} \cdot \begin{bmatrix} l_{1,1} & l_{2,1} & \cdots & l_{N,1} \\ 0 & l_{2,2} & \cdots & l_{N,2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & l_{N,N} \end{bmatrix}.$$

i.e. by solving the following system of linear equations:

$$l_{1,1}^2 = a_{1,1}, \quad l_{n,1} \cdot l_{1,1} = a_{n,1} \quad \text{for} \quad n = 2,...,N$$

$$l_{2,1}^2 + l_{2,2}^2 = a_{22}, \quad l_{n,1} \cdot l_{2,1} + l_{n,2} \cdot l_{2,2} = a_{n,2} \quad \text{for} \quad n = 3,...,N$$

etc.

Its solution has the form:

$$\left.\begin{aligned} l_{n,n} &= \sqrt{a_{n,n} - \sum_{i=1}^{n-1} l_{n,i}^2} \\ l_{\nu,n} &= \frac{a_{\nu,n} - \sum_{i=1}^{n-1} l_{\nu,i} \cdot l_{n,i}}{l_{n,n}} \quad \text{for} \quad \nu = n+1, \ldots, N \end{aligned}\right\} \text{ for } \quad n = 1, \ldots, N$$

The number of operations: $L(+,-) \propto O\left(\frac{1}{6}N^3\right)$, $L(*,/) \propto= O\left(\frac{1}{6}N^3\right)$ and $L\left(\sqrt{\ }\right) = N$.

Residual correction (iterative improvement) of the solution

The solution $\hat{\mathbf{x}}^{(1)}$, obtained by means of the LU or LL^T factorisation, may be given the form:

$$\hat{\mathbf{x}}^{(1)} = \dot{\mathbf{x}} + \Delta\mathbf{x}$$

where $\dot{\mathbf{x}}$ is the exact solution of $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$.

The equation resulting from the substitution of $\dot{\mathbf{x}} = \hat{\mathbf{x}}^{(1)} - \Delta\mathbf{x}$ to $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$:

$$\mathbf{A}\cdot\left(\hat{\mathbf{x}}^{(1)} - \Delta\mathbf{x}\right) = \mathbf{b}$$

may be given the form:

$$\mathbf{A}\cdot\Delta\mathbf{x} = \mathbf{A}\cdot\hat{\mathbf{x}}^{(1)} - \mathbf{b}$$

An estimate $\Delta\hat{\mathbf{x}}$ of $\Delta\mathbf{x}$ may be obtained by solving this equation, using the already obtained results of the LU or LL^T factorisation of $\mathbf{A}$. It may be then used for correction of the solution $\hat{\mathbf{x}}^{(1)}$:

$$\hat{\mathbf{x}}^{(2)} = \hat{\mathbf{x}}^{(1)} - \Delta\hat{\mathbf{x}}$$

3.3. Computing determinants and inverse matrices

To calculate the determinant $\det(\mathbf{A})$ with a possibly small numerical error, one should use the LU or LL^T factorisation of the matrix $\mathbf{A}$ in the following way:

$$\det(\mathbf{A}) = \det(\mathbf{L}\cdot\mathbf{U}) = \det(\mathbf{L})\cdot\det(\mathbf{U}) = \det(\mathbf{U}) = \prod_{n=1}^{N} u_{n,n}$$

$$\det(\mathbf{A}) = \det(\mathbf{L}\cdot\mathbf{L}^T) = \left[\det(\mathbf{L})\right]^2 = \left(\prod_{n=1}^{N} l_{n,n}\right)^2$$

To calculate the inverse matrix $\mathbf{A}^{-1}$ in an efficient and accurate way, one should use the equality $\mathbf{A}\cdot\mathbf{A}^{-1}=\mathbf{I}$ which may be re-written in the form:

$$\mathbf{A}\cdot\begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,N} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N,1} & y_{N,2} & \cdots & y_{N,N} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

$$\begin{matrix} \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_N \end{matrix} \qquad \begin{matrix} \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_N \end{matrix}$$

or in the form of a set of N systems of linear equations with the same matrix $\mathbf{A}$:

$$\mathbf{A}\cdot\mathbf{y}_1=\mathbf{e}_1$$

$$\mathbf{A}\cdot\mathbf{y}_2=\mathbf{e}_2$$

$$\cdots$$

$$\mathbf{A}\cdot\mathbf{y}_N=\mathbf{e}_N$$

The use of the LU (or LL^T) factorisation is recommended for solving those equations:

$$\mathbf{L}\cdot\mathbf{U}\cdot\mathbf{y}_1=\mathbf{e}_1$$

$$\mathbf{L}\cdot\mathbf{U}\cdot\mathbf{y}_2=\mathbf{e}_2$$

$$\cdots$$

$$\mathbf{L}\cdot\mathbf{U}\cdot\mathbf{y}_N=\mathbf{e}_N$$

3.4. Iterative methods for solving $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$

A distinctive feature of an iterative algorithm: the repetition of identical or similar sequences of operations (from Latin *iterare* = to repeat).

Example: The equation $a \cdot x = b$ $(a \neq 0)$ may be solved by means of the following iterative algorithm:

$$x^{(i+1)} = x^{(i)} + \gamma \cdot \left[a \cdot x^{(i)} - b \right] \quad \text{for} \quad i = 0, 1, \ldots$$

The speed of its convergence depends on γ because:

$$\dot{x} + \Delta x^{(i+1)} = \dot{x} + \Delta x^{(i)} + \gamma \cdot \left[a \cdot \left(\dot{x} + \Delta x^{(i)} \right) - b \right], \text{ where } \Delta x^{(i)} \equiv x^{(i)} - \dot{x},\ \Delta x^{(i+1)} \equiv x^{(i+1)} - \dot{x}$$

$$\Delta x^{(i+1)} = \Delta x^{(i)} + \gamma \cdot a \cdot \Delta x^{(i)} = \left(1 + \gamma \cdot a\right) \cdot \Delta x^{(i)}$$

$$\left|\Delta x^{(i+1)}\right| < \left|\Delta x^{(i)}\right| \quad \text{if} \quad \left|1 + \gamma \cdot a\right| < 1$$

The convergence is guaranteed if:

$$-1 < 1 + \gamma \cdot a < 1 \ \Leftrightarrow\ -2 < \gamma \cdot a < 0 \ \Leftrightarrow\ -2 < \gamma \cdot sign(a) \cdot |a| < 0$$

$$\Leftrightarrow\ -\frac{2}{|a|} < \gamma \cdot sign(a) < 0 \ \Leftrightarrow\ \begin{cases} \gamma \in \left(0, -2/a\right) & \text{for } a < 0 \\ \gamma \in \left(-2/a, 0\right) & \text{for } a > 0 \end{cases}$$

General properties of iterative methods

A linear iterative method for solving $\mathbf{A}\cdot\mathbf{x}=\mathbf{b}$ is defined by the formula:

$$\hat{\mathbf{x}}^{(i+1)}=\mathbf{M}\cdot\hat{\mathbf{x}}^{(i)}+\mathbf{w} \text{ for } i=0,1,...,$$

where $\hat{\mathbf{x}}^{(0)}$ is the initial approximation of the solution, while $\mathbf{M}$ and $\mathbf{w}$ satisfy the coherence condition:

$$\mathbf{x}=\mathbf{M}\cdot\mathbf{x}+\mathbf{w}.$$

The method is convergent if $\mathrm{sr}(\mathbf{M})<1$; this condition follows from the following reasoning:

$$\mathbf{x}+\Delta\hat{\mathbf{x}}^{(i+1)}=\mathbf{M}\cdot\left(\mathbf{x}+\Delta\hat{\mathbf{x}}^{(i)}\right)+\mathbf{w}\Rightarrow\Delta\hat{\mathbf{x}}^{(i+1)}=\mathbf{M}\cdot\Delta\hat{\mathbf{x}}^{(i)}$$

$$\mathbf{M}=\mathbf{V}\cdot\mathbf{\Lambda}\cdot\mathbf{V}^{-1}\Rightarrow\ \Delta\hat{\mathbf{x}}^{(i+1)}=\mathbf{V}\cdot\mathbf{\Lambda}\cdot\mathbf{V}^{-1}\cdot\Delta\hat{\mathbf{x}}^{(i)}$$

where $\mathbf{\Lambda}=diag\{\lambda_1,\lambda_2,...\}$ is the matrix of the eigenvalues of $\mathbf{M}$, and $\mathbf{V}$ is the matrix of its eigenvectors; hence:

$$\underbrace{\mathbf{V}^{-1}\cdot\Delta\hat{\mathbf{x}}^{(i+1)}}_{\Delta\mathbf{z}^{(i+1)}}=\mathbf{\Lambda}\cdot\underbrace{\mathbf{V}^{-1}\cdot\Delta\hat{\mathbf{x}}^{(i)}}_{\Delta\mathbf{z}^{(i)}}\ \Leftrightarrow\Delta z_m^{(i+1)}=\lambda_m\Delta z_m^{(i)} \text{ for } m=1,2,...$$

which means that the convergence is guaranteed if $|\lambda_m|<1$ for $m=1,2,...$

The speed of convergence, *i*.e. the speed of approaching zero by:

$$\left\|\Delta\hat{\mathbf{x}}^{(i)}\right\| = \left\|\hat{\mathbf{x}}^{(i)} - \dot{\mathbf{x}}\right\| \text{ for } i = 0, 1, ...,$$

increases if $\mathrm{sr}\left(\mathbf{M}\right)$ goes down.

Typical stop conditions:

$$\frac{\left\|\hat{\mathbf{x}}^{(i+1)} - \hat{\mathbf{x}}^{(i)}\right\|}{\left\|\hat{\mathbf{x}}^{(i)}\right\|} \le \delta x \text{ and } \frac{\left\|\mathbf{A}\cdot\hat{\mathbf{x}}^{(i+1)} - \mathbf{b}\right\|}{\left\|\mathbf{b}\right\|} \le \delta b$$

where δx and δb are indicator of admissible aggregated errors.

Jacobi method

The Jacobi method is based on the following decomposition of $\mathbf{A}$:

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}$$

where $\mathbf{L}$ is a lower-triangular matrix, $\mathbf{D}$ is a diagonal matrix and $\mathbf{U}$ is an upper-triangular matrix, e.g.:

$$\underbrace{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}}_{\mathbf{A}} = \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{bmatrix}}_{\mathbf{L}} + \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{bmatrix}}_{\mathbf{D}} + \underbrace{\begin{bmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{bmatrix}}_{\mathbf{U}}$$

A heuristic development of the Jacobi method includes the following steps:

$$\mathbf{D} \cdot \mathbf{x} = -\left(\mathbf{L} + \mathbf{U}\right) \cdot \mathbf{x} + \mathbf{b}$$

$$\mathbf{D} \cdot \hat{\mathbf{x}}^{(i+1)} = -\left(\mathbf{L} + \mathbf{U}\right) \cdot \hat{\mathbf{x}}^{(i)} + \mathbf{b} \text{ for } i = 0, 1, \ldots$$

$$\hat{\mathbf{x}}^{(i+1)} = -\mathbf{D}^{-1} \cdot \left(\mathbf{L} + \mathbf{U}\right) \cdot \hat{\mathbf{x}}^{(i)} + \mathbf{D}^{-1} \cdot \mathbf{b} \text{ for } i = 0, 1, \ldots$$

Hence:

$$\mathbf{M} \equiv -\mathbf{D}^{-1} \cdot \left(\mathbf{L} + \mathbf{U}\right) \text{ and } \mathbf{w} \equiv \mathbf{D}^{-1} \cdot \mathbf{b}$$

The Jacobi method is convergent for any $\mathbf{A}$ with a strongly dominant diagonal, *i*.e. if one of the following conditions is satisfied:

$$|a_{n,n}| > \sum_{\substack{\nu=1 \\ \nu \neq n}}^{N} |a_{n,\nu}| \text{ for } n = 1, \ldots, N \quad \text{or} \quad |a_{n,n}| > \sum_{\substack{\nu=1 \\ \nu \neq n}}^{N} |a_{\nu,n}| \text{ for } n = 1, \ldots, N$$

Warning: The Jacobi method is not convergent for all positive definite matrices $\mathbf{A}$!

Gauss-Seidel method

The Gauss-Seidel method is based on the following decomposition of $\mathbf{A}$:

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}$$

where $\mathbf{L}$ is a lower-triangular matrix, $\mathbf{D}$ is a diagonal matrix, $\mathbf{U}$ is an upper-triangular matrix.

A heuristic development of the Gauss-Seidel method includes the following steps:

$$\left(\mathbf{L} + \mathbf{D}\right) \cdot \mathbf{x} = -\mathbf{U} \cdot \mathbf{x} + \mathbf{b}$$

$$\left(\mathbf{L} + \mathbf{D}\right) \cdot \hat{\mathbf{x}}^{(i+1)} = -\mathbf{U} \cdot \hat{\mathbf{x}}^{(i)} + \mathbf{b} \quad \text{for} \quad i = 0, 1, 2, \ldots$$

$$\hat{\mathbf{x}}^{(i+1)} = -\left(\mathbf{L} + \mathbf{D}\right)^{-1} \cdot \mathbf{U} \cdot \hat{\mathbf{x}}^{(i)} + \left(\mathbf{L} + \mathbf{D}\right)^{-1} \cdot \mathbf{b} \quad \text{for} \quad i = 0, 1, 2, \ldots$$

i.e.: $\mathbf{M} \equiv -\left(\mathbf{L} + \mathbf{D}\right)^{-1} \cdot \mathbf{U}$ and $\cdot \mathbf{w} \equiv \left(\mathbf{L} + \mathbf{D}\right)^{-1} \cdot \mathbf{b}$

The computing algorithm for an iteration consists in solving the following equation with a triangular matrix:

$$\left(\mathbf{L}+\mathbf{D}\right)\cdot\hat{\mathbf{x}}^{(i+1)}=-\mathbf{U}\cdot\hat{\mathbf{x}}^{(i)}+\mathbf{b}\quad\text{for}\quad i=0,1,2,\ldots$$

By transformations

$$\mathbf{D}\cdot\hat{\mathbf{x}}^{(i+1)}=-\mathbf{L}\cdot\hat{\mathbf{x}}^{(i+1)}-\mathbf{U}\cdot\hat{\mathbf{x}}^{(i)}+\mathbf{b}\quad\text{for}\quad i=0,1,2,\ldots$$

$$\hat{\mathbf{x}}^{(i+1)}=-\overline{\mathbf{L}}\cdot\hat{\mathbf{x}}^{(i+1)}-\overline{\Delta\mathbf{b}}^{(i)}\quad\text{for}\quad i=0,1,2,\ldots$$

where:

$$\overline{\mathbf{L}}=\mathbf{D}^{-1}\cdot\mathbf{L},\ \overline{\Delta\mathbf{b}}=\mathbf{D}^{-1}\cdot\left(\mathbf{U}\cdot\hat{\mathbf{x}}^{(i)}-\mathbf{b}\right)$$

one may get the following scalar implementation:

$$\hat{x}_1^{(i+1)}=-\Delta b_1^{(i)}$$

$$\hat{x}_2^{(i+1)}=-\bar{l}_{21}\cdot\hat{x}_1^{(i+1)}-\Delta b_2^{(i)}$$

$$\hat{x}_3^{(i+1)}=-\bar{l}_{31}\cdot\hat{x}_1^{(i+1)}-\bar{l}_{32}\cdot\hat{x}_2^{(i+1)}-\Delta b_3^{(i)}$$

etc.

The Gauss-Seidel method is convergent for any $\mathbf{A}$ with a strongly dominant diagonal, *i*.e. if one of the following conditions is satisfied:

$$\left|a_{n,n}\right| > \sum_{\substack{\nu=1 \\ \nu \neq n}}^{N} \left|a_{n,\nu}\right| \text{ for } n = 1, \ldots, N \quad \text{or} \quad \left|a_{n,n}\right| > \sum_{\substack{\nu=1 \\ \nu \neq n}}^{N} \left|a_{\nu,n}\right| \text{ for } n = 1, \ldots, N$$

The Gauss-Seidel method is convergent for any positive definite matrix $\mathbf{A}$.

Roman Z. Morawski
phone: (22) 234-7721, e-mail: *roman.morawski@pw.edu.pl*

Numerical Methods (ENUME)

4. SOLVING NONLINEAR ALGEBRAIC EQUATIONS

Lecture notes for Spring Semester 2022/2023

4.1. System of nonlinear algebraic equations

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \text{ where: } \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \text{ and } \mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_N(\mathbf{x}) \end{bmatrix}$$

Example:

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \sin(x_1 + x_2) - 1 \\ \cos(x_2 + x_3) + 1 \\ \tan(x_3 + x_4) - 1 \\ x_1 + x_2 + x_3 + x_4 \end{bmatrix}$$

4.2. General characteristics of iterative algorithms

An iterative algorithm (IA) for solving a numerical problem $\mathbf{d} \to \mathbf{w}$ has the form:

$$\mathbf{w}^{(i+1)} = \varphi\left(\mathbf{w}^{(i)}, \mathbf{w}^{(i-1)}, \ldots; \mathbf{d}\right) \quad \text{for } i=0,1,\ldots$$

where:

$\varphi_i(\cdot)$ is an operator defining IA

$\mathbf{w}^{(i)}$ is the ith approximation of the solution to the problem $\mathbf{d} \to \mathbf{w}$

Examples:

The Heron algorithm for solving $w^2 - d = 0$ $(d > 0)$, *i*.e. for computing $\sqrt{d}$:

$$w^{(0)} = \max\{d, 1\}, \qquad w^{(i+1)} = \tfrac{1}{2}\left(w^{(i)} + \frac{d}{w^{(i)}}\right) \quad \text{for } i = 0,1,\ldots$$

The Newton algorithm for solving $f(w; \mathbf{d}) = 0$:

$$w^{(i+1)} = w^{(i)} - \frac{f\left(w^{(i)}; \mathbf{d}\right)}{f'\left(w^{(i)}; \mathbf{d}\right)} \quad \text{for } i = 0,1,\ldots$$

An IA is *globally convergent* if there exists a non-empty set $\mathbb{W}_0$ of the starting points $\mathbf{w}^{(0)}$ that:

$$\left\|\Delta\mathbf{w}^{(i)}\right\| = \left\|\mathbf{w}^{(i)} - \dot{\mathbf{w}}\right\| \xrightarrow[i\to\infty]{} 0 \quad \text{for} \quad i = 0,1,...$$

An IA is *locally convergent* if it is convergent for any (even very small) vicinity $\mathbb{W}_0$ of the convergence point $\dot{\mathbf{w}}$. The analysis of local convergence may be performed by means of the error function:

$$\Delta\mathbf{w}^{(i+1)} = \Phi_i\left(\Delta\mathbf{w}^{(i)}; \mathbf{d}\right) \quad \text{for} \quad i = 0,1,...$$

taking into account that the error should diminish to zero for $i \to \infty$. The local convergence of a scalar IA is characterised by an approximation of the function Φ_i in the vicinity of $\dot{w}$ by:

$$\left\|\Delta\mathbf{w}^{(i+1)}\right\| \cong C \cdot \left\|\Delta\mathbf{w}^{(i)}\right\|^{\rho} \text{ (or } \left|\Delta w^{(i+1)}\right| \cong C \cdot \left|\Delta w^{(i)}\right|^{\rho} \text{ in the scalar case)}$$

where:

$C > 0$ is the *coefficient of local convergence* and $\rho \in [1,\infty)$ *is the exponent of local convergence*

The attainable (relative) accuracy of a scalar IA is characterised by an interval:

$$\frac{\left\|\Delta\mathbf{w}^{(i)}\right\|}{\left\|\dot{\mathbf{w}}^{(i)}\right\|} \in \left[-K \cdot eps, +K \cdot eps\right] \text{ (or } \delta\left[w^{(i)}\right] \in \left[-K \cdot eps, +K \cdot eps\right] \text{ in the scalar case)}$$

where K is the *indicator of attainable accuracy*

4.3. Analysis of scalar one-point IAs

A scalar one-point IA is defined by the formula:

$$w^{(i+1)} = \varphi\left(w^{(i)}; \mathbf{d}\right) \quad \text{for} \quad i = 0, 1, \ldots$$

which will be further processed using a simplified notation:

$$w_{i+1} = \varphi\left(w_i\right) \quad \text{for} \quad i = 0, 1, \ldots \text{ and } \Delta w^{(i)} \equiv \Delta_i$$

The parameters of local convergence may be determined using the Taylor series expansion of $\varphi\left(w_i\right)$ in the vicinity of $\dot{w}$:

$$w_{i+1} = \underbrace{\varphi\left(\dot{w}\right)}_{\dot{w}} + \varphi'\left(\dot{w}\right)\left(w_i - \dot{w}\right) + \tfrac{1}{2}\varphi''\left(\dot{w}\right)\left(w_i - \dot{w}\right)^2 + \ldots$$

$$\underbrace{w_{i+1} - \dot{w}}_{\Delta_{i+1}} = \underbrace{\varphi\left(\dot{w}\right) - \dot{w}}_{0} + \varphi'\left(\dot{w}\right)\left(\underbrace{w_i - \dot{w}}_{\Delta_i}\right) + \tfrac{1}{2}\varphi''\left(\dot{w}\right)\left(\underbrace{w_i - \dot{w}}_{\Delta_i}\right)^2 + \ldots$$

Hence:

$$\Delta_{i+1} = \varphi'\left(\dot{w}\right)\Delta_i + \tfrac{1}{2}\varphi''\left(\dot{w}\right)\Delta_i^2 + \ldots$$

For "small" $\Delta_i \quad (\Delta_i \to 0)$, the following approximation of:

$$\Delta_{i+1} = \varphi'(\dot{w})\Delta_i + \tfrac{1}{2}\varphi''(\dot{w})\Delta_i^2 + \tfrac{1}{6}\varphi'''(\dot{w})\Delta_i^3 + \ldots$$

is justified:

$$\varphi'(\dot{w}) \neq 0 \quad \Rightarrow \quad \Delta_{i+1} \cong \varphi'(\dot{w})\Delta_i$$

$$\varphi'(\dot{w}) = 0 \quad \text{and} \quad \varphi''(\dot{w}) \neq 0 \quad \Rightarrow \quad \Delta_{i+1} \cong \tfrac{1}{2}\varphi''(\dot{w})\Delta_i^2$$

$$\varphi'(\dot{w}) = \varphi''(\dot{w}) = 0 \quad \text{and} \quad \varphi'''(\dot{w}) \neq 0 \quad \Rightarrow \quad \Delta_{i+1} \cong \tfrac{1}{6}\varphi'''(\dot{w})\Delta_i^3$$

Thus, the parameters of local convergence C and ρ, *i*.e. the parameters of the function $|\Delta_{i+1}| \cong C \cdot |\Delta_i|^{\rho}$ are:

$$C = \frac{1}{\rho!}\left|\varphi^{(\rho)}(\dot{w})\right| \text{ and } \rho = 1, 2, \ldots$$

A scalar one-point IA is convergent if:

$$C < 1 \text{ for } \rho = 1 \text{ and } C < \infty \text{ for } \rho = 2, 3, \ldots$$

Example: Analysis of the local convergence of IA for computing $\sqrt[3]{d}$, *i*.e. for solving the equation $w^3 = d$:

$$w_{i+1} = w_i - \tfrac{2}{15}\left(w_i^3 - d\right) \quad \text{for} \quad d \in [1, 8]$$

Since $\varphi(w) \equiv w - \tfrac{2}{15}\left(w^3 - d\right)$, the exponent of convergence is $\rho = 1$ because:

$$\varphi'(w) = 1 - \tfrac{2}{15}\left(3w^2 - 0\right) \xrightarrow[w \to \sqrt[3]{d}]{} 1 - \tfrac{2}{5} d^{\frac{2}{3}} \neq 0$$

Consequently:

$$C = \left|1 - \tfrac{2}{5} d^{\frac{2}{3}}\right| \leq \tfrac{3}{5} \text{ for } d \in [1, 8]$$

So, AI is convergent because $C < 1$.

Example: Analysis of the local convergence of the Heron algorithm for computing $\sqrt{d}$:

$$w_0 = \max\{d, 1\} \text{ and } w_{i+1} = \tfrac{1}{2}\left(w_i + \frac{d}{w_i} \right) \text{ for } i = 0,1,...$$

Since $\varphi(w) \equiv \tfrac{1}{2}\left(w + \frac{d}{w} \right)$, the exponent of convergence is $\rho = 2$ because:

$$\varphi'(w) = \tfrac{1}{2}\left(1 - \frac{d}{w^2} \right) \xrightarrow[w \to \sqrt{d}]{} 0 \text{ and } \varphi''(w) = \frac{d}{w^3} \xrightarrow[w \to \sqrt{d}]{} \frac{1}{\sqrt{d}} \neq 0$$

Thus:

$$C = \frac{1}{2\sqrt{d}}$$

Evaluation of attainable accuracy

The evaluation of attainable accuracy is based on the following principle:

$$\tilde{w}_i = w_i\left(1+\vartheta_i\right) \xrightarrow[i\to\infty]{} \dot{w}\left(1+\vartheta_i\right)$$

where ϑ_i is the error resulting from propagation and accumulation of rounding errors introduced during the iterations number $1, ..., i$. If $\left|\vartheta_{i+1}\right| << 1$, then:

$$\left|\vartheta_{i+1}\right| \le C \cdot \left|\dot{w}\right|^{\rho-1} \cdot \left|\vartheta_i\right|^{\rho} + \left|\Delta\vartheta_i\right|$$

where:

$C \cdot \left|\dot{w}\right|^{\rho-1} \cdot \left|\vartheta_i\right|^{\rho}$ is the error component "inherited" from the previous iteration

$\left|\Delta\vartheta_i\right|$ is the error component introduced during the current iteration

Example: Evaluation of attainable accuracy of the IA for computing $\sqrt[3]{d}$:

$$w_{i+1} = w_i - \tfrac{2}{15}\left(w_i^3 - d\right) \quad \text{dla } d \in [1,8]$$

$$\tilde{w}_{i+1} = \left\{\tilde{w}_i - \tfrac{2}{15}\left[\tilde{w}_i^3\left(1+\eta_{pi}\right) - d\right]\left(1+\eta_{mi}\right)\left(1+\eta'_{oi}\right)\right\}\left(1+\eta''_{oi}\right)$$

For $i \to \infty$:

$$\dot{w}\left(1+\vartheta_{i+1}\right) = \left\{\dot{w}\left(1+\vartheta_i\right) - \tfrac{2}{15}\left[\dot{w}^3\left(1+\vartheta_i\right)^3\left(1+\eta_{pi}\right) - \dot{w}^3\right]\left(1+\eta_{mi}+\eta'_{oi}\right)\right\}\left(1+\eta''_{oi}\right)$$

$$\dot{w}\left(1+\vartheta_{i+1}\right) = \left\{\dot{w}\left(1+\vartheta_i\right) - \tfrac{2}{15}\left[\dot{w}^3\left(1+3\vartheta_i+\eta_{pi}\right) - \dot{w}^3\right]\left(1+\eta_{mi}+\eta'_{oi}\right)\right\}\left(1+\eta''_{oi}\right)$$

$$\dot{w}\left(1+\vartheta_{i+1}\right) = \dot{w}\left\{1+\vartheta_i - \tfrac{2}{15}\dot{w}^2\left(3\vartheta_i+\eta_{pi}\right)\left(1+\eta_{mi}+\eta'_{oi}\right)\right\}\left(1+\eta''_{oi}\right)$$

$$\dot{w}\left(1+\vartheta_{i+1}\right) = \dot{w}\left\{1+\left(1-\tfrac{2}{5}\dot{w}^2\right)\vartheta_i - \tfrac{2}{15}\dot{w}^2\eta_{pi} + \eta''_{oi}\right\}$$

$$\vartheta_{i+1} = \left(1-\tfrac{2}{5}\dot{w}^2\right)\vartheta_i - \tfrac{2}{15}\dot{w}^2\eta_{pi} + \eta''_{oi}$$

$$\left|\vartheta_{i+1}\right| \le C\left|\vartheta_i\right| - \tfrac{2}{15}\dot{w}^2\left|\eta_{pi}\right| + \left|\eta''_{oi}\right| \;\Rightarrow\; \left|\vartheta_{i+1}\right| \le \tfrac{3}{5}\left|\vartheta_i\right| + \tfrac{23}{15}eps \le \;\ldots\; \le \left(\tfrac{3}{5}\right)^{i+1}\left|\vartheta_0\right| + \left[\sum_{\nu=0}^{i}\left(\tfrac{3}{5}\right)^{\nu}\right]\tfrac{23}{15}eps$$

$$\left|\vartheta_{i+1}\right| \le \frac{\frac{23}{15}eps}{1-\frac{3}{5}} = \tfrac{23}{6}eps \cong 4eps \quad \Rightarrow \quad K = 4$$

- For all AI with $\rho = 1$:

$$K \cdot eps \cong \frac{\sup\left\{ \left|\Delta \vartheta_i(\mathbf{d})\right| \,\middle|\, \mathbf{d} \in \mathbb{D} \right\}}{1 - \sup\left\{ C(\mathbf{d}) \,\middle|\, \mathbf{d} \in \mathbb{D} \right\}}$$

- For all AI with $\rho = 2, 3, \ldots$:

$$K \cdot eps \cong \sup\left\{ \left|\Delta \vartheta_i\right| \,\middle|\, \mathbf{d} \in \mathbb{D} \right\}$$

(in this case, the error component "inherited" from the previous iteration $C \cdot \left|\dot{w}\right|^{\rho-1} \cdot \left|\vartheta_i\right|^{\rho}$ is negligible in comparison to the error component introduced during the current iteration $\left|\Delta \vartheta_i\right|$)

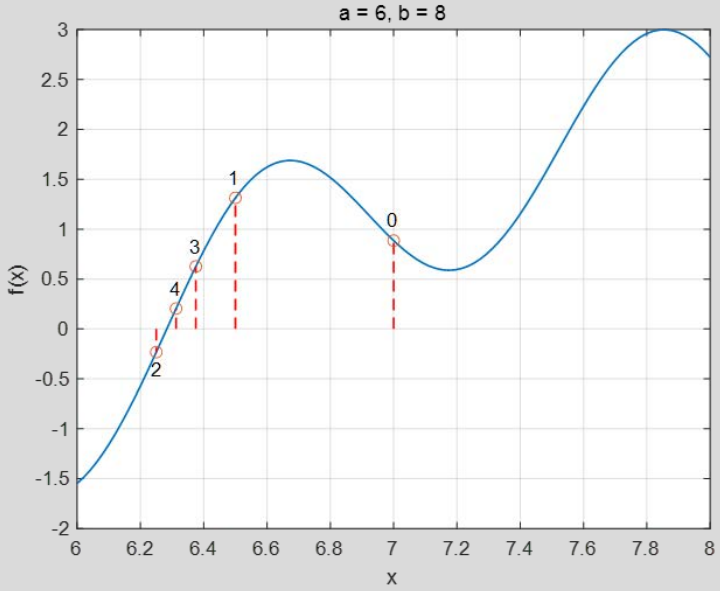
4.4. Methods for solving scalar equations $f(x)=0$

Bisection method

For $f(x)$ continuous in $[a_0,b_0]$ and satisfying the condition $f(a_0)\cdot f(b_0)<0$, the bisection method is defined by the formulae: $x_0=\frac{1}{2}(a_0+b_0)$ and:

$$x_{i+1}=\tfrac{1}{2}(a_{i+1}+b_{i+1}) \text{ with } \begin{cases} a_{i+1}=a_i,\ b_{i+1}=x_i & \text{gdy } f(a_i)\cdot f(x_i)<0 \\ a_{i+1}=x_i,\ b_{i+1}=b_i & \text{gdy } f(a_i)\cdot f(x_i)>0 \end{cases} \text{ for } i=0,1,2,\ldots$$

```
% Function defining equation
  f=@(x) 2*sin(x)+sin(5*x);a=6;b=8;
  xx=linspace(6,8,100);yy=f(xx);
  plot(xx,yy,'LineWidth',1);hold on;grid on
  xlabel('x');ylabel('f(x)');set(gca,'FontSize',12);
  title(['a = ',num2str(a),', b = ',num2str(b)],...
                                'FontWeight','normal');
  set(gca,'FontSize',10);
% Implementation of bisection method
  for i=1:5
    z(i)=(a+b)/2;
    if sign(f(z(i)))==sign(f(a)),a=z(i);else b=z(i);end
    line([z(i),z(i)],[0,f(z(i))],'LineStyle','--',...
                              'Color','red','LineWidth',1);
    text(z(i)-0.02,f(z(i))+0.2,num2str(i-1))
  end
  plot(z,f(z),'o');
```



