NUMERICKÉ RECEPTY

Numerické modelování (scientific computing, computational science)
Témata
– otevřena v dávnověku (Newton *1643, Euler *1707, Lagrange *1736, Fourier *1768, Gauss *1777 aj.)
– lineární algebra, aproximace (interpolace), integrování, řešení nelineárních rovnic, hledání extrémů, (Fourierova) transformace, soustavy (obyčejných, parciálních) diferenciálních rovnic
Praxe (černé skříňky)
– knihovny (převážně) pro C a Fortran: (lin. algebra) BLAS/LAPACK, (rychlé) MKL, ACML, (velké) IMSL, NAG aj.
– interaktivní numerické systémy: MATLAB, GNU Octave, Mathematica, Python s NumPy a SciPy aj.
– paralelizace: vláknová paralelizace OpenMP (rozšíření překladačů) pro vícejádrový procesor se sdílenou pamětí
meziprocesová komunikace MPI (knihovna pro zasílání zpráv) pro klastry s distribuovanou pamětí
Meze
– omezená přesnost reálné aritmetiky v hardwaru (při 4/8 B mantisa 24/53 bitů a 7/15 desítkových míst)
problém: odčítání blízkých čísel a (katastrofická) ztráta přesnosti
– asymptotická časová složitost algoritmů vs. výkon reálné aritmetiky na CPU/GPU (~10^4 Kč) ~10^1/10^3 GFlops
problém: algoritmy s časovou složitostí O(n^2), O(n^3), např. lineární algebra
– paměťové nároky 2D a 3D problémů vs. sdílená paměť ~10^1 GB (např. 8 B x 812^3 ~ 4 GB)
problém: nevyhnutelnost (MPI) paralelizace, komunikační úzká hrdla, škálování výkonu (Flops/počet CPU)
Kniha Numerical Recipes (NR, www.nr.com, www.nrbook.com/a/bookcpdf.php, www.nrbook.com/a/bookfpdf.php)
Press W. H. et al., Numerical Recipes in C/Fortran, 1st ed. 1986, 2nd ed. 1992, 3rd ed. 2007
– učebnice se sadou zdrojových kódů (C, Fortran, Pascal jen 1st ed.), široké pokrytí témat, volná PDF (2nd ed.)
GNU Octave
– volně dostupná obdoba MATLABu, efektivní zápis lineární algebry, dobré pokrytí ostatních numerických témat

Mini-algoritmy, NR kap. 5
Řešení kvadratické rovnice, NR kap. 5.6
Běžný vzorec vede při $b^2 \gg 4ac$ k odčítání blízkých čísel ve výrazu $(-b \pm \sqrt{b^2-4ac})$ a ztrátě přesnosti jednoho kořenu, zatímco při q=-(b+sign(b)*sqrt(b*b-4*a*c))/2; x1=q/a, x2=c/q budou oba kořeny přesné (NR 5.6.4-5).
Octave: a=1; b=1e7; c=1; roots([a b c])
Numerické derivování, NR kap. 5.7
Tentýž problém s odčítáním blízkých čísel (NR 5.7.5-6): pro vzorec $f'(x) = [f(x+h) - f(x)]/h$ lze odhadnout optimální volbu kroku $h_{\mathrm{opt}} \sim x\sqrt{\epsilon(f)}$, ovšem i s optimálním krokem je relativní chyba derivace $\epsilon(f') \sim \sqrt{\epsilon(f)}$, tj. z 15 platných číslic zůstane 7. Jsou i přesnější vzorce, ale ztráta přesnosti je při derivování nevyhnutelná.
Octave: x=0:.1:pi*3; y=sin(x); dydx=diff(y)./diff(x); plot(x,y); hold on; plot(x(1:end-1),dydx);
Komplexní aritmetika, NR kap. 5.4
Nutné tehdy, když komplexní aritmetika v jazyce/překladači není nebo má vady (např. přetéká, když nemusí).
Komplexní („C") sčítání/odčítání provedou 2 reálná („R") sčítání/odčítání, $(a+ib) \pm (c+id) = (a \pm c) + i(b \pm d)$.
C-násobení se může provádět pomocí běžných 4 R-násobení a 2 R-sčítání/odčítání nebo pomocí 3 R-násobení a 5 R-sčítání/odčítání, tj. více operací, ale méně (někdy možná pomalých) násobení (NR 5.4.2):
$$(a+ib)(c+id) = (ac-bd) + i(bc+ad) = (ac-bd) + i[(a+b)(c+d) - ac - ad]$$
C-absolutní hodnota zahrnuje vytýkání před odmocninu kvůli možnému přetečení (NR 5.4.4, pozor i na dělení 0):
Octave: kdybychom pro komplexní z nevolali snadno abs(z), tak raději ne: z=1e300; sqrt(real(z)^2+imag(z)^2),
ale: a=real(z); b=imag(z); if abs(a)>abs(b), abs(a)*sqrt(1+(b/a)^2), else abs(b)*sqrt(1+(a/b)^2), end
C-dělení: (NR 5.4.5) nebo $x/y = x \cdot y^{-1}$, kde $y^{-1} = (a+ib)^{-1} = (a-ib)/|a^2+b^2|$, C-odmocňování: (NR 5.4.6-7)
Octave: format long; abs(1+1i), sqrt(2i), ans*ans, ans*ans
Vyčíslení polynomu – Hornerovo schéma, NR kap. 5.3
NR o vyčíslování s umocňováním, p=c(1)*x*x+c(2)*x+c(3): „all persons found guilty of such criminal behavior will be summarily executed", a to pro časovou složitost O(n^2). Řešení s vytýkáním společných výrazů má složitost O(n):
c=[1 0 0]; x=0.5; p=(c(1)*x+c(2))*x+c(3) nebo cyklem n=2; p=c(1); for i=2:n+1, p=p*x+c(i); end; p
Varianta pro společný výpočet P(x) a P'(x): n=2; p=c(1); dp=0; for i=2:n+1, dp=dp*x+p; p=p*x+c(i); end; p, dp
Octave: polyval([1 0 0],1:5), x=0:.1:2; p=[0.5 0 -1]; hold off; plot(x,polyval(p,x))
Vyčíslení řetězového zlomku – Lentzův algoritmus, NR kap. 5.2 (ukázky: functions.wolfram.com/Constants/Pi)
Vyčíslení řetězového zlomku zprava (pomocí cyklu nebo rekurze) bez možnosti efektivního odhadu chyby.
Lentzova metoda: vyčíslení zleva s možností sledování konvergence užitím „numerické Cauchyovy podmínky", tj. zda k předepsanému eps existuje index, od kterého abs(a(p)-a(q))<eps, nebo jen abs(a(p)-a(p-1))<eps.

Soustavy lineárních algebraických rovnic, NR kap. 2

Úlohou je řešit soustavu

$$\mathbf{A}\cdot\mathbf{x}=\mathbf{b} \qquad \text{neboli po řádcích} \qquad \sum_{j=1}^{n} a_{ij}x_j = b_i,\ i=1,\dots,n,$$

kde $\mathbf{A}$ je čtvercová matice soustavy, $\mathbf{x}$ vektor neznámých a $\mathbf{b}$ pravá strana.
Gaussova eliminační metoda sice bývá algebraickou metodou první volby, v praktickém počítání jsou však upřednostněny (algebraicky ekvivalentní) faktorizační metody. Vedle těchto tzv. přímých metod stojí metody iterační, určené pro velké soustavy s řídkými maticemi.
Octave: operátor \ („dělení maticí zleva"): A=[0 1;1 0]; b=[1;0]; x=A\b. Zkouška A*x má vrátit b. Alternativou je funkce linsolve(A,b,opts), umožňující svým třetím argumentem upřesnit typ matice a tím volbu vhodné metody. Obě cesty volají přímé řešiče z osvědčených numerických knihoven pro plné i řídké matice (LAPACK a UMFPACK).

Triviální případy

Snadno řešitelnými případy, co do složitosti algoritmu i časové složitosti řešení, jsou soustavy s diagonální a třídiagonální maticí (časová složitost O(n)) a soustavy s trojúhelníkovými maticemi (časová složitost $O(n^2)$).
Soustava s diagonální maticí, $\mathbf{D}\cdot\mathbf{x}=\mathbf{b}$, $d_{ij}=0$ pro $i\neq j$ (a samozřejmě $d_{ii}\neq 0$)
řešení $x_i = b_i/d_{ii},\ i=1,\dots,n$
Octave: n=5000; D=diag(rand(1,n)); b=ones(n,1); tic; x=D\b; toc % 0.00 s
Dopředná substituce pro soustavu s dolní (lower) trojúhelníkovou maticí, $\mathbf{L}\cdot\mathbf{x}=\mathbf{b},\ l_{ij}=0$ pro $i<j$
neznámé x_i se postupně získávají řešením rovnic směrem od první k poslední,

$$x_1 = b_1/l_{11},\ x_i = \left(b_i - \textstyle\sum_{j=1}^{i-1} l_{ij}x_j\right)/l_{ii},\ i=2,\dots,n$$

Octave: n=5000; L=tril(rand(n))+n*eye(n); b=ones(n,1); tic; x=L\b; toc; % 0.31 s
Zpětná substituce pro soustavu s horní (upper) trojúhelníkovou maticí, $\mathbf{U}\cdot\mathbf{x}=\mathbf{b},\ u_{ij}=0$ pro $i>j$
neznámé x_i se postupně získávají řešením rovnic směrem od poslední k první (NR 2.2.4),

$$x_n = b_n/u_{nn},\ x_i = \left(b_i - \textstyle\sum_{j=i+1}^{n} u_{ij}x_j\right)/u_{ii},\ i=n-1,\dots,1$$

Octave: n=5000; U=triu(rand(n))+n*eye(n); b=ones(n,1); tic; x=U\b; toc; % 0.31 s
opts.UT=true; tic; x=linsolve(U,b,opts); toc; % 0.16 s
Soustava s třídiagonální maticí, s nulami všude vyjma prvků $a_{i-1,i}$, a_{ii} a $a_{i+1,i}$
dvouprůchodově: eliminace subdiagonálních prvků, zpětná substituce pro matici se superdiagonálou
Octave: n=5000; A=triu(tril(rand(n),1),-1)+eye(n); b=ones(n,1); tic; x=A\b; toc; % 3.79 s
As=sparse(A); tic; xs=As\b; toc; max(abs(x-xs)) % 0.001 s
V ukázkách pro Octave vidíme, že diagonální matici a v podstatě i trojúhelníkové matice operátor \ rozezná a úlohu řeší efektivně O(n) nebo $O(n^2)$ algoritmem; třídiagonální matici nerozeznal a strávil s ní $O(n^3)$ času jako s plnou maticí. Tehdy má smysl aktivovat interní řešič pro řídké matice, zde převodem matice soustavy voláním sparse.

