Genetic algorithms: A possible model for COVID-19

Master degree in Modelling for Science and Engineering

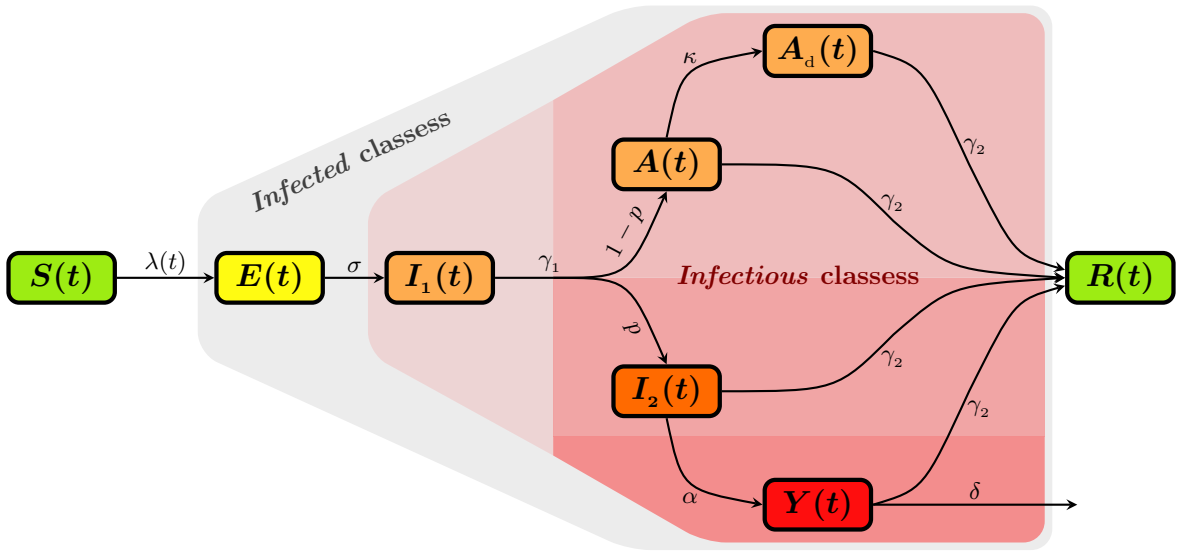
January 3, 2021

1 A possible relatively simple model for COVID-19

The proposed model is based on a continuous SEIR model formulated by means of the variables[1]

Variable	Size of population class at time t
$S(t)$	*susceptible* to infection,
$E(t)$	already *exposed* to infection,
$R(t)$	*recovered*,
$I_1(t)$	infectious class before the onset of symptoms (presymptomatic infectious class),
$A(t)$	almost asymptomatic infectious class with so mild symptoms that are never detected by the health system,
$A_{\mathrm{d}}(t)$	almost asymptomatic infectious class that is detected by the health system,
$I_2(t)$	infectious class with strong symptoms,
$Y(t)$	infectious class with serious symptoms.

It can be schematised by:



where P is the (known) *population size* and[2]

$$\lambda(t) := \beta \frac{\phi I_1(t) + A(t) + (1-\varepsilon_I)\left[A_{\mathrm{d}}(t) + I_2(t)\right] + (1-\varepsilon_Y)Y(t)}{P}$$

is the *strength of the infection at time* t, and the meaning of the remaining 11 parameters is:

[1] Individuals with strong or severe symptoms are isolated from the whole population at home or in the hospital, respectively, with a different degree of isolation effectiveness, and, therefore, they contribute less to transmission.

[2] The normalization of $\lambda(t)$ by dividing it by P, has a lot of meaning from the theoretical point of view but may seem trivial from the computational point of view. This normalization can also be interpreted as a normalization of the parameter β relative to the population size. This has the beneficial effect of highly decreasing the sensitivity of the parameter β.