The local convergence of bisection method is characterised by the parameters: $\rho = 1$ oraz $C = \frac{1}{2}$ because:

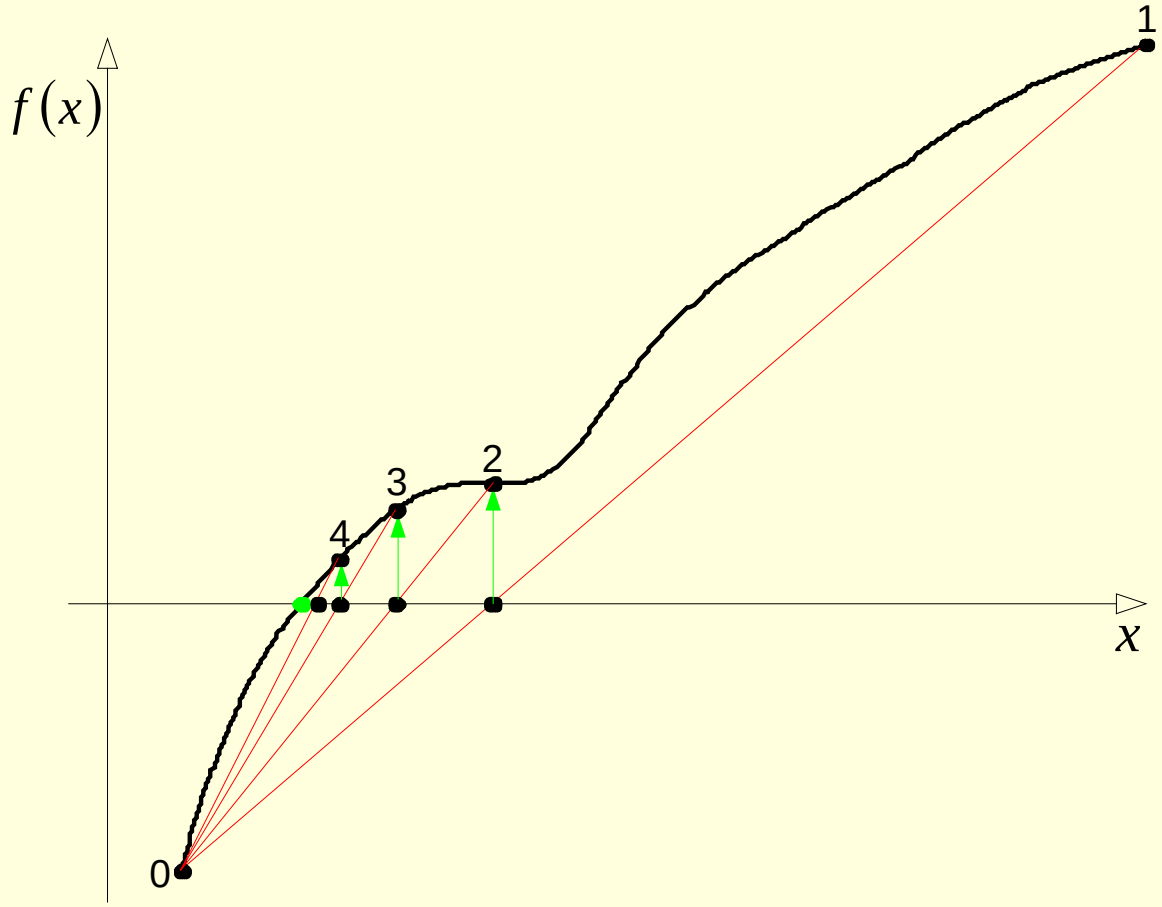
$$x_{i+1} = \varphi(x_i) \equiv \left\{ \begin{array}{ll} \frac{1}{2}(a_i + x_i) & \text{if } f(a_i) \cdot f(x_i) < 0 \\ \frac{1}{2}(x_i + b_i) & \text{if } f(a_i) \cdot f(x_i) > 0 \end{array} \right\} \text{ and } \varphi'(\dot{x}) = \frac{1}{2}$$

The number of iterations necessary for reducing the error below Δ is:

$$I = \min \left\{ i > \log_2 \frac{|b-a|}{\Delta} \right\}$$

Regula falsi method

$$x_{i+1} = x_i - \frac{x_i - x_0}{f(x_i) - f(x_0)} f(x_i) \text{ for } i = 1, 2, \ldots \Rightarrow \begin{cases} \rho = 1 \\ C = \dfrac{f'(\dot{x})}{f(x_0)}(\dot{x} - x_0) + 1 \end{cases}$$



The parameters ρ and C for the *regula falsi* method may be estimated as follows:

$$\varphi(x) \equiv x - \frac{x - x_0}{f(x) - f(x_0)} f(x) = \frac{x_0 f(x) - x f(x_0)}{f(x) - f(x_0)}$$

$$\varphi'(x) = \frac{\left[x_0 f(x) - x f(x_0)\right]' \cdot \left[f(x) - f(x_0)\right] - \left[x_0 f(x) - x f(x_0)\right] \cdot \left[f(x) - f(x_0)\right]'}{\left[f(x) - f(x_0)\right]^2}$$

$$= \frac{\left[x_0 f'(x) - f(x_0)\right] \cdot \left[f(x) - f(x_0)\right] - \left[x_0 f(x) - x f(x_0)\right] \cdot f'(x)}{\left[f(x) - f(x_0)\right]^2}$$

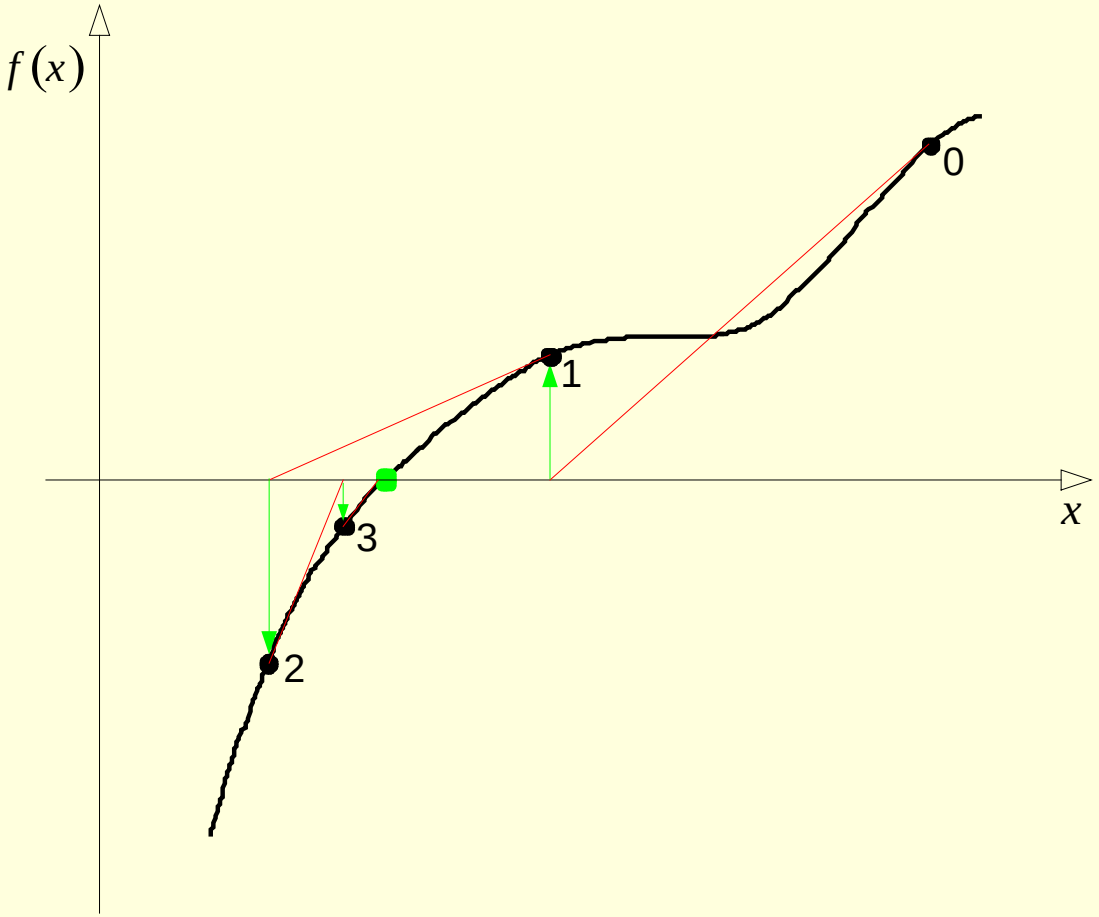
Since $f(\dot{x}) = 0$:

$$\varphi'(\dot{x}) = \frac{\left[x_0 f'(\dot{x}) - f(x_0)\right] \cdot \left[-f(x_0)\right] - \left[-x f(x_0)\right] \cdot f'(\dot{x})}{\left[-f(x_0)\right]^2} = \frac{f'(\dot{x})}{f(x_0)}(\dot{x} - x_0) + 1$$

Thus: $\rho = 1$ and $C = \frac{f'(\dot{x})}{f(x_0)}(\dot{x} - x_0) + 1$

Newton's (tangent) method

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad \text{for} \quad i = 0, 1, \ldots \quad \Rightarrow \begin{cases} \rho = 2 \\ C = -\frac{1}{2}\frac{f''(\dot{x})}{f'(\dot{x})} \end{cases}$$



The parameters ρ and C for the Newton's method may be estimated as follows:

$$\varphi(x) \equiv x - \frac{f(x)}{f'(x)} \Rightarrow \varphi'(x) = 1 - \frac{[f'(x)]^2 - f(x)f''(x)}{[f'(x)]^2} = \frac{f(x)f''(x)}{[f'(x)]^2} \xrightarrow[x \to \dot{x}]{} 0$$

$$\varphi''(x) = \frac{[f'(x)f''(x) + f(x)f'''(x)] \cdot [f'(x)]^2 - f(x)f''(x) \cdot 2f'(x)f''(x)}{[f'(x)]^4}$$

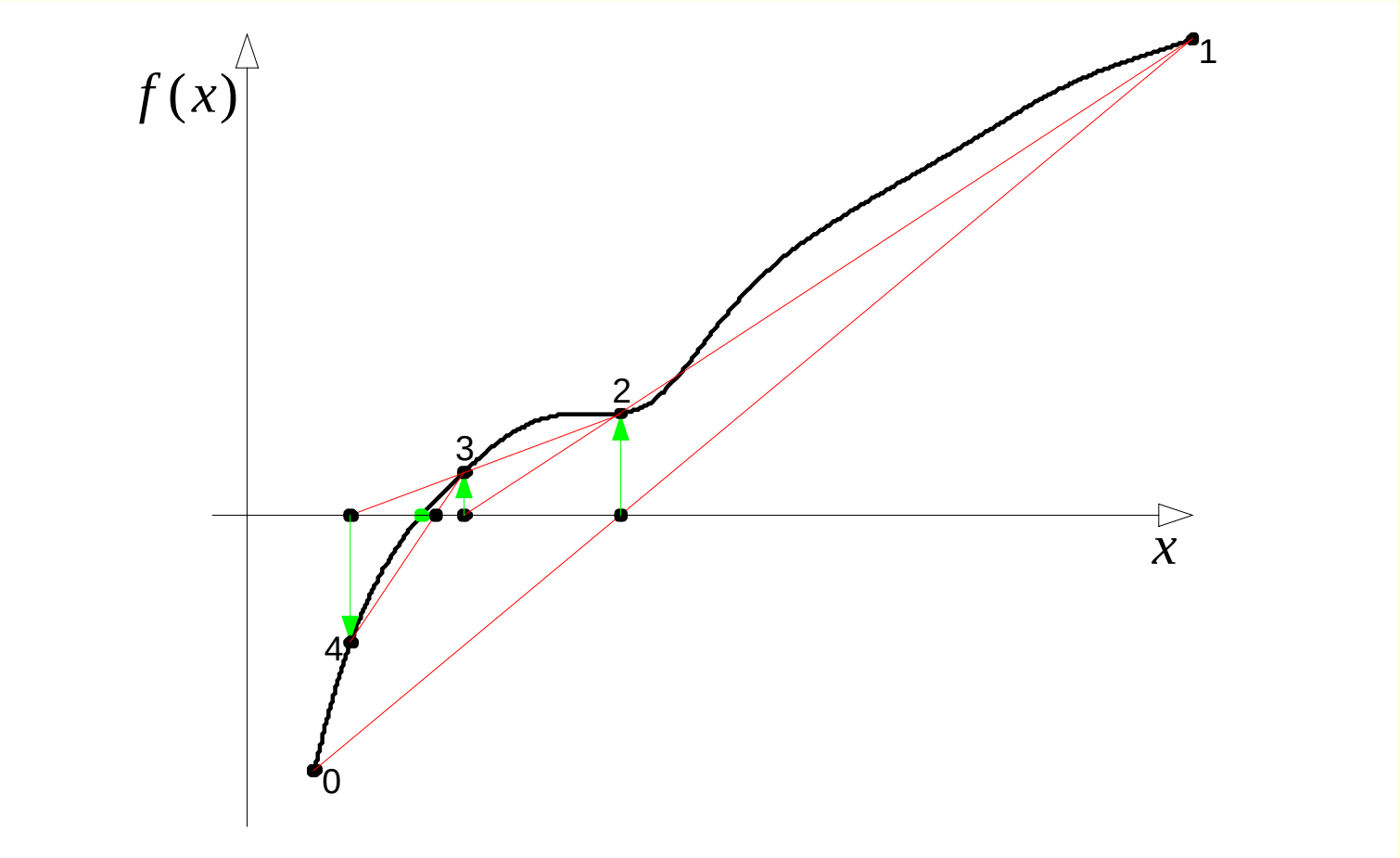
Since $f(\dot{x}) = 0$:

$$\varphi''(\dot{x}) = \frac{[f'(\dot{x})f''(\dot{x})] \cdot [f'(\dot{x})]^2}{[f'(\dot{x})]^4} = \frac{f''(\dot{x})}{f'(\dot{x})}$$

Thus: $\rho = 2$ and $C = \frac{1}{2}\varphi''(\dot{x}) = \frac{1}{2}\frac{f''(\dot{x})}{f'(\dot{x})}$

Secant method

$$x_{i+1} = x_i - \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i) \text{ for } i = 1, 2, \ldots \Rightarrow \begin{cases} \rho = \frac{1}{2}\left(\sqrt{5}+1\right) \cong 1.618 \\ C = \left[\frac{1}{2}\frac{f''(\dot{x})}{f'(\dot{x})}\right]^{\frac{1}{2}\left(\sqrt{5}-1\right)} \end{cases}$$



The parameters ρ and C for the secant method may be estimated as follows:

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})} \quad \text{for} \quad i = 1, 2, \ldots$$

$$\Delta_{i+1} = \Delta_i - \frac{\left[f(\dot{x}) + f'(\dot{x})\Delta_i + \frac{1}{2} f''(\dot{x})\Delta_i^2 + \ldots \right](\Delta_i - \Delta_{i-1})}{f(\dot{x}) + f'(\dot{x})\Delta_i + \frac{1}{2} f''(\dot{x})\Delta_i^2 + \ldots - f(\dot{x}) - f'(\dot{x})\Delta_{i-1} - \frac{1}{2} f''(\dot{x})\Delta_{i-1}^2 + \ldots}$$

Since $f(\dot{x}) = 0$:

$$\Delta_{i+1} \cong \Delta_i - \frac{\left[f'(\dot{x})\Delta_i + \frac{1}{2} f''(\dot{x})\Delta_i^2 \right](\Delta_i - \Delta_{i-1})}{f'(\dot{x})(\Delta_i - \Delta_{i-1}) + \frac{1}{2} f''(\dot{x})(\Delta_i^2 - \Delta_{i-1}^2)}$$

After dividing by $\Delta_i - \Delta_{i-1}$:

$$\Delta_{i+1} \cong \Delta_i - \frac{\left[f'(\dot{x})\Delta_i + \frac{1}{2} f''(\dot{x})\Delta_i^2 \right]}{f'(\dot{x}) + \frac{1}{2} f''(\dot{x})(\Delta_i + \Delta_{i-1})}$$

$$= \frac{f'(\dot{x})\Delta_i + \frac{1}{2} f''(\dot{x})\left(\Delta_i^2 + \Delta_{i-1}\Delta_i\right) - f'(\dot{x})\Delta_i - \frac{1}{2} f''(\dot{x})\Delta_i^2}{f'(\dot{x}) + \frac{1}{2} f''(\dot{x})(\Delta_i + \Delta_{i-1})} = \hat{C}\Delta_i\Delta_{i-1}$$

where $\hat{C} \equiv \frac{1}{2}\frac{f''(\dot{x})}{f'(\dot{x})}$. Since $|\Delta_{i+1}| \cong C|\Delta_i|^\rho \Rightarrow |\Delta_i| \cong C|\Delta_{i-1}|^\rho \Rightarrow |\Delta_{i-1}| \cong \left(\frac{1}{C}|\Delta_i|\right)^{\frac{1}{\rho}}$:

$$|\Delta_{i+1}| \cong |\hat{C}||\Delta_i||\Delta_{i-1}| \cong |\hat{C}||\Delta_i|\left(\frac{1}{C}|\Delta_i|\right)^{\frac{1}{\rho}} = |\hat{C}|C^{-\frac{1}{\rho}}|\Delta_i|^{1+\frac{1}{\rho}}$$

The following condition, resulting from the equality $|\Delta_{i+1}| \cong C|\Delta_i|^\rho$, must be satisfied:

$$\frac{|\Delta_{i+1}|}{C|\Delta_i|^\rho} \cong |\hat{C}|C^{-\frac{1}{\rho}-1}|\Delta_i|^{1+\frac{1}{\rho}-\rho} = 1 \text{ for } i \to \infty$$

This is possible if and only if $1 + \frac{1}{\rho} - \rho = 0$ and $|\hat{C}|C^{-\frac{1}{\rho}-1} = 1$. Hence:

$$\rho = \frac{1}{2}\left(1+\sqrt{5}\right) \text{ and } C = |\hat{C}|^{\frac{1}{2}\left(\sqrt{5}-1\right)}$$

Example: A comparison of the convergence speed of three methods when applied for solving the equation $x^2 - x - 2 = 0$.

METHOD	NUMBER OF ITERATIONS					
	1	2	3	4	5	6
BISECTION ($\rho = 1$) $x_0 = 3,\ x_1 = 1.5$	$1.3 \cdot 10^{-1}$	$6.3 \cdot 10^{-2}$	$3.1 \cdot 10^{-2}$	$1.6 \cdot 10^{-2}$	$7.8 \cdot 10^{-3}$	$3.9 \cdot 10^{-3}$
SECANT ($\rho = 1.618$) $x_0 = 3,\ x_1 = 1.5$	$7.1 \cdot 10^{-2}$	$1.5 \cdot 10^{-2}$	$7.5 \cdot 10^{-4}$	$7.5 \cdot 10^{-6}$	$3.8 \cdot 10^{-9}$	$1.9 \cdot 10^{-14}$
NEWTON ($\rho = 2$) $x_0 = 3$	$1.0 \cdot 10^{-1}$	$5.9 \cdot 10^{-3}$	$2.3 \cdot 10^{-5}$	$3.5 \cdot 10^{-10}$	*K eps*	*K eps*

Muller's method

The method is based on approximation of $f(x)$ around x_i by means of a quadratic function:

$$\hat{f}(x-x_i) = a_i(x-x_i)^2 + b_i(x-x_i) + c_i$$

- version I – the approximation by means of the Taylor series referring to $f(x_i)$, $f'(x_i)$ and $f''(x_i)$ (a generalised tangent method using second derivative):

$$a_i = \frac{1}{2} f''(x_i),\ b_i = f'(x_i),\ c_i = f(x_i)$$

- version II – the interpolation of $f(x_{i-2})$, $f(x_{i-1})$ and $f(x_i)$ (a generalised secant method derived from three-point interpolation):

$$\begin{bmatrix} a_i \\ b_i \end{bmatrix} = \begin{bmatrix} -(x_i - x_{i-1})^2 & (x_i - x_{i-1}) \\ -(x_i - x_{i-2})^2 & (x_i - x_{i-2}) \end{bmatrix}^{-1} \cdot \begin{bmatrix} f(x_i) - f(x_{i-1}) \\ f(x_i) - f(x_{i-2}) \end{bmatrix} \text{ and } c_i = f(x_i)$$

The iterative formula, implemented in complex arithmetic, is identical for both versions:

$$x_{i+1} = x_i - \frac{2c_i}{b_i + \operatorname{sgn}(b_i)\sqrt{b_i^2 - 4a_i c_i}}$$

The convergence exponent is: $\rho = 3$ for version I, and $\rho = 1.84$ for version II.

4.5. Newton-Raphson method for solving system of equations $\mathbf{f}(\mathbf{x}) = \mathbf{0}$

The method is based on the assumption that:

$$\mathbf{f}(\mathbf{x}) \equiv \left[f_1(\mathbf{x}) \, f_2(\mathbf{x}) \ldots f_N(\mathbf{x}) \right]^T$$

where $\mathbf{x} \equiv \left[x_1 \; x_2 \; \ldots \; x_N \right]^T$. The method is defined by the following formula:

$$\left.\begin{array}{c} \mathbf{x}_{i+1} = \mathbf{x}_i - \left[\mathbf{f}'(\mathbf{x}_i) \right]^{-1} \mathbf{f}(\mathbf{x}_i) \\ \Uparrow \\ x_{i+1} = x_i - \left[f'(x_i) \right]^{-1} f(x_i) \end{array}\right\} \text{ for } i = 0, 1, \ldots; \text{ where } \mathbf{f}'(\mathbf{x}) \equiv \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_N}{\partial x_1} & \cdots & \dfrac{\partial f_N}{\partial x_N} \end{bmatrix}$$

The corresponding algorithm has the form:

❶ compute $\mathbf{f}(\mathbf{x}_i)$ and $\mathbf{f}'(\mathbf{x}_i)$;

❷ solve $\left[\mathbf{f}'(\mathbf{x}_i) \right] \Delta\mathbf{x}_i = -\mathbf{f}(\mathbf{x}_i)$ with respect to $\Delta\mathbf{x}_i = \mathbf{x}_{i+1} - \mathbf{x}_i$;

❸ compute $\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta\mathbf{x}_i$;

❹ return to ❶ if $\left\| \Delta\mathbf{x}_i \right\| > \Delta x$ or $\left\| \Delta\mathbf{x}_i \right\| / \left\| \mathbf{x}_i \right\| > \delta x$.

4.6. Methods for finding roots (zeros) of polynomials

The roots $\dot{x}_n$ of a polynomial:

$$f_N(x) \equiv a_N x^N + a_{N-1} x^{N-1} + \ldots + a_1 x + a_0 = a_N \prod_{n=1}^{N} (x - \dot{x}_n)$$

may be: real and/or complex; single and/or multiple.

Finding real roots and linear deflation

Any method for solving scalar nonlinear equations, described in § 3.3, may be used for finding real zeros of a polynomial. A zero already found may be eliminated using the so-called *linear deflation*.

For any single real root $\dot{x}$ of $f_N(x)$, the following relationship holds:

$$f_N(x) = f_{N-1}(x) \cdot (x - \dot{x}), \text{ where } f_{N-1}(x) = b_{N-1} x^{N-1} + b_{N-2} x^{N-2} + \ldots + b_1 x + b_0$$

The coefficients $b_0, ..., b_{N-1}$ may be determined using an algorithm for dividing the polynomial $f_N(x)$ by $(x - \dot{x})$, called the Horner's algorithm:

$$b_{N-1} = a_N, \text{ and } b_n = a_{n+1} + \dot{x} \cdot b_{n+1} \text{ for } n = N-2, ..., 0$$

Finding complex roots and quadratic deflation

A pair of complex roots $\dot{x}$ and $\dot{x}^*$ may be found using the Mueller's method. They may be eliminated using the so-called *quadratic deflation*. The coefficients of the quadratic polynomial:

$$m(x) \equiv (x - \dot{x})(x - \dot{x}^*) = x^2 - px - r$$

whose values are $p = 2\,\mathrm{Re}(\dot{x})$ i $r = -|\dot{x}|^2$, are used for finding the polynomial :

$$f_{N-2}(x) = b_{N-2}x^{N-2} + \ldots + b_1 x + b_0$$

such that:

$$f_N(x) = (x^2 - px - r)\, f_{N-2}(x)$$

The solution to this problem has the form:

$$b_{N-2} = a_N$$

$$b_{N-3} = a_{N-1} + pb_{N-2}$$

$$b_n = a_{n+2} + pb_{n+1} + rb_{n+2} \text{ for } n = N-4, \ldots, 0$$

Accuracy of roots estimation

Example: The errors ε_n of the estimates $\tilde{x}_n = \dot{x}_n\left(1+\varepsilon_n\right)$ of roots x_n $(n=1,2,3)$ of the polynomial:

$$f\left(x\right) = x^3 - 30x^2 + 299x - 990$$

may be evaluated in the following way:

$$f\left(\tilde{x}_n\right) = \tilde{x}_n^3 - 30\tilde{x}_n^2 + 299\tilde{x}_n - 990 = \rho_n$$

$$\dot{x}_n^3\left(1+\varepsilon_n\right)^3 - 30\dot{x}_n^2\left(1+\varepsilon_n\right)^2 + 299\dot{x}_n\left(1+\varepsilon_n\right) - 990 = \rho_n$$

$$\dot{x}_n^3\left(1+3\varepsilon_n\right) - 30\dot{x}_n^2\left(1+2\varepsilon_n\right) + 299\dot{x}_n\left(1+\varepsilon_n\right) - 990 \cong \rho_n$$

$$\left[\dot{x}_n^3 - 30\dot{x}_n^2 + 299\dot{x}_n - 990\right] + \left(3\dot{x}_n^3 - 60\dot{x}_n^2 + 299\dot{x}_n\right)\varepsilon_n \cong \rho_n$$

Since $\dot{x}_n{}^3 - 30\dot{x}_n^2 + 299\dot{x}_n - 990 = 0$, the errors may be determined as follows:

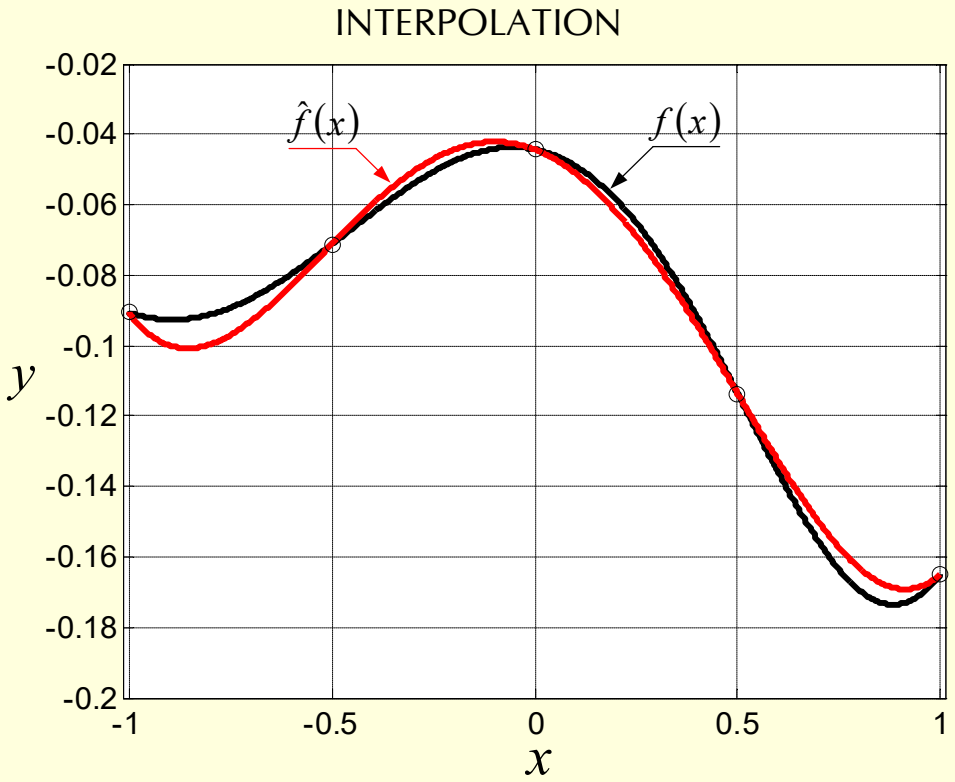
$$\varepsilon_n \cong \frac{\rho_n}{3\dot{x}_n^3 - 60\dot{x}_n^2 + 299\dot{x}_n} \cong \frac{\rho_n}{3\tilde{x}_n^3 - 60\tilde{x}_n^2 + 299\tilde{x}_n}$$

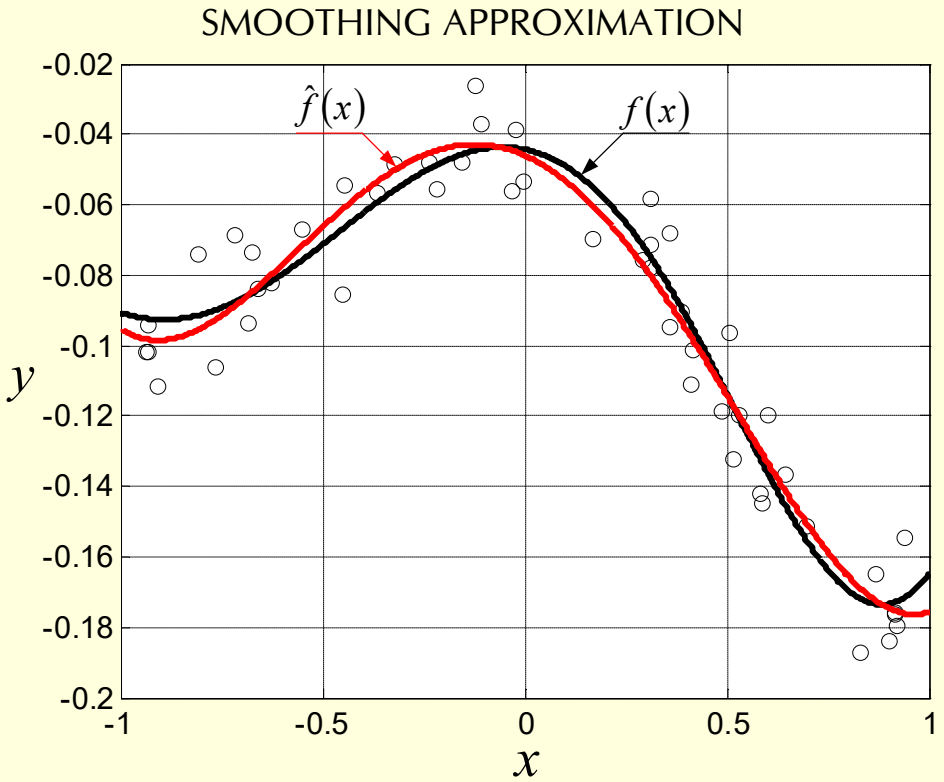
Roman Z. Morawski
phone: (22) 234-7721, e-mail: *roman.morawski@pw.edu.pl*

Numerical Methods (ENUME)

5. APPROXIMATION OF FUNCTIONS

Lecture notes for Spring Semester 2022/2023


INTERPOLATION
$\hat{f}(x)$
$f(x)$
-0.02
-0.04
-0.06
-0.08
-0.1
-0.12
-0.14
-0.16
-0.18
-0.2
y
-1
-0.5
0
0.5
1
x


SMOOTHING APPROXIMATION
$\hat{f}(x)$
$f(x)$
-0.02
-0.04
-0.06
-0.08
-0.1
-0.12
-0.14
-0.16
-0.18
-0.2
y
-1
-0.5
0
0.5
1
x

5.1. Formulation of the interpolation problem

Having $N+1$ pairs:

$$\left\langle x_n, f\left(x_n\right)\right\rangle \text{ for } n=0,...,N$$

determine a function $\hat{f}_N\left(x\right)$ interpolating a function $f\left(x\right)$, *i*.e. satisfying the following conditions:

$$\hat{f}_N\left(x_n\right)=f\left(x_n\right) \text{ for } n=0,...,N$$

The solution of the interpolation problem consists in estimation of the parameters $\mathbf{p}$ of a known function $\hat{f}_N\left(x\right)\equiv\hat{f}_N\left(x;\mathbf{p}\right)$, where $\hat{f}_N\left(x\right)$ is (most frequently):

- an algebraic polynomial $\hat{f}_N\left(x;\mathbf{p}\right)=\sum_{n=0}^{N}p_n x^n$ with $\mathbf{p}\equiv\left[p_0\ ...\ p_N\right]^T$
- a polynomial spline function of order 2, 3 or 5
- a trigonometric polynomial: $\hat{f}_N\left(x;\mathbf{p}\right)=\sum_{n=0}^{N}p_n\cos\left(nx\right)$ with $\mathbf{p}\equiv\left[p_0\ ...\ p_N\right]^T$
- a rational function: $\hat{f}_N\left(x;\mathbf{p}\right)=\frac{a_0+a_1x+...+a_{N-M-1}x^{N-M-1}}{1+b_1x+...+b_Mx^M}$ with $\mathbf{p}\equiv\left[a_0\ ...\ a_{N-M-1}\ b_1\ ...\ b_M\right]^T$

5.2. Interpolation of discrete values of an unknown function with an algebraic polynomial

There exists a unique polynomial $\hat{f}_N(x;\mathbf{p}) = \sum_{n=0}^{N} p_n x^n$ of order N, such that:

$$\hat{f}_N(x) = f(x_n) \text{ for } n = 0, ..., N$$

If $f(x)$ is $N+1$ times differentiable in the interval $[x_0, x_N]$, then the error of interpolation $\hat{f}_N(x;\mathbf{p}) - f(x)$ is subject to the following assessment:

$$\left|\hat{f}_N(x;\mathbf{p}) - f(x)\right| \le \frac{Y_{N+1}}{(N+1)!}\prod_{n=0}^{N}(x - x_n), \text{ where } Y_{N+1} = \sup\left\{\left|f^{(N+1)}(x)\right| \;\middle|\; x \in [x_0, x_N]\right\}$$

Conclusion: a "large" error of interpolation may happen in the vicinity of "large" changes of $f(x)$ or its derivatives.