Gaussova eliminační metoda, NR kap. 2.1–2.2

Nulováním nediagonálních, resp. poddiagonálních prvků matice soustavy pomocí odčítání vhodných násobků jiných řádků (eliminačními transformacemi) se kráčí ke tvaru

$$\mathbf{D}\cdot\mathbf{x}=\mathbf{b}', \qquad \text{resp.}\quad \mathbf{U}\cdot\mathbf{x}=\mathbf{b}'',$$

kde $\mathbf{D}$ je diagonální matice a $\mathbf{U}$ horní trojúhelníková matice. Časová složitost eliminace je $O(n^3)$, neboť n^2-n, resp. $(n^2-n)/2$ prvků se eliminuje odčítáním n-prvkových řádků, časová složitost řešení pro $\mathbf{D}$, resp. $\mathbf{U}$ uvedena výše. Pro numerickou stabilitu je nezbytná sloupcová (záměna jen sloupců), řádková (záměna jen řádků) nebo úplná pivotace tak, aby v odčítané rovnici prvek (pivot) s největší absolutní hodnotou (normalizovaných rovnic) ležel na diagonále.

Faktorizační metody, NR kap. 2.3, 2.6, 2.9, 2.10

LU rozklad

Krok „A": Nalezne se rozklad matice soustavy na součin dvou faktorů, $\mathbf{A}=\mathbf{L}\cdot\mathbf{U}$, kde $\mathbf{L}$ je dolní (lower) trojúhelníková (zde s jedničkami na diagonále, $l_{ii}=1$) a $\mathbf{U}$ horní (upper) trojúhelníková matice. Úloha je vlastně soustavou n^2 nelineárních rovnic pro n^2 neznámých $l_{ij},\ u_{ij}$,

$$\begin{pmatrix} 1 & 0 & 0 \\ & \ddots & 0 \\ l_{ik} & & 1 \end{pmatrix} \cdot \begin{pmatrix} u_{11} & & u_{kj} \\ 0 & \ddots & \\ 0 & 0 & u_{nn} \end{pmatrix} = \begin{pmatrix} & a_{ij} & \\ & & \end{pmatrix},$$

a je přímočará při vhodné volbě řazení rovnic. Croutův algoritmus „po sloupcích zleva" (NR obr. 2.3.1) s časovou složitostí O(n³) zajišťuje, že v každé rovnici $\sum_{k=1}^{\min(i,j)} l_{ik}u_{kj} = a_{ij}$ se objeví právě jeden dosud nespočtený prvek matic $\mathbf{L}$ a $\mathbf{U}$. Hledané prvky tak lze snadno získat v pořadí $u_{11},\ l_{21},\ldots,\ l_{n1},\ u_{12},\ u_{22},\ldots,\ l_{n2}$, atd.
Krok „b": Soustavu $\mathbf{A}\cdot\mathbf{x} = (\mathbf{L}\cdot\mathbf{U})\cdot\mathbf{x} = \mathbf{L}\cdot(\mathbf{U}\cdot\mathbf{x}) = \mathbf{b}$ pak lze řešit jako dvě soustavy s trojúhelníkovými maticemi,

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

každou s časovou složitostí O(n²). Vektor $\mathbf{b}$ potřeba až v kroku „b".
Pro determinant platí $\det\mathbf{A} = \det\mathbf{L}\cdot\det\mathbf{U} = \prod_{i=1}^{n} u_{ii}$, neboť determinanty trojúhelníkových matic jsou snadné.
Pásové matice o šířce pásu $m = m_1 + 1 + m_2$, kde m_1 je počet nenulových prvků ve sloupci pod diagonálou (či vlevo od) a m_2 totéž nad diagonálou, mají vlastní variantu LU rozkladu s časovou složitostí O(n), závislou ovšem také (nelineárně) na m. LU rozklad nezachovává pásovost v maticích $\mathbf{L}$ a $\mathbf{U}$ (jejich nenulových prvků je více než nenulových prvků $\mathbf{A}$). Varianta ILU (incomplete LU) ignorující nenulové prvky mimo pás se užívá pro zvýšení efektivity iteračních metod.
Choleského rozklad $\mathbf{A} = \mathbf{L}\cdot\mathbf{L}^T$ („$\mathbf{L}$ je odmocninou $\mathbf{A}$") s poloviční časovou složitostí proti LU rozkladu lze použít pro symetrické pozitivně definitní matice, metoda SVD (singular value decomposition, $\mathbf{A} = \mathbf{U}\cdot\mathbf{W}\cdot\mathbf{V}^T$) je vhodná pro špatně podmíněné úlohy nebo přeurčené a podurčené problémy s obdélníkovými maticemi.
Knihovny přímých řešičů: LAPACK je volně dostupnou a kvalitně optimalizovanou knihovnou s přímými řešiči soustav s plnými, pásovými a symetrickými maticemi v reálném a komplexním oboru a řešiči pro vlastní čísla a vektory. Existuje v řadě variant, někdy i komerčních (CULA pro GPU, ScaLAPACK pro klastry s MPI), je součástí větších knihoven (MKL, ACML ad.). UMFPACK je jednou z volně dostupných knihoven s přímými řešiči pro řídké matice. Obě knihovny lze volat z C a Fortranu a jsou užity v MATLABu i Octavu pro implementaci operátoru \ a funkce linsolve.

Iterační metody

Jsou užívány pro řešení velkých soustav ($n \gg 10^3$), které ovšem mají řídkou matici (s mnoha nulovými prvky). Soustava $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$ se upraví na tvar

$$\mathbf{x}^{(k+1)} = \mathbf{H}\cdot\mathbf{x}^{(k)} + \mathbf{g},$$

kde $\mathbf{H}$ se nazývá iterační maticí; iterace konvergují k řešení $\mathbf{x}$ tehdy, je-li spektrální poloměr $\mathbf{H}$ (maximum z vlastních čísel v absolutní hodnotě) menší než 1. Prakticky pak úspěch (časová složitost) spočívá v efektivní realizaci součinu iterační matice s obecným vektorem, $\mathbf{H}\cdot\mathbf{v}$, a na kvalitě odhadu $\mathbf{x}^{(0)}$.
Otázka volby $\mathbf{x}^{(0)}$ je někdy snadná, např. při řešení úlohy vyvíjející se v čase lze iterace v nové časové hladině startovat konečným řešením z předchozí časové hladiny, $\mathbf{x}^{(0)}(t+dt) = \mathbf{x}^{(\mathrm{final})}(t)$, jindy může stačit řešení $\mathbf{D}\cdot\mathbf{x}^{(0)} = \mathbf{b}$ nebo řešení pro ILU rozklad matice $\mathbf{A}$.
Rychlost konvergence iteračních metod se často zlepšuje zaváděním předpodmiňujících matic (preconditioners) $\tilde{\mathbf{A}}$, které se podobají matici $\mathbf{A}$, ale lze je snáze invertovat (např. diagonála $\mathbf{A}$, třídiagonální pás z $\mathbf{A}$ nebo původní $\mathbf{A}$ invertované ILU rozkladem); řeší se tak vlastně soustava

$$(\tilde{\mathbf{A}}^{-1}\cdot\mathbf{A})\cdot\mathbf{x} = \tilde{\mathbf{A}}^{-1}\cdot\mathbf{b}.$$

Efektivní iterační řešiče a předpodmiňující metody jsou součástí řady knihoven, často vyvíjených pro klastry s MPI. Za nejrychlejší iterační metodu se považuje multigridová (MG) metoda, populární jsou varianty metody sdružených gradientů (CG). Dvě následující metody jsou jako samostatné metody spíše učebnicovými ukázkami, vyskytující se však frekventovaně jako komponenty složitějších metod.

Jacobiova a Gaussova-Seidelova metoda

(Jacobi) Pro rozklad $\mathbf{A} = \mathbf{L}' + \mathbf{D} + \mathbf{U}'$, kde $\mathbf{L}'$ a $\mathbf{U}'$ jsou trojúhelníkové matice s nulami na diagonále, lze v řešené soustavě vše kromě $\mathbf{D}\cdot\mathbf{x}$ převést na pravou stranu a doplněním iteračních indexů získat iterační vzorec

$$\mathbf{D}\cdot\mathbf{x}^{(k+1)} = -(\mathbf{L}' + \mathbf{U}')\cdot\mathbf{x}^{(k)} + \mathbf{b}.$$

Na pořadí vyčíslování prvků vektoru $\mathbf{x}^{(k+1)}$ nezáleží, metoda je vhodná pro paralelizaci.
(Gauss-Seidel) Převodem $\mathbf{U}'\cdot\mathbf{x}$ na pravou stranu a doplněním iteračních indexů se získá soustava s dolní trojúhelníkovou maticí řešitelná dopřednou substitucí, tedy sériově, od prvního prvku k poslednímu,

$$(\mathbf{L}' + \mathbf{D})\cdot\mathbf{x}^{(k+1)} = -\mathbf{U}'\cdot\mathbf{x}^{(k)} + \mathbf{b}.$$

Obě metody konvergují pro ostře diagonálně dominantní matice, Gaussova-Seidelova metoda i pro symetrické pozitivně definitní matice, zkusit je lze i v jiných případech. Rychlost konvergence obou metod je však malá.
Octave: n=5; A=rand(n)+n*eye(n); b=A*(1:n)'; L=tril(A,-1); Dvec=diag(A); U=triu(A,1); D=diag(Dvec);
x=zeros(n,1); (opakovat do konvergence) x=D\(-(L+U)*x+b); x' (5 platných míst po 11 iteracích)
x=zeros(n,1); (opakovat do konvergence) x=(L+D)\(-U*x+b); x' (5 platných míst po 5 iteracích)

Polynomická aproximace, NR kap. 3

Úlohou je nalézt aproximační funkci, která přesně nebo přibližně vystihuje tabelované hodnoty $(x_i, f_i),\ i = 0, \ldots, n,$ $x_i \neq x_k$ pro $i \neq k$, nebo která nahrazuje složitější funkci, kterou lze tabelovat. Vyčíslování aproximační funkce v intervalu $< x_0, x_n >$ se nazývá interpolace, mimo tento interval extrapolace. První a frekventovanou volbou pro aproximační funkci jsou polynomy. Úlohu požadující přesně vystihnout tabelované hodnoty (interpolační podmínky v interpolačních uzlech) pomocí jediného polynomu řeší polynomická interpolace, jindy se aproximuje po částech polynomy nižšího stupně spojitými (až hladkými) v interpolačních uzlech, jindy se jen minimalizují odchylky aproximované a aproximační funkce a hovoří se o polynomické regresi. Tyto aproximační úlohy vedou vesměs k řešení soustav lineárních algebraických rovnic; Octave to zakrývá funkcemi polyfit a interp1.