Parameter	Meaning $\left(\text{unit is } \frac{\text{per capita rate}}{\text{day}}\right)$
β	average infectivity,
$\phi < 1$	pre-symptomatic infectivity factor,
ε_I	isolation effectiveness of strong cases,
ε_Y	isolation effectiveness of serious cases,
σ	rate of appearance of infectious ability,
γ_1	rate of appearance of symptoms,
γ_2	rate of recovery,
κ	detectability rate of asymptomatic people,
p	probability for strong symptoms (probability of the health system to clinically detect cases),
α	rate of appearance of serious symptoms (i.e. rate at which individuals with strong symptoms become so seriously ill that they are in need of hospitalization and a fraction of those will require intensive care and respirators),
δ	disease-induced mortality.

Non-negativity: All parameters and variables are non-negative.

1.1 The model in Differential Equations

The model described in the above schema can be formulated in terms of the following system of Differential Equations in dimension 8:

$$
(1) \qquad \begin{cases} \dfrac{dS(t)}{dt} = -\lambda(t)S(t), \\ \dfrac{dE(t)}{dt} = \lambda(t)S(t) - \sigma E(t), \\ \dfrac{dI_1(t)}{dt} = \sigma E(t) - \gamma_1 I_1(t), \\ \dfrac{dA(t)}{dt} = \gamma_1(1-p)I_1(t) - (\kappa+\gamma_2)A(t), \\ \dfrac{dA_{\mathrm{d}}(t)}{dt} = \kappa A(t) - \gamma_2 A_{\mathrm{d}}(t), \\ \dfrac{dI_2(t)}{dt} = \gamma_1 p I_1(t) - (\alpha+\gamma_2)I_2(t), \\ \dfrac{dY(t)}{dt} = \alpha I_2(t) - (\delta+\gamma_2)Y(t), \\ \dfrac{dR(t)}{dt} = \gamma_2\left(A(t) + A_{\mathrm{d}}(t) + I_2(t) + Y(t)\right). \end{cases}
$$

Observation 1. *Model* (1) *has the serious drawback of assuming isotropic behaviour and parameters according to age strata and mobility habits, and distribution of classes among cities and neighbourhoods.*

The total population at time t is

$$P(t) := S(t) + E(t) + I_1(t) + A(t) + A_{\mathrm{d}}(t) + I_2(t) + Y(t) + R(t).$$

Hence, the population size is P coincides with $P(0) = S(0) + E(0) + I_1(0) + A(0) + A_{\mathrm{d}}(0) + I_2(0) + Y(0) + R(0)$, and the accumulated number of disease-induced deaths at time t is:

$$D(t) := P - P(t) \geq 0.$$

Clearly, if $E(t)$, $I_1(t)$, $A(t)$, $A_{\mathrm{d}}(t)$, $I_2(t)$, $Y(t)$ and $R(t)$ are known, then $S(t)$ can be determined by

$$S(t) := P - \big(E(t) + I_1(t) + A(t) + A_{\mathrm{d}}(t) + I_2(t) + Y(t) + R(t)\big) - D(t).$$

Observe also that, usually, the known data is $A_{\mathrm{d}}(t)$, $I_2(t)$, $Y(t)$, and also $R(t)$ and $D(t)$ (assuming that there are no under-reported cases). The values of $E(t)$, $I_1(t)$ and $A(t)$ are unknown by assumption. Consequently $S(t)$ is also unknown.

2 The exercise

There has been a COVID-19 pandemic (pandemonium?), that we suppose governed by the Model (1) with unknown parameters. For simplicity we assume that there are no under-reported cases among the publicly known variables. In particular the value $R(t)$ is considered to be true.