The Lagrange polynomial: $\hat{f}_N(x) = \sum_{n=0}^{N} f(x_n)\cdot L_n(x)$ with $L_n(x) \equiv \prod_{\substack{\nu=0 \\ \nu\neq n}}^{N} \frac{x - x_\nu}{x_n - x_\nu}$

5.3. Interpolation of discrete values of an unknown function with a polynomial spline function

The function $s(x; x_0, ..., x_N)$, where $x_0 < x_1 < ... < x_N$, is a polynomial spline function of order M, if:

- it is a polynomial of order not higher than M in each subinterval $[x_n, x_{n+1}]$, *i*.e.:

$$s(x) = a_{n,M}(x - x_n)^M + ... + a_{n,1}(x - x_n) + a_{n,0} \text{ for } x \in [x_n, x_{n+1}],\ n = 0, 1, ..., N-1$$

- it is continuous with $M-1$ derivatives in $[x_0, x_N]$

The function $s(x) \equiv s(x; x_0, ..., x_N)$ is an interpolating spline function if:

$$s(x_n) = y_n \text{ for } n = 0, 1, ..., N$$

The coefficients of an interpolating spline function $a_{n,M}, ..., a_{n,1}, a_{n,0}$, whose number is $N(M+1)$, have to satisfy the following $MN - M + N + 1$ equations:

- the $N+1$ interpolation conditions
- the $M(N-1)$ continuity conditions for the derivatives of order $0, ..., M-1$ at $x_1, ..., x_{N-1}$

The remaining degrees of freedom, whose number is:

$$N(M+1)-(MN-M+N+1)=M-1$$

may be used for satisfying some additional boundary conditions:

- for a spline function of order 2 (alternatively):
  - a) $s'(x_0^+)=\alpha_1$
  - b) $s'(x_N^-)=\beta_1$
  - c) $s'(x_0^+)=s'(x_N^-)$
- for a spline function of order 3 (alternatively):
  - a) $s''(x_0^+)=\alpha_2, s''(x_N^-)=\beta_2$
  - b) $s'(x_0^+)=\alpha_1, s'(x_N^-)=\beta_1$
  - c) $s'(x_0^+)=s'(x_N^-), s''(x_0^+)=s''(x_N^-)$

Example: Design a spline function of order 2:

$$s(x) = a_n (x - x_n)^2 + b_n (x - x_n) + c_n \quad \text{for } x \in [x_n, x_{n+1}],\ n = 0, 1, 2$$

interpolating the following points:

n	0	1	2	3
x_n	0	1	2	3
y_n	1	2	4	3

and satisfying the following boundary condition: $s'(x_0^+) = 0$.

The interpolation conditions have the form:

- $s(x_n) \equiv a_n (x_n - x_n)^2 + b_n (x_n - x_n) + c_n = y_n \quad \Rightarrow \; c_n = y_n$ for $n = 0, 1, 2$
- $s(x_3) \equiv a_2 (x_3 - x_2)^2 + b_2 (x_3 - x_2) + c_2 = y_3 \quad \Rightarrow \; a_2 + b_2 + c_2 = y_3$

The continuity conditions for the function have the form:

- $s(x_1^-) \equiv a_0 (x_1 - x_0)^2 + b_0 (x_1 - x_0) + c_0 = a_1 (x_1 - x_1)^2 + b_1 (x_1 - x_1) + c_1 \equiv s(x_1^+)$

  $\Rightarrow a_0 + b_0 + c_0 = c_1$
- $s(x_2^-) \equiv a_1 (x_2 - x_1)^2 + b_1 (x_2 - x_1) + c_1 = a_2 (x_2 - x_2)^2 + b_2 (x_2 - x_2) + c_2 \equiv s(x_2^+)$

  $\Rightarrow \; a_1 + b_1 + c_1 = c_2$

The continuity conditions for the first derivative of the function have the form:

- $s'(x_1^-) \equiv 2a_0 (x_1 - x_0) + b_0 = 2a_1 (x_1 - x_1) + b_1 \equiv s'(x_1^+) \quad \Rightarrow \; 2a_0 + b_0 = b_1$
- $s'(x_2^-) \equiv 2a_1 (x_2 - x_1) + b_1 = 2a_2 (x_2 - x_2) + b_2 \equiv s'(x_2^+) \quad \Rightarrow \; 2a_1 + b_1 = b_2$

The boundary condition has the form:

- $s'(x_0^+) \equiv 2a_0 (x_0 - x_0) + b_0 = 0 \quad \Rightarrow \; b_0 = 0$

The values of c_n result from the interpolation conditions: $c_0 = 1$, $c_1 = 2$ and $c_2 = 4$.

The remaining conditions assume the form of algebraic equations with respect to a_n and b_n:

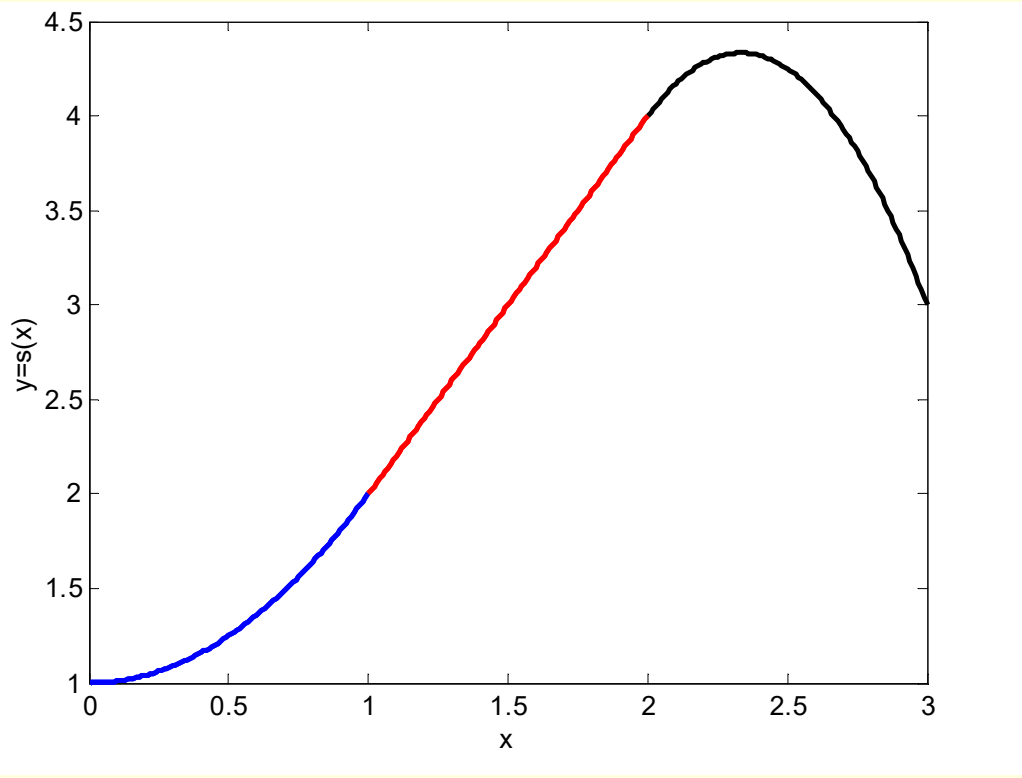
$$a_2 + b_2 + 4 = 3 \quad \Rightarrow \quad a_2 + b_2 = -1 \qquad 2a_0 + b_0 = b_1$$
$$a_0 + b_0 + 1 = 2 \quad \Rightarrow \quad a_0 + b_0 = 1 \qquad 2a_1 + b_1 = b_2$$
$$a_1 + b_1 + 2 = 4 \quad \Rightarrow \quad a_1 + b_1 = 2 \qquad b_0 = 0$$

The solution of those equations has the form:

$$a_0 = 1,\ a_1 = 0 \text{ and } a_2 = -3$$
$$b_0 = 0,\ b_1 = 2 \text{ and } b_2 = 2$$

Hence:

$$s(x) = \begin{cases} x^2 + 1 & \text{for} \quad x \in [0,1] \\ 2(x-1) + 2 & \text{for} \quad x \in [1,2] \\ -3(x-2)^2 + 2(x-2) + 4 & \text{for} \quad x \in [2,3] \end{cases}$$



5.4. Least-squares approximation of a function on the basis of its discrete values

A vector of parameters $\mathbf{p} = \left[p_1 \ldots p_K \right]^T$ is sought for, the vector that makes the linear combination of linearly independent functions $\left\{ \varphi_k(x) \middle| k = 1, 2, \ldots, K \right\}$:

$$\hat{f}(x;\mathbf{p}) = \sum_{k=1}^{K} p_k \varphi_k(x)$$

best approximate a sequence of discrete values of an unknown function $f(x)$:

$$\left\{ f(x_n) \middle| n = 1, 2, \ldots, N \right\}$$

with respect to the following (least-squares) criterion:

$$J_2(\mathbf{p}) = \sum_{n=1}^{N} \left[\hat{f}(x_n;\mathbf{p}) - f(x_n) \right]^2 \quad \text{or} \quad \overline{J}_2(\mathbf{p}) = \sqrt{\frac{1}{N} \sum_{n=1}^{N} \left[\hat{f}(x_n;\mathbf{p}) - f(x_n) \right]^2}$$

Examples of $\varphi_k(x)$:

$$\left\{ x^{k-1} \middle| k = 1, \ldots, K \right\}, \quad \left\{ \cos\left((k-1)x \right) \middle| k = 1, \ldots, K \right\}, \quad \left\{ \operatorname{sinc}\left(x - \widehat{x}_k \right) \middle| k = 1, \ldots, K \right\}$$

The necessary condition of the minimum has the form:

$$\frac{\partial J_2(\mathbf{p})}{\partial p_k} = 0 \quad \text{for} \quad k = 1, ..., K$$

$$\Updownarrow$$

$$\mathbf{\Phi}^T \cdot \mathbf{\Phi} \cdot \mathbf{p} = \mathbf{\Phi}^T \cdot \mathbf{y}$$

where:

$$\mathbf{\Phi} = \begin{bmatrix} \varphi_1(x_1) & \varphi_2(x_1) & \cdots & \varphi_K(x_1) \\ \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_K(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(x_N) & \varphi_2(x_N) & \cdots & \varphi_K(x_N) \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} f(x_1) & f(x_2) & \ldots & f(x_N) \end{bmatrix}^T$$

The solution may be presented in an analytical form:

$$\hat{\mathbf{p}} = \left(\mathbf{\Phi}^T \cdot \mathbf{\Phi}\right)^{-1} \cdot \mathbf{\Phi}^T \cdot \mathbf{y}$$

or numerical form:

$$\hat{\mathbf{p}} = \arg_{\mathbf{p}} \left\{ \mathbf{\Phi}^T \cdot \mathbf{\Phi} \cdot \mathbf{p} = \mathbf{\Phi}^T \cdot \mathbf{y} \right\}$$

The Cholesky-Banachiewicz method may be used because $\mathbf{\Phi}^T \cdot \mathbf{\Phi}$ is a positive definite matrix.

5.5. Uniform approximation of a function on the basis of its discrete values

A vector of parameters $\mathbf{p} = \left[p_1 \ldots p_K \right]^T$ is sought for, the vector that makes the linear combination of linearly independent functions $\left\{ \varphi_k(x) \middle| \, k = 1, 2, \ldots, K \right\}$:

$$\hat{f}(x;\mathbf{p}) = \sum_{k=1}^{K} p_k \varphi_k(x)$$

best approximate a sequence of discrete values of an unknown function $f(x)$:

$$\left\{ f(x_n) \middle| \, n = 1, 2, \ldots, N \right\}$$

with respect to the following criterion:

$$J_\infty(\mathbf{p}) = \sup\left\{ \left| \hat{f}(x_n;\mathbf{p}) - f(x_n) \right| \;\; n = 1, \ldots, N \right\}$$

Numerical implementation:

- the algorithms of non-differentiable optimisation
- the Remez algorithm
- the least-squares approximation based on the first-kind Chebyshev polynomials

5.6. Least-squares approximation of a known function by means of algebraic polynomials

A vector of parameters $\mathbf{p} = \left[p_1 \ldots p_K \right]^T$ is sought for, the vector that makes the linear combination of linearly independent functions $\left\{ \varphi_k(x) \middle| k = 1, 2, \ldots, K \right\}$:

$$\hat{f}(x;\mathbf{p}) = \sum_{k=1}^{K} p_k \varphi_k(x)$$

best approximate a given function $f(x)$ with respect to the following criterion:

$$J_2(\mathbf{p}) = \int_a^b \left[\hat{f}(x;\mathbf{p}) - f(x) \right]^2 dx \quad \text{or} \quad \bar{J}_2(\mathbf{p}) = \sqrt{\frac{1}{b-a} \int_a^b \left[\hat{f}(x;\mathbf{p}) - f(x) \right]^2 dx}$$

where the approximated function $f(x)$ is assumed to satisfy the condition:

$$\int_a^b f^2(x)\, dx < \infty$$

The necessary condition of the minimum has the form:

$$\frac{\partial J_2(\mathbf{p})}{\partial p_k} = 0 \ \text{ for } \ k = 1, 2, ..., K \quad \Leftrightarrow \quad \mathbf{A} \cdot \mathbf{p} = \mathbf{b}$$

where:

$$\left.\begin{aligned} a_{k,j} &= \int_a^b \varphi_k(x) \cdot \varphi_j(x)\, dx \equiv \langle \varphi_k \,|\, \varphi_j \rangle \quad \text{for } j = 1, 2, ..., K \\ b_k &= \int_a^b f(x) \cdot \varphi_k(x)\, dx \equiv \langle f \,|\, \varphi_k \rangle \end{aligned}\right\} \text{ for } \quad k = 1, 2, ..., K$$

A particularly simple solution may be obtained if $\{\varphi_k\}$ is a subset of an orthogonal basis, *i*.e.:

$$\langle \varphi_k \,|\, \varphi_j \rangle \equiv \int_a^b \varphi_k(x) \cdot \varphi_j(x)\, dx = 0 \ \text{ for } \ k \neq j$$

Then $a_{k,j} = 0$ for $k, j = 1, 2, ..., K;\ j \neq k$, *i*.e. the matrix $\mathbf{A} \equiv diag\{a_{k,k}\}$.

If, moreover, $\|\varphi_k\|_2 = \sqrt{\langle \varphi_k, \varphi_k \rangle} = 1$ for $k = 1, ..., K$, then:

$$\mathbf{A} \equiv \mathbf{I} \ \Rightarrow \ \mathbf{p} = \mathbf{b} \ \Leftrightarrow \ p_k = \langle f \,|\, \varphi_k \rangle \ \text{ for } \ k = 0, 1, ..., K$$

A sequence of linearly independent functions $\{\varphi_1(x), \varphi_2(x), ...\}$ may be transformed into a sequence of orthogonal functions $\{\psi_1(x), \psi_2(x), ...\}$ by means of the following procedure called *Gram-Schmidt orthogonalisation*.

The inner product of functions in the above procedure is defined as follows:

$$\langle \varphi_k \,|\, \psi_j \rangle_w \equiv \int_a^b w(x) \cdot \varphi_k(x) \cdot \psi_j(x)\, dx$$

where $w(x)$ is a weighing function such that $w(x) \geq 0$ and $\int_a^b w(x)\, dx < \infty$.

The orthogonalisation of $\{\varphi_k(x) = x^k \,|\, k = 0, 1, ...\}$ for various intervals $[a,b]$ weighing functions $w(x)$ generates various families of *orthogonal polynomials*.

Example: Legendre polynomials – orthogonal in $[-1,1]$ with $w(x)=1$:

$$P_0(x)=1,\ P_1(x)=x$$

$$P_k(x)=\frac{2k-1}{k}xP_{k-1}(x)-\frac{k-1}{k}P_{k-2}(x)\ \text{ for } k=2,3,\ldots$$

$$\|P_k\|_2^2=\int_{-1}^{1}P_k^2(x)\,dx=\frac{2}{2k+1}\quad \text{for } k=0,1,\ldots$$

Example: Chebyshev polynomials of the I kind – orthogonal in $[-1,1]$ with $w(x)=\frac{1}{\sqrt{1-x^2}}$:

$$T_0(x)=1,\ T_1(x)=x$$

$$T_k(x)=2xT_{k-1}(x)-T_{k-2}(x)\ \text{ for } k=2,3,\ldots$$

$$\int_{-1}^{1}w(x)T_k(x)T_{\ j}(x)dx=\begin{cases}\pi & \text{for}\quad k=j=0\\ \pi/2 & \text{for}\quad k=j\neq 0\\ 0 & \text{for}\quad k\neq j\end{cases}$$

Example: Hermite polynomials – orthogonal in $(-\infty,+\infty)$ with $w(x)=\exp\left(-x^2\right)$:

$$H_0(x)=1,\ H_1(x)=2x$$

$$H_k(x)=2xH_{k-1}(x)-(2k-2)H_{k-2}(x) \quad \text{for } k=2,3,\ldots$$

$$\|H_k\|_2^2=\int_{-\infty}^{+\infty}\exp\left(-x^2\right)\cdot H_k^2(x)\,dx=2^k\,k!\sqrt{\pi} \quad \text{for } k=2,3,\ldots$$

Notice: If the functions $\{\psi_1(x),\psi_2(x),\ldots\}$ are orthogonal with respect to the inner product:

$$\langle\psi_k\,|\,\psi_j\rangle_w\equiv\int_a^b w(x)\cdot\psi_k(x)\cdot\psi_j(x)\,dx$$

then the functions $\bar{\psi}_k(x)=\sqrt{w(x)}\cdot\psi_k(x)$ are orthogonal with respect to the inner product:

$$\langle\bar{\psi}_k\,|\,\bar{\psi}_j\rangle_w\equiv\int_a^b\bar{\psi}_k(x)\cdot\bar{\psi}_j(x)\,dx$$

5.7. Approximation of a known function by means of a rational function (Padé method)

The parameters $\mathbf{p} = \left[a_0 \ \ldots \ a_L \ \ b_1 \ \ldots \ b_M\right]^T$ of a rational function:

$$\hat{f}\left(x;\mathbf{p}\right) = \frac{a_0 + a_1 x + \ldots + a_L x^L}{1 + b_1 x + \ldots + b_M x^M} \quad \text{for } L{=}0, 1, \ldots;\ M = 1, 2, \ldots;\ L \le M$$

are determined in such a way as to satisfy the equality up to $L + M + 1$ terms:

$$\left(1 + b_1 x + \ldots + b_M x^M\right)\left(c_0 + c_1 x + c_2 x^2 + c_4 x^3 + \ldots\right) \cong a_0 + a_1 x + \ldots + a_L x^L$$

where $c_0 + c_1 x + c_2 x^2 + c_4 x^3 + \ldots$ is the MacLaurin expansion of $f\left(x\right)$.

Thus, the parameters $a_0, \ldots, a_L, b_1, \ldots, b_M$ should satisfy the following equations:

$$c_0 = a_0, c_l + \sum_{k=0}^{l-1} c_k b_{l-k} = a_l \qquad \text{for } l = 1, \ldots, L$$

$$c_l + \sum_{k=0}^{l-1} c_k b_{l-k} = 0 \qquad \text{for } l = L+1, \ldots, M$$

$$c_l + \sum_{k=l-M}^{l-1} c_k b_{l-k} = 0 \qquad \text{for } l = M+1, \ldots, M+L$$

5.8. Comparison of selected methods of interpolation and approximation

Two methods of interpolation, *viz.*:

- the interpolation by means of a Lagrange polynomial
- the interpolation by means of a polynomial spline of order 3

and two methods of approximation, *viz.*:

- the least-squares approximation by means of an algebraic polynomial
- the least-squares approximation by means of a linear combination of Chebyshev polynomials

are applied for interpolation and approximation of the function:

$$y = f(x) \equiv \sin\left(\frac{24}{x^2+4}\right) * \frac{e^{z/10}\sin(z/4) + e^{-z/10}\cos(z/4)}{10 + e^{-z/10}\sin(z/4) + e^{z/10}\cos(z/4)}, \text{ where } z = 5x+5,\ x \in [-1,1]$$

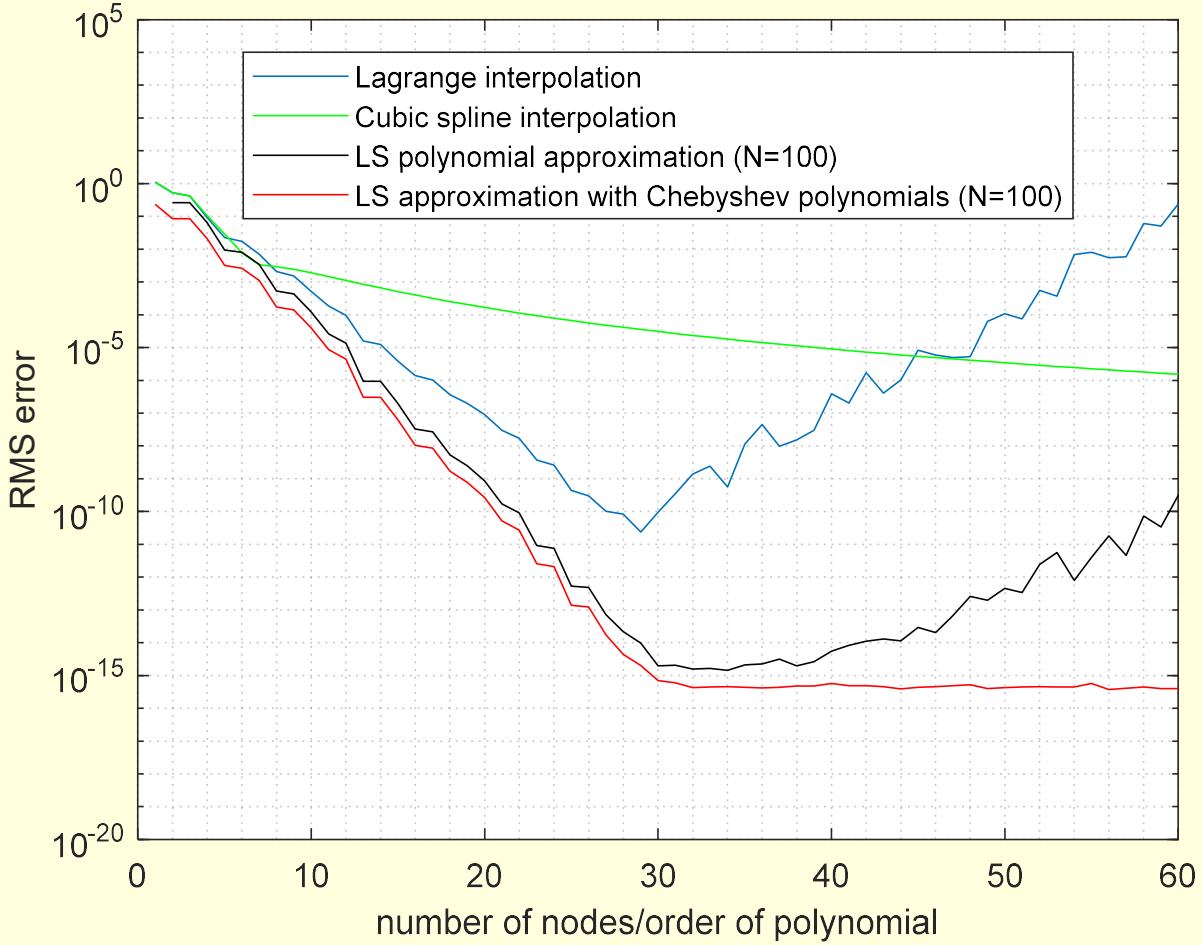
The data for this purpose are generated after the formula:

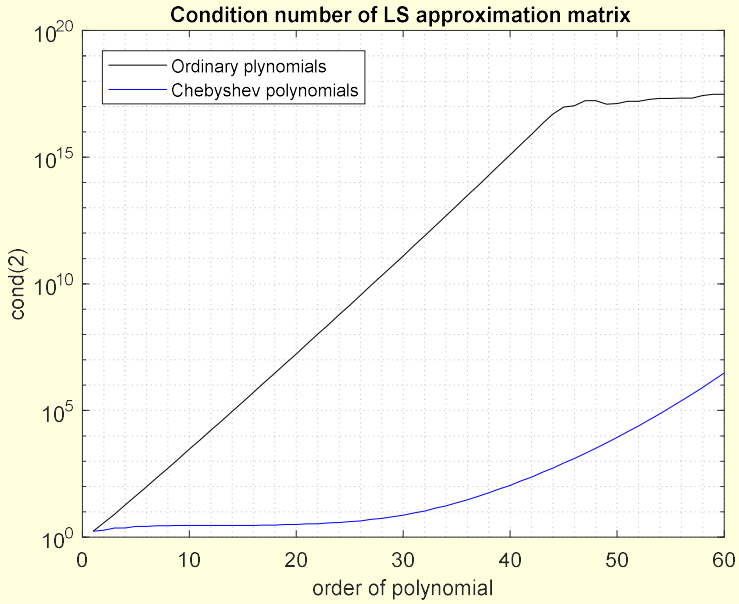
$$x_n = -1 + 2\frac{n-1}{N-1},\ y_n = f(x_n) \text{ for } n = 1, ..., N$$

The accuracy of interpolation and approximation has been assessed using the criterion:

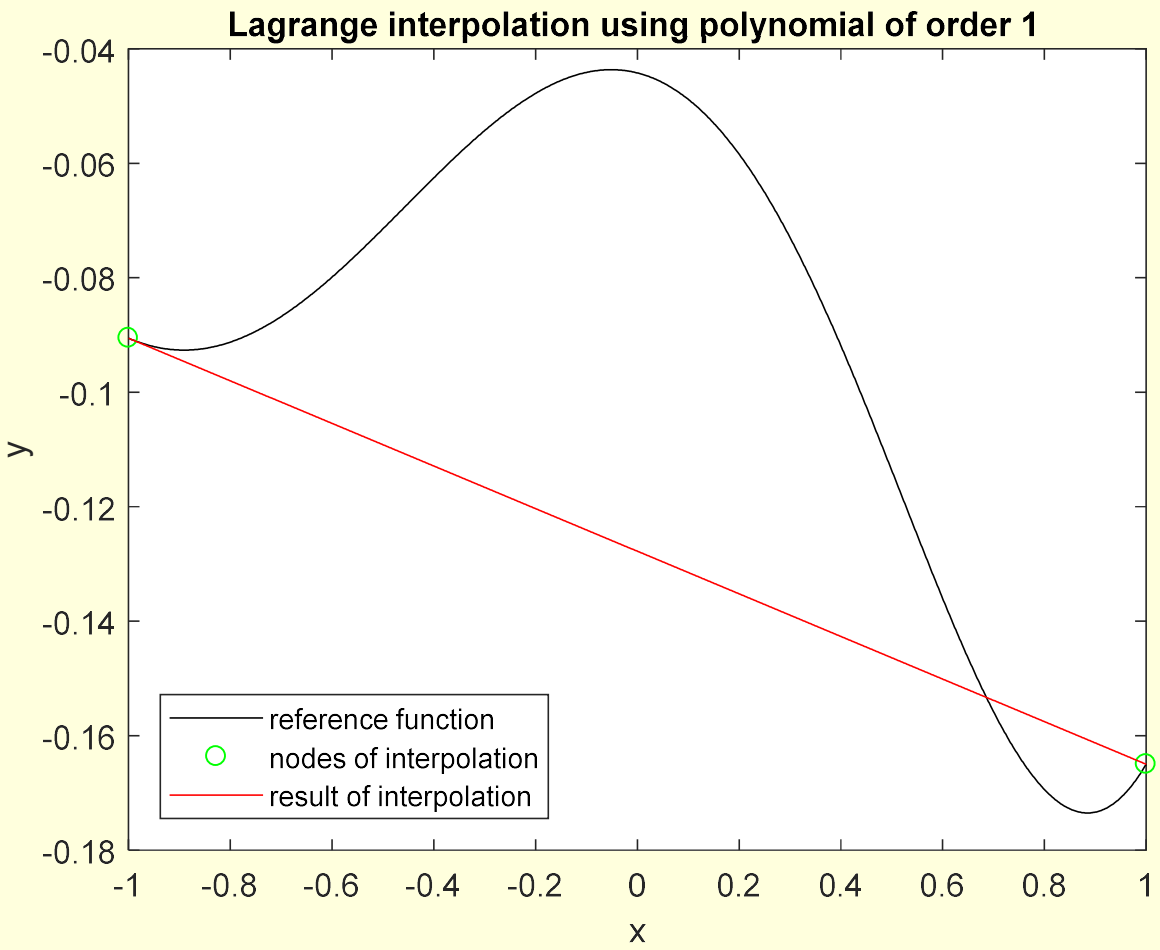
$$\Delta f \equiv \sqrt{\frac{1}{2}\int_{-1}^{+1}\left[\hat{f}(x) - f(x)\right]^2 dx} \quad \text{(root-mean-square error = RMS error)}$$