Polynomická interpolace jedním polynomem, NR kap. 3.1, 3.5
Aproximační funkce je $P_n(x)$, polynom obecně n-tého stupně pro $n+1$ interpolačních podmínek. Existuje právě jeden takový polynom, zapsat jej je možné více způsoby.

Lagrangeův interpolační polynom (NR 3.1.1)
Lagrangeův interpolační polynom $L_n(x) = f_0 l_n^{(0)}(x) + \cdots + f_n l_n^{(n)}(x)$ je tvořen váženým součtem elementárních Lagrangeových polynomů $l_n^{(i)}(x)$ stupně n, splňujících $l_n^{(i)}(x_i) = 1$ a $l_n^{(i)}(x_k) = 0$ pro $i \neq k$, interpolační podmínky jsou tak zjevně splněny. Polynomy $l_n^{(i)}(x)$ lze (opět zjevně) zapsat jako normované součiny kořenových činitelů,
$l_n^{(i)}(x) = [(x-x_0)(x-x_1)...(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)]/[(x_i-x_0)(x_i-x_1)...(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)]$.

Polynom ve standardním tvaru (NR 3.5.2)
Pro interpolační polynom $P_n(x) = c_0 + c_1 x + \cdots + c_n x^n$ tvoří interpolační podmínky $P_n(x_i) = f_i$ soustavu $n+1$ lineárních algebraických rovnic pro koeficienty c_i, $\mathbf{V} \cdot \mathbf{c} = \mathbf{f}$,

$$\begin{pmatrix} 1 & x_0 & \ldots & x_0^n \\ 1 & x_1 & \ldots & x_1^n \\ \vdots & & & \vdots \\ 1 & x_n & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix}.$$

Pro speciální tvar (Vandermondovy) matice soustavy existuje úsporná metoda s časovou složitostí O(n²).
Octave: n=3; x=[1;2;3;4]; f=[1;-1;1;-1]; A=[ones(n+1,1),x,x.^2,x.^3]; p=flipud(A\f) nebo A=vander(x); p=A\f
nebo p=polyfit(x,f,n) a kreslení: xi=0:.1:5; plot(xi,polyval(p,xi),x,f,'o')

Newtonův interpolační polynom
Hledání koeficientů polynomu $N_n(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + \ldots + a_n(x-x_0)\ldots(x-x_{n-1})$ vede k soustavě $n+1$ lineárních algebraických rovnic s dolní trojúhelníkovou maticí, $\mathbf{L} \cdot \mathbf{a} = \mathbf{f}$,

$$\begin{pmatrix} 1 & 0 & \ldots & 0 \\ 1 & x_1 - x_0 & \ldots & 0 \\ \vdots & & & \vdots \\ 1 & x_n - x_0 & \ldots & (x_n-x_0)(x_n-x_1)\ldots(x_n-x_{n-1}) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix}.$$

Soustava je snadno řešitelná dopřednou substitucí s časovou složitostí O(n²). Přidávání interpolačních podmínek nevede ke změně dosud získaných koeficientů a_i (vlastnost permanence).
Octave: n=3; A=zeros(n+1); for i=0:n, for j=0:i, A(i+1,j+1)=prod(x(i+1)-x(1:j)); end, end; newton=A\f
Koeficienty Newtonova polynomu lze také zapsat pomocí poměrných diferencí k-tého řádu: $a_k = f[x_0, x_1, \ldots, x_k]$, kde $f[x_i] = f(x_i)$ a $f[x_i, \ldots, x_{i+k}] = (f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}])/(x_{i+k} - x_i)$. Z toho lze na ekvidistantní síti s krokem h odvodit explicitní zápis Newtonova polynomu: $N_n(t) = \sum_{k=0}^{n} \binom{t}{k} \Delta^k f_0 = \sum_{k=0}^{n} (-1)^k \binom{-t}{k} \nabla^k f_n$, kde $t = \frac{x-x_0}{h}$ v prvním a $t = \frac{x-x_n}{h}$ v druhém případě, $\binom{t}{0} = 1$, $\binom{t}{k} = \frac{t(t-1)\ldots(t-k+1)}{k!}$ a dopředné a zpětné diference k-tého řádu jsou definovány vztahy: $\Delta^k f_i = \Delta^{k-1} f_{i+1} - \Delta^{k-1} f_i$, $\nabla^k f_i = \nabla^{k-1} f_i - \nabla^{k-1} f_{i-1}$, $\Delta^0 f_i = \nabla^0 f_i = f_i$.

Nevillův algoritmus (NR 3.1.3) umožňuje vyčíslit hodnotu $P_n(x)$ pro dané x bez hledání koeficientů polynomu. Časová složitost je stále O(n²).
Hermitův interpolační polynom H_{2n+1} splňuje interpolační podmínky kladené na funkční hodnoty a první derivace.
Ekvidistantní sítě: interpolace polynomem vyššího stupně je na ekvidistantních sítích nevhodná pro přílišné oscilace, zvláště u krajů (Rungeův fenomenon). Aproximuje se proto buď po částech polynomy nízkého stupně nebo na speciálních nerovnoměrných sítích tvořených kořeny ortogonálních (Čebyševových, Lagrangeových) polynomů.

Octave: hat=@(x) 1-sign(x).*x; (polynomická aproximace na ekvidistantní síti a na uzlech OG polynomů)
n=10; x=-1:2/n:1; f=hat(x); peq=polyfit(x,f,n); hold off; xi=-1:.01:1; plot(xi,polyval(peq,xi),'-r',x,f,'dr')
n=11; x=cos(pi*((1:n)-0.5)/n); f=hat(x); pog=polyfit(x,f,n); hold on; plot(xi,polyval(pog,xi),'--b',x,f,'ob')

Polynomická interpolace po částech

Po částech lineární interpolace

Aproximační funkcí jsou po částech lineární polynomy spojité v interpolačních uzlech, jako při spojování podle pravítka. Časová složitost nalezení aproximované hodnoty odpovídá časové složitosti nalezení příslušného podintervalu, ideálně O(1) až O(log n).
Octave: n=20; x=linspace(0,1,n); f=rand(1,n); dx=.001; xi=0:dx:1; fi=interp1(x,f,xi,'linear'); plot(xi,fi,x,f,'o')
Gnuplot: file='data'; plot file with linespoints

Interpolace kubickými spliny, NR kap. 3.3

Aproximační funkcí jsou po částech kubické polynomy spojité v interpolačních uzlech až do druhých derivací, jako při spojování podle křivítka. Funkce je na n podintervalech definována pomocí 4n neznámých koeficientů; požadavek splnění interpolačních podmínek, spojitosti nulté, první a druhé derivace a 2 dodatečných podmínek tvoří 4n rovnic, redukovatelných na soustavu n lineárních rovnic s třídiagonální maticí. Časová složitost sestrojení kubických splinů je tak lineární, O(n).
Octave: n=20; x=linspace(0,1,n); f=rand(1,n); dx=.001; xi=0:dx:1; fi=interp1(x,f,xi,'spline'); plot(xi,fi,x,f,'o')
Gnuplot: file='data'; plot file smooth csplines, file with points

Polynomická regrese metodou nejmenších čtverců

Aproximační funkcí nechť je lineární kombinace $m+1$ bázových funkcí, např. polynomů $x^0,\dots,x^m$. Úmyslem není splnit $n+1$ (obvykle $n \gg m$) interpolačních podmínek přesně (třeba pro jejich nepřesnost), ale minimalizovat odchylku hodnot aproximované a aproximační funkce. Odchylka se ve smyslu metody nejmenších čtverců definuje jako (vážený) součet druhých mocnin („čtverců") rozdílů hodnot aproximované a aproximační funkce (L_2 aproximace). Minimalizace odchylky vzhledem k $m+1$ koeficientům aproximační funkce vyžaduje podle nich odchylku derivovat, což vede k soustavě $m+1$ lineárních (tzv. normálních) rovnic pro koeficienty, $\mathbf{G}\cdot\mathbf{c}=\mathbf{b}$. Při volbě lineárně nezávislých bázových funkcí je (Gramova) matice soustavy symetrická a pozitivně definitní, pro ortogonální bázové funkce je dokonce diagonální. Případu s polynomickými bázovými funkcemi se říká polynomická regrese, při $m=1$ lineární regrese (proložení přímky).
Octave: m=1; n=3; x=[1;2;3;4]; f=[0;3;4;2]; p=polyfit(x,f,m), xi=0:.1:5; plot(xi,polyval(p,xi),x,f,'o')
m=2; p=polyfit(x,f,m), plot(xi,polyval(p,xi),x,f,'o')
Gnuplot: f(x)=a*x+b; fit f(x) 'xf.dat' via a,b; plot 'xf.dat',f(x)

