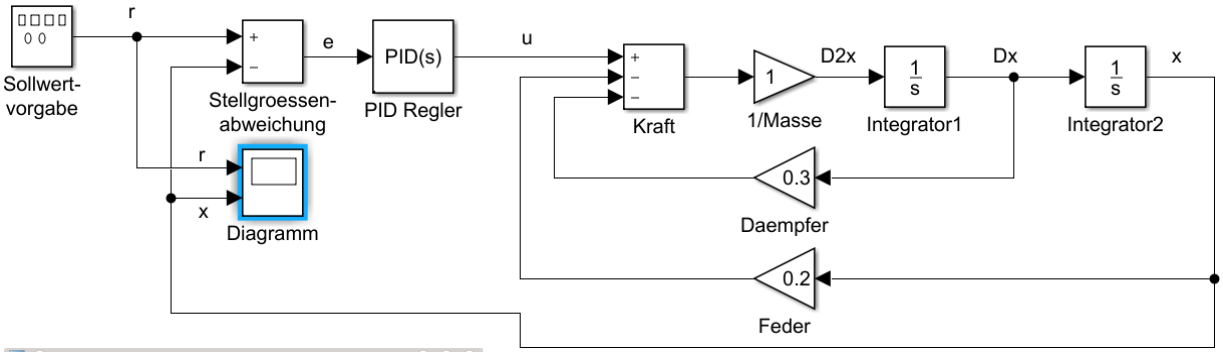
Werkzeuge zur Behandlung von Anfangswertproblemen

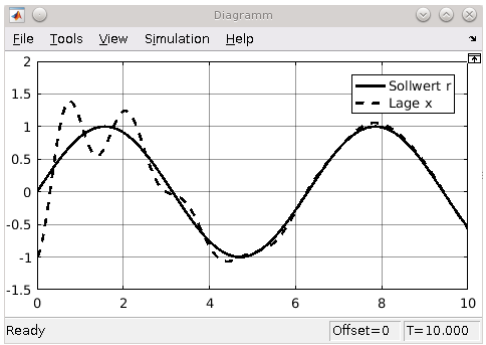
		Bibliotheken	Abgeschlossene Programme
symbolisch			Maple Mathematica MuPad ...
numerisch		IMSL NAG ODEPACK Shampine & Gordon MBSPACK MBSSIM Numerical Recipes ...	Matlab/Simulink Scilab NEWMOS ...

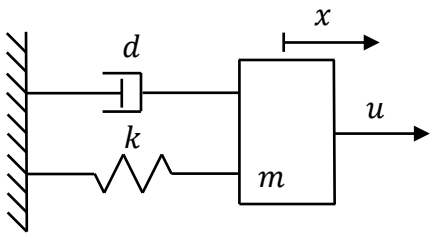
Simulink

Simulink ist Bestandteil der Matlab–Umgebung und eignet sich zur blockorientierten Modellierung und Simulation dynamischer Systeme. Über eine graphische Oberfläche wird der Rechenplan des Modells erstellt, wobei auf Standardblöcke zurückgegriffen werden kann. Für die numerische Integration stehen in Simulink die Einschrittverfahren Euler, RK3 und RK5 zur Verfügung. Ferner werden die Mehrschrittverfahren Adams (PECE) und Gear (ein BDF–Code) angeboten.

Beispiel: Lageregelung eines Einmassenschwingers durch PID–Regelung







Maple

Mit Maple können Differentialgleichungen symbolisch (oder falls alle Zahlenwerte gegeben sind auch numerisch) integriert werden. Dieser Ansatz ist auf einfache Probleme beschränkt.

```
int(x^2+x^3, x);
int(x^2+x^3, x=1..100);
```

NEWMOS

NEWMOS ist ein Simulator für modulare Systeme, der am Institut B für Mechanik entwickelt wurde. Er erlaubt die Simulation von differential–algebraischen Systemen und die Durchführung von Multirate–Simulationen.