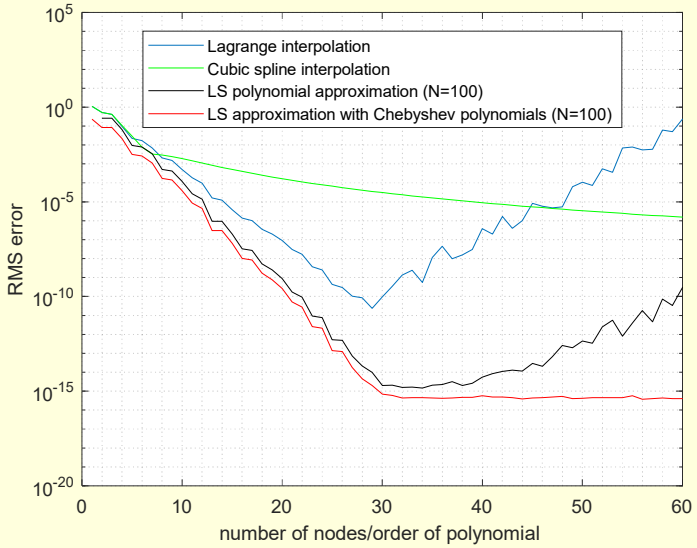
Error-free data



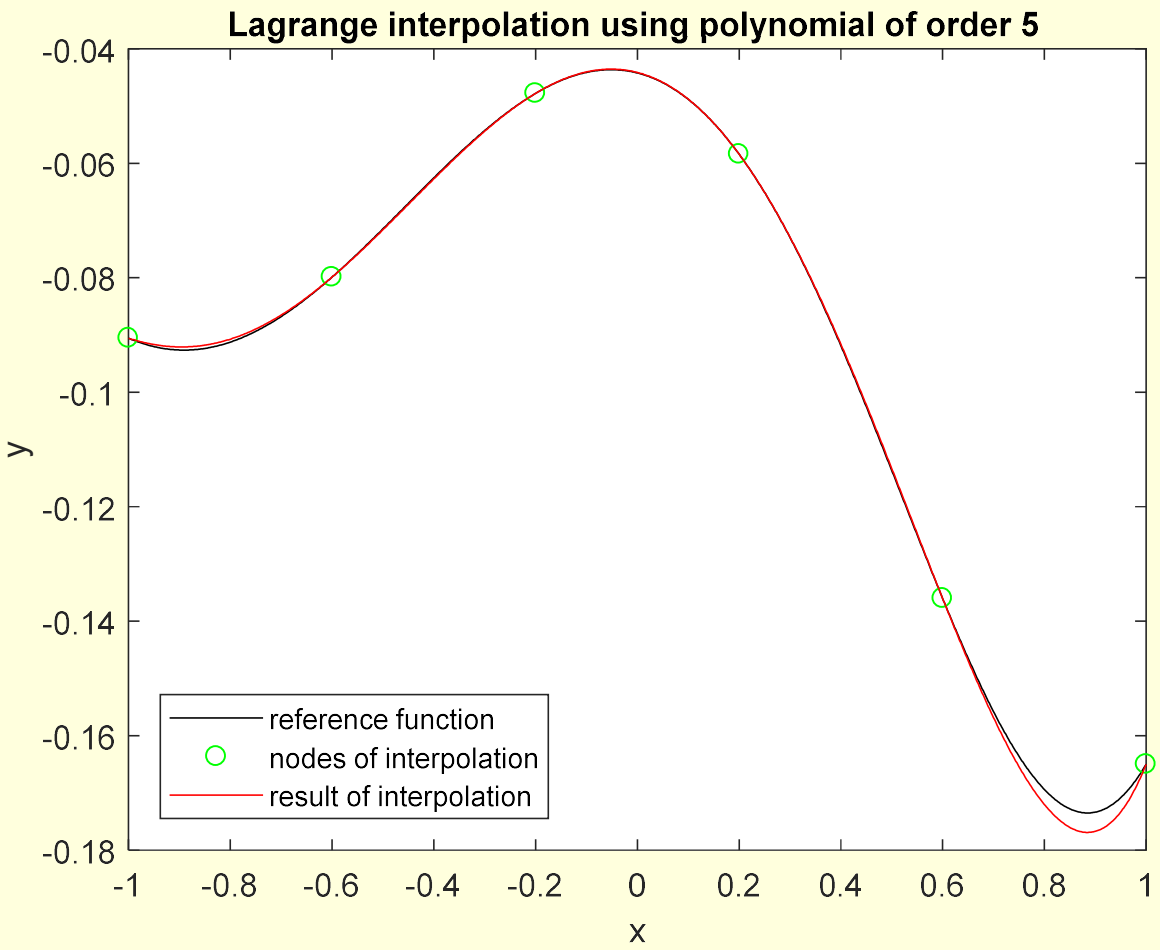


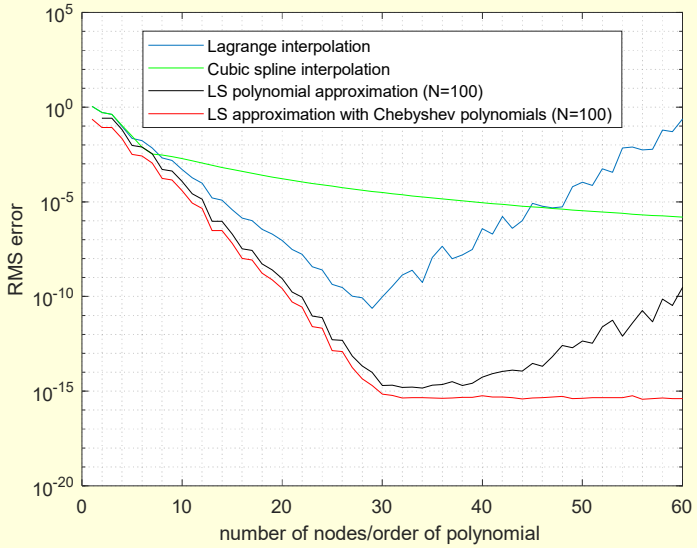
Error-free data



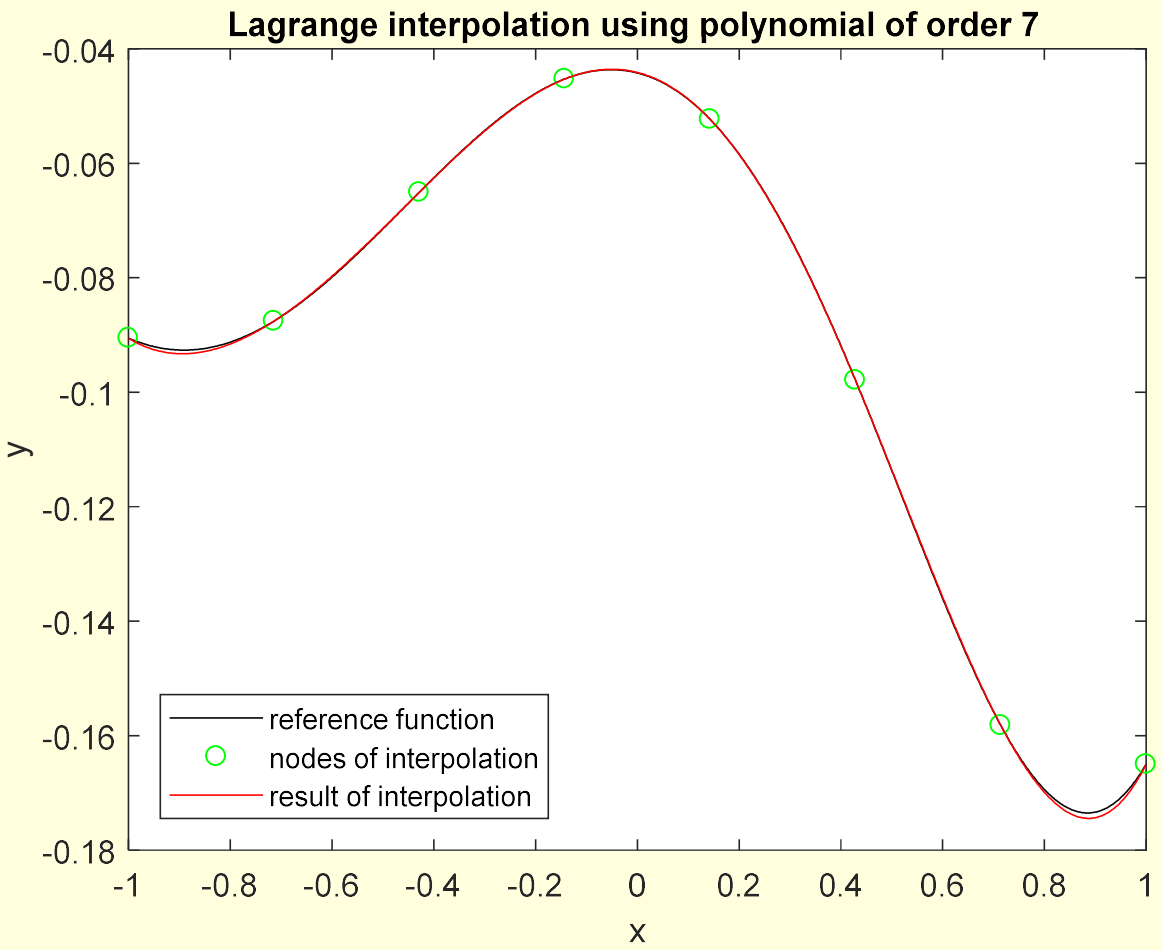


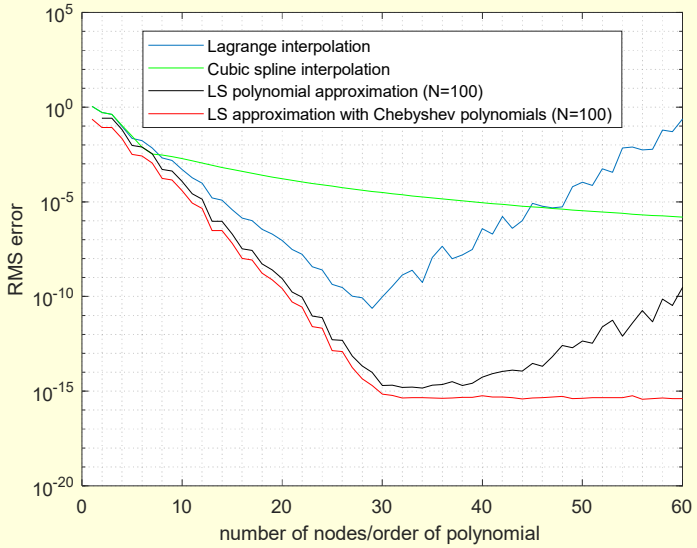
Error-free data



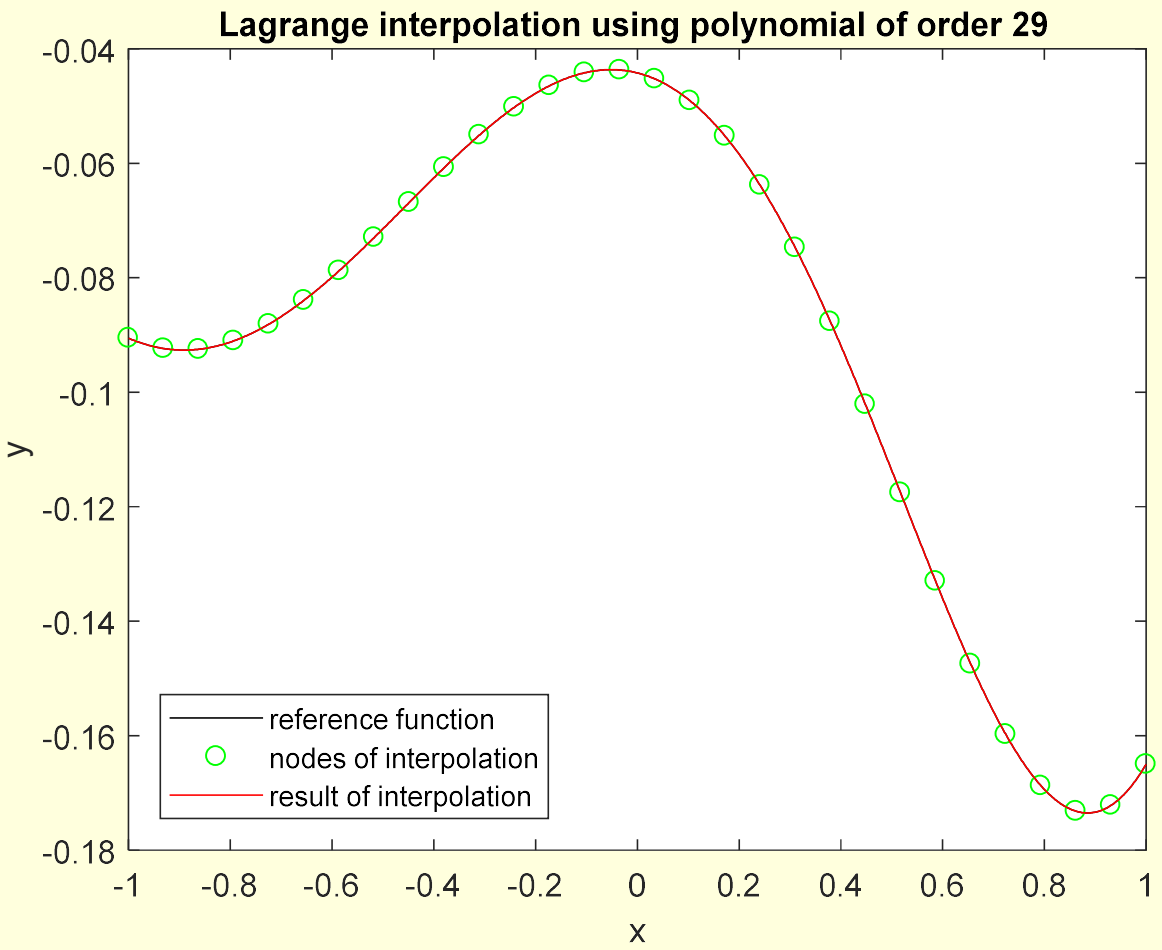


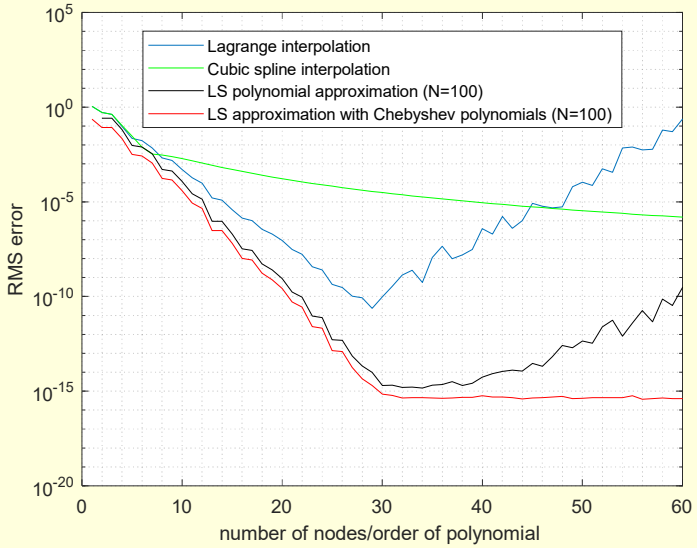
Error-free data



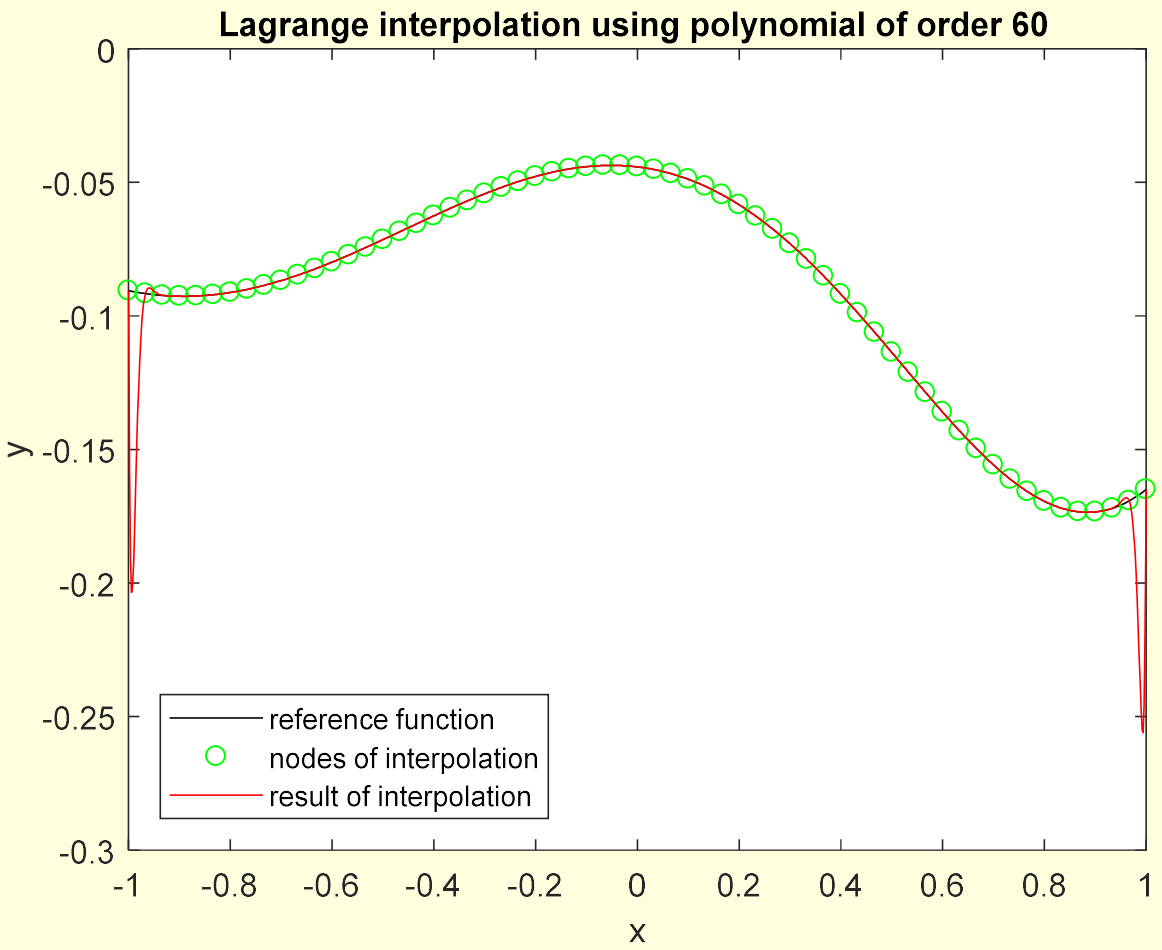


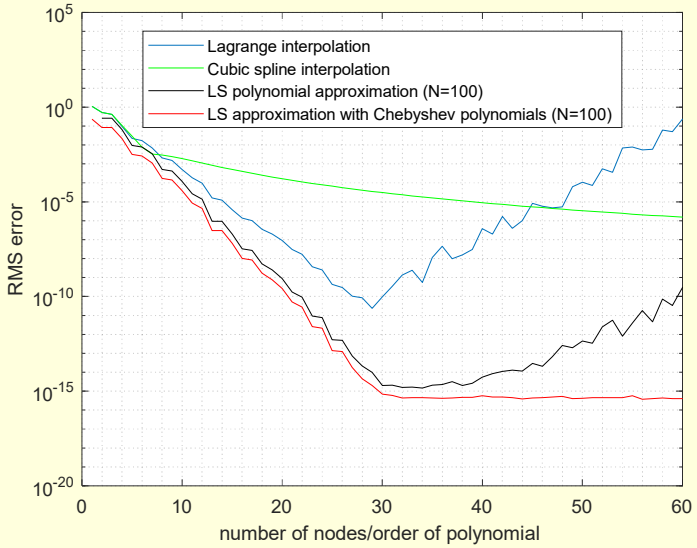
Error-free data



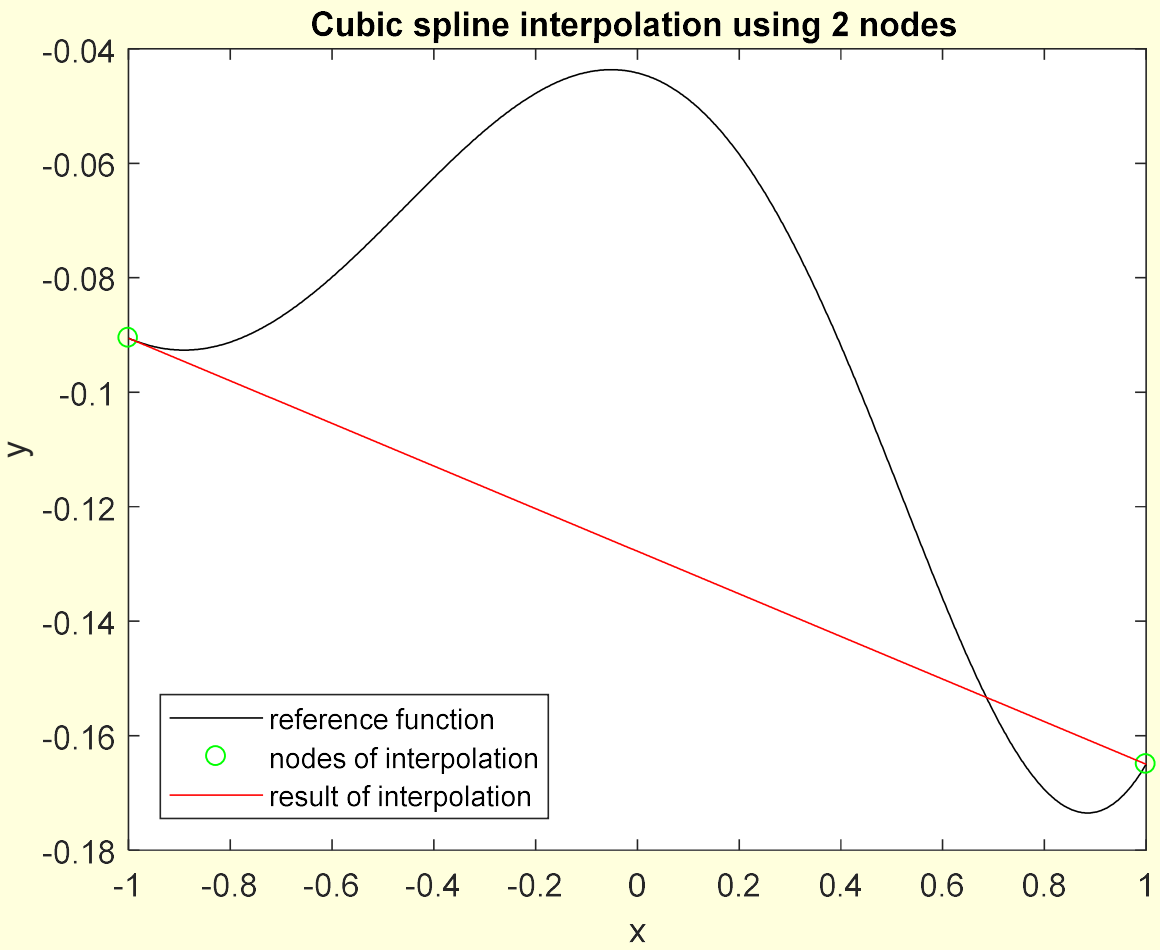


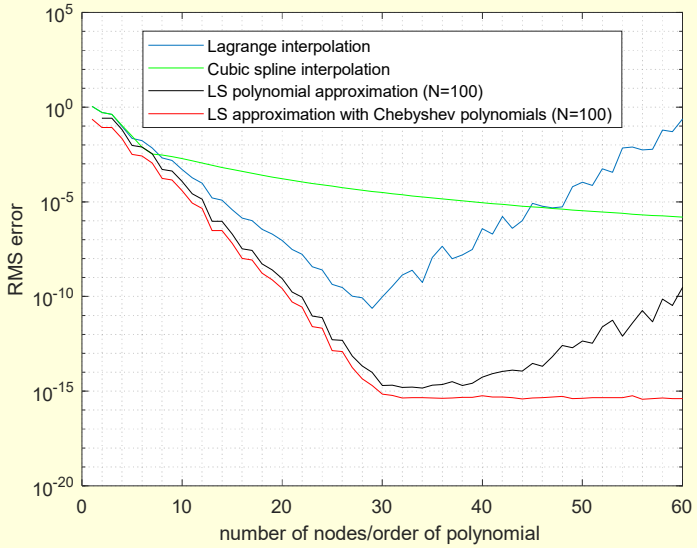
Error-free data



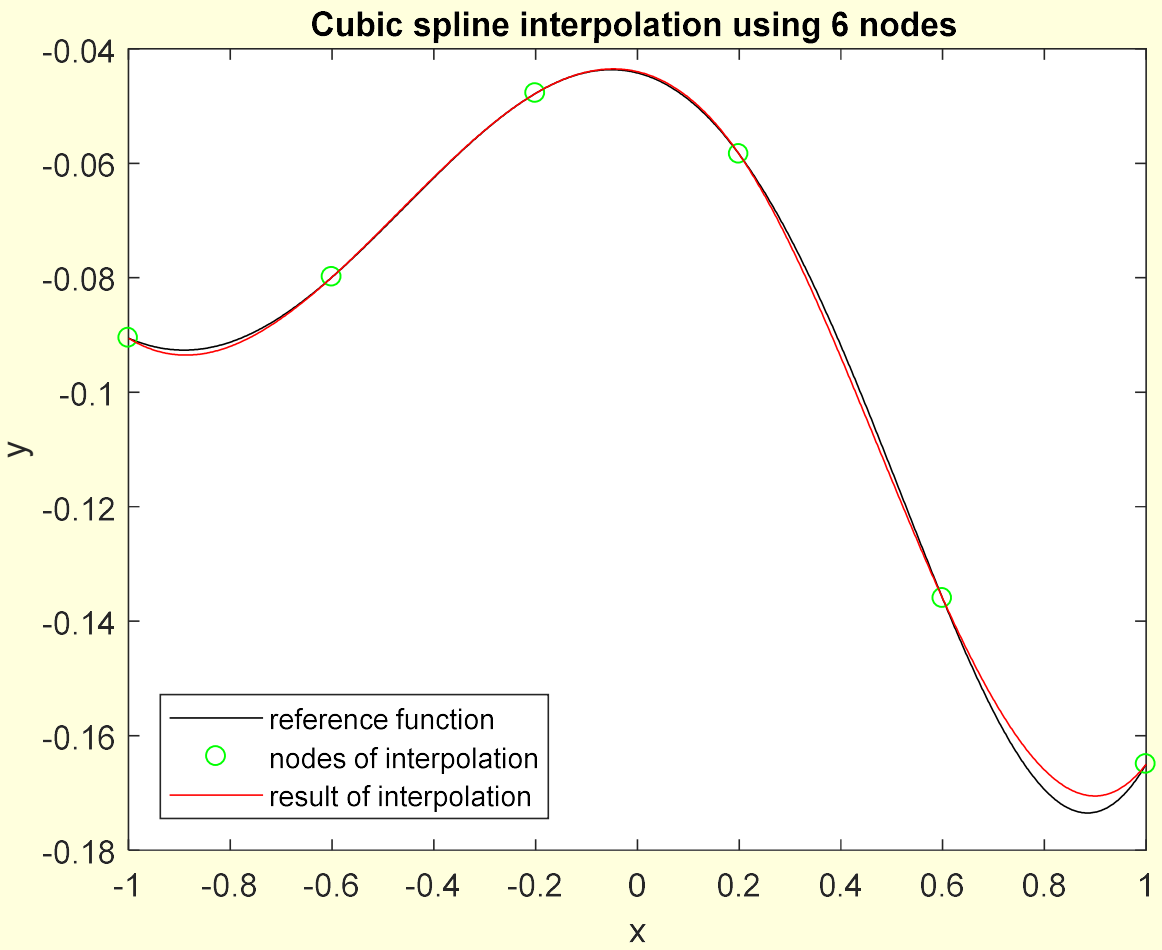


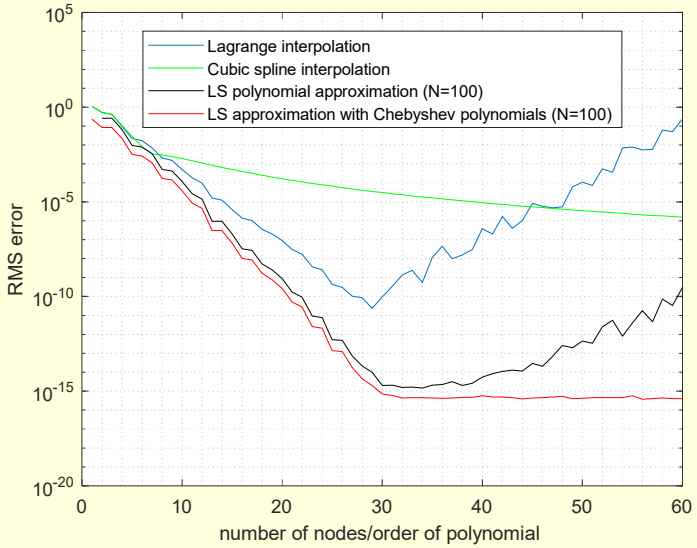
Error-free data



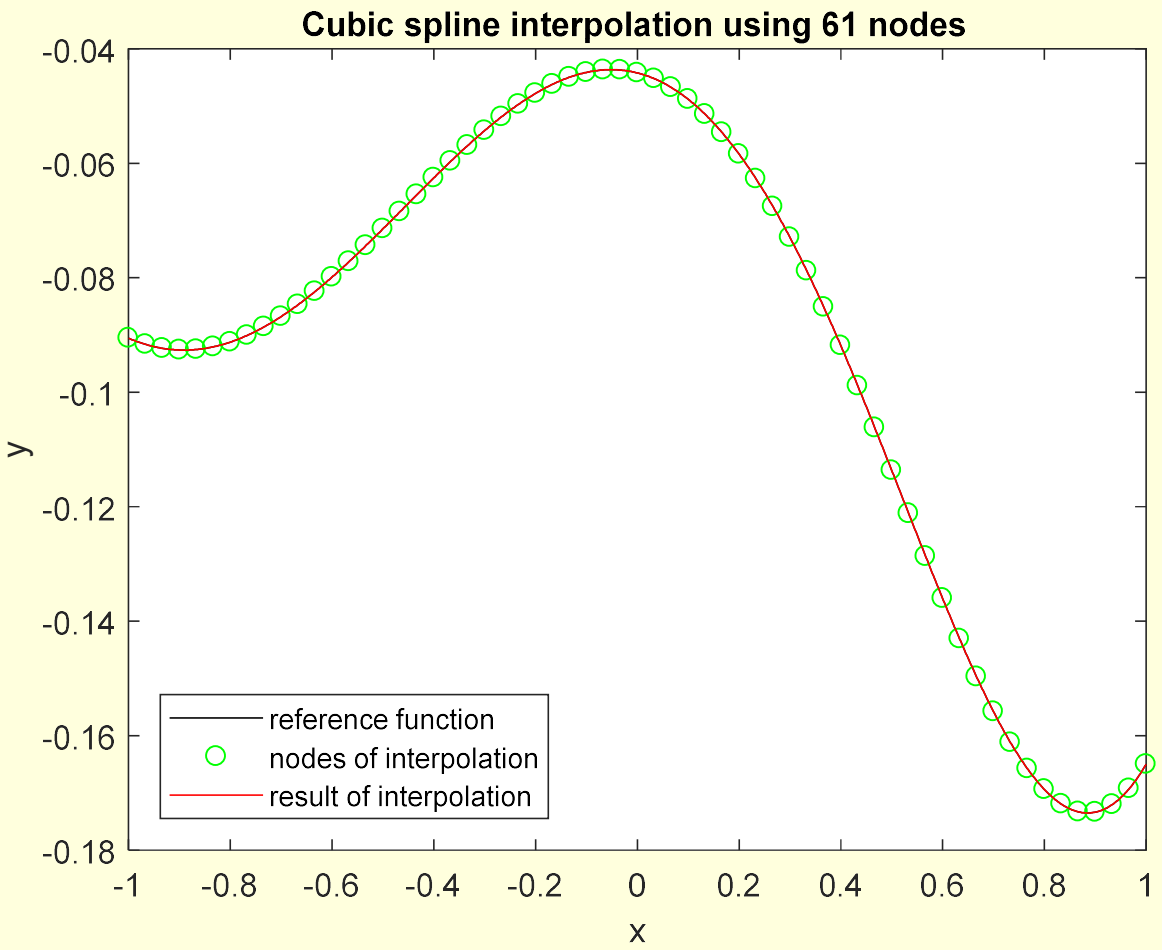


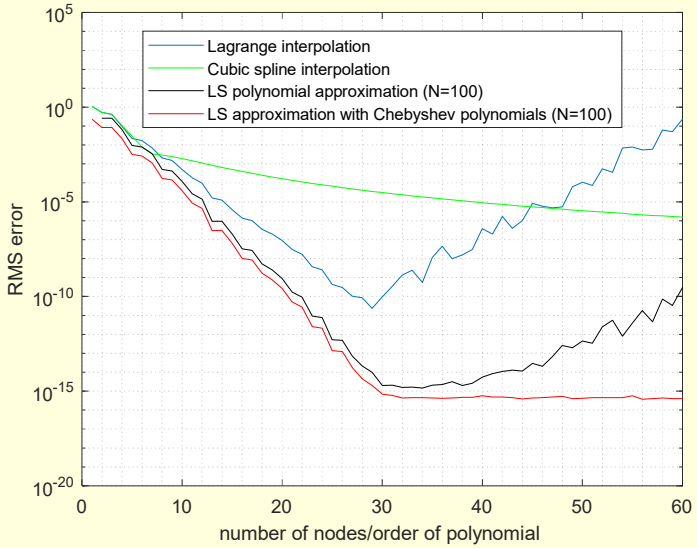
Error-free data



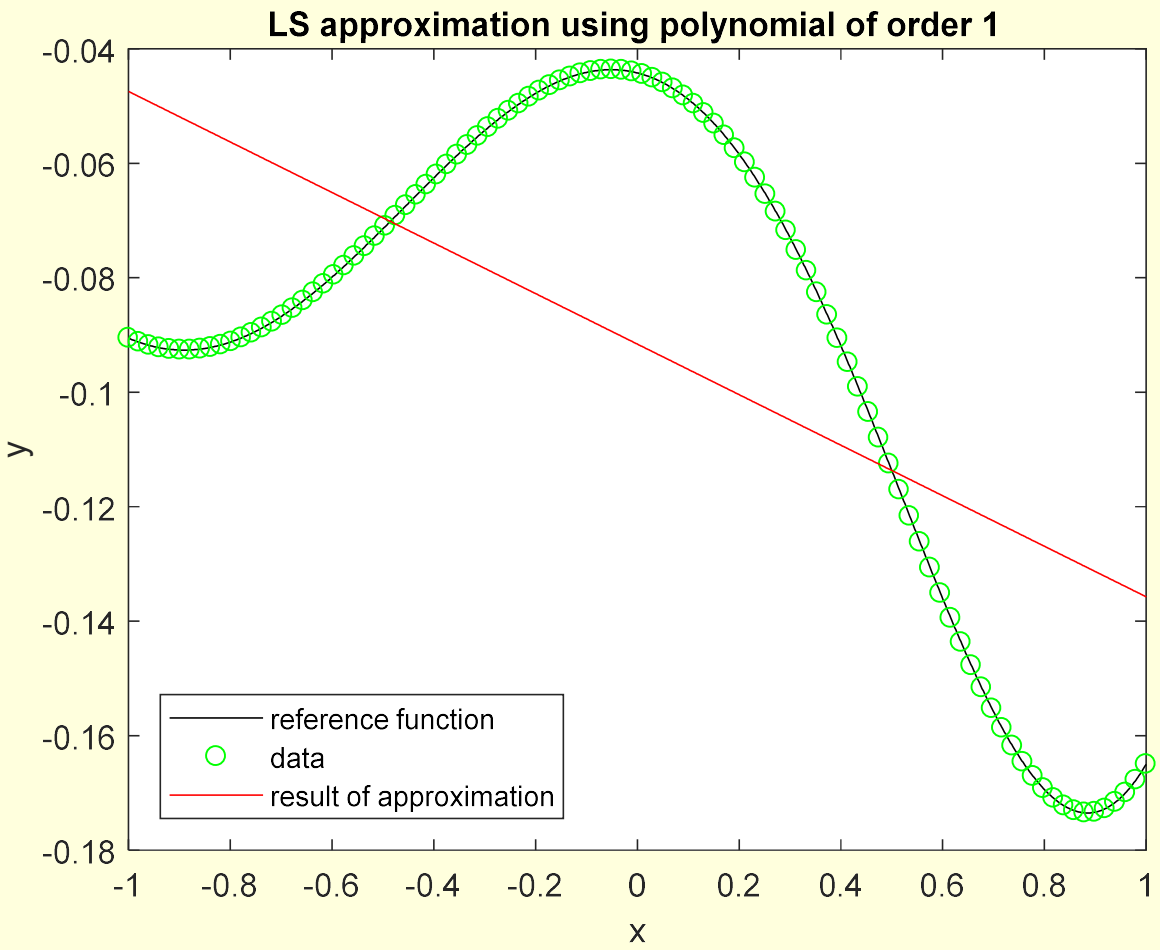


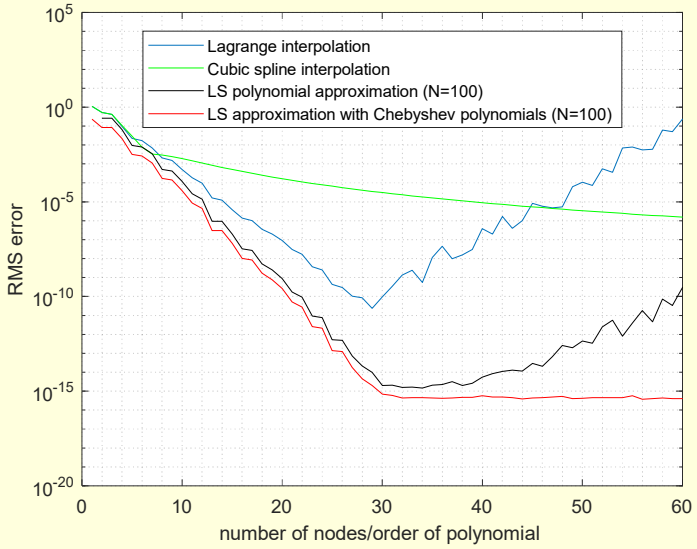
Error-free data



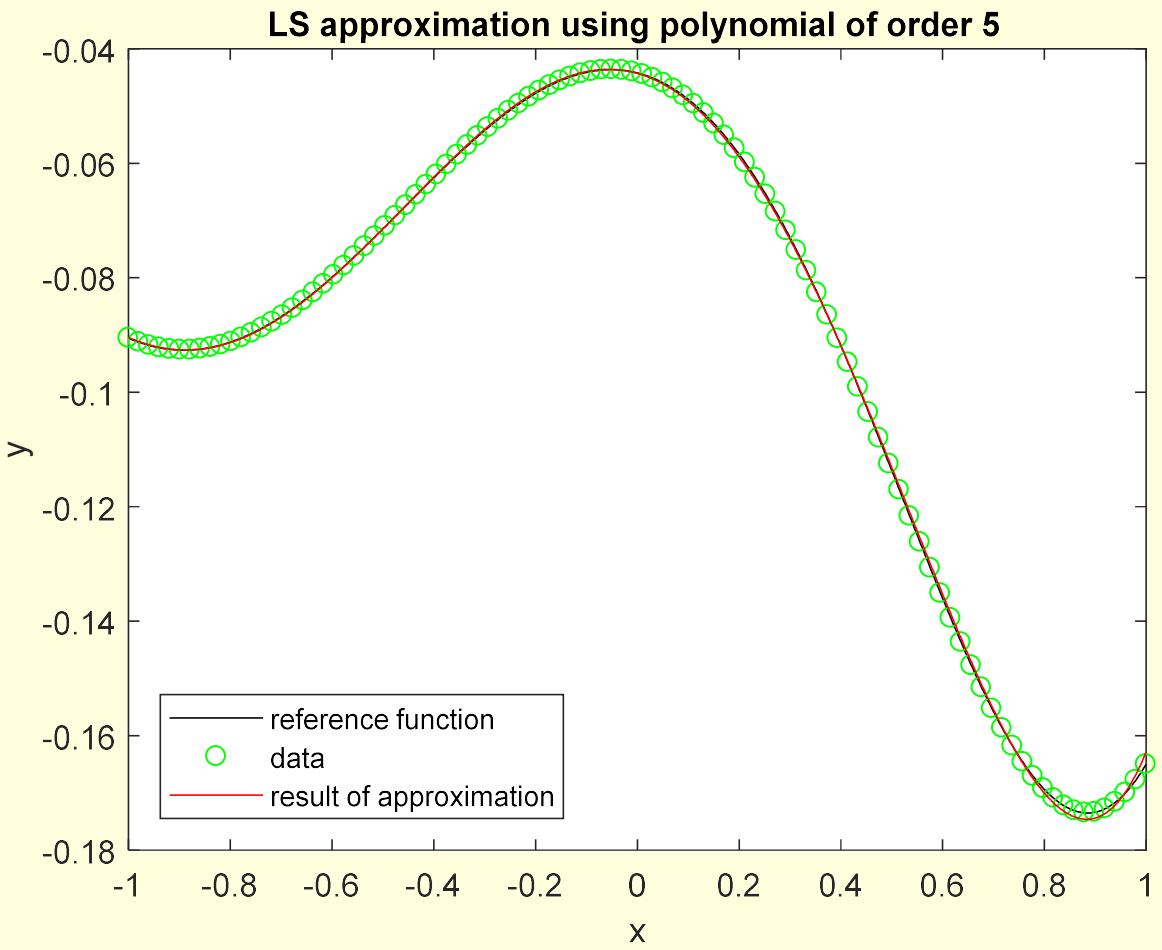


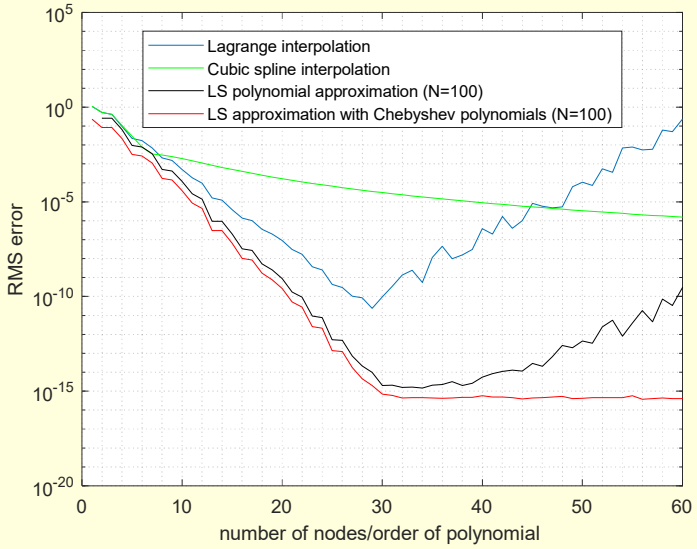
Error-free data



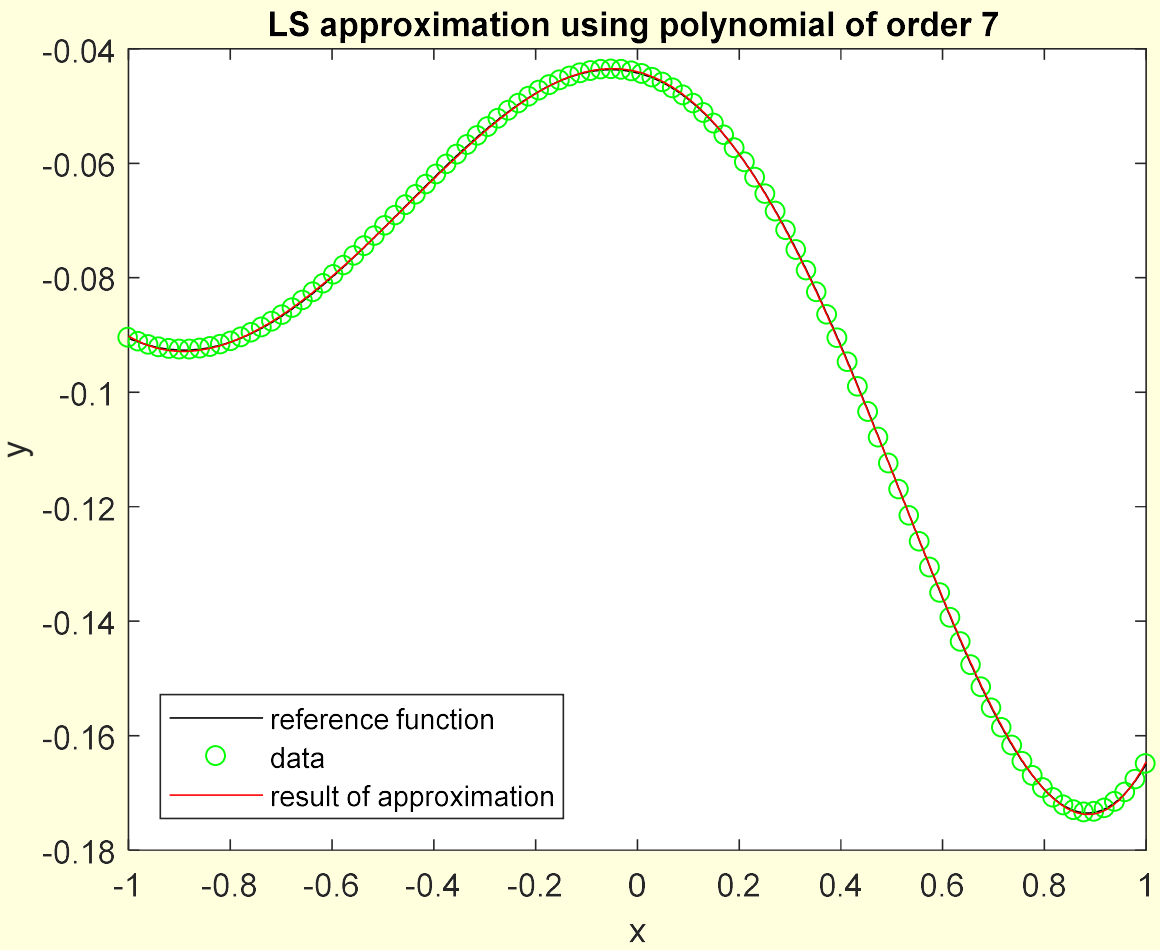


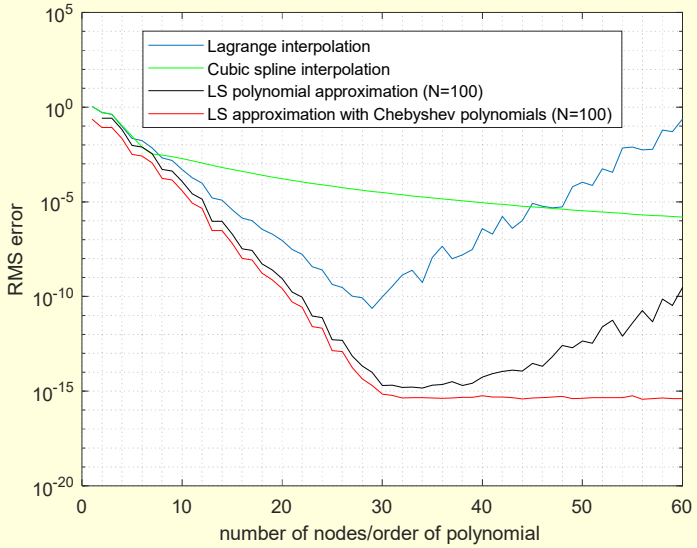
Error-free data



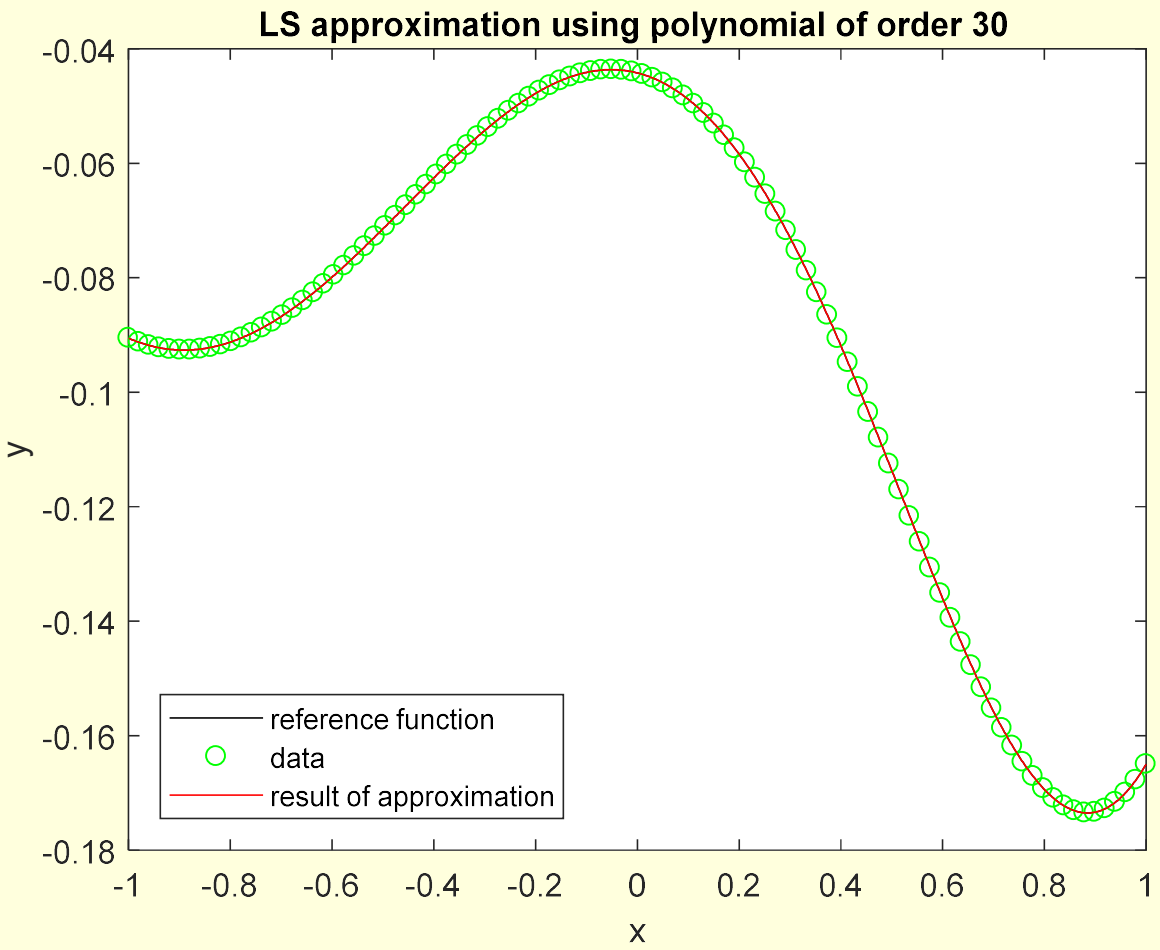


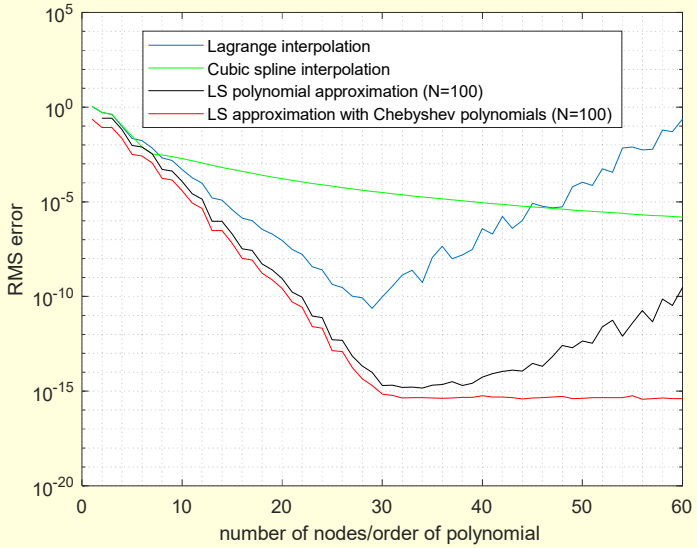
Error-free data



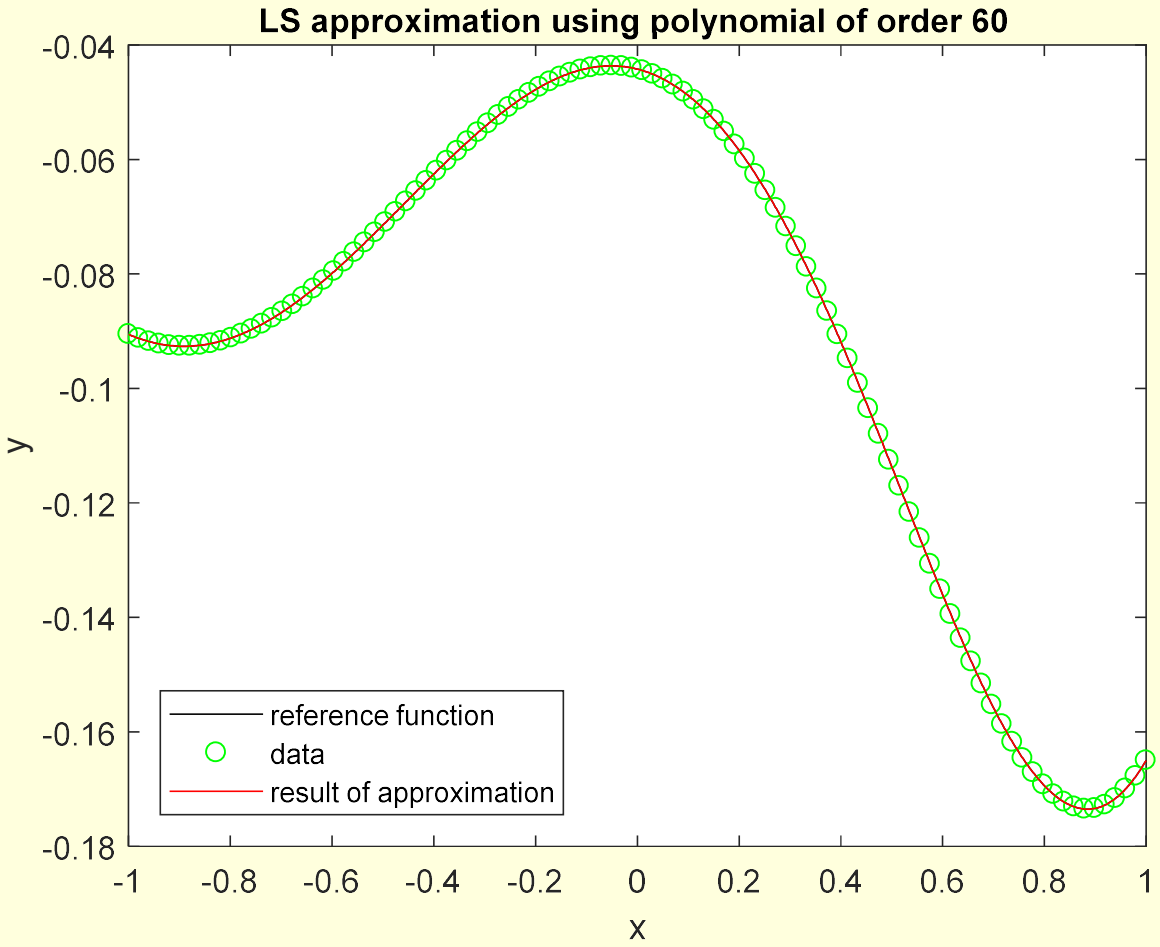


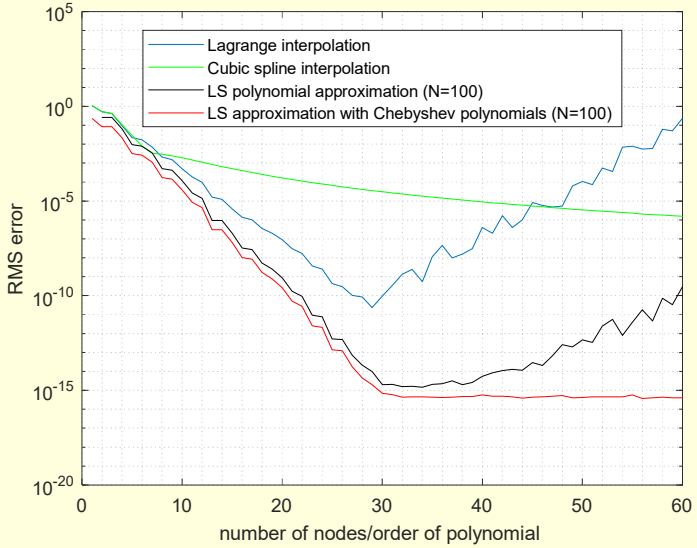
Error-free data



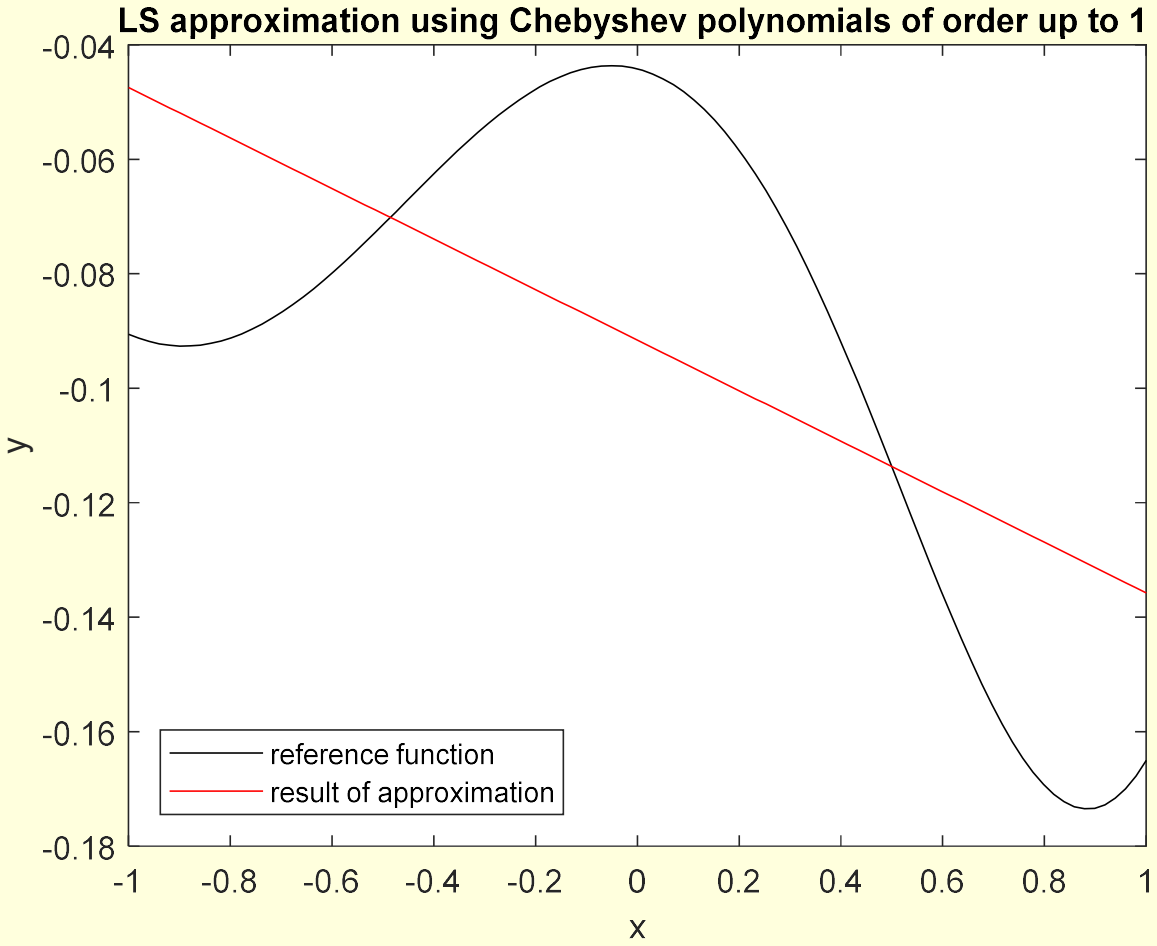


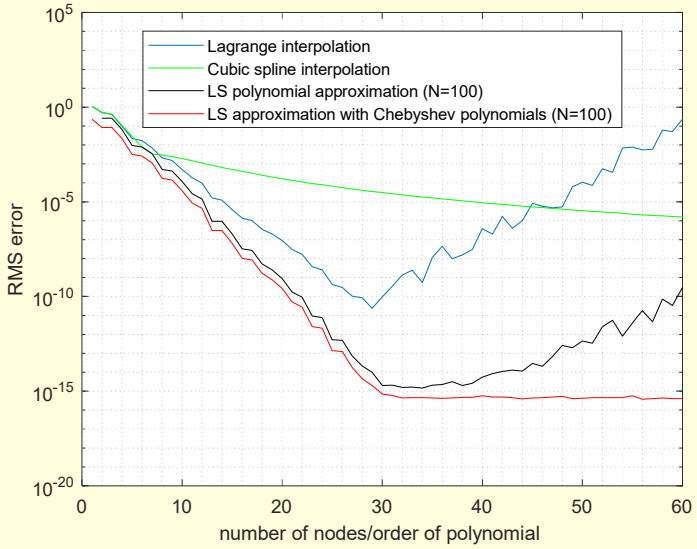
Error-free data



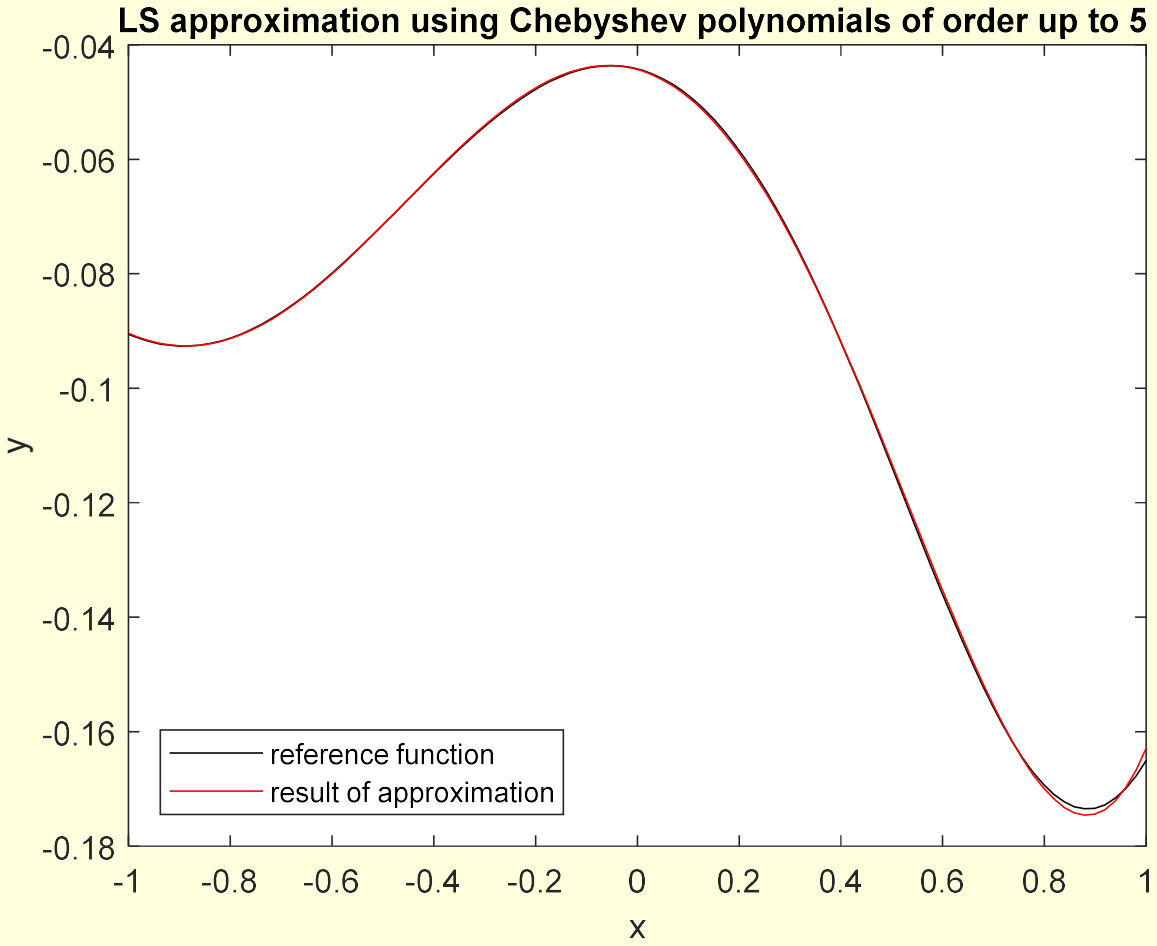


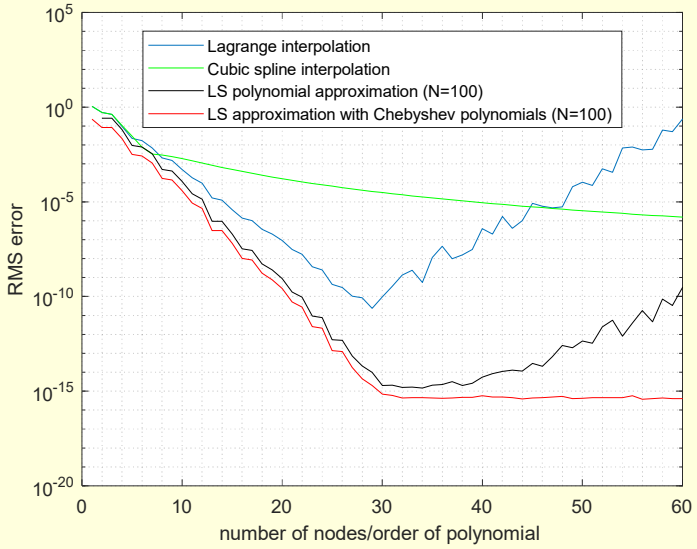
Error-free data



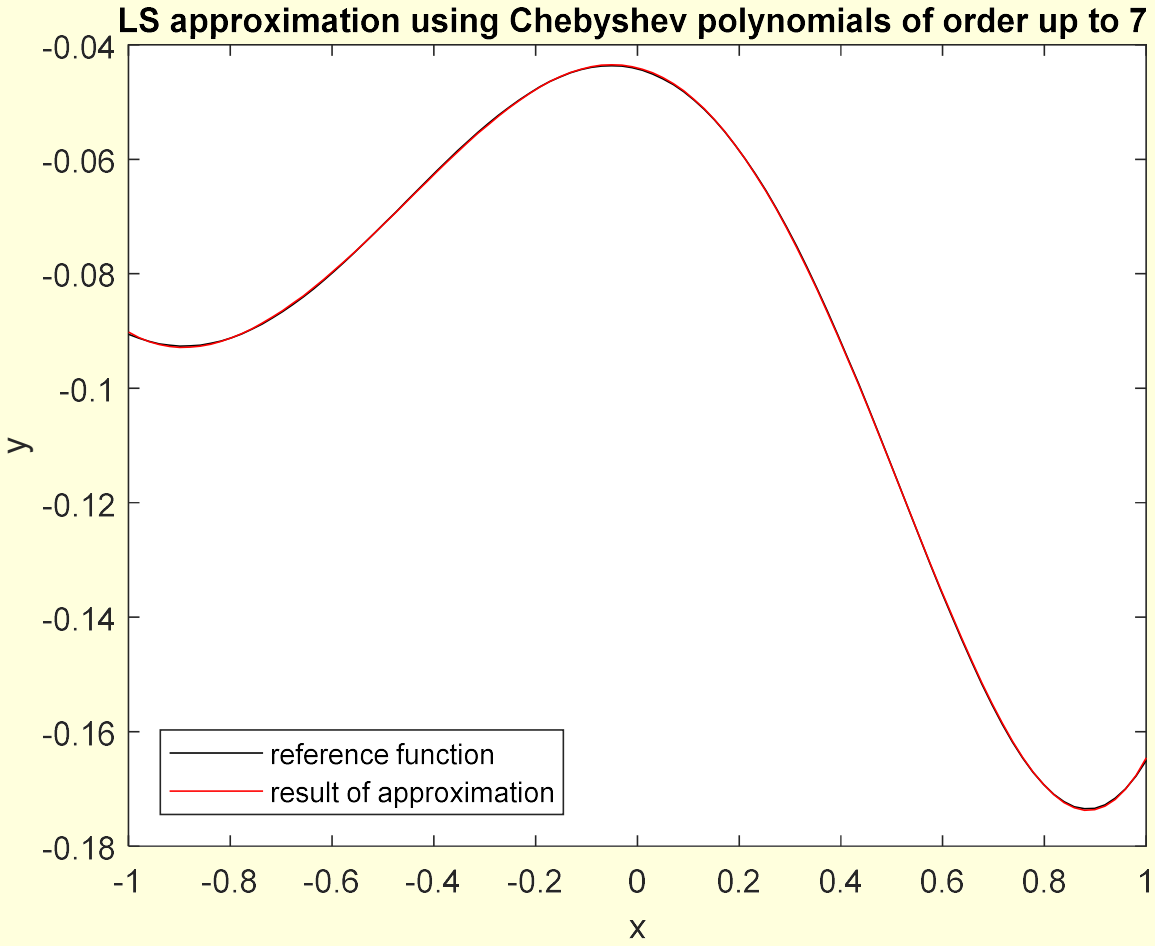


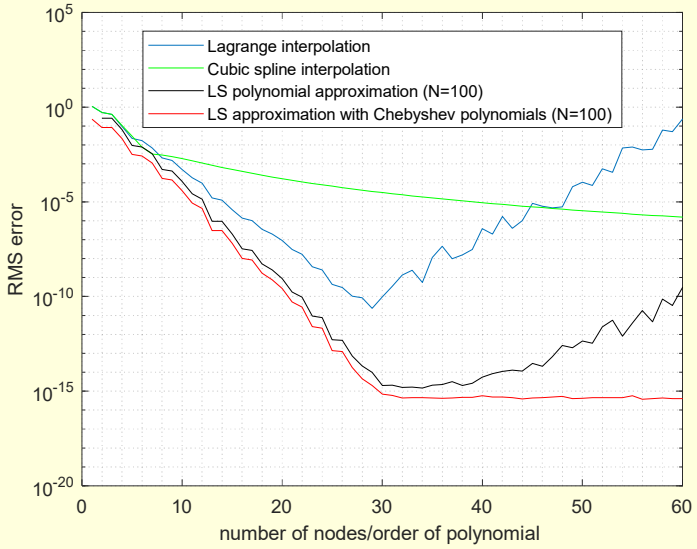
Error-free data



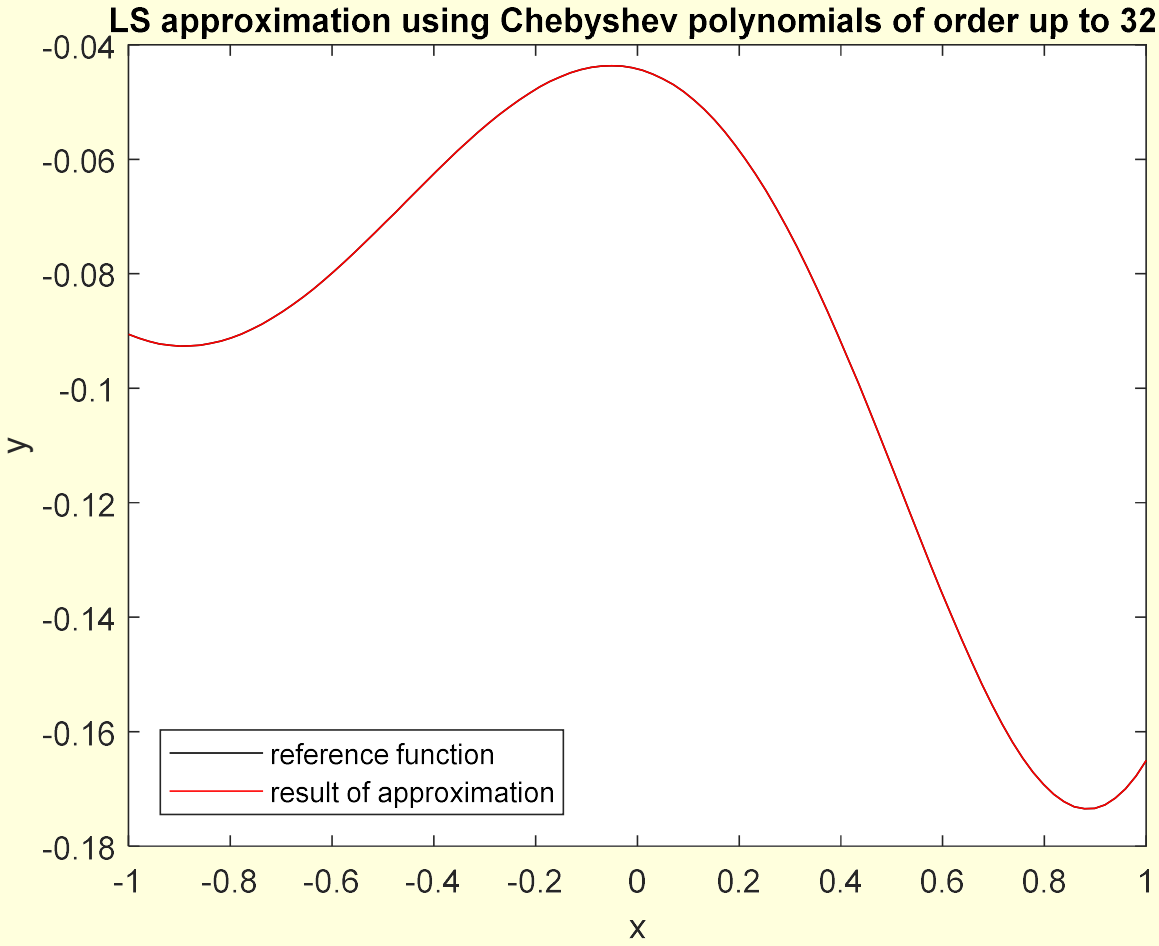


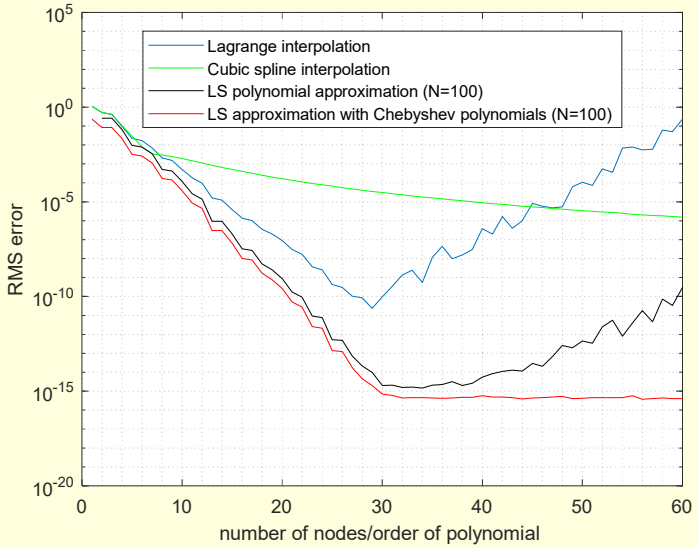
Error-free data



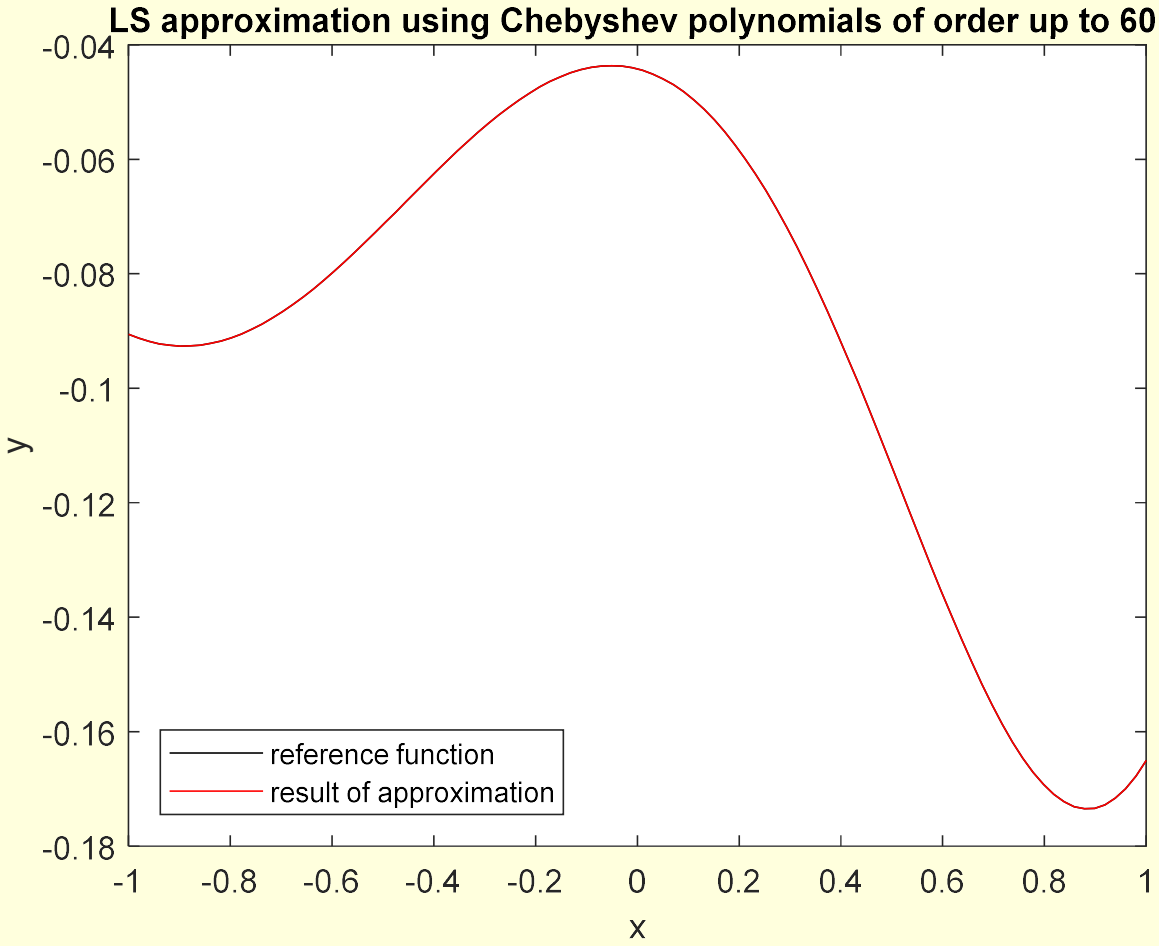


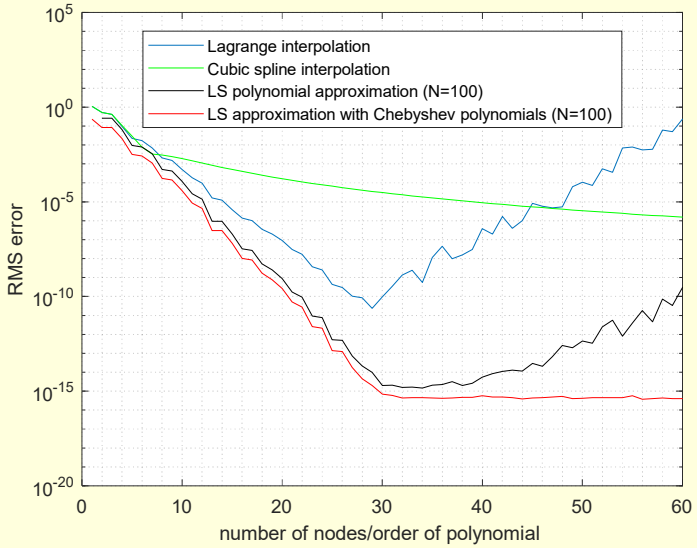
Error-free data