Odvození pro aproximační funkci $\varphi(x)=\sum_{j=0}^m c_j\varphi_j(x)$. Minimalizuje se výraz $\sum_{k=0}^n \omega_k\left[\varphi(x_k)-f_k\right]^2$ vzhledem k c_i:

$$\frac{\partial}{\partial c_i}\sum_{k=0}^n \omega_k\left[\sum_{j=0}^m c_j\varphi_j(x_k)-f_k\right]^2 = \sum_{k=0}^n 2\omega_k\varphi_i(x_k)\left[\sum_{j=0}^m c_j\varphi_j(x_k)-f_k\right]=0,\quad i=0,\dots,m,$$

$$\sum_{j=0}^m\left[\sum_{k=0}^n \omega_k\varphi_i(x_k)\varphi_j(x_k)\right]c_j=\sum_{k=0}^n \omega_k f_k\varphi_i(x_k),\quad i=0,\dots,m.$$

Pro polynomy $\varphi_j(x)=x^j$ a $\omega_k=1$

$$\begin{pmatrix} \sum_k 1 & \sum_k x_k & \dots & \sum_k x_k^m \\ \sum_k x_k & \sum_k x_k^2 & \dots & \sum_k x_k^{m+1} \\ \vdots & & & \vdots \\ \sum_k x_k^m & \sum_k x_k^{m+1} & \dots & \sum_k x_k^{2m} \end{pmatrix}\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_m \end{pmatrix}=\begin{pmatrix} \sum_k f_k \\ \sum_k f_k x_k \\ \vdots \\ \sum_k f_k x_k^m \end{pmatrix}.$$

Pro ortogonální Čebyševovy polynomy $\varphi_j(x)=T_j(x)=\cos(j\arccos x)$, $\omega_k=1$ a uzly x_k v kořenech $T_{n+1}(x)$

$$\begin{pmatrix} m+1 & 0 & \dots & 0 \\ 0 & \frac{m+1}{2} & \dots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \dots & \frac{m+1}{2} \end{pmatrix}\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_m \end{pmatrix}=\begin{pmatrix} \sum_k f_k T_0(x_k) \\ \sum_k f_k T_1(x_k) \\ \vdots \\ \sum_k f_k T_m(x_k) \end{pmatrix},\quad x_k=\cos[\pi(k+\tfrac12)/(n+1)],\ k=0,\dots,n,$$

neboť ortogonalita zde znamená $\sum_k T_i(x_k)T_j(x_k)=0$ pro $i\neq j$.

Numerické integrování, NR kap. 4

Na polynomické aproximaci integrandu jsou založeny kvadraturní vzorce neboli vážené součty hodnot integrandu v interpolačních uzlech. Newtonovy-Cotesovy kvadraturní vzorce jsou formulovány na ekvidistantních sítích, kde interpolační polynomy vyšších stupňů příliš oscilují; používají se proto složené N.-C. vzorce, odpovídající polynomické interpolaci po částech (pravidlo lichoběžníkové, Simpsonovo ad.). Přesnější Gaussovy kvadraturní vzorce jsou definovány na sítích tvořených kořeny ortogonálních polynomů. Nejen hazardní hráče přitahuje méně přesná, ale účinná metoda Monte Carlo. Octave poskytuje integrátor quad a několik dalších podobného jména.

Obecný kvadraturní vzorec

Odvozuje se analytickým integrováním Lagrangeova polynomu $L_n(x)$ interpolujícího integrand na $n+1$ uzlech,

$$\int_{x_0}^{x_n} f(x)\,dx \approx I_n \equiv \int_{x_0}^{x_n} L_n(x)\,dx = \int_{x_0}^{x_n} \sum_{i=0}^{n} f_i l_n^{(i)}(x)\,dx = \sum_{i=0}^{n} w_i f_i, \quad \text{kde} \quad w_i = \int_{x_0}^{x_n} l_n^{(i)}(x)\,dx, \quad f_i = f(x_i).$$

Newtonovy-Cotesovy kvadraturní vzorce, NR kap. 4.1

Na intervalu $< a, b >$ se zavede ekvidistantní síť $x_i = x_0 + ih$, kde $i = 0, \ldots, n$. Pokud krajní body intervalu jsou interpolačními uzly, $a = x_0$ a $b = x_n$, je krok v síti $h = (b-a)/n$ a výsledný kvadraturní vzorec se nazývá uzavřený, jinak jde o vzorec otevřený. Pokud se interpoluje jedním polynomem n-tého stupně na $n+1$ uzlech, říká se kvadraturnímu vzorci jednoduchý, pokud se interpoluje po částech více polynomy, jde o vzorec složený. Analytickým integrováním Lagrangeova interpolačního polynomu se získají následující uzavřené Newtonovy-Cotesovy vzorce rostoucí přesnosti (ve smyslu interpolace integrandu polynomy vyšších stupňů):

$n = 1$: jednoduché lichoběžníkové pravidlo $\int_{x_0}^{x_1} f(x)\,dx \approx \frac{1}{2}h(f_0 + f_1)$, neboť

$$\int_{x_0}^{x_1} \left(f_0\, l_1^{(0)} + f_1\, l_1^{(1)}\right) dx = \int_{x_0}^{x_1} \left(f_0 \tfrac{x-x_1}{x_0-x_1} + f_1 \tfrac{x-x_0}{x_1-x_0}\right) dx = \tfrac{f_0}{h}\left[x_1 x - \tfrac{x^2}{2}\right]_{x_0}^{x_1} + \tfrac{f_1}{h}\left[\tfrac{x^2}{2} - x_0 x\right]_{x_0}^{x_1} = \tfrac{1}{2}h(f_0 + f_1)$$

$n = 2$: jednoduché Simpsonovo pravidlo $\int_{x_0}^{x_2} f(x)\,dx \approx \frac{1}{3}h(f_0 + 4f_1 + f_2)$

$n = 3$: jednoduché tříosminové pravidlo $\int_{x_0}^{x_3} f(x)\,dx \approx \frac{3}{8}h(f_0 + 3f_1 + 3f_2 + f_3)$

V případě lichoběžníkového pravidla je integrand aproximován lineárně a plocha pod integrandem je obsahem lichoběžníka, Simpsonovo pravidlo integrand aproximuje kvadraticky, tříosminové pravidlo kubicky.

Složením N/n jednoduchých vzorců pro $x_0, \ldots, x_N$ (N sudé pro Simpsonovo pravidlo, dělitelné 3 pro $\frac{3}{8}$ pravidlo):

složené lichoběžníkové pravidlo $L_N \approx h(\frac{1}{2}f_0 + f_1 + f_2 + \cdots + f_{N-1} + \frac{1}{2}f_N)$

složené Simpsonovo pravidlo $S_N \approx \frac{1}{3}h\,(f_0 + 4f_1 + 2f_2 + 4f_3 + \cdots + 4f_{N-1} + f_N)$

složené tříosminové pravidlo $T_N \approx \frac{3}{8}h\,(f_0 + 3f_1 + 3f_2 + 2f_3 + 3f_4 + \cdots + 3f_{N-1} + f_N)$

Jméno má i kvadraturní vzorec založený na (zde pravostranné) aproximaci integrandu konstantním polynomem:

obdélníkové pravidlo $\int_{x_0}^{x_1} f(x)\,dx \approx hf_1, \quad O_N \approx h(f_1 + f_2 + \cdots + f_{N-1} + f_N)$

Složené obdélníkové i lichoběžníkové pravidlo lze snadno použít i na neekvidistantních sítích, $h_i = x_i - x_{i-1}$:

$O'_N \approx \sum_{i=1}^{N} h_i f_i\,,$ $\qquad L'_N \approx \frac{1}{2}\sum_{i=1}^{N} h_i(f_{i-1} + f_i)\,.$

Rombergova metoda, NR kap. 4.3

Praktický postřeh založený na vzorci pro rozvoj chyby složeného lichoběžníkového pravidla (NR 4.2.1) praví, že složené Simpsonovo pravidlo na $N+1$ uzlech lze vyčíslit pomocí lichoběžníkových pravidel na $N+1$ a $N/2+1$ uzlech:

$S_N = \frac{4}{3}L_N - \frac{1}{3}L_{N/2}$, jak lze ověřit dosazením:

$$\tfrac{4}{3}L_N - \tfrac{1}{3}L_{N/2} = \tfrac{4}{3}h\left(\tfrac{f_0}{2} + f_1 + f_2 + \cdots + \tfrac{f_N}{2}\right) - \tfrac{1}{3}2h\left(\tfrac{f_0}{2} + f_2 + \cdots + \tfrac{f_N}{2}\right) = \tfrac{h}{3}\left(f_0 + 4f_1 + 2f_2 + \cdots + f_N\right) = S_N$$

Vyčíslováním L_N pro $N = 1,\ 2,\ 4,\ 8,\ \ldots$ tak lze touto a obdobnými kombinacemi výsledky levně zpřesňovat.

Gaussovy kvadraturní vzorce, NR kap. 4.5

Kvadraturní vzorec $I_n = \sum w_i f_i$ má při cca $2n$ stupních volnosti (váhy w_i a uzly x_i) potenciální řád přesnosti (tj. stupeň polynomu integrovatelného přesně) rovněž cca $2n$. Triviální volbou cca n uzlů (ekvidistantní síť N.-C. vzorců) se řád přesnosti sníží o cca n, odpovědnou volbou uzlů se řád přesnosti může uchovat. Jinak řečeno: jednoduché N.-C. vzorce s aproximačním polynomem stupně n dovolují při $n+1$ ekvidistantních uzlech integrovat přesně polynomy do stupně n. Gaussovy vzorce s aproximačním polynomem téhož stupně, ale vyčísleným na n speciálně volených uzlech dovolují integrovat přesně polynomy do stupně $2n-1$. Vhodnými uzly jsou kořeny ortogonálních polynomů $p_n(x)$, splňujících požadavek ortogonality s vahou $w(x)$, $\int w(x)p_i(x)p_j(x)\,dx = 0$ pro $i \neq j$. Gaussův kvadraturní vzorec je pak $I_n = \int w(x)f(x)\,dx = \sum w_i f_i$, kde váhy w_i a uzly x_i pro vyčíslení f_i jsou dány

v učebnicích nebo je lze odvodit. Gaussovy vzorce se mohou zvláště vyplatit, když součást integrandu $w(x)$ je vahou klasických ortogonálních polynomů a zbytek integrandu $f(x)$ je polynomem aproximovatelným přesně.

