Genetic algorithms application

Master degree in Modelling for Science and Engineering

January 14, 2020

This assignment is based on [McC05, Example 5.3].

1 A model for cancer treatment

Consider $N(t)$ the number of tumor cells at time t, which, according to the Gompertz growth model, can be modeled with the following differential equation *defined only for* $N(t) \geq 0$:

$$\begin{aligned} N'(t) &= N(t)\left(\lambda \ln\left(\frac{\Theta}{N(t)}\right) - \sum_{j=1}^{d} k_j \sum_{i=0}^{n-1} C_{ij}\chi_{\left(\tau_i,\tau_{i+1}\right]}(t)\right) \\ &= N(t)\left(\lambda\ln(\Theta) - \lambda\ln\big(N(t)\big) - \mathsf{drift}_i\right) \quad \text{whenever } t \in \left(\tau_i,\tau_{i+1}\right], \end{aligned}$$

where:

- $\mathsf{drift}_i = \sum_{j=1}^{d} k_j C_{ij}$;
- λ and Θ are general parameters of the model, being λ the sensitivity or "evolution speed" of the tumour and Θ a kind of a carrying capacity;
- n is the number of doses or treatment sessions, given at times $\tau_0 = 0 < \tau_1 < \cdots < \tau_{n-1}$. The dose at time τ_{n-1} is supposed to act until a given time $\tau_n > \tau_{n-1}$;
- $\chi_{\left(\tau_i,\tau_{i+1}\right]}$ is the *indicator function on the interval* $\left(\tau_i,\tau_{i+1}\right]$:

$$\chi_{\left(\tau_i,\tau_{i+1}\right]}(t) = \begin{cases} 1 & \text{if } t \in \left(\tau_i,\tau_{i+1}\right], \text{ and} \\ 0 & \text{otherwise;} \end{cases}$$

- d is the number of drugs in each dose;
- C_{ij} with $i \in \{0,1,\ldots,n-1\}$ and $j \in \{1,2,\ldots,d\}$ denotes the concentration of drug j at the dose i;
- k_j with $j \in \{1,2,\ldots,d\}$ is the "effectivity" of the drug j and is estimated by means of clinical trial results.

Observe that, the expression $N'(t) = N(t)\left(\lambda\ln(\Theta) - \lambda\ln\big(N(t)\big) - \mathsf{drift}\right)$ vanishes at $\Theta\cdot\exp\left(\frac{-\mathsf{drift}}{\lambda}\right)$, whenever $N(t) > 0$. So, the solution $N(t)$ of the above ODE is

$$\begin{cases} \text{strictly increasing whenever} & \Theta\cdot\exp\left(\frac{-\mathsf{drift}_i}{\lambda}\right) > N(0), \text{ and} \\ \text{strictly decreasing whenever} & \Theta\cdot\exp\left(\frac{-\mathsf{drift}_i}{\lambda}\right) < N(0). \end{cases}$$

In any case, $\Theta\cdot\exp\left(\frac{-\mathsf{drift}}{\lambda}\right)$ is a horizontal assymptote of $N(t)$ as t goes to infinity.

There are also some general restrictions for the above model:

A priori constraints (on the dose concentrations):

Constraint 1: There is a maximum concentration for each drug $C_{\max,j}$, so that $C_{ij} \leq C_{\max,j}$ for $i = 0,1,\ldots,n-1$.

Constraint 2: There is also a maximum of the cumulative concentracion for each drug $C_{\text{cum},j}$. So, $\sum_{i=0}^{n-1} C_{ij} \leq C_{\text{cum},j}$.

Constraint 3: As a side effect, there are m important organs such that the organ $k \in \{1, 2, \ldots, m\}$ can assume a maximum $C_{\text{s-eff},k}$ of damage, and each drug j damages the organ k an amount of η_{kj} per concentration unit. So,

$$C_{\text{s-eff},k} \geq \sum_{j=1}^{d} \eta_{kj} C_{ij}$$

for every i, k such that $0 \leq i \leq n-1$ and $1 \leq k \leq m$. Of course, we can multiply the parameters n_{jk} by a constant depending on k such that we can normalize $C_{\text{s-eff},k} = 1$.

Constraints on the effect of the doses:

Constraint 4: There is a maximum in the number of tumour cells $N_{\max}$, so, $N_{\max} \geq N(t)$ for all t.

2 Aim of the exercise

At first we want to find a curative treatment which, among the curative options, it is the most effective in the sense that it *minimises* the following fitness function:

$$\int_{\tau_0}^{\tau_\ell} N(t)\mathrm{d}t.$$

We consider that the solution is curative if $N(\tau_i) \leq 1000$ for 3 consecutive values $\tau_{\ell-2}, \tau_{\ell-1}$ and τ_ℓ with $\ell \in \{2, 4, \ldots, n\}$ (from [McC99]).

If a curative treatment does not exist we want to find a palliative treatment that *maximises* the lifespan of the patient. A palliative treatment has no dose or treatment neither at time τ_n nor later on. So, the term $\sum_{j=1}^{d} k_j C_{ij}$ in Gompertz model must be taken equals to zero for $t > \tau_n$.

We consider that the lifespan of the patient terminates when Constraint 4 is violated.