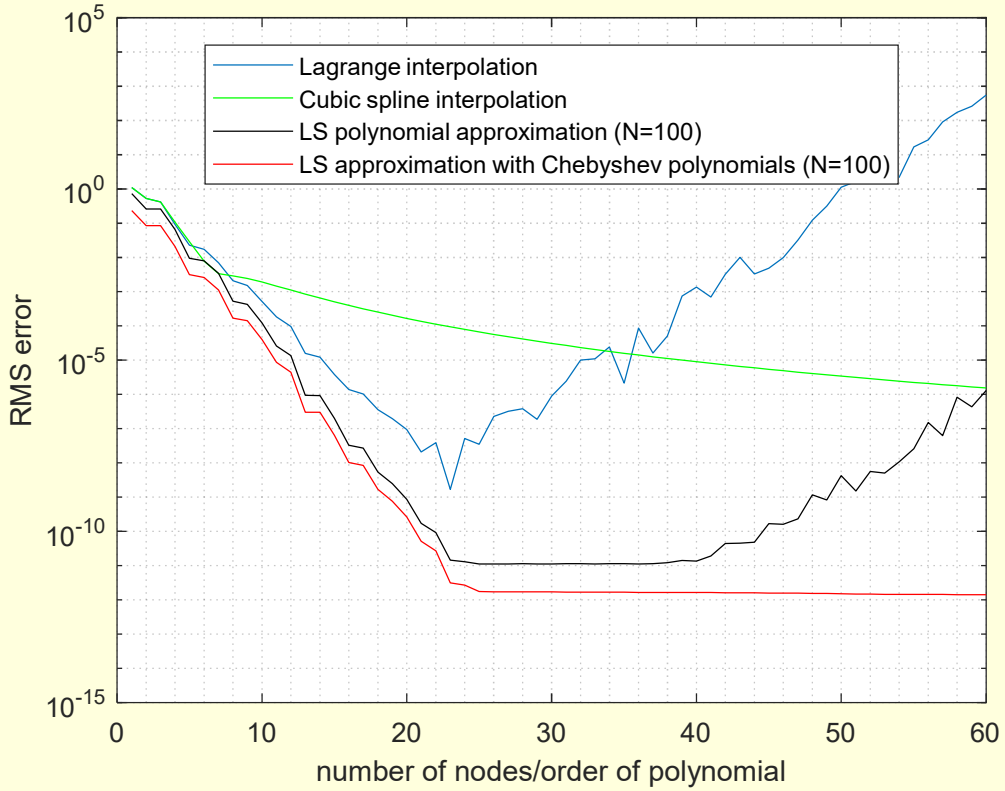
Error-free data



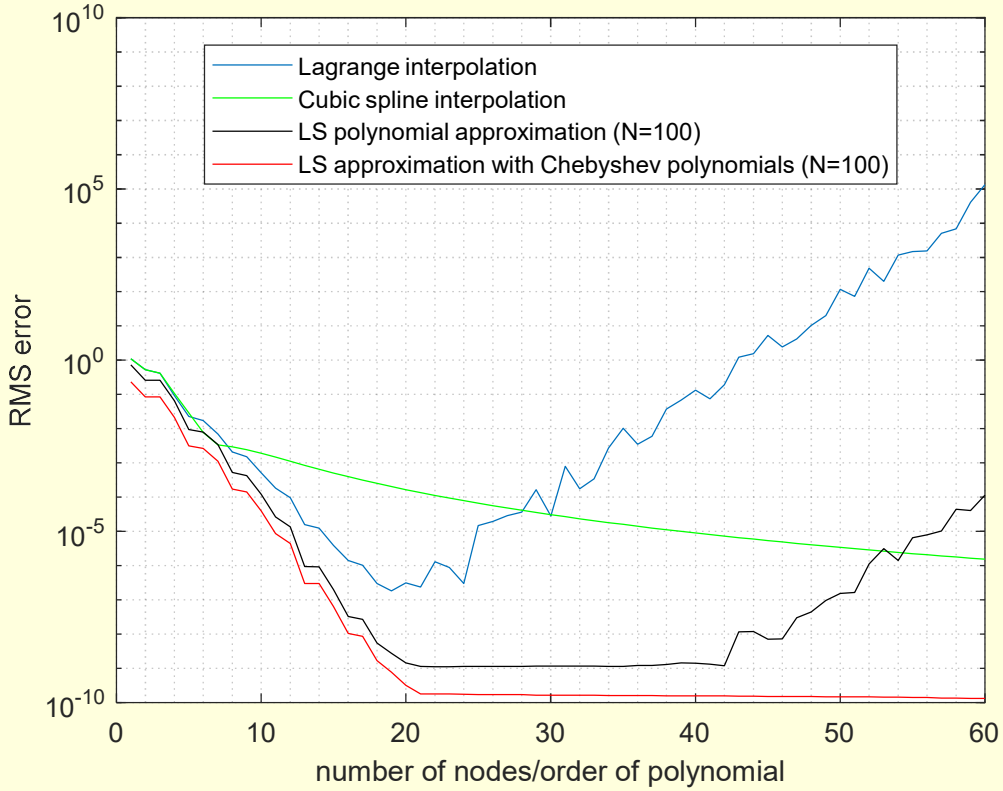


Error-corrupted data

$\sigma = 10^{-12}$

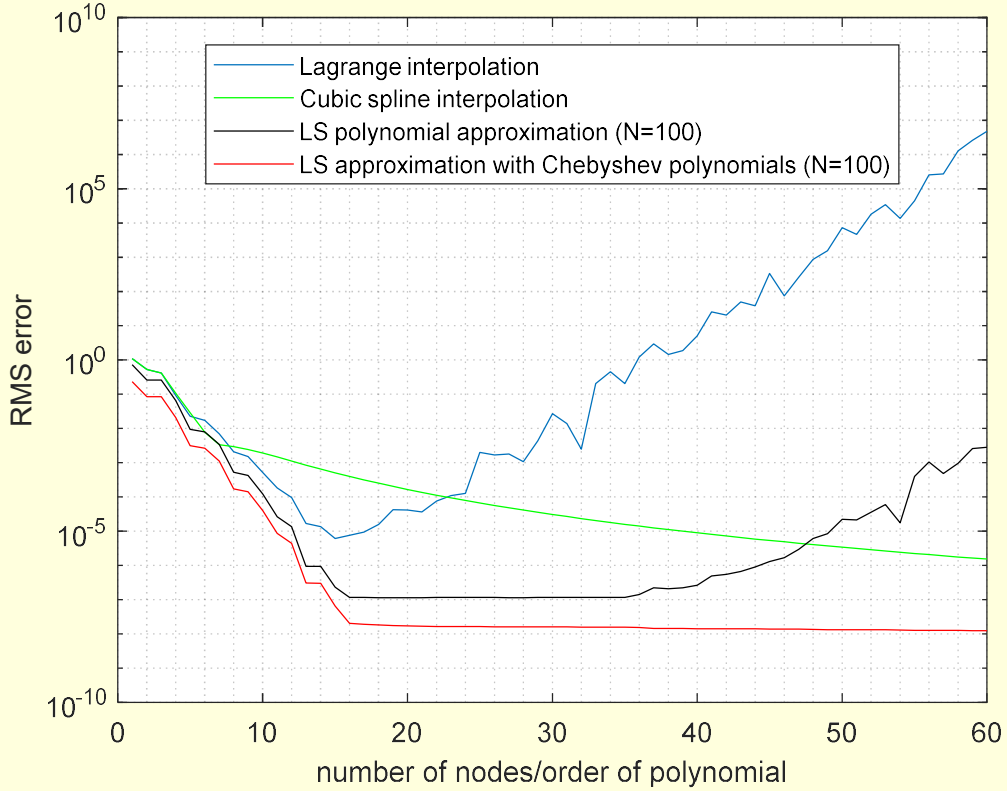


$\sigma = 10^{-10}$

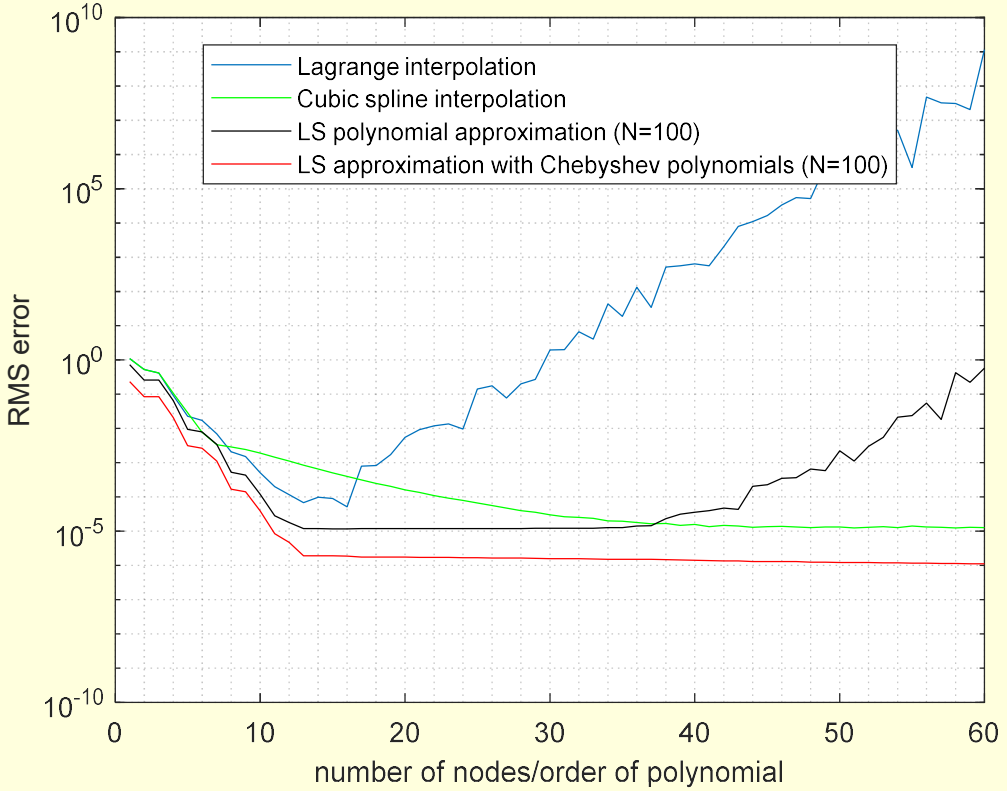


Error-corrupted data

$\sigma = 10^{-8}$



$\sigma = 10^{-6}$



Roman Z. Morawski
phone: (22) 234-7721, e-mail: *roman.morawski@pw.edu.pl*

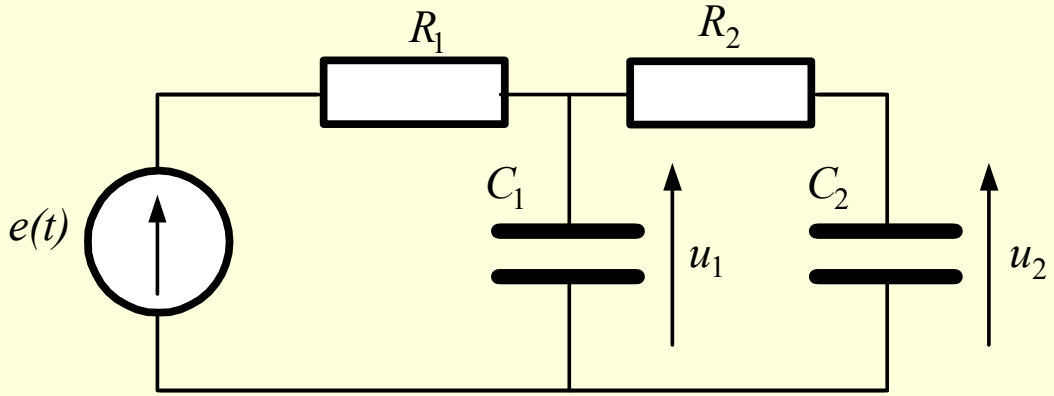
Numerical Methods (ENUME)

6. SOLVING ORDINARY DIFFERENTIAL EQUATIONS

Lecture notes for Spring Semester 2022/2023

6.1. Introductory example: dynamics of an electrical network

Using the elemental equations and Kirchhoff's laws, one may model the following network:



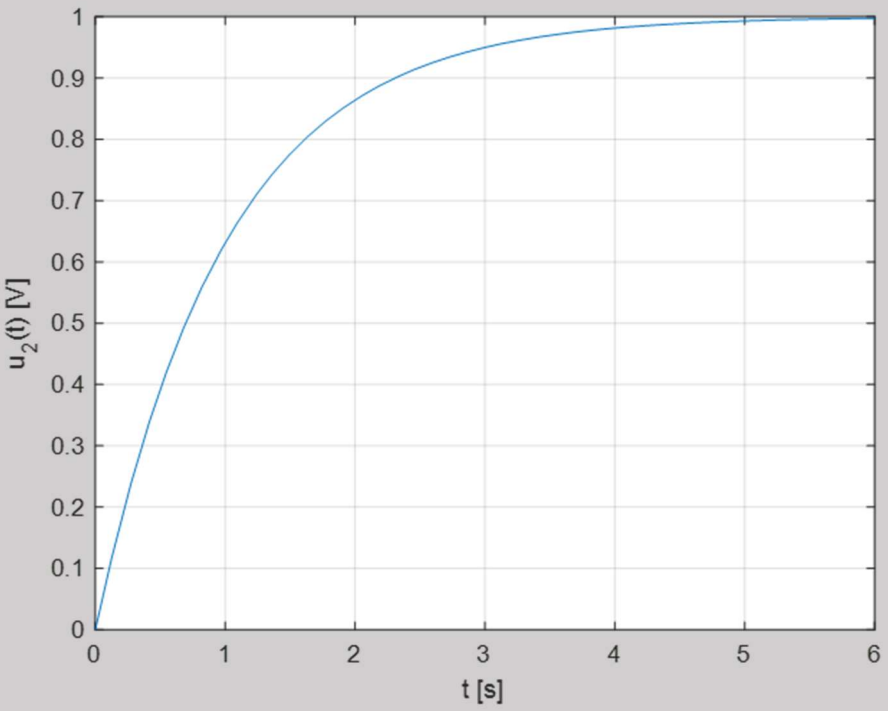
with two ordinary differential equations (ODEs):

$$\begin{bmatrix} u_1'(t) \\ u_2'(t) \end{bmatrix} = \begin{bmatrix} -\dfrac{R_1+R_2}{R_1R_2C_1} & \dfrac{1}{R_2C_1} \\ \dfrac{1}{R_2C_2} & -\dfrac{1}{R_2C_2} \end{bmatrix} \cdot \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} + \begin{bmatrix} \dfrac{1}{R_1C_1} \\ 0 \end{bmatrix} \cdot e(t) \quad \text{for } t \in [0,T]$$

which for $R_1 = 1\,\mathrm{k\Omega}$, $R_2 = 1\,\mathrm{M\Omega}$, $C_1 = C_2 = 1\,\mu\mathrm{F}$ and $e(t) = \mathbb{1}(t)\ \mathrm{V}$ takes on the form:

$$\underbrace{\begin{bmatrix} u_1'(t) \\ u_2'(t) \end{bmatrix}}_{\mathbf{u}'(t)} = \underbrace{\begin{bmatrix} -1001 & 1 \\ 1 & -1 \end{bmatrix}}_{\mathbf{A}} \cdot \underbrace{\begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}}_{\mathbf{u}(t)} + \underbrace{\begin{bmatrix} 10^3 \\ 0 \end{bmatrix}}_{\mathbf{b}} \quad \text{for } t > 0$$

```
% Symbolic solution of ODE
A=[-1001 1;1 -1];b=[1000;0];syms u(t) [2 1]
ODE=diff(u,t)==A*u(t)+b;IniCon=u(0)==[0;0];
uSol=dsolve(ODE,IniCon);u2(t)=simplify(uSol.u2);
fprintf('u2(t) = %s\n',vpa(u2(t)));