Legendreovy ortogonální polynomy $P_n(x)$, též NR kap. 6.8

Polynomy ortogonální na intervalu $\langle -1,1\rangle$ s vahou $w(x)=1$, tj. $\int_{-1}^{1} P_i(x)P_j(x)\,dx = 0$ pro $i \neq j$

Rekurentní vzorec: $(n+1)P_{n+1} = (2n+1)xP_n - nP_{n-1}$, výchozí polynomy $P_0 = 1,\ P_1 = x$

Octave:
```
pp=@(p,pm,x,n) (p.*x*(2*n+1)-pm*n)/(n+1); x=-1.2:.01:1.2; p0=ones(size(x)); p1=x;
p2=pp(p1,p0,x,1); p3=pp(p2,p1,x,2); plot(x,p0,x,p1,x,p2,x,p3); axis([-1.2 1.2 -1.2 1.2]); grid on
```

Kořeny a váhy Gaussova-Legendreova kvadraturního vzorce se zjišťují numericky.

$n=1$, $P_1(x)=x$ na $\langle -1,1\rangle$, kořen $x_1=0$, váha $w_1=2$

Newton-Cotes $I_1 = \frac{1}{2}\cdot 2(f(-1)+f(1))$ na $\langle -1,1\rangle$ nebo $I_1 = \frac{1}{2}h(f(a)+f(b))$ na $\langle a,b\rangle$
integruje přesně polynomy stupně 1

Legendre $I_1 = 2f(0)$ na $\langle -1,1\rangle$ nebo $I_1 = (b-a)f(\frac{1}{2}(a+b))$ na $\langle a,b\rangle$ při substituci $y=\frac{2x-a-b}{b-a}$,
integruje přesně polynomy stupně $2n-1=1$

$n=2$, $P_2(x)=\frac{3}{2}x^2-\frac{1}{2}$ na $\langle -1,1\rangle$, kořeny $x_{1,2}=\pm\sqrt{\frac{1}{3}}$, váhy $w_{1,2}=1$

Newton-Cotes $I_2 = \frac{1}{3}\cdot 1(f(-1)+4f(0)+f(1))$ integruje přesně do stupně 2 (ze 3 uzlových hodnot)

Legendre $I_2 = f(-\sqrt{\frac{1}{3}})+f(\sqrt{\frac{1}{3}})$ integruje přesně do stupně $2n-1=3$ (ze 2 uzlových hodnot)

Čebyševovy ortogonální polynomy $T_n(x)$, též NR kap. 5.8

Polynomy ortogonální na intervalu $\langle -1,1\rangle$ s vahou $w(x)=\frac{1}{\sqrt{1-x^2}}$, tj. $\int_{-1}^{1} w(x)T_i(x)T_j(x)\,dx = 0$ pro $i \neq j$

Rekurentní vzorec: $T_{n+1} = 2xT_n - T_{n-1}$, výchozí polynomy $T_0 = 1,\ T_1 = x$, též: $T_n(x) = \cos(n \arccos x)$

Octave:
```
tt=@(t,tm,x,n) (t.*x*2-tm); x=-1.2:.01:1.2; t0=ones(size(x)); t1=x;
t2=tt(t1,t0,x,1); t3=tt(t2,t1,x,2); plot(x,t0,x,t1,x,t2,x,t3); axis([-1.2 1.2 -1.2 1.2]); grid on
```

Kořeny a váhy Gaussova-Čebyševova kvadraturního vzorce: $x_k = \cos\left[\pi(k-\frac{1}{2})/n\right],\ w_k = \pi/n,\ k=1,\ldots,n$

$n=1$, $T_1(x)=x$ na $\langle -1,1\rangle$, kořen $x_1=0$, váha $w_1=\pi$

uzavřený Newton-Cotes vyžaduje regularitu $w(x)f(x)$ pro $x=\pm 1$, přičemž $w(\pm 1)$ diverguje

Čebyšev $I_1 = \pi f(0)$ integruje na $\langle -1,1\rangle$ přesně výraz $w(x)f(x)$ pro $f(x)$ polynom stupně $2n-1=1$

$n=2$, $T_2(x)=2x^2-1$ na $\langle -1,1\rangle$, kořeny $x_{1,2}=\pm\sqrt{\frac{1}{2}}$, váhy $w_{1,2}=\frac{\pi}{2}$

Čebyšev $I_2 = \frac{\pi}{2}\left[f(-\sqrt{\frac{1}{2}})+f(\sqrt{\frac{1}{2}})\right]$ integruje na $\langle -1,1\rangle$ přesně $w(x)f(x)$ pro $f(x)$ stupně $2n-1=3$

Octave: kvadraturní integrátor quad(f,a,b,tol), kde f je funkce skalárního argumentu, též quadgk, quadcc, trapz ad.
```
f=@(x) sin(x); a=0; b=pi; tol=1e-10; quad(f,a,b,tol)                    % vrátí 2
format long; g=@(x) polyval([16 -16],x)./polyval([1 -2 0 4 -4],x); quad(g,0,1)   % vrátí pi
```

Integrování metodou Monte Carlo, NR kap. 7.6

Na vstupu metod Monte Carlo se objevují (pseudo)náhodná čísla. Monte Carlo výpočet vícerozměrného určitého integrálu vyčíslováním funkce $f(x)$ v náhodných bodech $x_i,\ i=1,\ldots,N$, je popsán vzorcem (NR 7.6.1):

$$\int f\,dV \approx V\langle f\rangle \pm V\sqrt{\frac{\langle f^2\rangle-\langle f\rangle^2}{N}}\ ,$$

kde V je integrační oblast a úhlové závorky značí aritmetický průměr, $\langle f\rangle = \frac{1}{N}\sum_{i=1}^{N} f_i,\ f_i = f(x_i)$. Chyba je popsána ve statistickém smyslu a je jí vystižena pomalá konvergence metody: počet platných míst výsledku roste s $\sqrt{N}$. (Chyba složeného lichoběžníkového pravidla L_N klesá s $\frac{1}{N^2}$.)

Pro jednorozměrný integrál nabude Monte Carlo vzorec tvaru (podobného vzorci pro obdélníkové pravidlo O_N):

$$\int_a^b f(x)\,dx \approx (b-a)\langle f\rangle = \frac{b-a}{N}\sum_{i=1}^{N} f_i\,.$$

Oblíbenou variantou je obklopit m-rozměrný integrand $(m+1)$-rozměrnou oblastí, tu pokrýt náhodnými body a nasčítat ty z nich, které leží „pod integrandem". V našem případě tak na intervalu $\langle a,b\rangle$ odhadneme $y_{\min} \le f(x)$ a $y_{\max} \ge f(x)$, zavedeme funkci $F(x,y)=1$ pro $y \le f(x),\ F(x,y)=0$ pro $y > f(x)$, a metodou Monte Carlo vyčíslíme dvourozměrný integrál na obdélníku,

$$\int_a^b\int_{y_{\min}}^{y_{\max}} F(x,y)\,dydx \approx S\langle F\rangle = SK/N,\ \text{kde } S=(b-a)(y_{\max}-y_{\min}) \text{ a } K=\sum_{i=1,y_i\le f_i}^{N} 1\,.$$

Je totiž
$$\int_a^b\int_{y_{\min}}^{y_{\max}} F(x,y)\,dydx = \int_a^b\int_{y_{\min}}^{f(x)} 1\,dydx = \int_a^b f(x)\,dx - (b-a)y_{\min}\,.$$

Octave:
```
format long; f=@(x) sin(x); a=0; b=pi; ymin=0; ymax=1;
N=1000000; x=a+(b-a)*rand(1,N); I1D=(b-a)*mean(f(x))                    % vrátí (např.) 1.999
y=ymin+(ymax-ymin)*rand(1,N); K=sum(y<=f(x)); I2D=(b-a)*ymin+(b-a)*(ymax-ymin)*K/N
N=1000; plot(x(1:N),f(x(1:N)),'o',x(1:N),y(1:N),'.'); axis([a b ymin ymax])
```

Hledání kořenů, NR kap. 9

Úlohou je řešit rovnici $f(x) = 0$ neboli hledat nulové body (kořeny) reálné funkce reálné proměnné. Při konečné přesnosti reálné aritmetiky jde spíše o řešení úlohy $|f(x)| < \delta$. Pozor však na situace podle NR obr. 9.1.1: blízkými kořeny, nespojitostmi či singularitami funkce $f(x)$ mohou být níže uvedené metody zmateny. Obvyklým požadavkem metod je lokalizace kořenu v intervalu $\langle a, b\rangle$ nebo, v některých případech, odhad polohy kořenu x_0. Kořeny se lokalizují analýzou funkce nebo jejím vykreslením (např. v gnuplotu) nebo postupy navrženými v NR kap. 9.1 (procedury zbrac pro expanzi intervalu a zbrak pro rozdělení intervalu). Některé metody lze zobecnit pro funkce komplexní proměnné nebo pro soustavy rovnic. Octave: fzero pro 1 rovnici, fsolve pro soustavu.