3 Parameters of the model

Some of the parameters are taken from [McC99], while others are (computationally) experimental:

- $\Theta = 10^{12}$ (from [McC99]).
- $\lambda = 0.336$.
- $N(0) = 20000$ (from the picture in [McC99]).
- Number of drugs: $d = 10$.
- Parameters $\vec{k} = [k_1, \ldots, k_{10}]$:

$$\vec{k} = [0.12, 0.0502, 0.0637, 0.1347, 0.0902, 0.0546, 0.0767, 0.1121, 0.0971, 0.0403].$$

- Number of doses: $n = 10$, with equal separation: $T = \tau_{i+1} - \tau_i = 3$ for $i = 0, 1, \ldots, n-2$. So, $\tau_0 = 0$, $\tau_1 = 3$, $\ldots \tau_8 = 24$ and the treatment finishes at $\tau_9 = 27$. We take $\tau_n = \tau_{10} = 33$ (invented).
- **A posteriori Constraint 4:** $N_{\max} = 0.95 \cdot \Theta = 9.510^{11}$ (invented).
- **A priori Constraint 1:** $C_{\max,j} = 15$ for all j.
- **A priori Constraint 2:** $C_{\text{cum},j} = 127$ for all j (invented).
- **A priori Constraint 3:** We consider $m = 4$ different organs which cannot be killed by the treatment with all $C_{\text{s-eff},k} = 1$. Here we propose two different cases of parameters η_{kj}:
  The first case corresponds to:

$k \setminus j$	1	2	3	4	5	6	7	8	9	10
1	0.0036	0.0098	0.0061	0.0009	0.0003	0.0108	0.0045	0.0021	0.0096	0.0125
2	0.0063	0.0082	0.0062	0.0062	0.0083	0.013	0.0039	0.0019	0.0015	0.005
3	0.0129	0.0018	0.0116	0.0021	0.009	0.0129	0.0054	0.0049	0.0093	0.0066
4	0.0053	0.0086	0.0067	0.0029	0.0089	0.0054	0.0042	0.0095	0.0112	0.0092

where we have been able to find curative treatments.

The second case corresponds to:

$k \setminus j$	1	2	3	4	5	6	7	8	9	10
1	0.00612	0.01666	0.01037	0.00153	0.00051	0.01836	0.00765	0.00357	0.01632	0.02125
2	0.01071	0.01394	0.01054	0.01054	0.01411	0.0221	0.00663	0.00323	0.00255	0.0085
3	0.02193	0.00306	0.01972	0.00357	0.0153	0.02193	0.00918	0.00833	0.01581	0.01122
4	0.00901	0.01462	0.01139	0.00493	0.01513	0.00918	0.00714	0.01615	0.01904	0.01564

where we have just found palliative treatments.

4 Proposed solution strategy

The idea is to try to solve the first problem (finding a curative treatment) and, if it turns to be infeasible, then solve the second one. Both problems are to be solved with minimising genetic algorithms with appropriate fitness functions. Notice that, for the second problem, an inversion of the lifespan must be done since we want to *maximise* it.

Inspired by the penalty methods, infeasible individuals will be assigned with a very high fitness (in any case, with a value larger than the maximum possible integral $\int_{\tau_0}^{\tau_{10}} N(t)\mathrm{d}t$ or inverse of lifespan.

To do this it is necessary to write two fitness functions that have as input an individual from the population and return the fitness of this individual. These functions must evaluate feasibility of the individual and penalize infeasibility, and compute the numerically the appropriate solution of the Gompertz model and evaluate either $\int_{\tau_0}^{\tau_\ell} N(t)\mathrm{d}t$ or lifespan.

A problem is considered to be *infeasible* when for 10 consecutive generations all individuals are infeasible.

The recommended stoping rule is that the best solution (best fitness) does not improve during four consecutive feasible generations (meaning that contain some feasible individuals).

Next we discuss different aspect implementation aspects of the strategy.

4.1 Individuals

Clearly, an individual in the population must be a vector of dimension $d \times n$ listing all independent variables C_{ij}; the dose concentrations for all the treatments. Taking into account Constraint 1: $C_{\max,j} = 15$ for all j, we will use unsigned integer of 4 bits for each entry of the vector, when written in binary (in other words Constraint 1 is "hardcoded"). Since there is no such datatype this vector must be of `unsigned char` type (and waste half of the memory — to avoid this wasting of memory seems too complicate). The suggested organization for an "individual" vector is:

$$\big[C_{0,1}, C_{0,2}, \ldots, C_{0,d}, C_{1,1}, C_{1,2}, \ldots, C_{1,d}, \ldots, C_{n-1,1}, C_{n-1,2}, \ldots, C_{n-1,d}\big].$$

4.2 Computing the integral $\int_{\tau_0}^{\tau_\ell} N(t)\mathbf{d}t$

The methods to integrate ODE's give its solution at discrete values of time:

$$N(t_0), N(t_1), \ldots, N(t_i), \ldots, N(t_r)$$

with $t_0 = 0 < t_1 < t_2 < \cdots < t_i < \cdots < t_r = t_{\text{fin}}$. Usually, neighbouring values of t_i are very close to each other. This implies that the trapezoidal rule is enough to compute a very good approximation of an integral. However, the very nice usual trapezoidal rule formulae cannot be applied here since, in general, the points t_i are not equispaced.