fplot(@(t) u2(t),[0,6]);grid on;
xlabel('t');ylabel('u_2(t)');
```




```
u2(t) = 9.9900e-04*exp(-1.0010e+03*t) - 1.0010*exp(-0.9990*t) + 1
```

6.2. Formulation of initial-value problem (IVP)

In scalar notation: Find the functions $y_1(t),\ldots,y_M(t)$ satisfying the following set of ODEs:

$$\frac{dy_m(t)}{dt} = f_m\left(t, y_1(t),\ldots,y_M(t)\right) \text{ for } t\in[0,T]$$

given the initial conditions, *i*.e. the values of $y_m(0)$ for $m=1,\ldots,M$.

In vector notation: Find a vector of functions $\mathbf{y}(t)\equiv\left[y_1(t)\ldots y_M(t)\right]^T$ satisfying the following system of ODEs:

$$\frac{d\mathbf{y}(t)}{dt} = \mathbf{f}\left(t,\mathbf{y}(t)\right) \text{ for } t\in[0,T]$$

given the initial condition, *i*.e. the vector $\mathbf{y}(0)\equiv\left[y_1(0)\ldots y_M(0)\right]^T$

6.3. Design of numerical methods for solving IVPs

A numerical method for solving an IVP is defined by a recursive operator:

$$\mathbf{y}_n \equiv \mathscr{S}\left(\mathbf{y}_{n-1}, \mathbf{y}_{n-2}, \ldots\right) \quad \text{for } n = 0, \ldots, N$$

generating a sequence $\{\mathbf{y}_n\}$ being an estimate of the sequence $\{\mathbf{y}(t_n)\}$, where:

$$0 = t_0 \le t_1 \le \cdots \le t_n \le \cdots \le t_N = T$$

Such a method may be designed by replacing the derivative in:

$$\frac{d\mathbf{y}(t)}{dt} = \mathbf{f}\left(t, \mathbf{y}(t)\right) \text{ for } t \in [0, T] \text{ with a formula of numerical differentiation}$$

Example: By replacing: $\left.\frac{dy(t)}{dt}\right|_{t=t_{n-1}}$ with $\frac{y(t_n)-y(t_{n-1})}{t_n-t_{n-1}}$ in $\left.\frac{dy(t)}{dt}\right|_{t=t_{n-1}} = f\left(t_{n-1}, y(t_{n-1})\right)$

the so-called forward Euler method is obtained, *viz.*:

$$y_n = y_{n-1} + h_{n-1} f\left(t_{n-1}, y_{n-1}\right)$$

where y_n is an estimate of $y(t_n)$ and $h_n = t_n - t_{n-1}$.

