

# Index

- adaptive Runge–Kutta method, 275–283
- arithmetic operators, in Python, 6
- arrays
  - accessing/changing, 20
  - copying, 23
  - creating, 19
  - functions, 21–22
  - operations on, 20–21
- augmented assignment operators, 7
- augmented coefficient matrix, 28
  
- backward finite difference approximations, 179
- banded matrix, 54–63
- bisection, 142–143
- bisection method, for equation root, 142–145
- Brent’s method, 175
- Bulirsch–Stoer method, 278–279, 283
  - algorithm, 280–284
  - midpoint method, 277–278
  - Richardson extrapolation, 278–279
- bulStoer, 281
  
- choleski(a), 46–47
- Choleski’s decomposition, 44–47
- cmath module, 18
- coefficient matrices, symmetric/banded, 54–63
  - symmetric, 57–58
  - symmetric/pentadiagonal, 58–61
  - tridiagonal, 55–57
- comparison operators, in Python, 7
- composite Simpson’s 1/3 rule, 148
- composite trapezoidal rule, 195–197
- conditionals, in Python, 8
- conjGrad, 86, 87
- conjugate gradient method, 84–87
  - conjugate directions, 383–384
  - Powell’s method, 382–387
- continuation character, 6
- cubicSpline, 117–118
- cubic splines, 114–118, 195
- curve fitting. *See* interpolation/curve fitting
  
- deflation of polynomials, 169
- diagonal dominance, 65
- docstring, 26
- Doolittle’s decomposition, 41–44
- downhill simplex method, 392–395
- downhill, 393–395
  
- eigenvals3, 364
- eigenvalue problems. *See* symmetric matrix eigenvalue problems
- elementary operations, linear algebra, 30
- embedded integration formula, 269
- equivalent linear equation, 30
- error
  - control in Python, 14–15
  - in finite difference approximations, 181–182
- Euhler’s method, 250
- evalPoly, 168–169
- evaluation of polynomials, 167–169
- exponential functions, fitting, 129–130
  
- false position method, roots of equations, 145–146
- finite difference approximations, 177–181
  - errors in, 181–182
  - first central difference approximations, 178–179
  - first noncentral, 179–180
  - second noncentral, 180–181
- finite elements, 228
- first central difference approximations, 182–183
- fourth-order Runge–Kutta method, 252–253
- functions, in Python, 15–16
  
- gaussElimin, 37
- Gauss elimination method, 33–38
  - algorithm for, 35–38
    - back substitution phase, 36
    - elimination phase, 35–36
  - multiple sets of equations, 37, 38
- Gauss elimination with scaled row pivoting, 65–68
- Gaussian integration, 211–221
  - abscissas/weights for Gaussian quadratures, 216–219
    - Gauss–Chebyshev quadrature, 217
    - Gauss–Hermite quadrature, 218
    - Gauss–Laguerre quadrature, 217–218
    - Gauss–Legendre quadrature, 216–217
    - Gauss quadrature with logarithmic singularity, 219
  - determination of nodal abscissas/weights, 214–216
  - orthogonal polynomials, 212–214
- Gauss–Jordan elimination, 31–32
- Gauss–Legendre quadrature over quadrilateral element, 228–231
- gaussNodes, 219–220
- gaussPivot, 67–68

- gaussQuad, 220–221
- gaussQuad2, 230–231
- gaussSeidel, 84
- Gauss–Seidel method, 82–84
- gerschgorin, 361
- Gerschgorin's theorem, 361
- golden section search, 377–379
- goldSearch, 378–379
  