Clearly,

$$\int_0^{t_{\text{fin}}} N(t)\mathrm{d}t = \sum_{i=0}^{r-1} \int_{t_i}^{t_{i+1}} N(t)\mathrm{d}t \approx \sum_{i=0}^{r-1} \frac{N(t_{i+1}) + N(t_i)}{2}\big(t_{i+1} - t_i\big) = \frac{1}{2}\sum_{i=0}^{r-1}\big(N(t_{i+1}) + N(t_i)\big)\big(t_{i+1} - t_i\big).$$

4.3 Integrating and ODE

We will use the Runge-Kutta-Fehlberg method of order 7-8 with adaptive space. See the appendix to this document.

In the file `RKF78.c` (also needed `RKF78.h` for definitions and prototypes) there is an implementation for ODE's and another one for systems (see the implementation notes in `RKF78.c` for the meaning of parameters and how to use the procedure).

However as an example on how to use `RKF78` and write the fitness function we proivide here an implementation.

```
typedef struct { double drift_i; } ODE_Parameters;
void Gompertz(double t, double N, double *der, void *Params){
    *der = ((N < 1.e-16) ? 0.0 :
         N*(lambdalogTheta_par - lambda_par*log(N) - ((ODE_Parameters *) Params)->drift_i));
}
```

NOTE: The construction `((N < 1.e-16) ? 0.0 :` is crucial to avoid bad behaviour for rounding errors.

```
/* Tumour Parameters */
#define lambda_par 0.336
#define lambdalogTheta_par 9.284023094951992774680543277273261947643
#define NZero_par 20000
#define CCUMj_par 127
#define NMax_par 9.5e+11
#define m_par 4
#define d_par 10
#define n_par 11

static float k_j[d_par] = {0.12, 0.0502, 0.0637, 0.1347, 0.0902, 0.0546, 0.0767, 0.1121,
                           0.0971, 0.0403};
static float t_i[n_par] = {0.0, 3.0, 6.0, 9.0, 12.0, 15.0, 18.0, 21.0, 24.0, 27.0, 33.0};
static float eta_kj[m_par][d_par] = {
     {0.0036, 0.0098, 0.0061, 0.0009, 0.0003, 0.0108, 0.0045, 0.0021, 0.0096, 0.0125},
     {0.0063, 0.0082, 0.0062, 0.0062, 0.0083, 0.013, 0.0039, 0.0019, 0.0015, 0.005},
     {0.0129, 0.0018, 0.0116, 0.0021, 0.009, 0.0129, 0.0054, 0.0049, 0.0093, 0.0066},
     {0.0053, 0.0086, 0.0067, 0.0029, 0.0089, 0.0054, 0.0042, 0.0095, 0.0112, 0.0092} };

double Curative_Fitness(unsigned char *Cij){
  register unsigned char i, j;
  ODE_Parameters GompertzParams;
  double N = NZero_par, t = t_i[0];
  double hmin = 1.e-8, hmax = 1.0, h = 1.e-3, tol = 1.e-8;
  unsigned char curativecounter = 0U, npar = n_par - 1;
  double integral = 0.0, lastt = t, lastN = N;

  if(!TestIfConstraints2and3AreVerified(Cij)) return MAXDOUBLE;

  for(i=0; i < npar; i++){ double tfin = t_i[i+1]; // Implementing treatment i
    GompertzParams.drift_i = 0.0;
    for(j=0; j < d_par; j++) GompertzParams.drift_i += k_j[j] * *(Cij++);

    while(t+h < tfin) {
      RKF78(&t, &N, &h, hmin, hmax, tol, &GompertzParams, Gompertz);
      if(N > NMax_par) return MAXDOUBLE;
      integral += (lastN + N)*(t - lastt);
      lastt = t; lastN = N;
    }

    do { h = tfin - t;
         RKF78(&t, &N, &h, hmin, hmax, tol, &GompertzParams, Gompertz);
         if(N > NMax_par) return MAXDOUBLE;
         integral += (lastN + N)*(t - lastt);
         lastt = t; lastN = N;
    } while (t < tfin);
    if(N < 1000) { curativecounter++; if(curativecounter > 2) return integral/2.0; }
    else curativecounter = 0U;
  }
  return MAXDOUBLE;
}
```

For the second family of parameters consider to replace the corresponding code by:

```
static float eta_kj[m_par][d_par] =
  {{0.00612, 0.01666, 0.01037, 0.00153, 0.00051, 0.01836, 0.00765, 0.00357, 0.01632, 0.02125},
  {0.01071, 0.01394, 0.01054, 0.01054, 0.01411, 0.0221, 0.00663, 0.00323, 0.00255, 0.0085},
  {0.02193, 0.00306, 0.01972, 0.00357, 0.0153, 0.02193, 0.00918, 0.00833, 0.01581, 0.01122},
  {0.00901, 0.01462, 0.01139, 0.00493, 0.01513, 0.00918, 0.00714, 0.01615, 0.01904, 0.01564}
};
```