The algorithm for solving the system of ODEs from Section 6.1:

$$\mathbf{u}'(t) = \mathbf{A} \cdot \mathbf{u}(t) + \mathbf{b} \cdot e(t) \text{ for } t \in [0, T]$$

based on the above method takes on the form:

$$\mathbf{u}_n = \mathbf{u}_{n-1} + h \cdot \left[\mathbf{A} \cdot \mathbf{u}_{n-1} + \mathbf{b} \cdot e(t_{n-1})\right] \text{ for } n = 1, 2, \ldots \text{ and } \mathbf{u}_0 = \mathbf{u}(0)$$

6.4. General properties of numerical methods for solving IVPs

Convergence of methods for solving IVPs

A numerical method for solving an IVP is convergent if for any set of ODEs, having a unique solution $\mathbf{y}(t)$, the approximate solution $\{\mathbf{y}_n\}$, dependent on the step h, converges to $\{\mathbf{y}(t_n)\}$ for $h \to 0$, *i*.e. the global error of the numerical solution is diminishing to a zero:

$$\mathbf{e}_n \equiv \mathbf{y}_n - \mathbf{y}(t_n) \xrightarrow[h\to 0]{} \mathbf{0} \text{ for } n = 0, 1, \ldots$$

Local error and order of methods for solving IVPs

An estimate of the error generated by the operator:

$$\mathbf{y}_n \equiv \mathscr{S}\left(\mathbf{y}_{n-1}, \mathbf{y}_{n-2}, \dots\right) \quad \text{for } n = 0, \dots, N$$

during a single step of solving IVP is called *local error*:

$$\mathbf{r}_n \equiv \mathscr{S}\left(\mathbf{y}\left(t_{n-1}\right), \mathbf{y}\left(t_{n-2}\right), \dots\right) - \mathbf{y}\left(t_n\right)$$

Its dependence on the step of integration h:

$$\mathbf{r}_n \equiv \mathbf{r}_n\left(h\right) = \mathbf{r}_n\left(0\right) + \mathbf{r}_n'\left(0\right)h + \tfrac{1}{2}\mathbf{r}_n^{(2)}\left(0\right)h^2 + \dots$$

is used for the determination of the order of the method characterising its local accuracy.

A method is said to be of order p if:

$$\mathbf{r}_n\left(0\right) = 0,\ \mathbf{r}_n'\left(0\right) = 0,\ \dots,\ \mathbf{r}_n^{(p)}\left(0\right) = 0, \text{ but } \mathbf{r}_n^{(p+1)}\left(0\right) \neq 0$$

For the method of order p:

$$\mathbf{r}_n\left(h\right) \cong \frac{1}{\left(p+1\right)!}\mathbf{r}_n^{(p+1)}\left(0\right)h^{p+1}$$

Example: The local error and the order of the forward Euler method:

$$y_n = y_{n-1} + hf\left(t_{n-1}, y_{n-1}\right)$$

will be determined using a simplified notation:

$$y\left(t_n\right) \equiv \dot{y}_n,\ y'\left(t_n\right) \equiv \dot{y}'_n,\ y''\left(t_n\right) \equiv \dot{y}''_n,\ \ldots$$

$$f\left(t_n, y\left(t_n\right)\right) = f\left(t_n, \dot{y}_n\right) = \dot{y}'_n$$

The Taylor series of the RHS of the formula $y_n = y_{n-1} + hf\left(t_{n-1}, y_{n-1}\right)$ has the form:

$$y_n = \dot{y}_{n-1} + h\dot{y}'_{n-1} = \underbrace{\left(\dot{y}_n - \dot{y}'_n h + \tfrac{1}{2}\dot{y}''_n h^2 - \ldots\right)}_{y_{n-1}} + h\underbrace{\left(\dot{y}'_n - \dot{y}''_n h + \ldots\right)}_{y'_{n-1}}$$

Hence:

$$y_n = \dot{y}_n + \left(-\dot{y}'_n h + \dot{y}'_n h\right) + \left(\tfrac{1}{2}\dot{y}''_n h^2 - \dot{y}''_n h^2\right) + \ldots \cong \dot{y}_n - \tfrac{1}{2}\dot{y}''_n h^2$$

and the conclusion:

$$p = 1,\ r_n\left(h\right) \cong -\tfrac{1}{2}\dot{y}''_n h^2$$

Stability of methods for solving IVPs

For any square matrix $\mathbf{A}$, having no repeated eigenvalues $\lambda_1, \lambda_2, \ldots \in \mathbb{C}$:

$\mathbf{A} = \mathbf{V} \cdot \mathbf{\Lambda} \cdot \mathbf{V}^{-1}$, where $\mathbf{\Lambda} = diag\{\lambda_1, \lambda_2, \ldots\}$ and $\mathbf{V}$ is the matrix of eigenvectors

This decomposition may be used for transformation of a system of linear ODEs:

$$\mathbf{y}'(t) = \mathbf{A} \cdot \mathbf{y}(t) + \mathbf{B} \cdot \mathbf{u}(t)$$

into a set of independent scalar ODEs, *viz.*:

$$\mathbf{y}'(t) = \mathbf{V} \cdot \mathbf{\Lambda} \cdot \mathbf{V}^{-1} \cdot \mathbf{y}(t) + \mathbf{B} \cdot \mathbf{u}(t)$$

$$\underbrace{\mathbf{V}^{-1} \cdot \mathbf{y}'(t)}_{\mathbf{z}'(t)} = \mathbf{\Lambda} \cdot \underbrace{\mathbf{V}^{-1} \cdot \mathbf{y}(t)}_{\mathbf{z}(t)} + \mathbf{V}^{-1} \cdot \mathbf{B} \cdot \mathbf{u}(t)$$

$$\mathbf{z}'(t) = \mathbf{\Lambda} \cdot \mathbf{z}(t) + \mathbf{V}^{-1} \cdot \mathbf{B} \cdot \mathbf{u}(t), \text{ where: } \mathbf{z}(t) \equiv \mathbf{V}^{-1} \cdot \mathbf{y}(t).$$

Thus, the results of stability analysis obtained for a scalar ODE, called "test equation":

$y'(t) = \lambda \cdot y(t)$ with $\lambda \in \mathbb{C}$, for $y(0) = 1$ and $t \in [0, T]$

may be generalised on linear systems of ODEs.

Example: The forward (explicit) Euler method:

$$y_n = y_{n-1} + h \cdot f\left(t_{n-1}, y_{n-1}\right)$$

when applied to the test equation, $y'(t) = \lambda \cdot y(t)$, generates a difference equation:

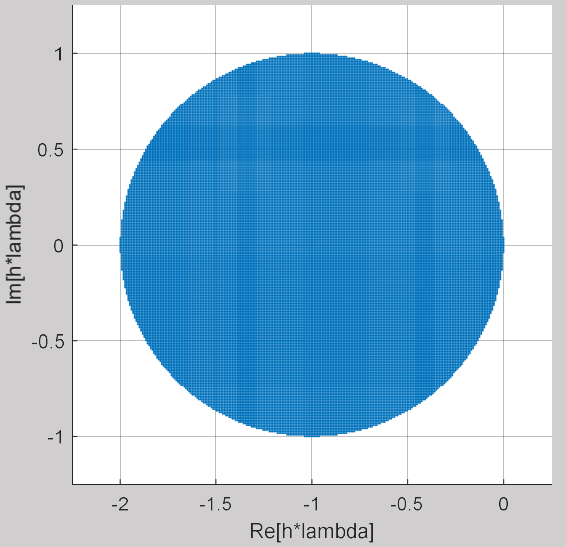
$$y_n = y_{n-1} + h\lambda y_{n-1} = \left(1 + h\lambda\right) y_{n-1}$$

Thus, it is stable if:

$$\left|1 + h\lambda\right| < 1 \;\Rightarrow\; \left|1 + h\lambda\right|^2 < 1 \;\Rightarrow\; \left[1 + \mathrm{Re}\left(h\lambda\right)\right]^2 + \left[\mathrm{Im}\left(h\lambda\right)\right]^2 < 1$$

i.e. for $h\lambda$ in the unite circle whose centre is located at $(-1, 0)$.

```
% REGION OF STABILITY FOR EXPLICIT EULER METHOD.
% NOTATION: h*lambda = x+j*y
% Range of x and y coordinates:
  N=300;xx=linspace(-2.25,0.25,N);yy=linspace(-1.25,1.25,N);
% Generation of grid with all points (x,y):
  [x,y] = meshgrid(xx,yy);
% Representation of inequality in logical matrix:
  hl=x+1j*y;I=abs(1+hl)<1;
% Plot of points (x,y) that satisfy inequality:
  scatter(x(I),y(I),0.3);grid on
  xlabel('Re[h*lambda]');ylabel('Im[h*lambda]');
  axis('equal');axis([-2.25 0.25 -1.25 1.25]);
```



Example: The backward (implicit) Euler method:

$$y_n = y_{n-1} + h \cdot f\left(t_n, y_n\right) \;\Rightarrow\; y_n = \arg_y \left\{ y = y_{n-1} + h \cdot f\left(t_n, y\right) \right\}$$

when applied to the test equation, $y'(t) = \lambda \cdot y(t)$, generates a difference equation:

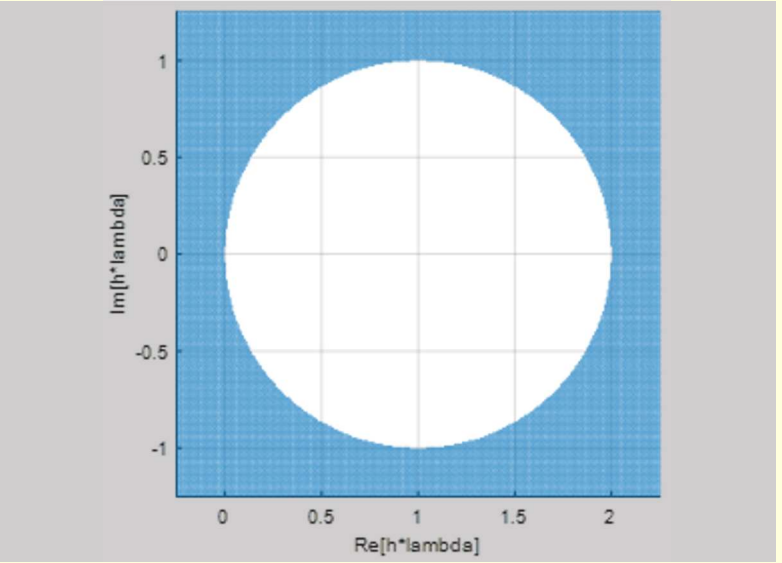
$$y_n = y_{n-1} + h\lambda y_n = \frac{1}{1-h\lambda} y_{n-1}$$

Thus, it is stable if:

$$\left| \frac{1}{1-h\lambda} \right| < 1 \;\Rightarrow\; \left|1-h\lambda\right| > 1 \;\Rightarrow\; \left[1-\mathrm{Re}\left(h\lambda\right)\right]^2 + \left[-\mathrm{Im}\left(h\lambda\right)\right]^2 > 1$$

i.e. for all $h\lambda$ with $\mathrm{Re}\left(h\lambda\right) < 0$.

```
% REGION OF STABILITY FOR IMPLICIT EULER METHOD.
% NOTATION: h*lambda = x+j*y
% Range of x and y coordinates:
  N=300;xx=linspace(-0.25,2.25,N);yy=linspace(-1.25,1.25,N);
% Generation of grid with all points (x,y):
  [x,y] = meshgrid(xx,yy);
% Representation of inequality in logical matrix:
  hl=x+1j*y;I=1./abs(1-hl)<1;
% Plot of points (x,y) that satisfy inequality:
  scatter(x(I),y(I),0.3);grid on
  xlabel('Re[h*lambda]');ylabel('Im[h*lambda]');
  axis('equal');axis([-0.25 2.25 -1.25 1.25]);
```



6.5. Single-step methods

Generalised Runge-Kutta formula for single-step methods

$$y_n = y_{n-1} + h \cdot \sum_{k=1}^{K} w_k f_k$$

where:

$$f_k = f\left(t_{n-1} + c_k h,\ y_{n-1} + h \cdot \sum_{\kappa=1}^{K} a_{k,\kappa} f_\kappa \right) \quad \text{for } k = 1, 2, \ldots, K$$

$$c_k \in [0,1] \text{ and } w_k \in [0,1] \text{ for } k = 1, 2, \ldots, K$$

$$\sum_{k=1}^{K} w_k = 1$$

Butcher's table of coefficients:

c_1	$a_{1,1}$	$a_{1,2}$	$\cdots$	$a_{1,K}$
c_2	$a_{2,1}$	$a_{2,2}$	$\cdots$	$a_{2,K}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_K	$a_{K,1}$	$a_{K,2}$	$\cdots$	$a_{K,K}$
	w_1	w_2	$\cdots$	w_K

Runge-Kutta formula for explicit single-step methods

If $a_{k,\kappa} = 0$ for $\kappa \geq k$, then the method is *explicit*, and the values of f_k may be determined recursively:

$$f_k = f\left(t_{n-1} + c_k h,\, y_{n-1} + h \cdot \sum_{\kappa=1}^{k-1} a_{k,\kappa} f_\kappa \right) \quad \text{for } k = 1, 2, \ldots, K;$$

Runge-Kutta formula for implicit single-step methods

Otherwise it is *implicit*, and therefore the values of f_k must be determined by solving the system of algebraic equations (nonlinear, if the function f is nonlinear:

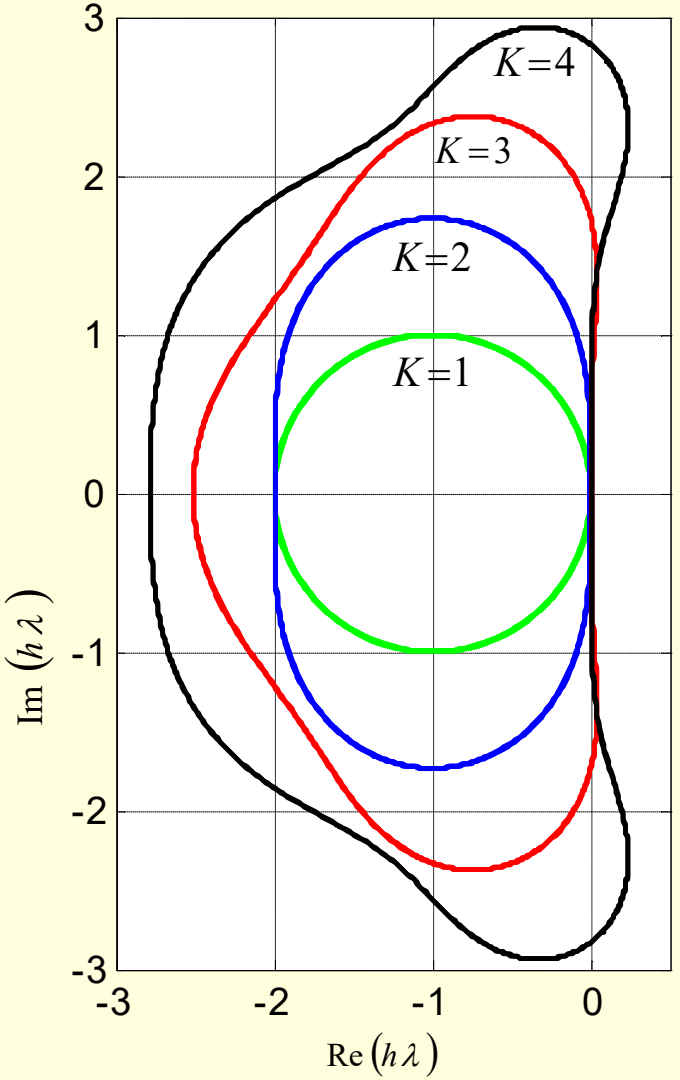
$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_K \end{bmatrix} = \begin{bmatrix} f\left(t_{n-1} + c_1 h,\, y_{n-1} + h \cdot \sum_{\kappa=1}^{K} a_{1,\kappa} f_\kappa \right) \\ f\left(t_{n-1} + c_2 h,\, y_{n-1} + h \cdot \sum_{\kappa=1}^{K} a_{2,\kappa} f_\kappa \right) \\ \vdots \\ f\left(t_{n-1} + c_K h,\, y_{n-1} + h \cdot \sum_{\kappa=1}^{K} a_{K,\kappa} f_\kappa \right) \end{bmatrix}$$

Order and stability of explicit single-step methods

The maximal possible order:

$$p(K)=\begin{cases} K & \text{for} \quad K=1,2,3,4 \\ K-1 & \text{for} \quad K=5,6,7 \\ K-2 & \text{for} \quad K\geq 8 \end{cases}$$

Regions of stability:



Lower-order explicit single-step methods

1. The forward Euler method ($K=1,\ p=1$):

$$y_n = y_{n-1} + h \cdot f\left(t_{n-1}, y_{n-1}\right)$$

0	0
	1

2. The modified (midpoint) Euler method ($K=2,\ p=2$):

$$y_n = y_{n-1} + hf\left(t_{n-1} + \tfrac{1}{2}h, y_{n-1} + \tfrac{1}{2}hf\left(t_{n-1}, y_{n-1}\right)\right)$$

0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0
	0	1

or

$$y_n = y_{n-1} + hf\left(t_{n-\frac{1}{2}}, y_{n-\frac{1}{2}}\right) \text{ with } y_{n-\frac{1}{2}} = y_{n-1} + \tfrac{1}{2}hf\left(t_{n-1}, y_{n-1}\right) \text{ and } t_{n-\frac{1}{2}} = t_{n-1} + \tfrac{1}{2}h$$

3. The Heun method ($K=2,\ p=2$):

$$y_n = y_{n-1} + \tfrac{1}{2}h\Big[f\left(t_{n-1}, y_{n-1}\right) + f\left(t_{n-1}+h, y_{n-1}+hf\left(t_{n-1}, y_{n-1}\right)\right)\Big]$$

0	0	0
1	1	0
	$\frac{1}{2}$	$\frac{1}{2}$

or

$$y_n = y_{n-1} + h\cdot\frac{\left[f\left(t_{n-1}, y_{n-1}\right) + f\left(t_n, \hat{y}_n\right)\right]}{2} \quad \text{with } \hat{y}_n = y_{n-1} + hf\left(t_{n-1}, y_{n-1}\right)$$

Example: The IVP from Section 6.1:

$$\mathbf{u}'(t) = \mathbf{A}\cdot\mathbf{u}(t) + \mathbf{b}\cdot e(t) \text{ for } t\in[0,T]$$

may be solved by means of the Heun method using the following algorithm:

$$\hat{\mathbf{u}}_n = \mathbf{u}_{n-1} + h\left[\mathbf{A}\cdot\mathbf{u}_{n-1} + \mathbf{b}\cdot e\left(t_{n-1}\right)\right] \quad \Leftarrow \quad \hat{y}_n = y_{n-1} + hf\left(t_{n-1}, y_{n-1}\right)$$

$$\mathbf{u}_n = \mathbf{u}_{n-1} + h\cdot\frac{\left[\mathbf{A}\cdot\mathbf{u}_{n-1} + \mathbf{b}\cdot e\left(t_{n-1}\right)\right] + \left[\mathbf{A}\cdot\hat{\mathbf{u}}_n + \mathbf{b}\cdot e\left(t_n\right)\right]}{2} \quad \Leftarrow \quad y_n = y_{n-1} + h\cdot\frac{\left[f\left(t_{n-1}, y_{n-1}\right) + f\left(t_n, \hat{y}_n\right)\right]}{2}$$

for $n = 1, 2, \ldots$ and $\mathbf{u}_0 = \mathbf{u}(0)$

4. The Kutta method ($K=3,\ p=3$):

$$y_n = y_{n-1} + \tfrac{1}{6}h\left(f_1 + 4f_2 + f_3\right)$$

with: $f_1 = f\left(t_{n-1}, y_{n-1}\right)$

$$f_2 = f\left(t_{n-1} + \tfrac{1}{2}h, y_{n-1} + \tfrac{1}{2}hf_1\right)$$

$$f_3 = f\left(t_{n-1} + h, y_{n-1} - hf_1 + 2hf_2\right)$$

0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0
1	−1	2	0
	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$

5. The "classical" Runge-Kutta method ($K=4,\ p=4$):

$$y_n = y_{n-1} + \tfrac{1}{6}h\left(f_1 + 2f_2 + 2f_3 + f_4\right)$$

with:

$$f_1 = f\left(t_{n-1}, y_{n-1}\right)$$

$$f_2 = f\left(t_{n-1} + \tfrac{1}{2}h, y_{n-1} + \tfrac{1}{2}hf_1\right)$$

$$f_3 = f\left(t_{n-1} + \tfrac{1}{2}h, y_{n-1} + \tfrac{1}{2}hf_2\right)$$

$$f_4 = f\left(t_{n-1} + h, y_{n-1} + hf_3\right)$$

0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0	0
$\frac{1}{2}$	0	$\frac{1}{2}$	0	0
1	0	0	1	0
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

Overview of implicit single-step methods

1. The backward Euler method ($K=1,\ p=1$):

$$y_n = y_{n-1} + h \cdot f\left(t_n, y_n\right)$$

1	1
	1

2. The family of implicit Butcher (Gauss-Legendre) methods: $p=2K$, stability for $\mathrm{Re}\left[h\lambda\right]<0$.

Example: The implicit Butcher method of order 4 ($K=2,\ p=4$):

$\frac{1}{2}-\frac{\sqrt{3}}{6}$	$\frac{1}{4}$	$\frac{1}{4}-\frac{\sqrt{3}}{6}$
$\frac{1}{2}+\frac{\sqrt{3}}{6}$	$\frac{1}{4}+\frac{\sqrt{3}}{6}$	$\frac{1}{4}$
	$\frac{1}{2}$	$\frac{1}{2}$

3. The family of implicit Radau methods: $p=2K-1$, stability for $\mathrm{Re}\left[h\lambda\right]<0$.

Example: The implicit Radau IIA method of order 3 ($K=2,\ p=3$):

$$y_n = y_{n-1} + h\left(\tfrac{3}{4} f_1 + \tfrac{1}{4} f_2\right)$$

with:

$$f_1 = f\left(t_{n-1} + \tfrac{1}{3}h,\ y_{n-1} + h\left(\tfrac{5}{12} f_1 - \tfrac{1}{12} f_2\right)\right)$$

$$f_2 = f\left(t_{n-1} + h,\ y_{n-1} + h\left(\tfrac{3}{4} f_1 + \tfrac{1}{4} f_2\right)\right)$$

$\frac{1}{3}$	$\frac{5}{12}$	$-\frac{1}{12}$
1	$\frac{3}{4}$	$\frac{1}{4}$
	$\frac{3}{4}$	$\frac{1}{4}$

When applied to the IVP from Section 6.1:

$$\mathbf{u}'(t) = \mathbf{A}\cdot\mathbf{u}(t) + \mathbf{b}\cdot e(t) \text{ for } t \in [0,T]$$

the Radau IIA method of order 3 yields:

$$\mathbf{u}_n = \mathbf{u}_{n-1} + h\left(\tfrac{3}{4}\mathbf{f}_1 + \tfrac{1}{4}\mathbf{f}_2\right) \text{ for } n = 1, 2, \ldots \text{ and } \mathbf{u}_0 = \mathbf{u}(0)$$

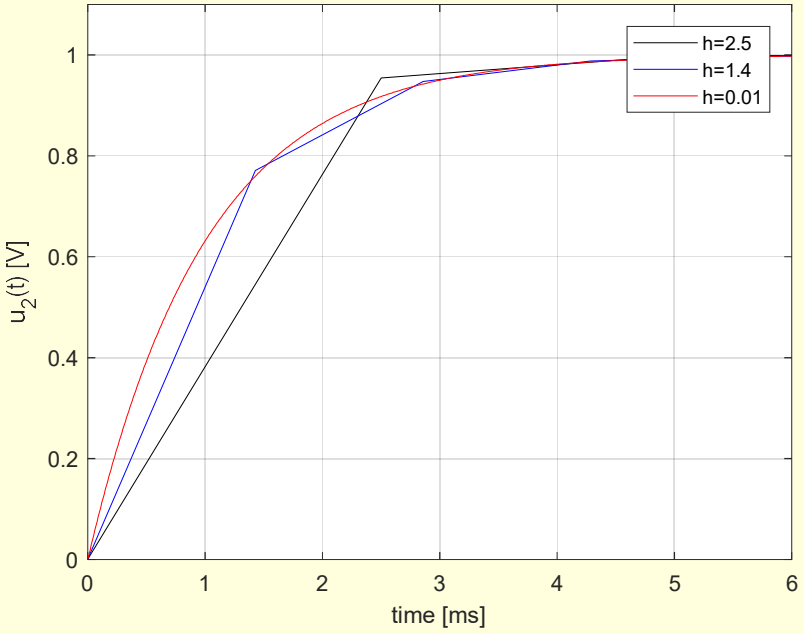
with:

$$\mathbf{f}_1 = \mathbf{A}\cdot\left[\mathbf{u}_{n-1} + h\left(\tfrac{5}{12}\mathbf{f}_1 - \tfrac{1}{12}\mathbf{f}_2\right)\right] + \mathbf{b}\cdot e\left(t_{n-1} + \tfrac{1}{3}h\right)$$

$$\mathbf{f}_2 = \mathbf{A}\cdot\left[\mathbf{u}_{n-1} + h\left(\tfrac{3}{4}\mathbf{f}_1 + \tfrac{1}{4}\mathbf{f}_2\right)\right] + \mathbf{b}\cdot e\left(t_n\right)$$

or:

$$\begin{bmatrix} \mathbf{I} - \frac{5}{12}h\mathbf{A} & \frac{1}{12}h\mathbf{A} \\ -\frac{3}{4}h\mathbf{A} & \mathbf{I} - \frac{1}{4}h\mathbf{A} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{A}\cdot\mathbf{u}_{n-1} + \mathbf{b}\cdot e\left(t_{n-1} + \frac{1}{3}h\right) \\ \mathbf{A}\cdot\mathbf{u}_{n-1} + \mathbf{b}\cdot e\left(t_n\right) \end{bmatrix}$$



4. The family of implicit Lobatto methods: $p = 2K - 2$, stability for $\mathrm{Re}\left[h\lambda\right] < 0$.

Example: The fourth-order implicit Lobatto IIIA method ($K = 3,\ p = 4$):

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{5}{24} & \frac{1}{3} & -\frac{1}{24} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

6.6. Linear multi-step methods

Definition of linear multi-step methods

- The K-step linear method is defined by the following formula:

$$y_n = \sum_{k=1}^{K_\alpha} \alpha_k y_{n-k} + h \sum_{k=\kappa}^{K_\beta} \beta_k \cdot f\left(t_{n-k}, y_{n-k}\right) \text{ with } y_0 = y(0),\ \ldots,\ y_{K-1} = y\left((K-1)h\right)$$

where $\kappa = 0$ or $\kappa = 1$, $K_\alpha \geq 1$, $K_\beta \geq \kappa$, $\max\left\{K_\alpha, K_\beta\right\} = K$.

- The method is called *explicit* if $\kappa = 1$ ($\beta_0 = 0$) since then y_n depends only on y_{n-1}, y_{n-2}, …, y_{n-K} and $f\left(t_{n-1}, y_{n-1}\right)$, $f\left(t_{n-2}, y_{n-2}\right)$,…, $f\left(t_{n-K}, y_{n-K}\right)$.

- The method is called *implicit* if $\kappa = 0$ ($\beta_0 \neq 0$) since then y_n depends also on $f\left(t_n, y_n\right)$; therefore, it is necessary to solve at each step a (nonlinear) algebraic equation:

$$y_n - \beta_0 \cdot f\left(t_n, y_n\right) = F\left(y_{n-1}, y_{n-2}, \ldots\right)$$

where:

$$F\left(y_{n-1}, y_{n-2}, \ldots\right) \equiv \sum_{k=1}^{K_\alpha} \alpha_k y_{n-k} + h \sum_{k=1}^{K_\beta} \beta_k \cdot f\left(t_{n-k}, y_{n-k}\right)$$

Design of linear multi-step methods and its local accuracy

For the sake of convenience, the formula defining a designed multi-step method will be re-written in the form:

$$y_n = \sum_{k=1}^{K_\alpha} \alpha_k y_{n-k} + h \sum_{k=\kappa}^{K_\beta} \beta_k \cdot y'_{n-k}$$

resulting from the substitution $y'_{n-1} \equiv f\left(t_{n-1}, y_{n-1}\right), \ldots, \; y'_{n-K} \equiv f\left(t_{n-K}, y_{n-K}\right)$.

The simplest method for determination of the parameters α_k and β_k, given K_α, κ and K_β, consists in:

- the development of the RHS of the above formula into the Taylor series at t_n;
- the solution of $K_\alpha + K_\beta - \kappa + 1$ linear algebraic equations resulting from setting the coefficients at the term without h to 1 and zeroing the coefficients at the terms with h^1, h^2, ..., $h^{K_\alpha + K_\beta - \kappa}$.

The above procedure provides, as a by-product, an estimate of the local error, *viz*. the term with $h^{K_\alpha + K_\beta - \kappa + 1}$.