On the initial condition (what happens at $t = 0$): We can assume that $R(0) = D(0) = 0$ and, more important, $A_{\rm d}(0) + I_2(0) + Y(0) > 0$. Then we must have $E(0), I_1(0) > 0$, and $A(0) > 0$ if $A_{\rm d}(0) > 0$; but these values are unknown. The value of $S(0)$ is also unknown but, since P is known,

$$S(0) := P - \big(E(0) + I_1(0) + A(0) + A_{\rm d}(0) + I_2(0) + Y(0)\big).$$

The data below is a figured pandemic public data about a period of 100 days (the fact that these values are, indeed, observed data of the corresponding variables at time t is indicated by the over-lines in the names of the variables). Clearly, the values of the 11 parameters of the model are unknown and, as explained above, $E(0), I_1(0), A(0)$ and $S(0)$ are also unknown.

Population size $P = 1,000,000$

t	$\overline{A_{\rm d}}(t)$	$\overline{I_2}(t)$	$\overline{Y}(t)$	$\overline{R}(t)$	$\overline{D}(t)$	t	$\overline{A_{\rm d}}(t)$	$\overline{I_2}(t)$	$\overline{Y}(t)$	$\overline{R}(t)$	$\overline{D}(t)$
0	1.000	1.000	0.000	0.000	0.000	51	278.160	416.566	112.197	751.997	16.215
1	1.841	1.253	0.056	0.348	0.000	52	306.763	459.370	123.746	830.294	17.924
2	2.285	1.607	0.123	0.733	0.002	53	338.296	506.551	136.480	916.637	19.809
3	2.571	2.041	0.203	1.169	0.004	54	373.057	558.552	150.519	1011.850	21.887
4	2.812	2.541	0.300	1.670	0.008	55	411.372	615.862	165.995	1116.839	24.180
5	3.059	3.096	0.414	2.249	0.013	56	453.603	679.015	183.056	1232.602	26.708
6	3.337	3.702	0.546	2.915	0.020	57	500.144	748.598	201.861	1360.238	29.496
7	3.655	4.356	0.698	3.681	0.029	58	551.431	825.258	222.588	1500.956	32.570
8	4.018	5.060	0.870	4.555	0.040	59	607.940	909.702	245.431	1656.087	35.960
9	4.428	5.814	1.062	5.548	0.054	60	670.196	1002.705	270.602	1827.096	39.697
10	4.889	6.626	1.276	6.673	0.071	61	738.773	1105.119	298.338	2015.593	43.818
11	5.402	7.499	1.513	7.941	0.091	62	814.301	1217.874	328.894	2223.347	48.361
12	5.972	8.442	1.773	9.364	0.115	63	897.473	1341.989	362.553	2452.305	53.369
13	6.603	9.462	2.058	10.959	0.143	64	989.042	1478.579	399.625	2704.605	58.889
14	7.299	10.570	2.369	12.739	0.175	65	1089.838	1628.858	440.449	2982.593	64.974
15	8.067	11.776	2.710	14.723	0.212	66	1200.765	1794.155	485.396	3288.848	71.680
16	8.914	13.091	3.083	16.931	0.253	67	1322.811	1975.914	534.874	3626.195	79.069
17	9.847	14.529	3.490	19.382	0.301	68	1457.053	2175.710	589.326	3997.735	87.212
18	10.875	16.103	3.934	22.102	0.355	69	1604.666	2395.253	649.239	4406.864	96.183
19	12.008	17.830	4.420	25.116	0.415	70	1766.929	2636.396	715.142	4857.302	106.065
20	13.257	19.725	4.952	28.454	0.483	71	1945.231	2901.150	787.612	5353.119	116.949