NOTE: The computation of drift_i with pointer arithmetic is based with:

$$\mathsf{drift}_i := \sum_{j=1}^{d} k_j * C_{ij} = \sum_{j=0}^{\texttt{d\_par}-1} \texttt{k\_j[j] * Cij[i*d\_par+j]}.$$

```
unsigned char TestIfConstraints2and3AreVerified(unsigned char *Cij){
  register unsigned char i, j, k;
  unsigned char npar = n_par - 1, *Cijofi;

  for(j=0; j < d_par; j++){ unsigned int ccumj = 0U;
    for(i=0; i < npar; i++) ccumj += *(Cij + i*d_par + j);
    if(ccumj > CCUMj_par) return 0U;
  }

  for(i=0, Cijofi=Cij; i < npar; i++, Cijofi += d_par) {
    for(k=0; k < m_par; k++){ double Cseffk = 0.0;
      for(j=0; j < d_par; j++) Cseffk += eta_kj[k][j] * Cijofi[j];
      if(Cseffk > 1.0) return 0U;
    }
  }
  return 1U;
}
```

References

[McC99] A McCall, John & Petrovski. A decision support system for cancer chemoterapy using genetic algorithms. *Computational Intelligence for Modelling Control & Automation. M. Mohammadian (Ed). IOS Press.*, pages 65–70, 1999.

[McC05] John McCall. Genetic algorithms for modelling and optimisation. *J. Comput. Appl. Math.*, 184(1):205–222, 2005.

Appendix A
Runge-Kutta Methods

The Runge-Kutta methods are an important family of iterative methods for the approximation of solutions of ODE's, that were develoved around 1900 by the german mathematicians C. Runge (1856–1927) and M.W. Kutta (1867–1944). We start with the considereation of the explicit methods. Let us consider an initail value problem (IVP)

$$\frac{d\,\mathbf{x}}{d\,t} = f(t, \mathbf{x}(t)), \tag{A.1}$$

$\mathbf{x}(t) = (x_1(t), x_2(t), \ldots x_n(t))^T$, $f \in [a,b] \times \mathbb{R}^n \to \mathbb{R}^n$, with an initial condition

$$\mathbf{x}(0) = \mathbf{x}_0\,. \tag{A.2}$$

We are interested in a numerical approximation of the continuously differentiable solution $\mathbf{x}(t)$ of the IVP (A.1)–(A.2) over the time interval $t \in [a,b]$. To this aim we subdivide the interval $[a,b]$ into M equal subintervals and select *the mesh points* t_j [11, 8]

$$t_j = a + j\,h\,, \qquad j = 0, 1, \ldots, M, \quad h = \frac{b-a}{M}\,. \tag{A.3}$$

The value h is called *a step size*.

The family of explicit Runge–Kutta (RK) methods of the m'th stage is given by [11, 9]

$$\mathbf{x}(t_{n+1}) := \mathbf{x}_{n+1} = \mathbf{x}_n + h\sum_{i=1}^{m} c_i k_i\,, \tag{A.4}$$

where

$$
\begin{aligned}
k_1 &= f(t_n, \mathbf{x}_n), \\
k_2 &= f(t_n + \alpha_2 h, \mathbf{x}_n + h\beta_{21} k_1(t_n, \mathbf{x}_n)), \\
k_3 &= f(t_n + \alpha_3 h, \mathbf{x}_n + h(\beta_{31} k_1(t_n, \mathbf{x}_n) + \beta_{32} k_2(t_n, \mathbf{x}_n))), \\
&\vdots \\
k_m &= f(t_n + \alpha_m h, \mathbf{x}_n + h \sum_{j=1}^{m-1} \beta_{mj} k_j).
\end{aligned}
$$

To specify a particular method, we need to provide the integer m (the number of stages), and the coefficients α_i (for $i = 2, 3, ..., m$), β_{ij} (for $1 \le j < i \le m$), and c_i (for $i = 1, 2, ..., m$). These data are usually arranged in a co-called *Butcher tableau* (after John C. Butcher) [11, 9]:

Table A.1 The Butcher tableau.

$$
\begin{array}{c|cccccc}
0 & & & & & & \\
\alpha_2 & \beta_{21} & & & & & \\
\alpha_3 & \beta_{31} & \beta_{32} & & & & \\
\vdots & \vdots & \vdots & \ddots & & & \\
\vdots & \vdots & \vdots & & & & \\
\alpha_m & \beta_{m1} & \beta_{m2} & \cdots\cdots & \beta_{mm-1} & & \\
\hline
 & c_1 & c_2 & \cdots\cdots & c_{m-1} & c_m &
\end{array}
$$

Examples

1. Let $m = 1$. Then

$$
\begin{aligned}
k_1 &= f(t_n, \mathbf{x}_n), \\
\mathbf{x}_{n+1} &= \mathbf{x}_n + h c_1 f(t_n, \mathbf{x}_n).
\end{aligned}
$$

   On the other hand, the Taylor expansion yields

$$
\mathbf{x}_{n+1} = \mathbf{x}_n + h\dot{\mathbf{x}}\big|_{t_n} + \cdots = \mathbf{x}_n + h f(t_n, \mathbf{x}_n) + \mathcal{O}(h^2) \Rightarrow c_1 = 1.
$$

   Thus, the first-stage RK-method is equivalent to the explicit Euler's method. Note that the Euler's method is of the first order of accuracy. Thus we can speak about the RK method of the first order.
2. Now consider the case $m = 2$. In this case Eq. (A.4) is equivalent to the system

$$
\begin{aligned}
k_1 &= f(t_n, \mathbf{x}_n)\,, \\
k_2 &= f(t_n + \alpha_2 h, \mathbf{x}_n + h\beta_{21} k_1)\,, \\
\mathbf{x}_{n+1} &= \mathbf{x}_n + h(c_1 k_1 + c_2 k_2)\,.
\end{aligned}
\tag{A.5}
$$