- Higher-order equations, shooting method, 296
- householder, 355–356
  - householder reduction to tridiagonal form, 351–356
  - accumulated transformation matrix, 354–355
  - householder matrix, 351–352
  - householder reduction of symmetric matrix, 352–359
- Idle (code editor), 3
- ill-conditioning, 28–29
- incremental search method, roots of equations, 140–141
- indirect methods. *See* iterative methods
- initial value problems
  - adaptive Runge–Kutta method, 269–273
  - Bulirsch–Stoer method, 277–281
    - algorithm, 280–281
    - midpoint method, 277–278
    - Richardson extrapolation, 278–279
  - multistep methods, 289
  - Runge–Kutta methods, 249–253
    - fourth-order, 252–253
    - second-order, 250–252
  - stability/stiffness, 266–268
  - stability of Euhler's method, 266–267
  - stiffness, 267–268
  - Taylor series method, 244–246
- Input/output
  - printing, 12–13
  - reading, 13–14
  - writing, 14
- integration order, 229
- interpolation, derivatives by, 185–186
  - cubic spline interpolant, 186
  - polynomial interpolant, 185–186
- interpolation/curve fitting
  - interpolation with cubic spline, 114–118
  - least-squares fit, 124–129
    - fitting a straight line, 125
    - fitting linear forms, 125–126
    - polynomial fit, 126–128
  - weighting of data, 128–130
    - fitting exponential functions, 129–130
    - weighted linear regression, 128–129
- polynomial interpolation, 99–107
  - Lagrange's method, 99–101
  - limits of, 106–107
  - Neville's method, 104–106
  - Newton's method, 101–103
  - rational function interpolation, 110–112
- interval halving method. *See* bisection method
- inversePower, 340
- inversePower3, 365–366
- iterative methods, 85–96
  - conjugate gradient method, 84–87
  - Gauss–Seidel method, 82–84
- jacobi, 326–327
- Jacobian matrix, 230
- Jacobi method, 321–327
  - Jacobi diagonalization, 323–326
  - Jacobi rotation, 322–323
  - similarity transformation, 322
  - transformation to standard form, 328–330
- Jenkins–Traub algorithm, 176
- knots of spline, 115
- Lagrange's method, 99–101
- Laguerre's method, 169–171
- lamRange, 362–363
- least-squares fit, 124–135
  - fitting linear forms, 125–126
  - fitting straight line, 125
  - polynomial fit, 126–128
  - weighting data, 128–130
    - fitting exponential functions, 129–130
    - weighted linear regression, 128–129
- linear algebraic equations systems. *See also* matrix algebra
  - back substitution, 32
  - direct methods overview, 31–33
  - elementary operations, 30
  - equivalent equations, 30
  - forward substitution, 32
  - Gauss elimination method, 33–40
    - algorithm for, 35–37
      - back substitution phase, 36
      - elimination phase, 35–36
    - multiple sets of equations, 37–38
  - ill-conditioning, 28–30
  - LU decomposition methods, 40–47
    - Choleski's decomposition, 44–47
    - Doolittle's decomposition, 41–44
  - matrix inversion, 79–80
  - pivoting, 64–70
    - diagonal dominance, 65
    - Gauss elimination with scaled row pivoting, 65–68
    - when to pivot, 70
  - QR decomposition, 98
  - singular value decomposition, 98
  - symmetric/banded coefficient matrices, 54–61
    - symmetric coefficient, 59–60
    - symmetric/pentadiagonal coefficient, 58–61
    - tridiagonal coefficient, 55–57
  - uniqueness of solution, 28
- linear forms, fitting, 125–126
- linear systems, 30
- linInterp, 292
- lists, 5–6
- loops, 8–10
- LR algorithm, 373
- LUdecomp, 43–44
- LUdecomp3, 56–57
- LUdecomp5, 61
- LU decomposition methods, 40–49
  - Choleski's decomposition, 44–47
  - Doolittle's decomposition, 41–44
- LUpivot, 68–70
- mathematical functions, 11
- math module, 17–18
- MATLAB, 2–3
- matrix algebra, 410–415
  - addition, 411
  - determinant, 412–413
  - inverse, 412
  - multiplication, 411–412