Example: The parameters α_1, α_2 and β_1 of the following two-step method for solving IVPs:

$$y_n = \alpha_1 y_{n-1} + \alpha_2 y_{n-2} + h\beta_1 y'_{n-1}$$

satisfy the set of equations:

$$\alpha_1 + \alpha_2 = 1, \quad -\alpha_1 - 2\alpha_2 + \beta_1 = 0, \quad \tfrac{1}{2}\alpha_1 + 2\alpha_2 - \beta_1 = 0$$

resulting from the development of the RHS into the Taylor series:

$$\begin{aligned} y_n = {} & \alpha_1 \left(\dot{y}_n - \dot{y}'_n h + \tfrac{1}{2}\dot{y}''_n h^2 - \tfrac{1}{6}\dot{y}'''_n h^3 + \ldots \right) \\ & + \alpha_2 \left(\dot{y}_n - 2\dot{y}'_n h + 2\dot{y}''_n h^2 - \tfrac{4}{3}\dot{y}'''_n h^3 + \ldots \right) \\ & + h\beta_1 \left(\qquad \dot{y}'_n \quad - \dot{y}''_n h \quad + \tfrac{1}{2}\dot{y}'''_n h^2 - \ldots \right) \end{aligned}$$

The local error may be estimated as follows:

$$\begin{aligned} r_n &\equiv y_n - \dot{y}_n \\ &\cong \underbrace{\left(\alpha_1 + \alpha_2 - 1\right)}_{=0} \dot{y}_n + \underbrace{\left(-\alpha_1 - 2\alpha_2 + \beta_1\right)}_{=0} \dot{y}'_n h + \underbrace{\left(\tfrac{1}{2}\alpha_1 + 2\alpha_2 - \beta_1\right)}_{=0} \dot{y}''_n h^2 + \left(-\tfrac{1}{6}\alpha_1 - \tfrac{4}{3}\alpha_2 + \tfrac{1}{2}\beta_1\right) \dot{y}'''_n h^3 \end{aligned}$$

The solution of the equations is: $\alpha_1 = 0, \alpha_2 = 1, \beta_1 = 2$. Therefore:

$$r_n \cong \alpha \left(-\tfrac{1}{6}\alpha_1 - \tfrac{4}{3}\alpha_2 + \tfrac{1}{2}\beta_1\right) \dot{y}'''_n h^3 = -\tfrac{1}{3}\dot{y}'''_n h^3$$

Stability of multi-step methods

The application of a multi-step method for solving $y'(t) = \lambda \cdot y(t)$ results in the difference equation:

$$y_n = \sum_{k=1}^{K} \alpha_k y_{n-k} + h \cdot \sum_{k=0}^{K} \beta_k \lambda y_{n-k} \quad \xrightarrow{\mathscr{Z}} \quad \sum_{k=0}^{K} \left[\alpha_k z^{-k} Y\left(z^{-1}\right) + h\lambda \beta_k z^{-k} Y\left(z^{-1}\right) \right] = F\left(\dot{y}_0, \dot{y}_{-1}, \dots\right)$$

Its solution in the $\mathscr{Z}$ domain has the form:

$$Y\left(z^{-1}\right) = \frac{F\left(\dot{y}_0, \dot{y}_{-1}, \dots\right)}{\sum_{k=0}^{K} \left(\alpha_k z^{-k} + h\lambda \beta_k z^{-k}\right)}, \text{ where } \alpha_0 \equiv -1, \text{ and some } \alpha_k \text{ and } \beta_k \text{ are zeros}$$

This solution is stable if the zeros of its denominator, *i*.e. of the equation:

$$\sum_{k=0}^{K} \alpha_k z^{-k} + h\lambda \sum_{k=0}^{K} \beta_k z^{-k} = 0$$

are located inside of the unit circle. The border of the region of stability, *i*.e. the set of $h\lambda$ values for which this condition is satisfied, may be determined by solving the above equation with respect to $h\lambda$ and substituting $z = e^{j\varphi}$:

$$h\lambda = -\frac{\sum_{k=0}^{K} \alpha_k e^{j(K-k)\varphi}}{\sum_{k=0}^{K} \beta_k e^{j(K-k)\varphi}} \quad \text{for} \quad \varphi \in [0, 2\pi]$$

Example: The so-called Shichman method (the Gear implicit method of order $p = K = 2$) is defined by the formula:

$$y_n = \tfrac{4}{3} y_{n-1} - \tfrac{1}{3} y_{n-2} + \tfrac{2}{3} h y'_n$$

The local accuracy of this method may be assessed as follows:

$$\begin{aligned} r_n \equiv y_n - \dot{y}_n \cong \tfrac{4}{3}\Big(\dot{y}_n - \quad \dot{y}'_n h + \tfrac{1}{2} \dot{y}''_n h^2 - \tfrac{1}{6} \dot{y}'''_n h^3 + \ldots \Big) \\ - \tfrac{1}{3}\Big(\dot{y}_n - 2\dot{y}'_n h + 2\dot{y}''_n h^2 - \tfrac{4}{3} \dot{y}'''_n h^3 + \ldots \Big) \\ + \tfrac{2}{3}\Big(\qquad \dot{y}'_n h \Big) - \dot{y}_n \end{aligned}$$

$$r_n \cong \tfrac{4}{3} \cdot \Big(-\tfrac{1}{6} \dot{y}'''_n h^3 \Big) - \tfrac{1}{3} \cdot \Big(-\tfrac{4}{3} \dot{y}'''_n h^3 \Big) = \Big(-\tfrac{2}{9} + \tfrac{4}{9} \Big) \dot{y}'''_n h^3 = \tfrac{2}{9} \dot{y}''' h^3$$

The equation for the border of the stability region of the method:

$$y_n = \frac{4}{3}y_{n-1} - \frac{1}{3}y_{n-2} + \frac{2}{3}hy'_n$$

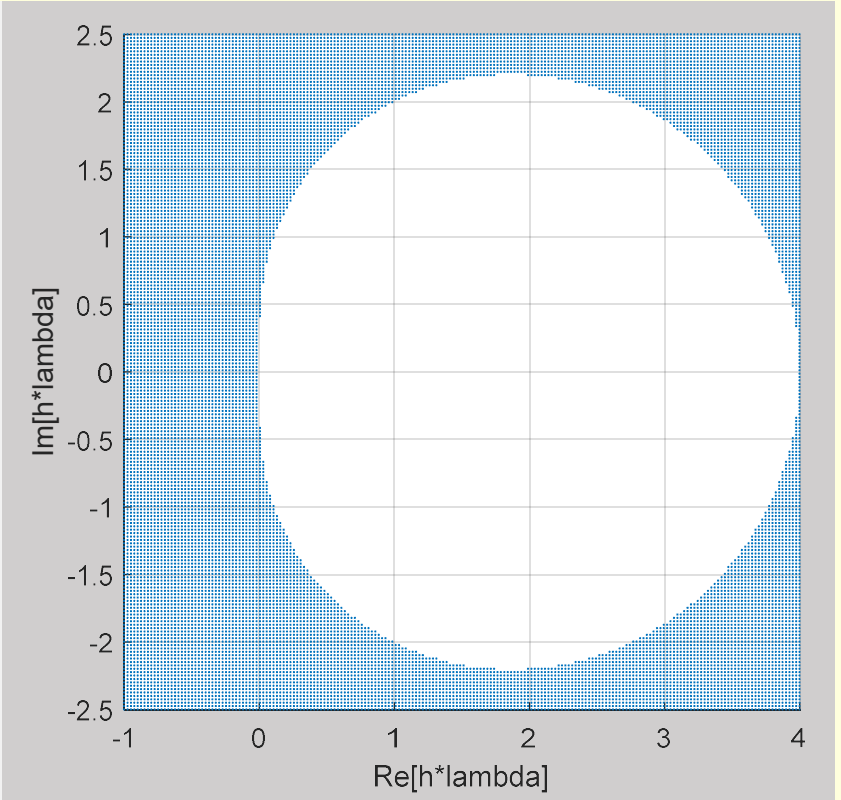
has the form:

$$h\lambda = \frac{1}{2}\left(3 - 4e^{-j\varphi} + e^{-j2\varphi}\right) \text{ for } \varphi \in [0, 2\pi]$$

or:

$$\mathrm{Re}(h\lambda) = \frac{1}{2}\left[3 - 4\cos(\varphi) + \cos(2\varphi)\right] \text{ and } \mathrm{Im}(h\lambda) = \frac{1}{2}\left[4\sin(\varphi) - \sin(2\varphi)\right] \text{ for } \varphi \in [0, 2\pi]$$

```
% REGION OF STABILITY FOR SHICHMAN METHOD.
% NOTATION: h*lambda = x+j*y
% Generation of x-y grid:
  N=200;xx=linspace(-1,4,N);yy=linspace(-2.5,2.5,N);
  [x,y]=meshgrid(xx,yy);hlambda=x+1j*y;
% Representation of inequality in logical matrix:
  syms z hl;eq=(3-2*hl)*z^2-4*z+1==0;zz=solve(eq,z);
  z1=double(subs(zz(1),hl,hlambda));
  z2=double(subs(zz(2),hl,hlambda));
  L=(abs(z1)<=1) & (abs(z2)<=1);
% Plot of points (x,y) that satisfy inequality:
  scatter(x(L),y(L),0.1);grid on
  xlabel('Re[h*lambda]');ylabel('Im[h*lambda]');
 axis('equal');axis([-1 4 -2.5 2.5]);
```



The application of the Shichman method for solving the IVP from Section 6.1:

$$\mathbf{u}'(t) = \mathbf{A} \cdot \mathbf{u}(t) + \mathbf{b} \cdot e(t) \text{ for } t \in [0, T]$$

requires the solution of the following system of linear algebraic equations at each step:

$$\mathbf{u}_n = \tfrac{4}{3}\mathbf{u}_{n-1} - \tfrac{1}{3}\mathbf{u}_{n-2} + \tfrac{2}{3}h \cdot \left[\mathbf{A} \cdot \mathbf{u}_n + \mathbf{b} \cdot e(t_n) \right] \text{ for } n = 1, 2, \ldots \text{ and } \mathbf{u}_0 = \mathbf{u}(0)$$

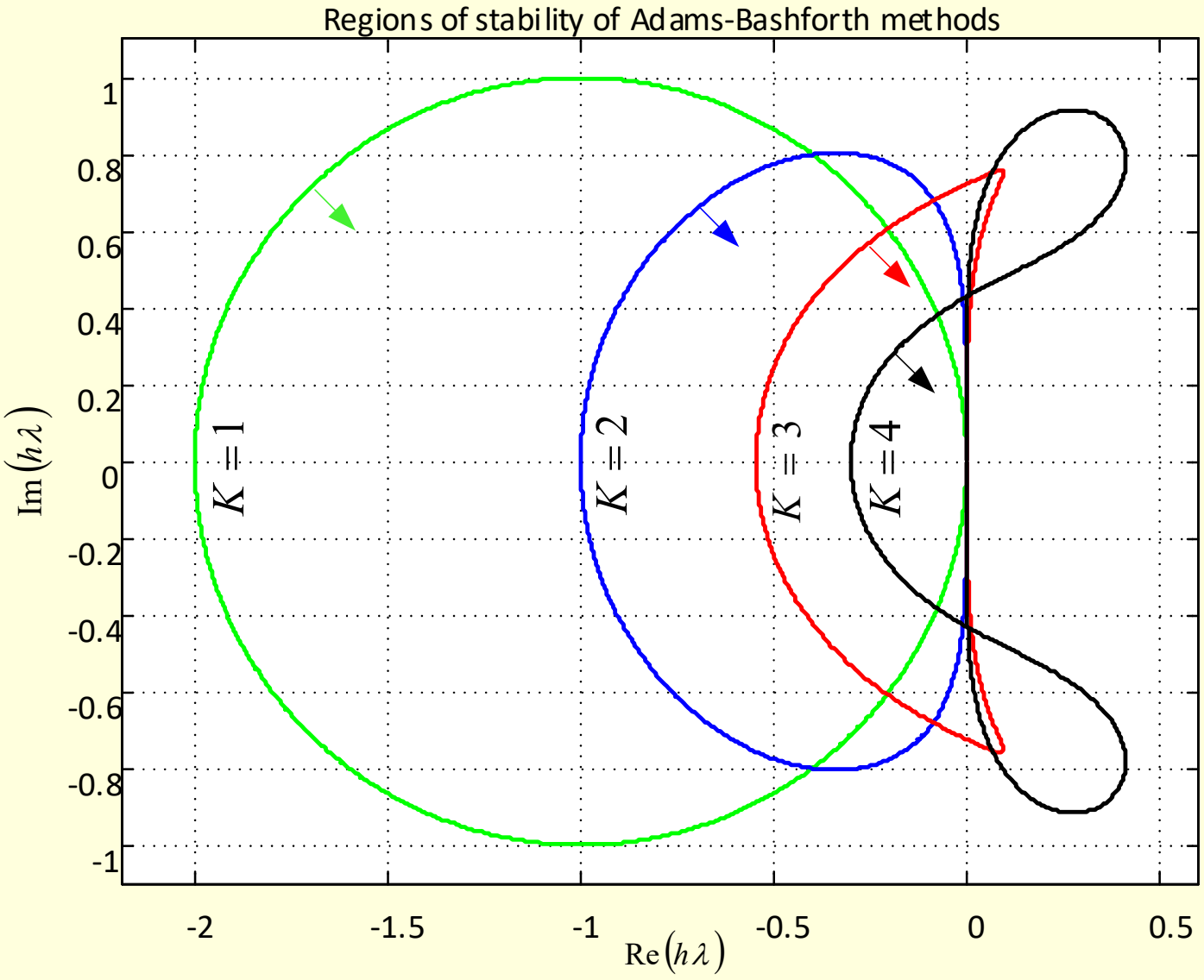
with respect to $\mathbf{u}_n$, *i*.e. of the system:

$$\left(\mathbf{I} - \tfrac{2}{3}h\mathbf{A} \right) \cdot \mathbf{u}_n = \tfrac{4}{3}\mathbf{u}_{n-1} - \tfrac{1}{3}\mathbf{u}_{n-2} + \tfrac{2}{3}h\mathbf{b}e(t_n) \text{ for } n = 1, 2, \ldots \text{ and } \mathbf{u}_0 = \mathbf{u}(0)$$

Explicit Adams methods (Adams-Bashforth methods)

$$y_n = y_{n-1} + h\sum_{k=1}^{K} \beta_k f\left(t_{n-k}, y_{n-k}\right), \quad r_n \cong c_{p+1} y_n^{(p+1)} h^{p+1}$$

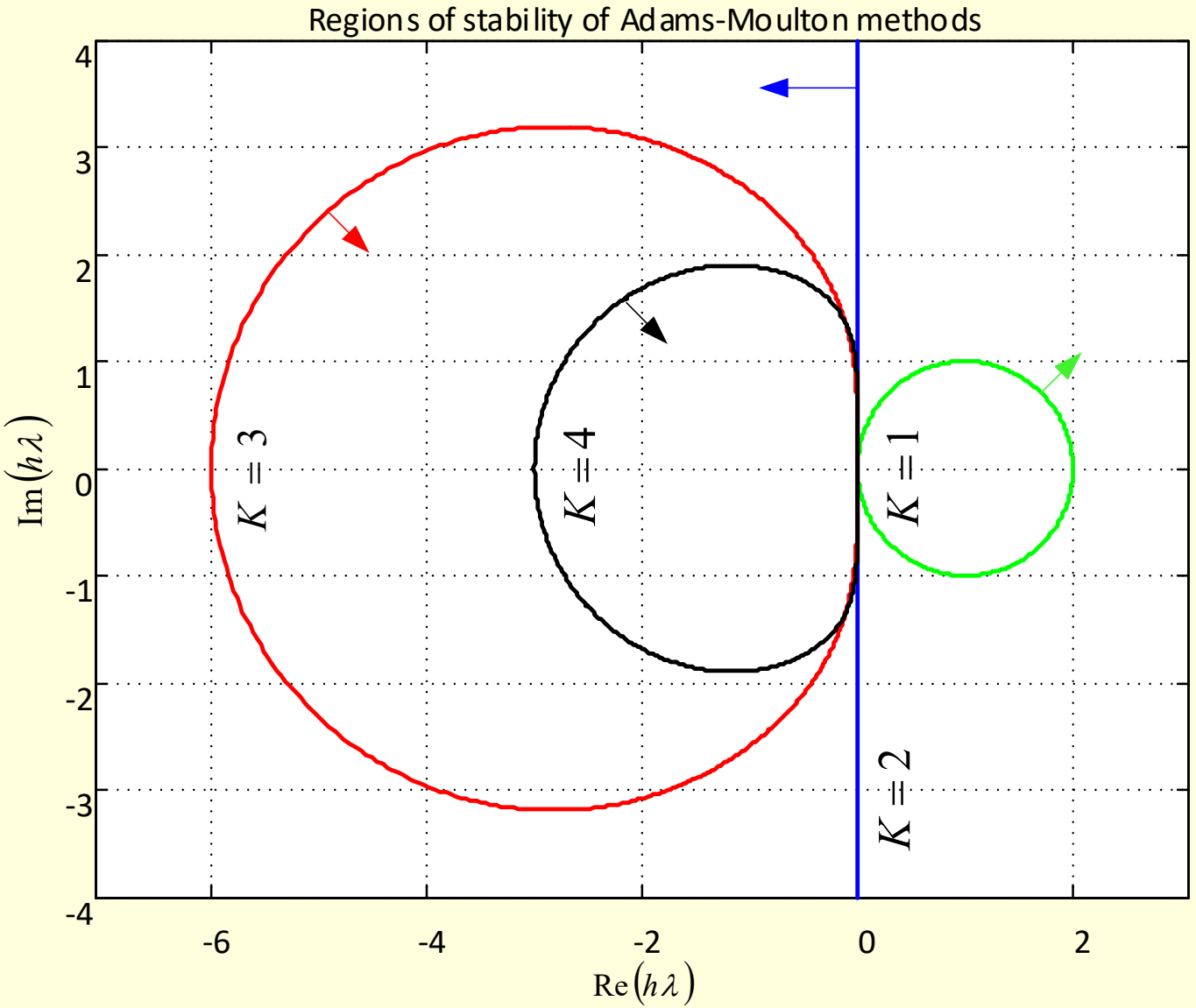
K	p	c_{p+1}	β_1	β_2	β_3	β_4	β_5	β_6	β_7
1	1	$-\frac{1}{2}$	1						
2	2	$-\frac{5}{12}$	$\frac{3}{2}$	$-\frac{1}{2}$					
3	3	$-\frac{3}{8}$	$\frac{23}{12}$	$-\frac{16}{12}$	$\frac{5}{12}$				
4	4	$-\frac{251}{720}$	$\frac{55}{24}$	$-\frac{59}{24}$	$\frac{37}{24}$	$-\frac{9}{24}$			
5	5	$-\frac{95}{288}$	$\frac{1901}{720}$	$-\frac{2774}{720}$	$\frac{2616}{720}$	$-\frac{1274}{720}$	$\frac{251}{720}$		
6	6	$-\frac{19087}{60480}$	$\frac{4277}{1440}$	$-\frac{7923}{1440}$	$\frac{9982}{1440}$	$-\frac{7298}{1440}$	$\frac{2877}{1440}$	$-\frac{475}{1440}$	


Regions of stability of Adams-Bashforth methods
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-2
-1.5
-1
-0.5
0
0.5
Im(hλ)
Re(hλ)
K = 1
K = 2
K = 3
K = 4

Implicit Adams methods (Adams-Moulton methods)

$$y_n = y_{n-1} + h\sum_{k=0}^{K} \beta_k \cdot f\left(t_{n-k}, y_{n-k}\right), \quad r_n \cong c_{p+1} y_n^{(p+1)} h^{p+1}$$

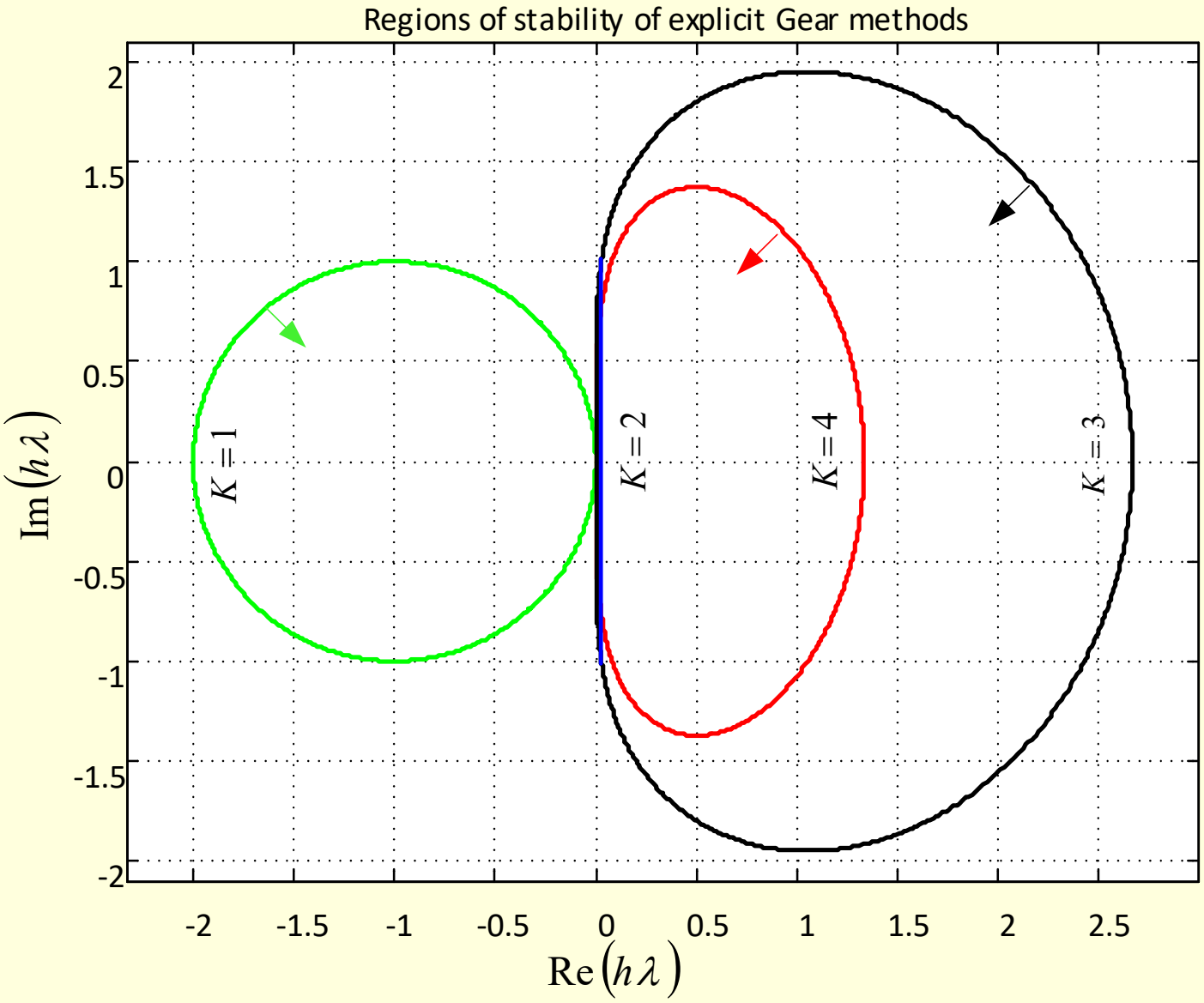
K	p	c_{p+1}	β_0	β_1	β_2	β_3	β_4	β_5	β_6	β_7
0	1	$\frac{1}{2}$	1							
1	2	$\frac{1}{12}$	$\frac{1}{2}$	$\frac{1}{2}$						
2	3	$\frac{1}{24}$	$\frac{5}{12}$	$\frac{8}{12}$	$-\frac{1}{12}$					
3	4	$\frac{19}{720}$	$\frac{9}{24}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$				
4	5	$\frac{3}{360}$	$\frac{251}{720}$	$\frac{646}{720}$	$-\frac{264}{720}$	$\frac{106}{720}$	$-\frac{19}{720}$			
5	6	$\frac{863}{60480}$	$\frac{475}{1440}$	$\frac{1427}{1440}$	$-\frac{798}{1440}$	$\frac{482}{1440}$	$-\frac{173}{1440}$	$\frac{27}{1440}$		
6	7	$\frac{275}{24192}$	$\frac{19087}{60480}$	$\frac{65112}{60480}$	$-\frac{46461}{60480}$	$\frac{37504}{60480}$	$-\frac{20211}{60480}$	$\frac{6312}{60480}$	$-\frac{863}{60480}$	


Regions of stability of Adams-Moulton methods
4
3
2
1
0
-1
-2
-3
-4
-6
-4
-2
0
2
Im(hλ)
Re(hλ)
K = 3
K = 4
K = 1
K = 2

Explicit Gear's methods

$$y_n = \sum_{k=1}^{K} \alpha_k y_{n-k} + h \cdot \beta_1 f\left(t_{n-1}, y_{n-1}\right), \quad r_n \cong c_{p+1} y_n^{(p+1)} h^{p+1}$$

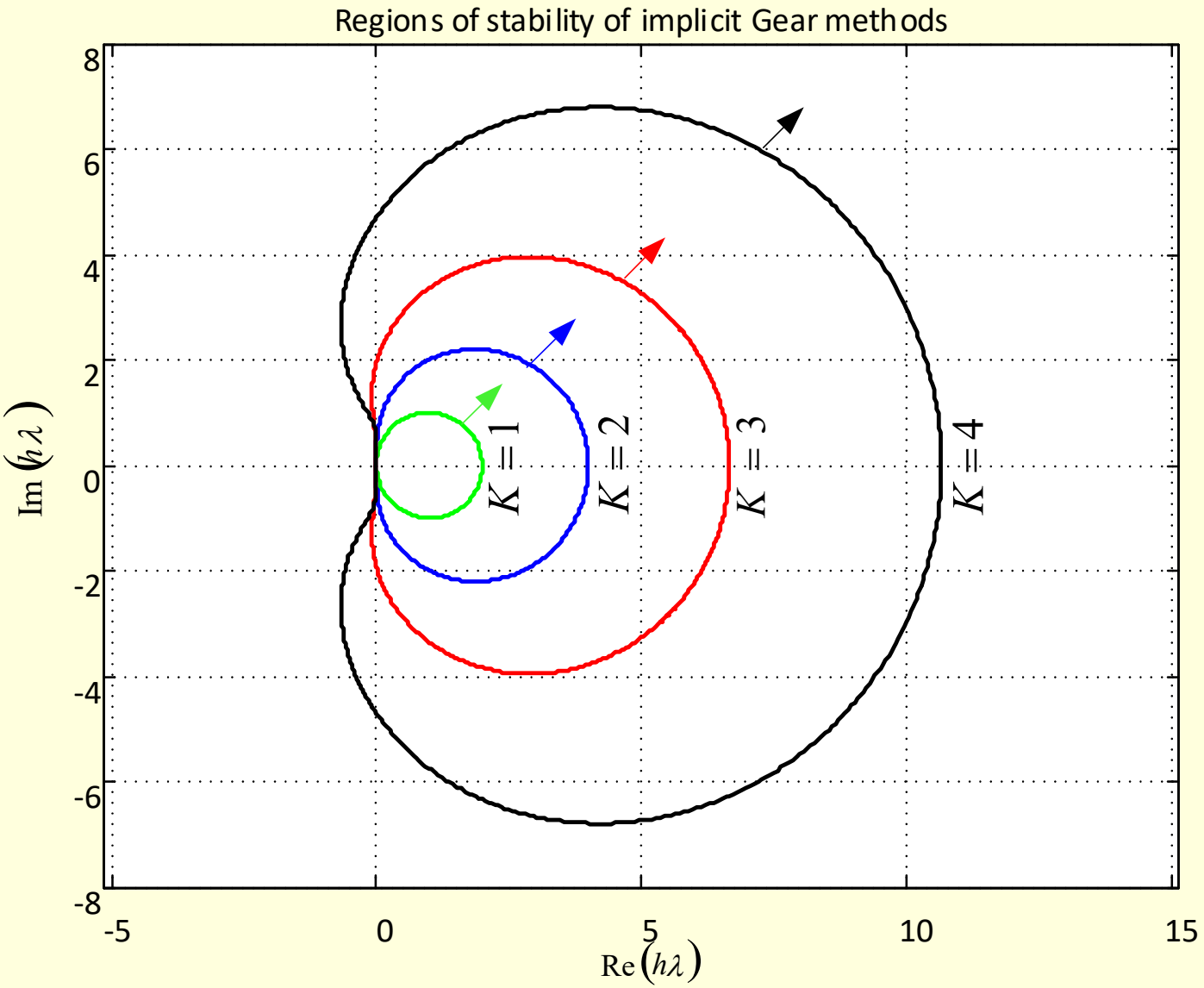
K	p	c_{p+1}	α_1	α_2	α_3	α_4	α_5	α_6	β_1
1	1	$-\frac{1}{2}$	1						1
2	2	$-\frac{1}{3}$	0	1					2
3	3	$-\frac{1}{4}$	$-\frac{3}{2}$	3	$-\frac{1}{2}$				3
4	4	$-\frac{1}{5}$	$-\frac{10}{3}$	6	-2	$\frac{1}{3}$			4
5	5	$-\frac{1}{6}$	$-\frac{65}{12}$	10	-5	$\frac{5}{3}$	$-\frac{1}{4}$		5
6	6	$-\frac{1}{7}$	$-\frac{77}{10}$	15	-10	5	$-\frac{3}{2}$	$\frac{1}{5}$	6


Regions of stability of explicit Gear methods
K = 1
K = 2
K = 4
K = 3
Im(hλ)
Re(hλ)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5

Implicit Gear's methods

$$y_n = \sum_{k=1}^{K} \alpha_k y_{n-k} + h \cdot \beta_0^* f\left(t_n, y_n\right), \quad r_n \cong c_{p+1} y_n^{(p+1)} h^{p+1}$$

K	p	c_{p+1}	α_1	α_2	α_3	α_4	α_5	α_6	β_0
1	1	$\frac{1}{2}$	1						1
2	2	$\frac{2}{9}$	$\frac{4}{3}$	$-\frac{1}{3}$					$\frac{2}{3}$
3	3	$\frac{3}{22}$	$\frac{18}{11}$	$-\frac{9}{11}$	$\frac{2}{11}$				$\frac{6}{11}$
4	4	$\frac{12}{125}$	$\frac{48}{25}$	$-\frac{36}{25}$	$\frac{16}{25}$	$-\frac{3}{25}$			$\frac{12}{25}$
5	5	$\frac{10}{137}$	$\frac{300}{137}$	$-\frac{300}{137}$	$\frac{200}{137}$	$-\frac{75}{137}$	$\frac{12}{137}$		$\frac{60}{137}$
6	6	$\frac{20}{343}$	$\frac{360}{147}$	$-\frac{450}{147}$	$\frac{400}{147}$	$-\frac{225}{147}$	$\frac{72}{147}$	$-\frac{10}{147}$	$\frac{60}{147}$


Regions of stability of implicit Gear methods
8
6
4
2
0
-2
-4
-6
-8
-5
0
5
10
15
Im (hλ)
Re (hλ)
K = 1
K = 2
K = 3
K = 4

6.7. Implementation of linear multi-step methods

Stiff systems of ODEs

A system of linear ODEs $\mathbf{y}' = \mathbf{A}\mathbf{y}$ is called stiff if:

$\frac{\sup\{|\lambda_1|, |\lambda_2|, \ldots\}}{\inf\{|\lambda_1|, |\lambda_2|, \ldots\}} >> 1$, where $\lambda_1, \lambda_2, \ldots$ are eigenvalues of $\mathbf{A}$.

Example: The system of ODEs:

$$\mathbf{y}'(t) = \begin{bmatrix} -667 & 333 \\ 666 & -334 \end{bmatrix} \cdot \mathbf{y}(t)$$

is stiff since $\lambda_1 = -1$ and $\lambda_2 = -1000$.

Its solution for $\mathbf{y}(0) = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$ has the form:

$y_1(t) = e^{-t} - e^{-1000t}$ and $y_2(t) = 2e^{-t} + e^{-1000t}$

The integration of a stiff system of ODEs is problematic due to the following step constraints:

- the stability requirement: $\sup\{|\lambda_1|,|\lambda_2|,...\}h < const. \Rightarrow h < \dfrac{const.}{\sup\{|\lambda_1|,|\lambda_2|,...\}}$;
- the required interval of integration: $T = Nh > \dfrac{1}{\inf\{|\lambda_1|,|\lambda_2|,...\}} \Rightarrow h > \dfrac{1}{N\inf\{|\lambda_1|,|\lambda_2|,...\}}$.

They imply a severe requirement for the number of steps:

$$\frac{1}{N\inf\{|\lambda_1|,|\lambda_2|,...\}} < h < \frac{const.}{\sup\{|\lambda_1|,|\lambda_2|,...\}} \Rightarrow N > \frac{1}{const.}\cdot\frac{\sup\{|\lambda_1|,|\lambda_2|,...\}}{\inf\{|\lambda_1|,|\lambda_2|,...\}}$$

If the forward Euler method is used for integration of the system from the previous example, then:

$$N > \frac{1}{const.}\cdot\frac{\sup\{|\lambda_1|,|\lambda_2|,...\}}{\inf\{|\lambda_1|,|\lambda_2|,...\}} = \frac{1}{2}\cdot\frac{1000}{1} = 500$$

to reproduce the response of the circuits within its one time constant.

If the number of steps needed to find a solution for a given interval $[0,T]$ is increasing, then:

- the number of arithmetic operations needed is increasing,
- the accumulated error due to the rounding errors of operations is increasing.

The above dilemma may be resolved by adaptation of the step size, the adaptation based on the evaluation of the local error.

Step-doubling method for step control

Let $y_n^{(1)}$ be an estimate of the solution obtained for h, and $y_n^{(2)}$ – for $\frac{h}{2}$:

$$y_n^{(1)} \cong y(t_n) + \underbrace{\gamma h^{p+1}}_{\text{the main part of the error}} + O\left(h^{p+2}\right) \quad \text{and} \quad y_n^{(2)} \cong y(t_n) + \underbrace{2 \cdot \gamma \left(\frac{h}{2}\right)^{p+1}}_{\text{the main part of the error}} + O\left(h^{p+2}\right)$$

The coefficient γ may be determined from those equations:

$$y_n^{(2)} - y_n^{(1)} \cong 2 \cdot \gamma \left(\frac{h}{2}\right)^{p+1} - \gamma h^{p+1} \quad \Rightarrow \quad \gamma \cong \frac{y_n^{(2)} - y_n^{(1)}}{\left(2^{-p} - 1\right) h^{p+1}}$$

and used for evaluation of the local error:

$$r_n^{(1)} \equiv y_n^{(1)} - y(t_n) \cong 2^p \frac{y_n^{(2)} - y_n^{(1)}}{2^p - 1} \quad \text{and} \quad r_n^{(2)} \equiv y_n^{(2)} - y(t_n) \cong \frac{y_n^{(2)} - y_n^{(1)}}{2^p - 1}$$

The rules for step selection are as follows:

- if $\left|r_n^{(1)}\right| < r_{\max}$, then double the step h;
- if $\left|r_n^{(1)}\right| \geq r_{\max}$, but $\left|r_n^{(2)}\right| < r_{\max}$, then continue without changing h;
- if $\left|r_n^{(2)}\right| \geq r_{\max}$, the divide the step h by 2.

Prediction-correction methodology

The main limitation of the step-size control, based on the local error assessment, is implied by the stability constraints.

Those constraints are much less severe for implicit methods than for explicit methods, but implicit methods require the solution of a nonlinear algebraic equation at each step.

The computational complexity of this operation may be significantly reduced by using an initial approximation of the solution obtained by means of a corresponding explicit method (prediction phase):

$$y_n^{(0)} = \sum_{k=1}^{K_\alpha} \alpha_k y_{n-k} + h \sum_{k=1}^{K_\beta} \beta_k f\left(t_{n-k}, y_{n-k}\right)$$

and next the equation:

$$y_n - h\beta_0^* f\left(t_n, y_n\right) = \sum_{k=1}^{K_\alpha} \alpha_k^* y_{n-k} + h \sum_{k=1}^{K_\beta} \beta_k^* f\left(t_{n-k}, y_{n-k}\right)$$

is solved with respect to y_n using an iterative method starting at $y_n^{(0)}$ (correction phase).