Now let us write down the Taylor series expansion of $\mathbf{x}$ in the neighborhood of t_n up to the h^2 term, i.e.,

$$
\mathbf{x}_{n+1} = \mathbf{x}_n + h \left.\frac{d\mathbf{x}}{dt}\right|_{t_n} + \frac{h^2}{2} \left.\frac{d^2\mathbf{x}}{dt^2}\right|_{t_n} + \mathscr{O}(h^3)\,.
$$

However, we know that $\dot{\mathbf{x}} = f(t, \mathbf{x})$, so that

$$
\frac{d^2\mathbf{x}}{dt^2} := \frac{df(t,\mathbf{x})}{dt} = \frac{\partial f(t,\mathbf{x})}{\partial t} + f(t,\mathbf{x}) \frac{\partial f(t,\mathbf{x})}{\partial \mathbf{x}}\,.
$$

Hence the Taylor series expansion can be rewritten as

$$
\mathbf{x}_{n+1} - \mathbf{x}_n = h f(t_n, \mathbf{x}_n) + \frac{h^2}{2} \left(\frac{\partial f}{\partial t} + f \frac{\partial f}{\partial \mathbf{x}} \right) \Bigg|_{(t_n, \mathbf{x}_n)} + \mathscr{O}(h^3)\,. \tag{A.6}
$$

On the other hand, the term k_2 in the proposed RK method can also expanded to $\mathscr{O}(h^3)$ as

$$
k_2 = f(t_n + \alpha_2 h, \mathbf{x}_n + h\beta_{21} k_1) = h f(t_n, \mathbf{x}_n) + h\,\alpha_2 \left.\frac{\partial f}{\partial t}\right|_{(t_n, \mathbf{x}_n)} + h\beta_{21} f \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{(t_n, \mathbf{x}_n)} + \mathscr{O}(h^3)\,.
$$

Now, substituting this relation for k_2 into the last equation of (A.5), we achieve the following expression:

$$
\mathbf{x}_{n+1} - \mathbf{x}_n = h\,(c_1 + c_2)\, f(t_n, \mathbf{x}_n) + h^2 c_2\,\alpha_2 \left.\frac{\partial f}{\partial t}\right|_{(t_n, \mathbf{x}_n)} + h^2 c_2 \beta_{21} f \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{(t_n, \mathbf{x}_n)} + \mathscr{O}(h^3)\,.
$$

Making comparision the last equation and Eq. (A.6) we can write down the system of algebraic equations for unknown coefficients

$$
\begin{aligned}
c_1 + c_2 &= 1\,, \\
c_2\,\alpha_2 &= \frac{1}{2}\,, \\
c_2\,\beta_{21} &= \frac{1}{2}\,.
\end{aligned}
$$

The system involves four unknowns in three equations. That is, one additional condition must be supplied to solve the system. We discuss two useful choices, namely

a) Let $\alpha_2 = 1$. Then $c_2 = 1/2$, $c_1 = 1/2$, $\beta_{21} = 1$. The corresponding Butcher tableau reads:

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 1/2 & 1/2 \end{array}$$

Thus, in this case the two-stages RK method takes the form

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{2}\Big(f(t_n, \mathbf{x}_n) + f(t_n + h, \mathbf{x}_n + h f(t_n, \mathbf{x}_n)) \Big),$$

and is equivalent to the Heun's method, so we refer the last method to as RK-method of the second order.

b) Now let $\alpha_2 = 1/2$. In this case $c_2 = 1$, $c_1 = 0$, $\beta_{21} = 1/2$. The corresponding Butcher tableau reads:

$$\begin{array}{c|cc} 0 & & \\ 1/2 & 1/2 & \\ \hline & 0 & 1 \end{array}$$

In this case the second-order RK method (A.4) can be written as

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\, f\big(t_n + \frac{h}{2}, \mathbf{x}_n + \frac{h}{2} f(t_n, \mathbf{x}_n)\big)$$

and is called the *RK2 method*.

RK4 Methods

One member of the family of Runge–Kutta methods (A.4) is often referred to as *RK4 method* or *classical RK method* and represents one of the solutions corresponding to the case $m = 4$. In this case, by matching coefficients with those of the Taylor series one obtains the following system of equations [8]

$$
\begin{aligned}
c_1 + c_2 + c_3 + c_4 &= 1\,, \\
\beta_{21} &= \alpha_2\,, \\
\beta_{31} + \beta_{32} &= \alpha_3\,, \\
c_2\alpha_2 + c_3\alpha_3 + c_4\alpha_4 &= \frac{1}{2}\,, \\
c_2\alpha_2^2 + c_3\alpha_3^2 + c_4\alpha_4^2 &= \frac{1}{3}\,, \\
c_2\alpha_2^3 + c_3\alpha_3^3 + c_4\alpha_4^3 &= \frac{1}{4}\,, \\
c_3\alpha_2\beta_{32} + c_4(\alpha_2\beta_{42} + \alpha_3\beta_{43}) &= \frac{1}{6}\,, \\
c_3\alpha_2\alpha_3\beta_{32} + c_4\alpha_4(\alpha_2\beta_{42} + \alpha_3\beta_{43}) &= \frac{1}{8}\,, \\
c_3\alpha_2^2\beta_{32} + c_4(\alpha_2^2\beta_{42} + \alpha_3^2\beta_{43}) &= \frac{1}{12}\,, \\
c_4\alpha_2\beta_{32}\beta_{43} &= \frac{1}{24}\,.
\end{aligned}
$$