21	14.633	21.808	5.534	32.147	0.559	72	2141.079	3191.686	867.280	5898.765	128.936
22	16.150	24.099	6.172	36.231	0.644	73	2356.106	3510.345	954.827	6499.101	142.133
23	17.823	26.620	6.871	40.746	0.738	74	2592.078	3859.646	1050.997	7159.429	156.661
24	19.667	29.395	7.638	45.735	0.844	75	2850.898	4242.291	1156.593	7885.531	172.651
25	21.701	32.453	8.480	51.245	0.960	76	3134.614	4661.169	1272.484	8683.701	190.246
26	23.943	35.822	9.406	57.331	1.090	77	3445.426	5119.362	1399.610	9560.781	209.600
27	26.416	39.535	10.422	64.051	1.233	78	3785.685	5620.137	1538.981	10524.199	230.885
28	29.143	43.628	11.540	71.469	1.392	79	4157.899	6166.949	1691.682	11582.004	254.286
29	32.151	48.141	12.770	79.656	1.569	80	4564.733	6763.432	1858.876	12742.907	280.004
30	35.469	53.116	14.124	88.693	1.763	81	5009.003	7413.384	2041.802	14016.312	308.258
31	39.128	58.603	15.615	98.666	1.979	82	5493.676	8120.751	2241.781	15412.350	339.286
32	43.164	64.654	17.256	109.670	2.217	83	6021.857	8889.602	2460.210	16941.914	373.345
33	47.616	71.328	19.064	121.812	2.480	84	6596.777	9724.093	2698.565	18616.677	410.714
34	52.527	78.688	21.056	135.210	2.770	85	7221.773	10628.429	2958.390	20449.122	451.691
35	57.944	86.805	23.252	149.991	3.091	86	7900.265	11606.813	3241.302	22452.549	496.601
36	63.918	95.758	25.671	166.298	3.446	87	8635.723	12663.380	3548.971	24641.082	545.789
37	70.508	105.632	28.338	184.289	3.837	88	9431.630	13802.127	3883.121	27029.662	599.628
38	77.777	116.523	31.278	204.136	4.269	89	10291.435	15026.829	4245.506	29634.031	658.513
39	85.795	128.535	34.520	226.031	4.745	90	11218.496	16340.939	4637.900	32470.693	722.867
40	94.638	141.782	38.094	250.183	5.271	91	12216.019	17747.481	5062.074	35556.871	793.138
41	104.391	156.393	42.035	276.826	5.851	92	13286.985	19248.930	5519.771	38910.428	869.799
42	115.149	172.506	46.380	306.216	6.492	93	14434.065	20847.086	6012.679	42549.783	953.349
43	127.013	190.277	51.172	338.635	7.198	94	15659.535	22542.936	6542.399	46493.794	1044.309
44	140.099	209.874	56.456	374.394	7.978	95	16965.181	24336.518	7110.407	50761.622	1143.224
45	154.529	231.485	62.282	413.837	8.838	96	18352.193	26226.782	7718.015	55372.566	1250.660
46	170.444	255.316	68.708	457.342	9.787	97	19821.068	28211.456	8366.327	60345.878	1367.197
47	187.994	281.594	75.793	505.327	10.833	98	21371.503	30286.926	9056.199	65700.547	1493.434
48	207.347	310.569	83.606	558.252	11.988	99	23002.296	32448.131	9788.183	71455.071	1629.977
49	228.687	342.516	92.222	616.623	13.261	100	24711.260	34688.480	10562.489	77627.195	1777.437
50	252.217	377.737	101.722	681.000	14.666						