Beispiel: Multirate–Simulation des Viertelfahrzeugs mit aktivem Dämpfer

```
define mbs_system (viertel)
   with parameters (m1, m2, k2) inputs (F, u) outputs (yap, yrp)
   by positions (ya, yr) velocities (yaps, yrps)
{
   M[0][0] = m1;
   M[1][1] = m2;
   qe[0] = F;
   qe[1] = -F - k2 * (yr - u);
   K[0][0] = 1;
   K[1][1] = 1;
   y[0] = yaps;
   y[1] = yrps;
};
viertel.m1 = 600;
viertel.m2 = 40;
viertel.k2 = 160000;

define ss_system (daempfer)
   with parameters (Td, D) inputs (yap, yrp) outputs (F) by states (Fs)
{
   xp[0] = (D * (yrp - yap) - Fs) / Td;
   y[0] = Fs;
};
daempfer.Td = 1.0e-5;
daempfer.D = 2400;

define ss_system (anregung)
   with outputs (u) by { y[0] = 0.1; };

connect(viertel.F, daempfer.F);
connect(viertel.u, anregung.u);
connect(daempfer.yap, viertel.yap);
connect(daempfer.yrp, viertel.yrp);
```

```
load integrator (int1) of type (ruku4) from file (ruku4.so);
int1.h = 1.0e-3;
load buffer (buf1) of type (extrapolation) from file (extrapol.so);
buf1.order = 3;

load integrator (int2) of type (ruku4) from file (ruku4.so);
int2.h = 15e-6;
load buffer (buf2) of type (extrapolation) from file (extrapol.so);
buf1.order = 3;

write (t, all states, all derivatives, all outputs) plain to
   file (viertel.out) with stepsize (0.001);

setup (set1) by (viertel, anregung, int1, buf1);
setup (set2) by (daempfer, int2, buf2);

simulate (set1, set2) from (0) to (1) with timeslice (0.001);
```

ODEPACK

Die numerische Bibliothek ODEPACK stellt eine umfassende Sammlung hochentwickelter Numerik–Unterprogramme für die Integration von Anfangswertproblemen dar. Dabei werden insbesondere Mehrschrittverfahren unterstützt. Es existieren auch Methoden zur automatischen Umschaltung zwischen steifen (BDF) und nichtsteifen (PECE) Bereichen. Ferner werden differential–algebraische Systeme in der Index–1–Form unterstützt.

```
      subroutine lsode (f, neq, y, t, tout, itol, rtol, atol, itask,
     1                  istate, iopt, rwork, lrw, iwork, liw, jac, mf)
c-----------------------------------------------------------------------
c this is the march 30, 1987 version of
c lsode.. livermore solver for ordinary differential equations.
c this version is in double precision.
c
c lsode solves the initial value problem for stiff or nonstiff
c systems of first order ode-s,
c       dy/dt = f(t,y) ,  or, in component form,
c       dy(i)/dt = f(i) = f(i,t,y(1),y(2),...,y(neq)) (i = 1,...,neq).
c lsode is a package based on the gear and gearb packages, and on the
c october 23, 1978 version of the tentative odepack user interface
c standard, with minor modifications.
c-----------------------------------------------------------------------
c reference..
c       alan c. hindmarsh,  odepack, a systematized collection of ode
c       solvers, in scientific computing, r. s. stepleman et al. (eds.),
c       north-holland, amsterdam, 1983, pp. 55-64.
c-----------------------------------------------------------------------
c author and contact.. alan c. hindmarsh,
c                      computing and mathematics research div., l-316
c                      lawrence livermore national laboratory
c                      livermore, ca 94550.
c-----------------------------------------------------------------------
c summary of usage.
c
```

```
c first provide a subroutine of the form..
c                subroutine f (neq, t, y, ydot)
c                   dimension y(neq), ydot(neq)
c which supplies the vector function f by loading ydot(i) with f(i).
c
c f       = name of subroutine for right-hand side vector f.
c           this name must be declared external in calling program.
c neq     = number of first order ode-s.
c y       = array of initial values, of length neq.
c t       = the initial value of the independent variable.
c tout    = first point where output is desired (.ne. t).
c itol    = 1 or 2 according as atol (below) is a scalar or array.
c rtol    = relative tolerance parameter (scalar).
c atol    = absolute tolerance parameter (scalar or array).
c itask   = 1 for normal computation of output values of y at t = tout.
c istate  = integer flag (input and output).  set istate = 1.
c iopt    = 0 to indicate no optional inputs used.
c rwork   = real work array of length at least..
c                  20 + 16*neq                    for mf = 10,
c                  22 +  9*neq + neq**2           for mf = 22,
c lrw     = declared length of rwork (in user-s dimension).
c iwork   = integer work array of length at least..
c                  20         for mf = 10,
c                  20 + neq   for mf = 21 or 22
c liw     = declared length of iwork (in user-s dimension).
c jac     = name of subroutine for jacobian matrix.
c mf      = method flag.  standard values are..
c             10 for nonstiff (adams) method, no jacobian used.
c             22 for stiff method, internally generated full jacobian.
c note that the main program must declare arrays y, rwork, iwork,
c and possibly atol.
c
c the output from the first call (or any call) is..
c      y  = array of computed values of y(t) vector.
c      t  = corresponding value of independent variable (normally tout).
c istate  = 2  if lsode was successful, negative otherwise.
c             -1 means excess work done on this call, perhaps wrong mf.
c             -2 means excess accuracy requested (tolerances too small).
c             -3 means illegal input detected (see printed message).
c             -4 means repeated error test failures (check all inputs).
c             -5 means repeated convergence failures, perhaps bad jac.
c                supplied or wrong choice of mf or tolerances).
c             -6 means error weight became zero during problem (solution
c                component i vanished, and atol or atol(i) = 0.)
```