The system involves thirteen unknowns in eleven equations. That is, two additional condition must be supplied to solve the system. The most useful choices is [9]

$$
\alpha_2 = \frac{1}{2}\,, \qquad \beta_{31} = 0\,.
$$

The corresponding Butcher tableau is presented in Table A.2. The tableau A.2 yields

Table A.2 The Butcher tableau corresponding to the RK4 method.

0				
1/2	1/2			
1/2	0	1/2		
1	0	0	1	
	1/6	1/3	1/3	1/6

the equivalent corresponding equations defining the classical RK4 method:

$$
\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{6}\big(k_1 + 2k_2 + 2k_3 + k_4\big), \tag{A.7}
$$

where

$$
\begin{aligned}
k_1 &= f(t_n, \mathbf{x}_n),\\
k_2 &= f(t_n + \frac{h}{2}, \mathbf{x}_n + \frac{h}{2}k_1),\\
k_3 &= f(t_n + \frac{h}{2}, \mathbf{x}_n + \frac{h}{2}k_2),\\
k_4 &= f(t_n + h, \mathbf{x}_n + hk_3).
\end{aligned}
$$

This method is reasonably simple and robust and is a good general candidate for numerical solution of ODE's when combined with an intelligent adaptive step-size routine or an embedded methods (,e.g., so-called Runge-Kutta-Fehlberg methods (RKF45)).

Remark:

Notice that except for the classical method (A.7), one can also construct other RK4 methods. We mention only so-called $3/8$-*Runge-Kutta method*. The Brutcher tableau, corresponding to this method is presented in Table A.3.

Table A.3 The Butcher tableau corresponding to the $3/8$- Runge-Kutta method.

0				
1/3	1/3			
2/3	-1/3	1		
1	1	-1	1	
	1/8	3/8	3/8	1/8

Geometrical interpretation of the RK4 method

Let us consider a curve $\mathbf{x}(t)$, obtained by (A.7) over a single time step from t_n to t_{n+1}. The next value of approximation $\mathbf{x}_{n+1}$ is obtained ty integrating the slope function, i.e.,

$$
\mathbf{x}_{n+1} - \mathbf{x}_n = \int_{t_n}^{t_{n+1}} f(t, \mathbf{x})\, dt\,. \tag{A.8}
$$

Now, if the Simpson's rule is applied, the approximation to the integral of the last equation reads [10]

$$
\int_{t_n}^{t_{n+1}} f(t, \mathbf{x})\, dt \approx \frac{h}{6}\left(f(t_n, \mathbf{x}(t_n)) + 4\, f(t_n + \frac{h}{2}, \mathbf{x}(t_n + \frac{h}{2})) + f(t_{n+1}, \mathbf{x}(t_{n+1}))\right). \tag{A.9}
$$

On the other hand, the values k_1, k_2, k_3 and k_4 are approximations for slopes of the curve $\mathbf{x}$, i.e., k_1 is the slope of the left end of the interval, k_2 and k_3 describe two estimations of the slope in the middle of the time interval, whereas k_4 corresponds to the slope at the right. Hence, we can choose $f(t_n, \mathbf{x}(t_n)) = k_1$ and $f(t_{n+1}, \mathbf{x}(t_{n+1})) = k_4$, whereas for the value in the middle we choose the average of k_2 and k_3, i.e.,

$$f(t_n + \frac{h}{2}, \mathbf{x}(t_n + \frac{h}{2})) = \frac{k_2 + k_3}{2}.$$

Then Eq. (A.8) becomes

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{6}\left(k_1 + \frac{4(k_2 + k_3)}{2} + k_4\right),$$

which is equivalent to the RK4 schema (A.7).

Stage versus Order

The local truncation error ε for the method (A.7) can be estimated from the error term for the Simpson's rule (A.9) and equals [10, 8]

$$\varepsilon_{n+1} = -h^5 \frac{\mathbf{x}^{(4)}}{2880}.$$

Now we can estimate the final global error E, if we suppose that only the error above is presented. After M steps the accumulated error for the RK4 method reads

$$E(\mathbf{x}(b), h) = -\sum_{k=1}^{M} h^5 \frac{\mathbf{x}^{(4)}}{2880} \approx \frac{b-a}{2880}\mathbf{x}^{(4)} h = \mathscr{O}(h^4).$$

That is, the RK4 method (A.7) is of the fourth order. Now, let us compare two appximations, obtained using the time steps h and $h/2$. For the step size h we have

$$E(\mathbf{x}(b), h) \approx K h^4,$$

with $K = \text{const}$. Hence, for the step $h/2$ we get

$$E(\mathbf{x}(b), \frac{h}{2}) = K\frac{h^4}{16} \approx \frac{1}{16}E(\mathbf{x}(b), h).$$

That is, if the step size in (A.7) is reduced by the factor of two, the global error of the method will be reduced by the factor of $1/16$.