Statement of the exercise. Compute values of the 11 parameters that are consistent with the observed provided data. To do this it is also necessary to determine the initial condition

$$\Big(S(0) = P - \big(E(0) + I_1(0) + A(0) + A_{\rm d}(0) + I_2(0) + Y(0)\big), E(0), I_1(0), A(0), A_{\rm d}(0), I_2(0), Y(0), R(0) = 0\Big);$$

that is, to determine the values of $E(0), I_1(0)$, and $A(0)$ that are consistent with the observed provided data.

3 Proposed solution strategy

The exercise is to be solved with a minimising genetic algorithm with an appropriate fitness function.

3.1 Individuals

Clearly, an individual in the population may have two chromosomes: the first one is a vector formed by a possible choice of the 3 unknown initial conditions, and the second one is a vector formed by a possible choice of the 11 parameters:

$$\big(E(0), I_1(0), A(0)\big), \text{ and}$$
$$\big(\beta, \phi, \varepsilon_I, \varepsilon_Y, \sigma, \gamma_1, \gamma_2, \kappa, p, \alpha, \delta\big).$$

As recommended repeatedly, this "phenotype" is better encoded with a two-chromosome genotype of integers in binary. In the following table we explain, for each element of the phenotype the range and desired precision (or sensitivity), thus fixing the range and discretization formula for the genotype.

		Phenotype		Genotype	
		upper limit	precision	upper limit	Factor from genotype to phenotype
Initial Conditions	$E(0)$, $I_1(0)$, $A(0)$	P	10^{-3}	2^{30}	$\frac{1}{1000}$
Parameters	β	1	10^{-12}	2^{40}	2^{-40}
	ϕ	1	10^{-6}	2^{20}	2^{-20}
	ε_I	1	10^{-3}	2^{10}	2^{-10}
	ε_Y	1	10^{-3}	2^{10}	2^{-10}
	σ	1	10^{-6}	2^{20}	2^{-20}
	γ_1	1	10^{-6}	2^{20}	2^{-20}
	γ_2	1	10^{-6}	2^{20}	2^{-20}
	κ	1	10^{-3}	2^{10}	2^{-10}
	p	1	10^{-6}	2^{20}	2^{-20}
	α	1	10^{-12}	2^{40}	2^{-40}
	δ	1	10^{-12}	2^{40}	2^{-40}

Observations:

- All upper limit and precision values for the phenotype have been set to "common sense reasonable values".
- All upper limit values of the genotype have been chosen to be powers of two with following condition that

  genotype upper limit of the form $2^n >$ phenotype upper limit/precision.

  For example, for the initial conditions the above formula gives

  $$2^{30} = 1,073,741,824 > 1,000,050,000 = 1,000,050 \cdot 10^3 = P/10^{-3},$$

  and for β,

  $$2^{40} = 1,099,511,627,776 > 10^{12} = \frac{1}{10^{-12}}.$$

  Observe that the number $2^n - 1$ when written in binary in 64 a bits representation, has a string of $64 - n$ consecutive zeroes at the left, and a string of n consecutive ones at the right. Moreover, the expression in binary of all integers in the range $[0, 2^n - 1]$ has a string of at least $64 - n$ consecutive zeroes at the left. This is very useful, when programming crossovers and mutations, to avoid complicate feasibility tests.

- All powers of two have exponent less than or equal to 40, and there are some with exponents larger than 32. So, the base data type for the genes in the chromosomes must be `unsigned long int`.

3.2 Fitness function

The Genetic Algorithm must identify an individual that could possibly have generated the observed pandemic. This is done by finding the "fittest" individual from the point of view of generating the observed data. In other words, the fitness function must measure how similar is the observed data to the pandemic data generated from an individual.

More precisely, an individual Ind (together with the row $t = 0$ of the observed data) provides all the necessary information to generate a pandemic by means of Model 1. This amounts building a full determined initial condition

$$\big(S(0), E(0), I_1(0), A(0), A_d(0), I_2(0), Y(0), R(0) = 0\big);$$

where $E(0), I_1(0)$, and $A(0)$ are provided by Ind, $A_d(0), I_2(0), Y(0)$, and $R(0) = 0$ are determined by the row $t = 0$ of the observed data, and $S(0)$ is computed with the help of the formula

$$S(0) = P - \big(E(0) + I_1(0) + A(0) + A_d(0) + I_2(0) + Y(0)\big).$$

Then, the values of the 11 parameters provided by Ind allow to numerically compute (as in the observed data)

$$\big(S(t), E(t), I_1(t), A(t), A_d(t), I_2(t), Y(t), R(t)\big) \qquad \text{for } t = 1, 2, \ldots, 100$$

and, in particular,

$$\big(A_d(t), I_2(t), Y(t), R(t), D(t)\big) \qquad \text{for } t = 1, 2, \ldots, 100.$$