Metoda půlení intervalu (bisekce), NR kap 9.1
Není-li o kořenu k dispozici jiná informace než meze intervalu a_1 a b_1, pro které ovšem platí $f(a_1)f(b_1) < 0$, omezujeme kořen posloupností intervalů $\langle a_1, b_1\rangle \supset \langle a_2, b_2\rangle \supset \cdots \supset \langle a_k, b_k\rangle \supset \ldots$ tak, že v každém kroku aktuální interval rozpůlíme a vybereme tu polovinu, pro kterou zůstává zachováno $f(a_k)f(b_k) < 0$. Půlení zastavíme při dosažení $|f(x)| < \delta$ nebo $|b_k - a_k| < \varepsilon$. Chybu v k-tém kroku lze ztotožnit s aktuální délkou intervalu, $\varepsilon_k = b_k - a_k$. Zjevně platí $\varepsilon_{k+1} = \varepsilon_k/2$, chyba metody se vyvíjí lineárně. Každé půlení dodá mantise 1 bit přesnosti, není tedy třeba více než 24, resp. 53 půlení při reálné přesnosti 4, resp. 8 bytů. Není-li funkce spojitá, metoda může lokalizovat místo kořenu nespojitost či singularitu. Metoda nemůže lokalizovaný kořen ztratit, je vždy konvergující.
Octave: format long; f=@(x) x^3-1; a=.5; b=2; eps=1e-7;
do x=(a+b)*.5, if f(x)*f(a)<0, b=x; else a=x; end until b-a<eps

Metoda regula falsi, NR kap 9.2
Proložíme sečnou body omezující kořen, $(a, f(a)),\ (b, f(b)),\ f(a)f(b) < 0$, vypočteme její kořen a ztotožníme jej s jednou z předchozích mezí a, b tak, aby spolu s druhou mezí i nadále platilo $f(a)f(b) < 0$ (NR obr. 9.2.2). Jde proto opět o vždy konvergující metodu. Rovnici sečny můžeme zapsat ve tvaru

$$y(x) = f(a) + (f(b) - f(a))(x - a)/(b - a),$$

odkud pro její kořen x, v němž $y(x) = 0$, plyne

$$x = (af(b) - bf(a))/(f(b) - f(a)).$$

Octave: format long; f=@(x) x.^3-1; a=.5, b=2, eps=1e-7; x=a; fa=f(a); fb=f(b);
do xo=x; x=(a*fb-b*fa)/(fb-fa), fx=f(x); if fx*fa<0, b=x; fb=fx; else a=x; fa=fx; end until abs(x-xo)<eps
plot: přidat před cyklus: hold off; z=a:(b-a)/100:b; plot(z,f(z),'g',z,zeros(size(z)),'k'); xlim([a b]);
přidat na začátek cyklu (za klíčové slovo do): hold on; plot([a b],[fa fb],'r',[a b],[fa fb],'r.');

Metoda sečen, NR kap 9.2
Proložíme sečnou body omezující kořen, $(x_1, f_1)),\ (x_2, f_2)$, kde $f_k = f(x_k)$, vypočteme její kořen a označíme jej x_3. Pokračujeme dál spojováním naposledy získaných dvojic bodů sečnami až k dosažení konvergence nebo divergence. Není zajištěno, aby každý interval $\langle x_k, x_{k+1}\rangle$ omezoval kořen $f(x)$, metoda proto není vždy konvergující (NR obr. 9.2.1). Může však konvergovat rychleji než předchozí dvě metody. Rovnice sečny je zde

$$y(x) = f_{k-1} + (f_k - f_{k-1})(x - x_{k-1})/(x_k - x_{k-1}),$$

odkud pro její kořen x_{k+1}, v němž $y(x_{k+1}) = 0$, plyne

$$x_{k+1} = (x_{k-1}f_k - x_k f_{k-1})/(f_k - f_{k-1}).$$

Octave: format long; f=@(x) x.^3-1; a=.5, b=2, eps=1e-7; x1=a; f1=f(a); x2=b; f2=f(b);
do x=(x1*f2-x2*f1)/(f2-f1), x1=x2; f1=f2; x2=x; f2=f(x); until abs(x2-x1)<eps
plot: přidat před cyklus: hold off; z=a:(b-a)/100:b; plot(z,f(z),'g',z,zeros(size(z)),'k'); xlim([a b]);
přidat na začátek cyklu (za klíčové slovo do): hold on; plot([x1 x2],[f1 f2],'r',[x1 x2],[f1 f2],'r.');

Newtonova metoda tečen, NR kap 9.4
Funkci $f(x)$ linearizujeme neboli aproximujeme v okolí odhadu polohy kořenu x_0 nultým a prvním členem Taylorova rozvoje neboli vedeme tečnu k $f(x)$ bodem (x_0, f_0) a nalezneme kořen této aproximace. Ten označíme x_1, bodem (x_1, f_1) vedeme další tečnu, nalezneme její kořen a pokračujeme takto až k dosažení konvergence nebo divergence. Rovnice pro kořen tečny vedené bodem x_k je tedy

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k) = 0,$$

odkud plyne vzorec Newtonovy metody,

$$x_{k+1} = x_k - f(x_k)/f'(x_k).$$

Pro chybu $\varepsilon_k = x_k - x$, kde x je hledaný kořen, zřejmě platí stejný vztah, $\varepsilon_{k+1} = \varepsilon_k - f(x_k)/f'(x_k)$. Dosadíme-li sem Taylorovy rozvoje pro $f(x_k)$ a $f'(x_k)$ kolem kořenu x, kde ovšem $f(x) = 0$, tedy $f(x_k) \approx \varepsilon_k f'(x) + \frac{1}{2}\varepsilon_k^2 f''(x)$ a $f'(x_k) \approx f'(x) + \varepsilon_k f''(x)$, dostaneme odhad

$$\varepsilon_{k+1} \approx \varepsilon_k - \frac{\varepsilon_k f'(x) + \frac{1}{2}\varepsilon_k^2 f''(x)}{f'(x) + \varepsilon_k f''(x)} \approx \varepsilon_k^2 \frac{f''(x)}{2f'(x)}\,.$$

Chyba metody se vyvíjí kvadraticky, pro $\varepsilon_k = 10^{-2}$ může pokračovat přes 10^{-4} a 10^{-8} k 10^{-16}, tedy dosáhnout plné 15místné přesnosti 8bytového datového typu jen několika kroky. Je-li lineární aproximace (tečna) nevhodná, metoda selže. (Kořen tečny bude mimo oblast.) Praktickým postupem je užívání rychle konvergující Newtonovy metody, od které se při jejím pokusu o opuštění omezujícího intervalu ustoupí ke vždy konvergující bisekci. Aproximujeme-li ve vzorci Newtonovy metody $f'(x_k)$ numericky, $f'(x_k) = [f(x_k) - f(x_{k-1})]/(x_k - x_{k-1})$, získáme vzorec metody sečen.

Octave: format long; f=@(x) x.^3-1; df=@(x) 3*x.^2; a=.5; b=2; eps=1e-7; x=(a+b)*.5;
x, do xold=x; x=xold-f(x)/df(x) until abs(x-xold)<eps

plot: přidat před cyklus: hold off; z=a:(b-a)/100:b; plot(z,f(z),'g',z,zeros(size(z)),'k'); xlim([a b]);
přidat na začátek cyklu (za klíčové slovo do): hold on; plot(z,f(x)+df(x)*(z-x),'r',[x],[0],'k.',[x],[f(x)],'r.');

Newtonovou metodou (i metodou sečen) lze hledat také kořeny funkcí komplexní proměnné. Graficky vděčnou úlohou je vykreslit ty body komplexní roviny, z nichž Newtonova metoda pro funkci $f(z) = z^3 - 1$ konverguje ke zvolenému kořenu, např. $z = 1$. (NR obr. 9.4.4 sice v PDF souboru chybí, ale vykreslí jej následující skript.)

Octave: ndot=250000; nstep=50; eps=1e-5; f=@(z) z.^3-1; df=@(z) 3*z.^2; root=1;
x1=-2; x2=2; y1=-2; y2=2; z0=x1+(x2-x1)*rand(ndot,1)+i*(y1+(y2-y1)*rand(ndot,1));
z=z0; for n=1:nstep z=z-f(z)./df(z); end; result=z0(abs(z-root)<eps);
disp(['Plotting ',num2str(prod(size(result))),' dots']); plot(result,'.','markersize',2)

Newtonova metoda pro soustavu rovnic, NR kap. 9.6

Uvažujme soustavu N nelineárních rovnic

$$\mathbf{F}(\mathbf{x}) = \mathbf{0} \qquad \text{neboli} \qquad F_i(x_1, x_2, \ldots, x_N) = 0,\ i = 1, \ldots, N.$$

Linearizujeme Taylorovým polynomem,

$$\mathbf{F}(\mathbf{x} + \delta\mathbf{x}) \approx \mathbf{F}(\mathbf{x}) + \frac{\partial \mathbf{F}(\mathbf{x})}{\partial \mathbf{x}} \cdot \delta\mathbf{x} \qquad \text{neboli} \qquad F_i(\mathbf{x} + \delta\mathbf{x}) \approx F_i(\mathbf{x}) + \sum_{j=1}^{N} \frac{\partial F_i(\mathbf{x})}{\partial x_j}\delta x_j,$$

kde budeme značit Jacobiovu matici $\mathbf{J}(\mathbf{x}) = \partial\mathbf{F}/\partial\mathbf{x}$, $J_{ij} = \partial F_i/\partial x_j$. Podobně jako v Newtonově metodě pro jednu rovnici hledáme $\delta\mathbf{x}$ tak, aby $\mathbf{F}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{0}$, tedy řešíme v každém kroku soustavu lineárních algebraických rovnic

$$\mathbf{J} \cdot \delta\mathbf{x} = -\mathbf{F}.$$

Vzorec Newtonovy metody pro soustavu rovnic má tak tvar

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{J}^{-1}(\mathbf{x}_k) \cdot \mathbf{F}(\mathbf{x}_k)\,.$$

Počáteční hodnotu $\mathbf{x}_0$ musíme odhadnout.