Remark:

In general there are two ways to improve the accuracy:

1. One can reduce the time step h, i.e., the amount of steps increases;
2. The method of the higher convergency order can be used.

However, increasing of the convergency order p is reasonable only up to some limit, given by so-called *Butcher barrier* [11], which says, that the amount of stages m grows faster, as the order p. In other words, *for $m \geq 5$ there are no explicit RK methods with the convergency order $p = m$ (the corresponding system is unsolvable)*. Hence, in order to reach convergency order five one needs six stages. Notice that further increasing of the stage $m = 7$ leads to the convergency order $p = 5$ as well.

A.0.1 Adaptive stepsize control and embedded methods

As mentioned above, one way to guarantee accuracy in the solution of (A.1)–(A.1) is to solve the problem twice using step sizes h and $h/2$. To illustrate this approach, let us consider the RK method of the order p and denote an exact solution at the point $t_{n+1} = t_n + h$ by $\widetilde{\mathbf{x}}_{n+1}$, whereas $\mathbf{x}_1$ and $\mathbf{x}_2$ represent the approximate solutions, corresponding to the step sizes h and $h/2$. Now let us perform one step with the step size h and after that two steps each of size $h/2$. In this case the true solution and two numerical approximations are related by

$$\widetilde{\mathbf{x}}_{n+1} = \mathbf{x}_1 + Ch^{p+1} + \mathscr{O}(h^{p+2}),$$
$$\widetilde{\mathbf{x}}_{n+1} = \mathbf{x}_2 + 2C\left(\frac{h}{2}\right)^{p+1} + \mathscr{O}(h^{p+2}).$$

That is,

$$|\mathbf{x}_1 - \mathbf{x}_2| = Ch^{p+1}\left(1 - \frac{1}{2^p}\right) \Leftrightarrow C = \frac{|\mathbf{x}_1 - \mathbf{x}_2|}{(1-2^{-p})h^{p+1}}.$$

Substituing the relation for C in the second estimate for the true solution we get

$$\widetilde{\mathbf{x}}_{n+1} = \mathbf{x}_2 + \varepsilon + \mathscr{O}(h^{p+2}),$$

where

$$\varepsilon = \frac{|\mathbf{x}_1 - \mathbf{x}_2|}{2^p - 1}$$

can be considered as a convenient *indicator* of the truncation error. That is, we have improved our estimate to the order $p+1$. For example, for $p = 4$ we get

$$\widetilde{\mathbf{x}}_{n+1} = \mathbf{x}_2 + \frac{|\mathbf{x}_1 - \mathbf{x}_2|}{15} + \mathscr{O}(h^6).$$

This estimate is accurate to fifth order, one order higter than with the original step h. However, this method is not efficient. First of all, it requires a significant amount

of computation (we should solve the equation three times at each time step). The second point is, that we have no possibility to control the truncation error of the method (higher order means not always higher accuracy).
However we can use an estimate ε for the *step size control*, namely we can compare ε with some *desired accuracy* ε_0 (see Fig A.1).

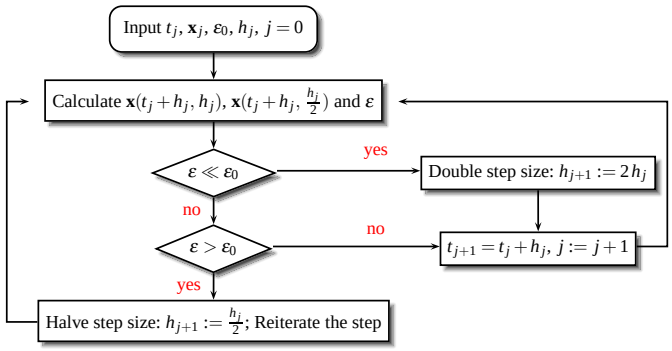

Fig. A.1 Flow diagramm of the step size control by use of the step doubling method.

Alternatively, using the estimate ε, we can try to formulate the following problem of the *adaptive step size control*, namely: Using the given values $\mathbf{x}_j$ and t_j, find the largest possible step size h_{new}, so that the truncation error after the step with this step size remains below some given desired accuracy ε_0, i.e,

$$C h_{new}^{p+1} \le \varepsilon_0 \Leftrightarrow \left(\frac{h_{new}}{h}\right)^{p+1} \frac{|\mathbf{x}_1 - \mathbf{x}_2|}{1-2^{-p}} \le \varepsilon_0 .$$

That is,

$$h_{new} = h\left(\frac{\varepsilon_0}{\varepsilon}\right)^{1/p+1} .$$

Then if the two answers are in close agreement, the approximation is accepted. If $\varepsilon > \varepsilon_0$ the step size has to be decreased, whereas the relation $\varepsilon < \varepsilon_0$ means, that the step size has to be increased in the next step.

