15

Numerical Algorithms (Draft), 2015-09-06

15.1 Numerical Integration of ODEs

For numerical considerations ordinary differential equations of higher are mostly reduced to a system of equations of first order. For example the second order equation

$$y'' = f(x, y, y')$$

can be rewritten as a first order system

$$\begin{aligned} y' &= z \\ z' &= f(x, y, z) \end{aligned}$$

for the function (y, z). This general property of ODEs allows general numerical routines to restrict to the solution of generic first order systems.

Definition 1 *Let $I \subset \mathbf{R}$ an interval, $\Omega \subset \mathbf{R}^n$ an open domain and $f : I \times \Omega \to \mathbf{R}^n$ a continuous map. Then*

$$\begin{aligned} y' &= f(x, y) \\ y(x_0) &= y_0 \end{aligned}$$

is the generic form of a first order ordinary differential equation *for the unkown vector-valued function $y : I \to \mathbf{\Omega}$.*

If f is lipschitz then y is uniquely determined as proved for example in the theorem of Picard-Lindelöf. In the following we introduce some single step methods for ODE integration which belong to the working horses of every numerical library.

15.1.1 Euler Method

The Euler method is the simplest ODE integrator and also least accurate method. Consider the power series expansion of y and suppress all terms of order greater than one. From the representation $y(x+h) = y(x) + hf(x,y) + O(h^2)$ we derive the Euler method

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

The geometric interpreation of the Euler method is to start at a given point (x_i, y_i) and to proceed to the next point by following the tangent direction $f(x_i, y_i)$ at (x_i, y_i) a given stepsize h as shown in figure **??**. The Euler method is very inaccurate and its most use is of theoretical nature.

15.1.2 Runge Kutta Method

The class of Runge Kutta methods is constructed to eliminate higher order error terms by evaluating differential equation in each step at additional intermediate points in the interval $[x_n, x_{n+1}]$. The most popular of these methods is the first order Runge Kutta method we will discuss in greater detail.

Definition 2 *A generic single step method is defined as*

$$y_{n+1} = y_n + h\phi(x_n, y_{n,}, f, h).$$

The function ϕ is called method function.

The fourth order Runge Kutta method requires four evaluations of the function f, one at x_n and at $x_n + h$ and two at $x_n + \frac{h}{2}$. One starts at x_n with the initial direction and proceeds to $x_n + \frac{h}{2}$. The tangent direction at that point is used to start at x_n again with the new tangent direction and proceeds again to $x_n + \frac{h}{2}$. One repeats the process, but in this last step one proceeds to $x_n + h$ and evaluates f there. Now one has determined four tangent directions and computes a weighted mean value which is then used to proceed from x_n to $x_n +$

h. The following equations show the construction formally, compare also figure **??** for a graphical explanation.

$$\begin{aligned} v_1 &= f(x_n, y_n) \\ v_2 &= f(x_n + \frac{h}{2}, y_n + \frac{h}{2} v_1) \\ v_3 &= f(x_n + \frac{h}{2}, y_n + \frac{h}{2} v_2) \\ v_4 &= f(x_n + h, y_n + h v_3). \end{aligned}$$

The fourth order Runge Kutta method is then given by

$$y_{n+1} = y_n + \frac{h}{6}(v_1 + 2v_2 + 2v_3 + v_4).$$

The following numerical routine `rungeKutta` computes a single Runge Kutta step based on the formulas above.

```
rungeKutta(double x, double [] y, int dim, double h, void (*deriv)())
{
  int i;
  double [][] dy = new double[4][dim];
  (*deriv)(x, y, dy[0]);
  for (i=0; i<dim; i++)
    ytemp[i] = y[i]+h/2*dy[0][i];
    (*deriv)(x+h/2, ytemp, dy[1]);
  for (i=0; i<dim; i++)
    ytemp[i] = y[i]+h/2*dy[1][i];
    (*deriv)(x+h/2, ytemp, dy[2]);
  for (i=0; i<dim; i++)
    ytemp[i] = y[i]+h*dy[2][i];
    (*deriv)(x+h, ytemp, dy[3]);
  for (i=0; i<dim; i++)
    yOut[i] = y[i]+h/6*(dy[0][i]+2dy[1][i]+2dy[2][i]+dy[3][i]);
}
```