Octave: řešení soustavy nelineárních rovnic

$$\begin{aligned} 2x_1x_2 + 3x_1x_3 - 93 &= 0 \\ 4x_1x_2 + 5x_2x_3 - 235 &= 0 \\ 6x_1x_3 + 7x_2x_3 - 371 &= 0 \end{aligned}$$

```
kmax=7; x=[1; 2; 3];
printf("%2d %15.10f %15.10f %15.10f\n",0,x)
for k=1:kmax
  F=[2*x(1)*x(2)+3*x(1)*x(3)-93; 4*x(1)*x(2)+5*x(2)*x(3)-235; 6*x(1)*x(3)+7*x(2)*x(3)-371];
  J=[2*x(2)+3*x(3) 2*x(1) 3*x(1); 4*x(2) 4*x(1)+5*x(3) 5*x(2); 6*x(3) 7*x(3) 6*x(1)+7*x(2)];
  x=x-J\F;
  printf("%2d %15.10f %15.10f %15.10f\n",k,x)
end
```

Octave: Pro hledání nulových bodů jedné funkce je určen řešič fzero(f,x0), kde f je skalární funkce skalárního argumentu a x0 odhad polohy nulového bodu (skalár nebo dvouprvkové pole mezí).

```
format long; f=@(x) sin(x); x0=3; fzero(f,x0), x0=[3 4]; fzero(f,x0)      % vrátí pí
```

Pro soustavy slouží řešič fsolve(F,x0,options), kde F je vektorová funkce a x0 vektor s počátečním odhadem.

```
F=@(x) [2*x(1)*x(2)+3*x(1)*x(3)-93; 4*x(1)*x(2)+5*x(2)*x(3)-235; 6*x(1)*x(3)+7*x(2)*x(3)-371];
x0=[1; 2; 3]; fsolve(F,x0)                                  % vrátí totéž jako skript výše
```

V MATLABu je fsolve součástí Optimization toolboxu.

Soustavy obyčejných diferenciálních rovnic, NR kap. 16

Úlohou je nalézt numerickou aproximaci funkcí, pro které je předepsána soustava R obyčejných diferenciálních rovnic prvního řádu,

$$\frac{dy_r(x)}{dx} = f_r(x, y_1(x), \ldots, y_R(x)),\ r = 1, \ldots, R, \quad \text{neboli} \quad \frac{d\mathbf{y}(x)}{dx} = \mathbf{f}(x, \mathbf{y}(x)),$$

s počáteční podmínkou

$$y_r(x_0) = y_{0r} \qquad \text{neboli} \quad \mathbf{y}(x_0) = \mathbf{y}_0.$$

Soustavy vyššího řádu lze převádět na soustavy prvního řádu, obdobně jako v následující ukázce.
Numerickým řešením úlohy na síti $x_n,\ n = 0, 1, \ldots,$ je pro každé $r = 1, \ldots, R$ posloupnost y_{nr}, která lépe či hůře aproximuje přesné řešení $y_r(x_n)$.

Ukázka: rovnice pro trajektorii $X(t) = (x(t), z(t))$ bodu o hmotnosti M v silovém poli $F = (F_x, F_z)$ se zadanou polohou a rychlostí $V(t) = (v_x, v_z)$ v počátečním čase t_0,

$$M\frac{d^2X(t)}{dt^2} = F(t, X(t)),\ X(t_0) = X_0,\ V(t_0) = V_0,$$

lze vyjádřit jako soustavu prvního řádu,

$$\frac{d}{dt}\begin{pmatrix} x(t) \\ z(t) \\ v_x(t) \\ v_z(t) \end{pmatrix} = \begin{pmatrix} v_x(t) \\ v_z(t) \\ F_x(t, x, z, v_x, v_z)/M \\ F_z(t, x, z, v_x, v_z)/M \end{pmatrix}.$$

Eulerova metoda, NR kap. 16.1
Dvě podoby nejjednodušší numerické metody lze získat použitím nultého a prvního členu Taylorova rozvoje pro $y(x)$ v okolí x_n, resp. x_{n+1} (vynecháváme index rovnice r):

$$y(x_n + h) \approx y(x_n) + hy'(x_n), \text{ resp. } y(x_{n+1} - h) \approx y(x_{n+1}) - hy'(x_{n+1}).$$

Když zde položíme $x_n + h = x_{n+1}$, nahradíme výrazy $y(x_n)$ pro přesné řešení odpovídajícími výrazy y_n pro numerické řešení a dosadíme za $y'(x)$ pravou stranu výchozích rovnic $f(x, y_1, \ldots, y_R)$, získáme následující:
explicitní Eulerova metoda pro 1 rovnici,

$$y_{n+1} = y_n + hf(x_n, y_n), \quad n = 0, 1, \ldots$$

implicitní Eulerova metoda pro 1 rovnici,

$$y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}).$$

Pro soustavu R rovnic má explicitní a implicitní Eulerova metoda tvar

$$y_{n+1,r} = y_{nr} + hf(x_n, y_{n1}, \ldots, y_{nR}),\ r = 1, \ldots, R, \quad \text{neboli} \qquad \mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(x_n, \mathbf{y}_n),$$
$$y_{n+1,r} = y_{nr} + hf(x_{n+1}, y_{n+1,1}, \ldots, y_{n+1,R}), \qquad \mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(x_{n+1}, \mathbf{y}_{n+1}).$$

Přesnost mají v taylorovském smyslu obě Eulerovy metody stejnou (prvního řádu), podstatně lepší je však stabilita implicitní varianty (což platí pro příbuzné explicitní a implicitní metody vcelku obecně). To lze ukázat na úloze

$$y' = -cy,\ y(0) = 1,\ c > 0,$$

s analytickým řešením $y(x) = e^{-cx}$ (viz NR kap. 16.6):
a) explicitní metoda: $y_{n+1} = y_n + h(-cy_n) = (1 - ch)y_n$, což dává nestabilní (divergující) numerické řešení při $|1 - ch| > 1$, čili při kroku $h > 2/c$;
b) implicitní metoda: $y_{n+1} = y_n + h(-cy_{n+1}) = y_n/(1 + ch)$, což nediverguje ani pro $h \to \infty$.
Octave: vývoj y_n pro $y' = -cy$ pomocí explicitní (pole ye, red) a implicitní (yi, blue) Eulerovy metody při $h > 2/c$

```
c=1; x0=0; y0=1; ye=y0; yi=y0; xmax=10.4; nmax=5; h=xmax/nmax; x=x0:h:xmax;
for n=1:nmax, ye(n+1)=(1-c*h)*ye(n); yi(n+1)=yi(n)/(1+c*h); end; plot(x,exp(-c*x),'g',x,ye,'r',x,yi,'b')
```

Cenou za lepší stabilitu implicitních metod je jejich větší numerická náročnost. Ta je způsobena buď několika iteracemi v každém kroku, kdy se po inicializaci $y_{n+1}^{[0]}$ explicitní metodou opakovaně volá implicitní metoda, $y_{n+1}^{[j+1]} = y_n + hf(x_{n+1}, y_{n+1}^{[j]}),\ j = 0, 1, \ldots,$ nebo řešením soustavy algebraických rovnic v každém kroku, k čemuž vede linearizace pravé strany soustavy (aplikace Newtonovy metody), zde $\mathbf{f} = \mathbf{f}(\mathbf{y})$, ve vzorci implicitní metody:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(\mathbf{y}_{n+1}) = \mathbf{y}_n + h(\mathbf{f}(\mathbf{y}_n) + \mathbf{J} \cdot (\mathbf{y}_{n+1} - \mathbf{y}_n)),$$ kde Jacobiovu matici $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}|_{\mathbf{y}_n}$ značíme $\mathbf{J}$, $J_{ij} = \frac{\partial f_i}{\partial y_j}$.

Pak $(\mathbf{I} - h\mathbf{J}) \cdot \mathbf{y}_{n+1} = (\mathbf{I} - h\mathbf{J}) \cdot \mathbf{y}_n + h\mathbf{f}(\mathbf{y}_n)$

a $\mathbf{y}_{n+1} = \mathbf{y}_n + h(\mathbf{I} - h\mathbf{J})^{-1} \cdot \mathbf{f}(\mathbf{y}_n).$

Odvozený vzorec se nazývá semi-implicitní Eulerovou metodou.

Rungovy-Kuttovy metody, NR kap. 16.1
Obecný vzorec (zde: explicitních) Rungových-Kuttových (RK) metod (zde: pro 1 rovnici) má tvar

$$\begin{aligned} y_{n+1} &= y_n + h\sum_{i=1}^{p} c_i k_i, \\ \text{kde} \quad k_1 &= f(x_n, y_n), \\ k_i &= f(x_n + h\alpha_i, y_n + h\sum_{j=1}^{i-1} \beta_{ij} k_j),\ i = 2, \ldots, p. \end{aligned}$$

Parametry $c_i,\ \alpha_i,\ \beta_{ij}$ konkrétní metody se volí tak, aby vážený průměr několika směrnic k_i podél hledané funkce $y(x)$ na intervalu $\langle x_n, x_{n+1}\rangle$ aproximoval Taylorův rozvoj $y(x)$ co nejvyššího řádu. Konstanty c_i jsou vahami směrnic, tedy $\sum_{i=1}^{p} c_i = 1$, konstanty α_i jsou z intervalu $\langle 0, 1\rangle$ a konstanty β_{ij} vyplňují dolní trojúhelníkovou matici. Počet p vyčíslení pravé strany odpovídá pro $p \le 4$ řádu přesnosti metody.

RK1 metoda prvního řádu přesnosti má parametry $p = 1,\ c_1 = 1$ a je to explicitní Eulerova metoda.
Pro každé $p > 1$ existuje ∞ RK metod, používá se jich však jen několik s jednoduše vyhlížejícími parametry nebo speciálními vlastnostmi. Dvě populární metody druhého řádu (NR 16.1.2):
RK2 metoda středního bodu (midpoint method) $y_{n+1} = y_n + hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hf(x_n, y_n))$,
RK2 Heunova metoda (Heun-Verfahren) $y_{n+1} = y_n + \frac{1}{2}h(f(x_n, y_n) + f(x_n + h, y_n + hf(x_n, y_n))$.