Notice that because our estimate of error is not exact, we should put some "safety" factor $\beta \simeq 1$ [11, 9]. Usually, $\beta = 0.8, 0.9$. The flow diagramm, corresponding to the the adaptive step size control is shown on Fig. A.2

Notice one additional technical point. The choise of the desired error ε_0 depends on the IVP we are interested in. In some applications it is convinient to set ε_0 propotional to h [9]. In this case the exponent $1/p+1$ in the estimate of the new time step is no longer correct (if h is reduced from a too-large value, the new predicted value h_{new} will fail to meet the desired accuracy, so instead of $1/p+1$ we should scale with $1/p$ (see [9] for details)). That is, the optimal new step size can be written as

$$h_{new} = \begin{cases} \beta h\left(\frac{\varepsilon_0}{\varepsilon}\right)^{1/p+1}, & \varepsilon \ge \varepsilon_0, \\ \beta h\left(\frac{\varepsilon_0}{\varepsilon}\right)^{1/p}, & \varepsilon < \varepsilon_0, \end{cases} \tag{A.10}$$

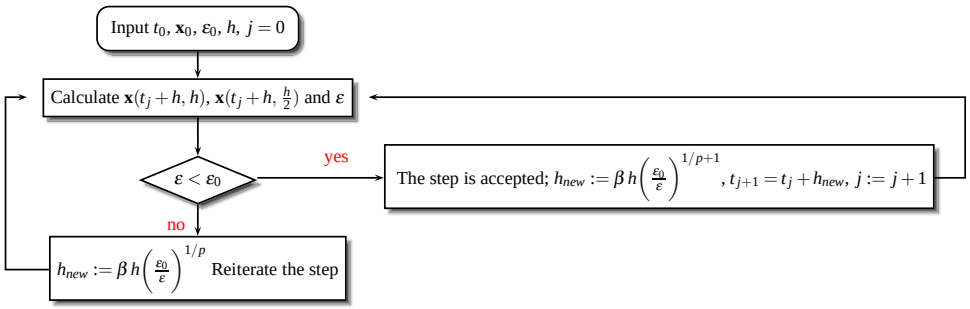

Fig. A.2 Flow diagramm of the adaptive step size control by use of the step doubling method.

where β is a "safety" factor.

Runge-Kutta-Fehlberg method

The alternative stepsize adjustment algorithm is based on the *embedded Runge-Kutta formulas*, originally invented by Fehlberg and is called *the Runge-Kutta-Fehlberg methods (RKF45)* [11, 10]. At each step, two different approximations for the solution are made and compared. Usually an fourth-order method with five stages is used together with an fifth-order method with six stages, that uses all of the points of the first one. The general form of a fifth-order Runge-Kutta with six stages is

$$\begin{aligned} k_1 &= f(t, \mathbf{x}), \\ k_2 &= f(t+\alpha_2 h, \mathbf{x}+h\beta_{21}k_1), \\ &\vdots \\ k_6 &= f(t+\alpha_6 h, \mathbf{x}+h\sum_{j=1}^{5}\beta_{6j}k_j). \end{aligned}$$

The embedded fourth-order formula is

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\sum_{i=1}^{6} c_i k_i + \mathcal{O}(h^5).$$

And a better value for the solution is determined using a Runge-Kutta method of fifth-order:

$$\mathbf{x}^*_{n+1} = \mathbf{x}_n + h\sum_{i=1}^{6} c^*_i k_i + \mathcal{O}(h^6)$$

The two particlular choises of unknown parametrs of the method are given in Tables A.4–A.5.

The error estimate is

$$\varepsilon = |\mathbf{x}_{n+1} - \mathbf{x}^*_{n+1}| = \sum_{i=1}^{6}(c_i - c^*_i)k_i.$$

Table A.4 Fehlberg parameters of the Runge-Kutta-Fehlberg 4(5) method.

1/4	1/4					
3/8	3/32	9/32				
12/13	1932/2197	-7200/2197	7296/2197			
1	439/216	-8	3680/513	-845/4104		
1/2	-8/27	2	-3544/2565	1859/4104	-11/40	
	25/216	0	1408/2565	2197/4104	-1/5	
	16/135	0	6656/12825	28561/56430	-9/50	2/55

Table A.5 Cash-Karp parameters of the Runge-Kutta-Fehlberg 4(5) method.

1/5	1/5					
3/10	3/40	9/40				
3/5	3/10	-9/10	6/5			
1	-11/54	5/2	-70/27	35/27		
7/8	1631/55296	175/512	575/13828	44275/110592	253/4096	
	37/378	0	250/621	125/594	512/1771	
	2825/27648	0	18575/48384	13525/55296	277/14336	1/4

As was mentioned above, if we take the current step h and produce an error ε, the corresponding "optimal" step h_{opt} is estimated as

$$h_{opt} = \beta h \left(\frac{\varepsilon_{tol}}{\varepsilon} \right)^{0.2},$$

where ε_{tol} is a desired accuracy and β is a "safety" factor, $\beta \simeq 1$. Then if the two answers are in close agreement, the approximation is accepted. If $\varepsilon > \varepsilon_{tol}$ the step size has to be decreased, whereas the relation $\varepsilon < \varepsilon_{tol}$ means, that the step size are to be increased in the next step.
Using Eq. (A.10), the optimal step can be often written as

$$h_{opt} = \begin{cases} \beta h \left(\frac{\varepsilon_{tol}}{\varepsilon} \right)^{0.2}, & \varepsilon \geq \varepsilon_{tol}, \\ \beta h \left(\frac{\varepsilon_{tol}}{\varepsilon} \right)^{0.25}, & \varepsilon < \varepsilon_{tol}, \end{cases}$$