As above, for $t = 1, 2, \ldots, 100$, let $\overline{A_d}(t), \overline{I_2}(t), \overline{Y}(t), \overline{R}(t)$, and $\overline{D}(t)$ denote, respectively, the values of the variables $A_d(t), I_2(t), Y(t), R(t)$, and $D(t)$ from the figured pandemic public data. Two possible norms that measure the agreement between the pandemic data generated by Ind and the figured pandemic public data are:

$$(2) \quad \max_{t=1,\ldots,100} \left\{ \Big(A_d(t) - \overline{A_d}(t)\Big)^2 + \Big(I_2(t) - \overline{I_2}(t)\Big)^2 + \Big(Y(t) - \overline{Y}(t)\Big)^2 + \Big(R(t) - \overline{R}(t)\Big)^2 + \Big(D(t) - \overline{D}(t)\Big)^2 \right\},$$

and

$$(3) \quad \sum_{t=1}^{100} W_t \left(\Big(A_d(t) - \overline{A_d}(t)\Big)^2 + \Big(I_2(t) - \overline{I_2}(t)\Big)^2 + \Big(Y(t) - \overline{Y}(t)\Big)^2 + \Big(R(t) - \overline{R}(t)\Big)^2 + \Big(D(t) - \overline{D}(t)\Big)^2 \right),$$

where $W_1, W_2, \ldots, W_{100} > 0$ are weights. Clearly, a value zero in the above fitness function indicates that Ind's phenotype *is* the one that drives the pandemic through Model 1.

Reasonable weights in the above norm (3) are $W_t = 1$, $W_t = t$, or $W_t = \exp(\nu t)$ for some value $\nu > 0$.

3.3 Integrating an ODE: Computing the values of $\big(S(t), E(t), I_1(t), A(t), A_d(t), I_2(t), Y(t), R(t)\big)$ for $t = 1, 2, \ldots, 100$

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` we provide here a full programmed implementation of the computation of the values

$$\big(S(t), E(t), I_1(t), A(t), A_d(t), I_2(t), Y(t), R(t)\big) \qquad \text{for } t = 1, 2, \ldots, 100.$$

First we need a structure (arbitrary) to store the ODE 11 parameters:

```
typedef struct{ double beta, phi, epsI, epsY;
                double sigma;
                double gamma1, gamma2;
                double kappa;
                double p;
                double alpha;
                double delta;
                double PopSize;
} ODE_Parameters;
```

Now, as explained in the implementation notes in `RKF78.c` we need a function the computes the Vector Field given by Model 1:

```
#define CoreModelDIM 8
void CoreModel(double t, double *x, unsigned dim, double *der, void *Params){
  ODE_Parameters *par = (ODE_Parameters *) Params; // To simplify the usage of Params (void pointer)
  double sigmae = par->sigma*x[1],
       gamma1i1 = par->gamma1*x[2],
         kappaA = par->kappa*x[3],
        alphai2 = par->alpha*x[5];
 der[0] = par->phi*x[2] + x[3] + (1-par->epsI)*(x[4]+x[5]) + (1-par->epsY)*x[6];
 der[0] = - par->beta * (x[0] * der[0])/par->PopSize;
 der[1] = -der[0] - sigmae;
 der[2] = sigmae - gamma1i1;
 der[3] = (1-par->p)*gamma1i1 - kappaA - par->gamma2*x[3] ;
 der[4] = kappaA - par->gamma2*x[4];
 der[5] = par->p*gamma1i1 - par->gamma2*x[5] - alphai2;
 der[6] = alphai2 - (par->gamma2+par->delta)*x[6];
 der[7] = par->gamma2*(x[3] + x[4] + x[5] + x[6]);
}
```