Odvození RK2 metod: Taylorův rozvoj $y(x)$ druhého řádu je $y_{n+1} \approx y_n + hy' + \frac{1}{2}h^2y''$, což při $y' = f(x_n, y_n) \equiv f$ a $y'' = f' = \partial_x f + f\partial_y f$ lze přepsat na tvar $y_{n+1} \approx y_n + hf + \frac{1}{2}h^2\partial_x f + \frac{1}{2}h^2 f\partial_y f$. Obecný vzorec RK2 metody $(p = 2)$ je $y_{n+1} = y_n + hc_1 f(x_n, y_n) + hc_2 f(x_n + h\alpha_2, y_n + h\beta_{21} f) \approx y_n + hc_1 f + hc_2(f + h\alpha_2\partial_x f + h\beta_{21} f\partial_y f)$, kde jsme použili Taylorův rozvoj funkce 2 proměnných. Srovnáním členů v obou vzorcích získáme soustavu 3 nelineárních rovnic pro 4 parametry metody: $c_1 + c_2 = 1,\ c_2\alpha_2 = \frac{1}{2},\ c_2\beta_{21} = \frac{1}{2}$. Můžeme volit $c_2 = C$, pak $c_1 = 1 - C$, $\alpha_2 = \beta_{21} = 1/(2C)$. Pro $C = 1$ máme metodu středního bodu, pro $C = \frac{1}{2}$ Heunovu metodu.

Populární RK4 metoda čtvrtého řádu (NR 16.1.3):

$$p = 4,\ c = (\tfrac{1}{6}, \tfrac{1}{3}, \tfrac{1}{3}, \tfrac{1}{6}),\ \alpha = (-, \tfrac{1}{2}, \tfrac{1}{2}, 1),\ \beta = (-, -, -, -|\ \tfrac{1}{2}, -, -, -|\ 0, \tfrac{1}{2}, -, -|\ 0, 0, 1, -)$$

neboli $y_{n+1} = y_n + h\left(\frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4\right)$,
kde $k_1 = f(x_n, y_n),\ k_2 = f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1),\ k_3 = f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2),\ k_4 = f(x_n + h, y_n + hk_3)$.
Odvození se vede podobně jako pro RK2, získá se soustava 11 nelineárních rovnic pro 13 parametrů metody.

Pro pravou stranu $f(x)$ nezávislou na $y(x)$ je krok Eulerovy metody ekvivalentní (levostrannému) obdélníkovému kvadraturnímu vzorci, $y_{n+1} = y_n + hf_n$, RK2 Heunova metoda odpovídá lichoběžníkovému pravidlu, $y_{n+1} = y_n + \frac{1}{2}h(f_n + f_{n+1})$, a RK4 metoda Simpsonovu pravidlu s krokem $\frac{h}{2}$: $y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$, kde $k_1 = f(x_n),\ k_2 = k_3 = f(x_n + \frac{h}{2}),\ k_4 = f(x_{n+1})$.

MATLAB: např. funkce ode45(f,tspan,y0) pro RK metodu zapouzdřující vzorce 4. a 5. řádu přesnosti (při $p = 6$). To umožňuje provádět v každém kroku odhad chyby a adaptivně tak regulovat délku (následujícího nebo revidovaného) kroku podle předepsané mezní chyby (NR kap. 16.2).
Octave: funkce lsode(fcn,y0,xvec) z volně dostupné knihovny ODEPACK, kde xvec je vektor popisující síť $(x_n,\ n = 0, 1, \ldots)$, y0 je počáteční podmínka $(y_{0r},\ r = 1, \ldots, R)$ a fcn(y,x) je funkce vyčíslující vektor pravých stran pro x = x a y = $(y_r,\ r = 1, \ldots, R)$. Cestou argumentu fcn lze dodat i funkci vracející Jacobiovu matici. Nastavení lsode (např. volba numerické metody) se děje voláním funkce lsode_options.

Příklad: 2D šikmý vrh $(x(t), z(t))$ bodu o hmotnosti M v gravitačním poli $F_g = (0, -Mg)$ při odporové síle $F_o = (-Kvv_x, -Kvv_z)$. Podle rovnic z ukázky v úvodu kapitoly volíme vektor neznámých v pořadí y = (x, z, v_x, v_z).
Soubor oderhs.m s funkcí vyčíslující vektor pravých stran:

```
function result = oderhs(y,t)                  % funkce pro pravé strany (right-hand sides)
global M g K;                                  % data tekoucí mimo rozhraní
v=sqrt(y(3)^2+y(4)^2);                         % velikost rychlosti
result=[y(3); y(4); -K/M*v*y(3); -g-K/M*v*y(4)];   % (vx, vz, -K/M.v.vx, -K/M.v.vz)
end
```

Příkazy pro výpočet a vykreslení balistické křivky:

```
global M g K; M=1; g=9.81; K=0.5; tvec=0:.1:2.5; y0=[0 0 7 7];  % parametry, síť, počáteční podmínka
y=lsode(@oderhs,y0,tvec); plot(y(:,1),y(:,2))                   % Livermore Solver for ODEs
```

Vícekrokové metody, NR kap. 16.7
Vícekrokové metody užívají k výpočtu y_{n+1} více hodnot $y_n,\ y_{n-1},\ldots,f_n,\ f_{n-1},\ldots,$ známých z předcházejících kroků. (Explicitní Rungovy-Kuttovy metody jsou v tomto smyslu jednokrokové.)
Obecný vzorec lineární p-krokové metody má tvar

$$y_{n+1} = \sum_{i=1}^{p} \alpha_i y_{n+1-i} + h \sum_{i=0}^{p} \beta_i f_{n+1-i},$$ přičemž $\alpha_p \neq 0$ nebo $\beta_p \neq 0.$

Při $\beta_0 = 0$ je metoda explicitní, jinak je implicitní.

Rodina Adamsových p-krokových metod je založena na proložení hodnot $f(x_i, y_i)$ na ekvidistantní síti x_i, $i = n, n-1, \ldots, n+1-p$ (v implicitních metodách včetně $n+1$), s krokem h interpolačním polynomem a jeho analytickém integrování na $\langle x_n, x_{n+1}\rangle$. Explicitní metody aproximují f polynomem s uzly v $x_n,\ x_{n-1},\ldots$ a ten extrapolují na interval $\langle x_n, x_{n+1}\rangle$, implicitní metody polynomicky aproximují na uzlech $x_{n+1},\ x_n,\ x_{n-1},\ldots$ a v intervalu $\langle x_n, x_{n+1}\rangle$ tak interpolují.
Explicitní Adamsovy-Bashforthovy metody:

$$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} L_{p-1}(x)\,dx = y_n + h \sum_{i=1}^{p} \beta_i f_{n+1-i}$$

Implicitní Adamsovy-Moultonovy metody:

$$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} L_{p}(x)\,dx \quad = y_n + h \sum_{i=0}^{p} \beta_i f_{n+1-i}$$

Řád přesnosti je p pro explicitní a $p+1$ pro implicitní metody. Je $\alpha_1 = 1$ a ostatní α_i jsou nulové. Koeficienty β_i metod prvního (Eulerovy metody) až čtvrtého řádu přesnosti jsou shrnuty v následující tabulce; odvodit je lze analytickým integrováním explicitního tvaru Lagrangeova nebo Newtonova interpolačního polynomu.

Tabulka koeficientů β_i Adamsových metod

	explicitní Adamsovy-Bashforthovy metody						implicitní Adamsovy-Moultonovy metody					
řád	p	i:	1	2	3	4	p	i:	0	1	2	3
1:	1	β_i:	1				1	β_i:	1			
2:	2	$2\beta_i$:	3	−1			1	$2\beta_i$:	1	1		
3:	3	$12\beta_i$:	23	−16	5		2	$12\beta_i$:	5	8	−1	
4:	4	$24\beta_i$:	55	−59	37	−9	3	$24\beta_i$:	9	19	−5	1

Metoda prediktor-korektor spočívá v inicializaci $y_{n+1}^{[0]}$ explicitní Adamsovou metodou (prediktor P) a střídavém vyčíslování pravé strany $f(x_{n+1}, y_{n+1}^{[j]})$ (evaluation E) a implicitní Adamsovy metody (obvykle téhož řádu) s dosazenou aktuální hodnotou $y_{n+1}^{[j]}$ (korektor C) pro $j = 0, 1, \ldots$ Varianty metody se liší počtem fází, vyjadřovaným symbolicky: PEC, PECE, P(EC)^{2}E aj. Explicitní prediktor ovšem činí z metody prediktor-korektor explicitní metodu (s menší stabilitou než použitý implicitní korektor).

Gearovy p-krokové metody (známé též jako BDF, backward differentiation formulas) se zvykově zapisují ve tvaru

$$\sum_{i=0}^{p} \alpha_i y_{n+i} = h\beta f(x_{n+p}, y_{n+p}),$$

jsou tedy implicitní. Řád přesnosti p-krokové BDF je p, BDF pro $p = 1$ je implicitní Eulerova metoda a BDF lze použít (jsou stabilní) jen pro $p \leq 6$. Poněkud magicky mohou působit následující návody:

$$\beta = \left(\sum_{i=1}^{p} \tfrac{1}{i}\right)^{-1}, \qquad \sum_{i=0}^{p} \alpha_i \omega^i = \beta \sum_{i=1}^{p} \tfrac{1}{i} \omega^{p-i} (\omega - 1)^i,$$

z nichž lze získat hodnoty koeficientů $\alpha_i,\ \beta$ shrnuté v následující tabulce.

Tabulka koeficientů $\alpha_i,\ \beta$ Gearových metod (BDF)

řád	p	β	i:	0	1	2	3	4	5	6
1:	1	1	α_i:	−1	1					
2:	2	2/3	$3\alpha_i$:	1	−4	3				
3:	3	6/11	$11\alpha_i$:	−2	9	−18	11			
4:	4	12/25	$25\alpha_i$:	3	−16	36	−48	25		
5:	5	60/137	$137\alpha_i$:	−12	75	−200	300	−300	137	
6:	6	60/147	$147\alpha_i$:	10	−72	225	−400	450	−360	147

Pro startování Adamsových a Gearových vícekrokových metod se používá posloupnost metod méněkrokových v kombinaci s Newtonovou metodou nebo metodou prediktor-korektor. Adaptivní řízení délky kroku podle předepsané mezní chyby je pro vícekrokové metody zjevně pracnější než pro metody jednokrokové.