15.2 Solving Variational Problems

15.2.1 Raleigh-Ritz-Galerkin Method

Consider the variational problem of minimizing the functional

$$J(v) := \frac{1}{2} a(v, v) - l(v) \qquad (15.1)$$

in an affine space V whose unique solution exists by theorem 15.4.99/2. Let $S_h \subset V$ be a finite dimensional affine subspace in which a and l are defined, and let $u_h \in S_h$ solve

$$a(u_h, v) = l(v) \quad \forall v \in S_h. \tag{15.2}$$

Let $\{\psi_1, ..., \psi_N\}$ be a basis of S_h such that

$$S_h = span\{\psi_1, ..., \psi_N\}.$$

Then there exists a unique coefficient vector $(u_1, ..., u_N) \subset \mathbb{R}^N$ of u_h such that

$$u_h(x) = \sum_{j=1}^{N} u_j \psi(x)$$

and $u_i = \psi(x_i)$. Using this representation, equation 15.2 is equivalent to the system of N equations

$$\sum_{j=1}^{N} a(\psi_j, \psi_i) u_j = l(\psi_i) \quad \forall i \in \{1, ..N\},$$

and can be written in matrix form

$$A \cdot u_h = b$$

with $A_{ij} := a(\psi_i, \psi_i)$ and $b_i := l(\psi_i)$. The matrix A is called the *stiffness matrix* or *system matrix*. A is positive definit since ${}^t u_h A u_h = a(u_h, u_h) \geqslant \alpha \|u_h\|_m^2 \ \forall u_h \in S_h$ if a is an H^m-elliptic bilinear form. The Raleigh-Ritz method solves 15.1 in S_h by directly solving

$$\nabla J(u) = \left(\frac{\partial}{\partial u_1} J(u), ..., \frac{\partial}{\partial u_N} J(u) \right) = \frac{1}{2} \left({}^t u \cdot A + {}^t (Au) \right) - {}^t b \overset{!}{=} 0$$

Instead of solving a linear system one prefers gradient methods if the dimension of S_h is higher. Here an initial guess u_0 is iteratively replaced by optimized values $u_{i+1} := u_i - h \cdot \widetilde{\nabla} J(u_i)$ where $\widetilde{\nabla} J(u_i)$ is a modified gradient of J at u_i, for example, the conjugate gradient.

Remark 3 *The solution $u_h \in S_h$ has shortest distance to the solution $u \in V$ if V is equipped with the bilinear form a as metric, i.e. $u - u_h$ is $a-$orthogonal to S_h*

$$a(u - u_h, v_h) = 0.$$

Proof. Since u is a solution in V we have

$$a(u, v_h) = l(v_h),$$

and since u_h is a solution in S_h

$$a(u_h, v_h) = l(v_h),$$

the assumption follows directly. □
Type: 22.4.99/2 - 23.4.99/3

15.2.2 Steepest Descent vs. Conjugate Gradient Method

Line Minimization: Given $p \in \mathbb{R}^n$, direction $n \in \mathbb{R}^n$ and an energy functional $E : \mathbb{R}^n \to \mathbb{R}$. Find a scalar λ that minimizes

$$E(p + \lambda n) \to \min.$$

Then replace p by $p + \lambda n$.
Conjugate Gradient Method: Taylor expansion around p gives

$$\begin{aligned} E(x) &= E(p) + \nabla E_p(x) + \frac{1}{2}\nabla^2 E_p(x, x) + \dots \\ &\approx c - bx + \frac{1}{2}x^t A x \end{aligned}$$

Quadratic function E:

$$\nabla E(x) = Ax - b$$

How does the gradient change along some direction ν?

$$\partial_v \nabla E = A \cdot \partial_v x = Av$$

Rule: Assume we have moved along some direction u to a minimum and now want to move along a new direction v. Then v shall not spoil our previous minimization, i.e. the change of the gradient shall be perpendicular to u, i.e.

$$0 =< u, \partial_v (\nabla E) >= uAv$$

Directions u and v are called conjugate.
Construction of Conjugate Directions
Let A be a positive-definite, symmetric nxn matrix. Let g_0 be an arbitrary vector, and $h_0 = g_0$. For $i = 0, 1, 2, \dots$ define the two sequences of vectors

$$\begin{aligned} g_{i+1} &= g_i - \lambda_i A h_i \\ h_{i+1} &= g_{i+1} + \gamma_i h_i, \end{aligned}$$

where λ_i resp. γ_i are chosen to fulfill $g_{i+1} \cdot g_i = 0$ resp. $h_{i+1} A h_i = 0$, i.e.

$$\lambda_i = \frac{g_i \cdot g_i}{g_i A h_i} \qquad \gamma_i = -\frac{g_{i+1} A h_i}{h_i A h_i}.$$

If denominators are zero take $\lambda_i = 0$ resp. $\gamma_i = 0$. Then

$$g_i \cdot g_{i+1} = 0 \qquad h_i A h_j = 0 \qquad \forall i \neq j$$

and the bi-orthogonalization procedure has produced a sequence g_i where each g_i is orthogonal and each h_i is conjugate to its set of predecessors.

Problem: A is generally not known!

Take: $g_i = -\nabla E|_{p_i}$ for some point p_i