- positive definiteness, 413
- transpose, 410
- useful theorems, 414
- matrix inversion, 79–80
- methods of feasible directions, 406
- midpoint, 277–278
- minimization along line, 376–379
  - bracketing, 376–377
  - golden section search, 377–379
- modules, in Python, 16–17
- multiple integrals, 227
  - Gauss–Legendre quadrature over quadrilateral element, 228–231
  - quadrature over triangular element, 234–237
- multistep methods, for initial value problems, 289
- Namespace, 24
- natural cubic spline, 115
- Nelder–Mead method, 392
- neville, 105
- Neville's method, 104–105
- Newton–Cotes formulas, 194–199
  - composite trapezoidal rule, 195–197
  - recursive trapezoidal rule, 197
  - Simpson's rules, 198–199
  - trapezoidal rule, 195
- newtonPoly, 103
- newtonRaphson, 152
- newtonRaphson2, 156–157
- Newton–Raphson method, 150–152, 155–157
- norm of matrix, 29
- notation, 27–28
- numpy module, 18–24
  - accessing/changing array, 20
  - array functions, 21–22
  - copying arrays, 23
  - creating an array, 19
  - linear algebra module, 22–23
  - operations on arrays, 20–21
  - vectorization, 23
- numerical differentiation
  - derivatives by interpolation, 185–186
    - cubic spline interpolant, 186
    - polynomial interpolant, 185–186
  - finite difference approximations, 177–182
    - errors in, 181–182
    - first central difference approximations, 178–179
    - first noncentral, 179–180
    - second noncentral, 180–181
  - Richardson extrapolation, 182–183
- numerical instability, 257
- numerical integration
  - Gaussian integration, 211–221
    - abscissas/weights for Gauss quadratures, 216–219
      - Gauss–Chebyshev quadrature, 217
      - Gauss–Hermite quadrature, 218
      - Gauss–Laguerre quadrature, 217–218
      - Gauss–Legendre quadrature, 216–217
      - Gauss quadrature with logarithmic singularity, 219
    - determination of nodal abscissas/weights, 214–216
    - orthogonal polynomials, 212–214
  - multiple integrals, 227–237
    - Gauss–Legendre quadrature over quadrilateral element, 228–231
    - quadrature over triangular element, 234–237
  - Newton–Cotes formulas, 194–199
    - composite trapezoidal rule, 195–197
    - recursive trapezoidal rule, 197–198
    - Simpson's rules, 198–199
    - trapezoidal rule, 195
  - Romberg integration, 202–205
- operators
  - arithmetic, 6–7
  - comparison, 7
- optimization
  - conjugate directions, 383–384
  - Powell's method, 382–387
  - minimization along line, 376–379
    - bracketing, 376–377
    - golden section search, 377–379
  - Nelder–Mead method. *See* simplex method
  - simplex method, 392–395
  - simulated annealing method, 406
- orthogonal polynomials, 212–214
- relaxation factor, 83
- pivoting, 64
  - diagonal dominance, 65–70
  - Gauss elimination with scaled row pivoting, 65–68
  - when to pivot, 70
- polyFit, 127–128
- polynomial fit, 126–128
- polynomial interpolation, 99–107
  - Lagrange's method, 99–101
  - limits of, 106–107
  - Neville's method, 104–106
  - Newton's method, 101–103
- polynomials, zeroes of, 166–172
  - deflation of polynomials, 169
  - evaluation of polynomials, 167–169
- Laguerre's method, 169–172
- polyRoots, 171–172
- powell, 386–387
- Powell's method, 382–387
- printing input, 12–13
- printSoln, 246
- Python
  - arithmetic operators, 6–7
  - cmath module, 18–19
  - comparison operators, 7–8
  - conditionals, 8
  - error control, 14–15
  - functions, 15–16
  - general information, 1–3
    - obtaining Python, 3
    - overview, 1–3
  - linear algebra module, 22–23
  - lists, 5–6
  - loops, 8–10
  - mathematical functions, 11
  - math module, 17–18
  - modules, 16–17
  - numpy module, 18–24
    - accessing/changing array, 21
    - array functions, 21–22
    - copying arrays, 23
    - creating an array, 19
    - operations on arrays, 20–21
  - printing output, 12–13
  - reading input, 11–12
  - scoping of variables, 24–25
  - strings, 4

