

///////////////////////////////////////////////////////////////////////////////

// Sample program to generate α and dα/dt curves by empirical model //

// //

// dα/dt=exp[p(α)-E(α)/(RT)](1-α) //

// //

// Here p(α) and E(α) are polynomials that are expressed as follows: //

// First α is mapped from range [0,1] to [-1,+1] by //

// x=2α-1 //

// then //

// p(α)= a₀ + a₁x + a₂x² + a₃x³ + a₄x⁴ +a₅x⁵ //

// E(α)= b₀ + b₁x + b₂x² //

// //

// The results are listed into tab-delimited text files with UTF-8 encoding. //

// The 1st output file contains t T 1-α dα/dt data //

// The 2nd output file contains t T 100\*(1-α) 100\*dα/dt data //

// This can be easily changed in the code. //

///////////////////////////////////////////////////////////////////////////////

// This is a source code for a consol application.

// **If Microsoft Visual Studio is used under Windows:** Use compiler option /utf-8

// and keep the very first program line (#define ... ) as it is:

#define \_CRT\_SECURE\_CPP\_OVERLOAD\_STANDARD\_NAMES 1

// **Under LINUX the GCC compiler may be used** with option -lm. For example:

// gcc filename\_of\_this\_program.cpp -lm -o filename\_of\_this\_program

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

////////////////////////////////////////////////////////////////////

// THE KINETIC PARAMETERS AND OTHER MODEL-SPECIFIC DATA COME HERE //

// Either copy them in the place of the next program line //

// or put them into include file "parameters.h". //

// They are given as constant global variables which are //

// accessible all over in the code. //

////////////////////////////////////////////////////////////////////

#include "parameters.h"

// A further global variable:

double heating\_rate\_per\_sec; // heating rate, °C s⁻¹

///////////////////////////////////////////////////////

// THE FUNCTIONS (PROCEDURES) FOLLOW: //

///////////////////////////////////////////////////////

double T\_Celsius(double t) // calculates T/°C from t/s :

{

// The ODE solution starts at 150°C

// because the reaction rate is insignificant below 150°C:

const double T\_Celsius\_start\_value=150;

return T\_Celsius\_start\_value+heating\_rate\_per\_sec\*t;

}

// Calculate the right hand side of the ODE,

// i.e. dα/dt as function of α and t:

double fct( double t, double alpha)

{

const double R=8.314462618E-3; // kJ mol⁻¹ K⁻¹

double x, p, E, T\_Kelvin;

// Polynomial evaluations by the Horner method:

x=2\*alpha-1;

p=a[0]+x\*(a[1]+x\*(a[2]+x\*(a[3]+x\*(a[4]+x\*a[5]))));

E=b[0]+x\*(b[1]+x\*b[2]);

// Calculate the right hand side of the ODE:

T\_Kelvin=T\_Celsius(t)+273.15;

return exp(p-E/(R\*T\_Kelvin))\*(1-alpha);

}

// The next function terminates the program whenever it is needed

// Informs the user about the termination and wait till the user hits an enter.

// Gives back a return code of 0 or 3 to the operating system.

void stop( int retcode )

{

char buff[128];

// Displays a line on the console:

puts( "Strike enter to terminate program:" );

// Inputs an ENTER from the console and cleans up whatever the user

// typed accidentally before the ENTER:

fgets(buff,sizeof(buff),stdin);

exit(retcode); // this C/C++ function terminates program

}

/////////////////////////////////////////////////////////////////////////

// Main program: ODE solving by 4th order Runge-Kutta method //

// Outputs the results into tab-delimited text files with UTF-8 //

// character encoding. These files can be opened by modern text //

// editors, or by Microsoft Excel, or by LibreOffice's Calc, etc. //

/////////////////////////////////////////////////////////////////////////

int main(void)

{

char filespec1[260], filespec2[260];

// Domain where the results will be printed:

const double alpha1=0.001, alpha2=0.999, eps=0.00002; // output alpha domain

double alpha, h, k1, k2, k3, k4, k, t, T\_Celsius\_value;

FILE \*out1, \*out2;

// format strings for the tab-delimited output of the data:

const char \*fmt1="%.3f\t%.1f\t%.5f\t%.4E\n"; // for t T 1-α dα/dt

const char \*fmt2="%.3f\t%.1f\t%.3f\t%.4E\n"; // for t T 100\*(1-α) 100\*dα/dt

// A loop to carry out the calculations for each of the given heating rates:

for (int iexper=0; iexper<nexper; ++iexper)

{

// The heating rate as a global variable in unit °C s⁻¹:

heating\_rate\_per\_sec=heating\_rate\_per\_min[iexper]/60.;

// h is the prefixed step size of ODE integration.

// T rises by 0.05°C during time h:

h=0.05/heating\_rate\_per\_sec; // time step for Runge-Kutta solution (s)

// Open output file for listing t, T, 1-α, and dα/dt

// Create file the name:

snprintf(filespec1,sizeof(filespec1),

outfile\_fmt,heating\_rate\_per\_min[iexper],"");

// Open the file

out1=fopen(filespec1,"w");

if( out1==NULL ) // stop in case of file-open error

{

puts("Cannot open new file:");

puts(filespec1);

stop(3);

}

// Open output file for listing t, T, 100\*(1-α), and 100\*dα/dt

// Add a " percent" tag to the corresponding file name:

snprintf(filespec2,sizeof(filespec2),

outfile\_fmt,heating\_rate\_per\_min[iexper]," percent");

// Open the file:

out2=fopen(filespec2,"w");

if( out2==NULL ) // stop in case of file-open error

{

puts("Cannot open new file:");

puts(filespec2);

stop(3);

}

// Write the filename to the file as a memo

// to avoid confusion later, when working with the results:

fprintf( out1, "File: \"%s\"\n\n", filespec1 );

fprintf( out2, "File: \"%s\"\n\n", filespec2 );

// Write a header line into both files.

fputs("t/min\tT/°C \t1-α \t(dα/dt)/s⁻¹\n",out1);

fputs("t/min\tT/°C \t100(1-α)\t100(dα/dt)/s⁻¹\n",out2);

// Runge-Kutta ODE solution:

alpha=0;

t=0;

while( alpha<alpha2+eps )

{

// This is the classical Runge-Kutta code:

k1=h\*fct(t,alpha);

k2=h\*fct(t+(h/2),alpha+(k1/2));

k3=h\*fct(t+(h/2),alpha+(k2/2));

k4=h\*fct(t+h,alpha+k3);

k=(1/6.)\*(k1+(2\*k2)+(2\*k3)+k4);

alpha=alpha+k;

t=t+h;

// output the results when needed

if(alpha > alpha1-eps )

{

T\_Celsius\_value=T\_Celsius(t);

// Output the corresponding data when T by 0.1°C

// round 1000×T\_Celsius\_value to integer and

// check when it is dividable by 100:

long i=(long)(1000.\*T\_Celsius\_value+0.5);

if( i%100==0 ) // output when i is dividable by 100

{

// output a line into file out1:

fprintf( out1, fmt1, t/60., T\_Celsius\_value, 1.-alpha,

fct(t,alpha) );

// output a line into file out2:

fprintf( out2, fmt2, t/60., T\_Celsius\_value, 100\*(1.-alpha),

100\*fct(t,alpha) );

}

}

}

fclose(out1); // close output file 1

fclose(out2); // close output file 2

}

puts("Ready");

stop(0);

}