We also need our computer individuals data-type (the order of the initial conditions is assumed the be the one specified above; the order of the parameters is assumed to be the one specified in `ODE_Parameters`, which coincides with the ordering specified in the table above):

```
#define IC_GENES_NUMBER 3
#define PARAMETERS_GENES_NUMBER 11
typedef struct {
    unsigned long IC[IC_GENES_NUMBER];
    unsigned long Pars[PARAMETERS_GENES_NUMBER];
    double fitness;
} individual;
```

Now, we can give the function that, by using `RKF78Sys`, integrates the EDO until $t = 100$ with the data provided by an `individual`. First we need to define the parameters `HMAX`, `HMIN` and `RKTOL` as required in the implementation notes in `RKF78.c`. The `DataForFitting` part is completely free and it is recommended that you use exactly the technology used to store the observed data (to make it easier to compare the observed data with the solution computed by the `RKF78Sys` function). The basic message at this stage is to store the predicted data instead of printing it.

```
#define HMAX  1.0
#define HMIN  1.e-3
#define RKTOL 1.e-5
int GeneratePredictionFromIndividual(double *xt, void *ODE_pars, DataForFitting *Pred) {
    register unsigned ndays;
    double t = 0.0, err, h = 1.e-3;

    for(ndays=1; ndays <= Pred->N_Days; ndays++) { int status;
        while(t+h < ndays) {
          status = RKF78Sys(&t, xt, CoreModelDIM, &h, &err, HMIN, HMAX, RKTOL, ODE_pars, CoreModel);
          if(status) return status;
        }
        h = ndays - t;
        status = RKF78Sys(&t, xt, CoreModelDIM, &h, &err, HMIN, HMAX, RKTOL, ODE_pars, CoreModel);
        if(status) return status;

        Pred->Data_Time_Series[ndays][0] = xt[4];
        Pred->Data_Time_Series[ndays][1] = xt[5];
        Pred->Data_Time_Series[ndays][2] = xt[6];
        Pred->Data_Time_Series[ndays][3] = xt[7];
        Pred->Data_Time_Series[ndays][4] = Pred->PopSize -
                                           (xt[0]+xt[1]+xt[2]+xt[3]+xt[4]+xt[5]+xt[6]+xt[7]);
    }
    return 0;
}
```

Another ingredient that you need is a function that converts an individual to data useful for Runge-Kutta prediction. In other words that converts genotype to phenotype and, after computing the prediction, it computes the fitness function by comparing the observed data with the predicted data with the individual phenotype with the help of the norms (2) or (3).

```
#define crom2IC(c)    (((double) c)/1000)
#define crom2HSPar(c) (((double) c)/1099511627776UL)
#define crom2Par(c)   (((double) c)/1048576U)
#define crom2LSPar(c) (((double) c)/1024U)

void CoreModelVersusDataQuadraticError(individual *ind, void *TheData) {
    DataForFitting *TDfF = (DataForFitting *) TheData;
    DataForFitting ThePrediction = { TDfF->PopSize, TDfF->N_Days, {
                                     { TDfF->Data_Time_Series[0][0], TDfF->Data_Time_Series[0][1],
                                       TDfF->Data_Time_Series[0][2], TDfF->Data_Time_Series[0][3],
                                       TDfF->Data_Time_Series[0][4] } } };
    double xt[CoreModelDIM] = { TDfF->PopSize, crom2IC(ind->IC[0]), crom2IC(ind->IC[1]),
                                crom2IC(ind->IC[2]),          TDfF->Data_Time_Series[0][0],
                                TDfF->Data_Time_Series[0][1], TDfF->Data_Time_Series[0][2],
                                TDfF->Data_Time_Series[0][3] };
                  xt[0] -= (xt[1]+xt[2]+xt[3]+xt[4]+xt[5]+xt[6]);
    ODE_Parameters ODE_pars = { crom2HSPar(ind->Pars[0]), crom2Par(ind->Pars[1]),
                                crom2LSPar(ind->Pars[2]), crom2LSPar(ind->Pars[3]),
                                crom2Par(ind->Pars[4]),   crom2Par(ind->Pars[5]),
                                crom2Par(ind->Pars[6]),   crom2LSPar(ind->Pars[7]),
                                crom2Par(ind->Pars[8]),   crom2HSPar(ind->Pars[9]),
                                crom2HSPar(ind->Pars[10]), TDfF->PopSize };

    if(GeneratePredictionFromIndividual(xt, &ODE_pars, &ThePrediction)) {
       ind->fitness = MAXDOUBLE;
       return;
    }

    ind->fitness =  ... Here compute the fitness function by comparing
    the observed data (i.e. TDfF or TheData) with the predicted data (ThePrediction)
    with the help of the norms (2) or (3) ...
```