- Python (*Continued*)
  - tuples, 4–5
  - type conversion, 10–11
  - variables, 3–4
  - vectorization, 23–24
  - writing/running programs, 25–26
- Python interpreter, 1
- QR algorithm, 380
- quadrature. *See* numerical integration
- quadrature over triangular element, 240–245
- rational function interpolation, 110–112
- reading input, 11–12
- recursive trapezoidal rule, 197–198
- relaxation factor, 89
- Richardson extrapolation, 182–183, 278–279
- Ridder's method, 146–150
- ridder, 147–148
- romberg, 204–205
- Romberg integration, 202–205
- root search, 141
- roots of equations
  - Brent's method, 175
  - false position method, 145
  - incremental search method, 140–141
  - Jenkins–Traub algorithm, 176
  - method of bisection, 142–143
  - Newton–Raphson method, 153–158
- Ridder's method, 146–150
- secant method, 145
- systems of equations, 155–157
  - Newton–Raphson method, 155–157
- zeroes of polynomials, 166–172
  - deflation of polynomials, 169
  - evaluation of polynomials, 167–168
  - Laguerre's method, 169–172
- Runge–Kutta–Fehlberg formulas, 270
- Runge–Kutta methods, 249–253
  - fourth-order, 252–253
  - second-order, 250–251
- run\_kut4, 252–253
- run\_kut5, 272–273
- scaled row pivoting, 65–68
- second noncentral finite difference
  - approximations, 180–181
- second-order Runge–Kutta method, 250–251
- shape functions, 229
- shooting method, 291–296
  - higher-order equations, 296
  - second-order differential equation, 291–292
- Shur's factorization, 373
- similarity transformation, 322
- Simpson's 3/8 rule, 199
- Simpson's rules, 198–199
- slicing operator, 3
- sortJacobi, 327–328
- sparsely populated matrix, 54
- stability/stiffness, 266–268
  - stability of Euler's method, 266–267
  - stiffness, 267–268
- stdForm, 329–330
- straight line, fitting, 125
- strings, 5
- Strum sequence, 358–360
- sturmSeq, 359–360
- swapCols, 67
- swapRows, 67
- symmetric/banded coefficient matrices, 54–62
  - symmetric coefficient matrix, 57–58
  - symmetric/pentadiagonal coefficient, 61–66
  - tridiagonal coefficient, 55–57
- symmetric matrix eigenvalue problems
  - eigenvalues of symmetric tridiagonal matrices, 358–366
    - bracketing eigenvalues, 362–363
    - computation of eigenvalues, 364
    - computation of eigenvectors, 365–366
    - Gerschgorin's theorem, 361
    - Strum sequence, 358–360
  - householder reduction to tridiagonal form, 351–356
    - accumulated transformation matrix, 354–355
    - householder matrix, 351–352
    - householder reduction of symmetric matrix, 352–354
- inverse power/power methods, 337–340
  - eigenvalue shifting, 338–339
  - inverse power method, 337–339
  - power method, 339
- Jacobi method, 321–330
  - Jacobi diagonalization, 323–328
  - Jacobi rotation, 322–323
  - similarity transformation/diagonalization, 321–322
    - transformation to standard form, 328–330
- LR algorithm, 373
- QR algorithm, 373
- Shur's factorization, 373
- symmetric/pentadiagonal coefficient matrix, 58–61
- synthetic division, 169
- systems of equations
  - Newton–Raphson method, 155–157
- taylor, 245–246
- Taylor series, 244, 407–408
  - function of several variables, 408
  - function of single variable, 407–408
- transpose, 410
- trapezoid, 197–198
- trapezoidal rule, 195
- triangleQuad, 236–237
- tridiagonal coefficient matrix, 55–57
- tuples, 5
- two-point boundary value problems
  - finite difference method, 305–314
    - fourth-order differential equation, 310–314
    - second-order differential equation, 306–310
  - shooting method, 291–301
    - higher-order equations, 296–301
    - second-order differential equation, 291–296
- type(a), 12
- type conversion, 10–11
- variables
  - Python, 3–4
  - scoping, 24–25
- vectorizing, 23–24
- weighted linear regression, 128–129
- writing/running programs, in Python, 25–26
- zeroes of polynomials, 166–172
  - deflation of polynomials, 169
  - evaluation of polynomials, 167–169
  - Laguerre's method, 169–172
- zero offset, 3