3.4 Infeasible individuals

In genetic algorithms, individuals have an important random ingredient. When an individual is used to determine the parameters and initial conditions of a System of ODE's we can easily expect that the integrator (`RKF78`) gets lost; which amounts to the fact that we cannot compute the fitness of the corresponding individual. In this situation it may happen that the generations (specially the initial population) could be almost completely infeasible, and this is very bad for the performance of the algorithm. In this regard it might be advisable:

- to introduce a count that gives the proportion of feasible individuals of every generation, and informs us when this proportion goes below a certain value.
- For the initial population this could be specially important, and so it is recommended to loop around the random building of the initial population until its proportion of feasible individuals is larger than a given security bound (for instance 0.7; meaning that at least 70% of the initial population must be feasible).

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:

0	
1	1
	1/2 1/2

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:

0	
1/2	1/2
	0 1

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).

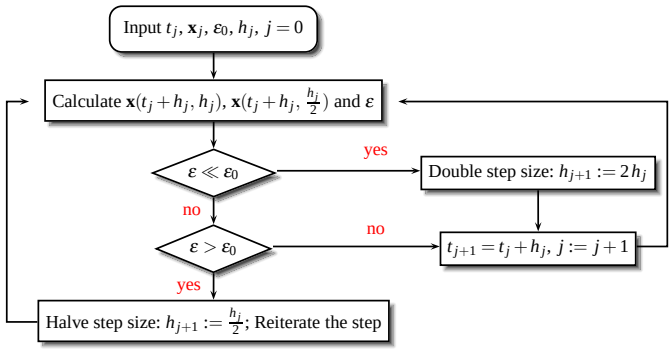


Fig. A.1 Flow diagramm of the step size control by use of the step doubling method.

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

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

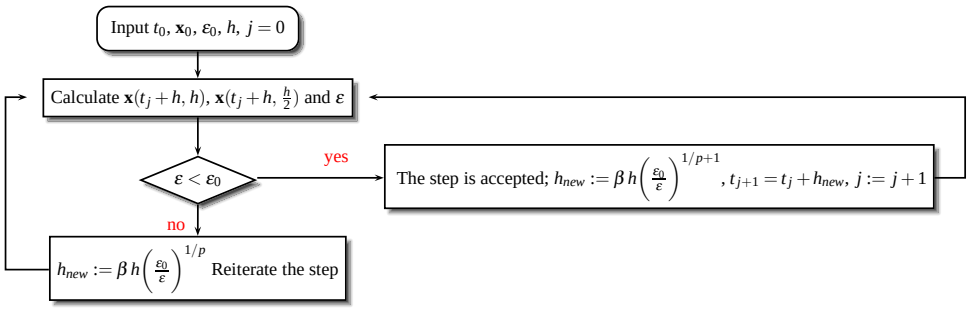


Fig. A.2 Flow diagramm of the adaptive step size control by use of the step doubling method.

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

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