OPENMP COMPUTING OF A REFERENCE SOLUTION FOR COUPLED LORENZ SYSTEM ON [0,400]

I. Hristov, R. Hristova, S. Dimova, P. Armyanov, N. Shegunov
FMI, Sofia University, Bulgaria

I. Puzynin, T. Puzynina, Z. Sharipov, Z. Tukhliev
MLIT, JINR, Dubna

Conference GRID'2021, July 5-9, MLIT, JINR, Dubna

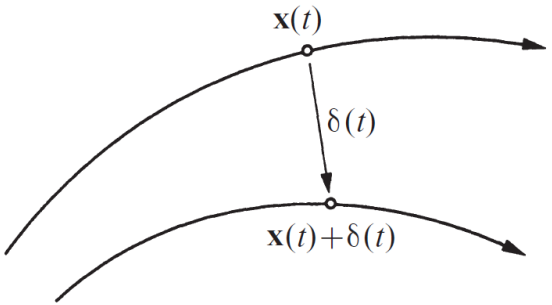
The model problem

We consider as a model problem a coupled Lorenz system:

$$\begin{aligned}
\frac{dx}{dt} &= a(y-x) \\
\frac{dy}{dt} &= r_s x - y - xz - \varepsilon_s XY \\
\frac{dz}{dt} &= xy - bz \\
\frac{dX}{dt} &= ca(Y-X) \\
\frac{dY}{dt} &= c(r_f X - Y - XZ) + \varepsilon_f Xy \\
\frac{dZ}{dt} &= c(XY - bZ)
\end{aligned} \tag{1}$$

where $a = 10$, $b = 8/3$, $c = 10$, $r_s = 28$, $r_f = 45$, $\varepsilon_s = 10^{-2}$, $\varepsilon_f = 10$. For these parameters the system has a chaotic attractor. The first three and last three equations are called slow and fast dynamics respectively.

Sensitive dependence on initial conditions



$$\delta(t) \sim \delta(0)e^{\lambda t}$$

$\lambda > 0$ is the Lyapunov exponent.
Predictability horizon (Lyapunov time) T is defined by:

$$T = \frac{1}{\lambda}\ln(\frac{tol}{\varepsilon})$$

where tol is our tolerance and ε is the round-off unit (precision).
For coupled Lorenz system $\lambda = 11.5$. If we use the standard double precision ($\varepsilon = 2^{-53}$) and tolerance $tol = 10^{-3}$, then $T \approx 2.5$.

What do we need to obtain a reliable long-term solution?

We need:

1. A multiple-precision floating point arithmetic.

2. A numerical method, which steps efficiently at the level of the precision, i.e. method, which allows arbitrary high order of accuracy.

3. If the time interval for the reference solution is very long, the computational problem can become large and we need parallelization of the algorithm.

In our work we use:

1. GMP library (The GNU Multiple Precision Arithmetic Library)

2. Taylor Series Method

3. OpenMP parallel technology for parallelization.

Taylor series method

Generally we want to solve numerically the initial value problem

$$\boldsymbol{X}'(\boldsymbol{t}) = \boldsymbol{f}(\boldsymbol{X}(\boldsymbol{t})), \boldsymbol{t} \in [\boldsymbol{0}, \boldsymbol{T}]$$
$$\boldsymbol{X}(\boldsymbol{0}) = \boldsymbol{X_0}$$

with the multiple-precision Taylor series method. We assume that f is analytic on its domain of definition and that $\boldsymbol{X}(\boldsymbol{t})$ is defined in $[\boldsymbol{0}, \boldsymbol{T}]$.

With N-th order Taylor series method the approximate solution $\tilde{\boldsymbol{X}}(\boldsymbol{t_n})$ is:

$$\tilde{\boldsymbol{X}}(\boldsymbol{t_0}) = \boldsymbol{X_0}$$

$$\tilde{\boldsymbol{X}}(\boldsymbol{t_{n+1}}) = \tilde{\boldsymbol{X}}(\boldsymbol{t_n}) + \tilde{\boldsymbol{X}}'(\boldsymbol{t_n})\boldsymbol{\tau} + \frac{\tilde{\boldsymbol{X}}''(\boldsymbol{t_n})}{\boldsymbol{2!}}\boldsymbol{\tau^2} + \ldots\ldots + \frac{\tilde{\boldsymbol{X}}^{(N)}(\boldsymbol{t_n})}{\boldsymbol{N!}}\boldsymbol{\tau^N}$$

$$\tau = \frac{e^{\text{-0.7/(N-1)}}}{e^2} \min\left\{ \left(\frac{1}{\|\mathbf{X_{N-1}}\|_\infty} \right)^{\frac{1}{N-1}}, \left(\frac{1}{\|\mathbf{X_N}\|_\infty} \right)^{\frac{1}{N}} \right\}$$

Clean Numerical Simulation (CNS)

Actually we use a numerical procedure, proposed by **Shijun Liao**, called "Clean Numerical Simulation", for obtaining a reliable long-term solution of a chaotic dynamical system [1]. The procedure is based on **multiple precision Taylor series method**.

[1] *Liao, Shijun."On the reliability of computed chaotic solutions of non-linear differential equations." Tellus A: Dynamic Meteorology and Oceanography 61.4 (2008).*

Computing the Taylor coefficients (the normalized derivatives)

We denote the normalized derivatives (the $i-th$ derivative divided by $i!$) with $x_i, y_i, z_i, X_i, Y_i, Z_i$ for $i = 0, ..., N-1$. From equation (1) we have

$$\begin{aligned} x_1 &= a(y_0 - x_0) \\ y_1 &= r_s x_0 - y_0 - x_0 z_0 - \varepsilon_s X_0 Y_0 \\ z_1 &= x_0 y_0 - b z_0 \\ X_1 &= ca(Y_0 - X_0) \\ Y_1 &= c(r_f X_0 - Y_0 - X_0 Z_0) - \varepsilon_f X_0 y_0 \\ Z_1 &= c(X_0 Y_0 - b Z_0). \end{aligned}$$

By applying Leibniz rule we obtain the following procedure for computing $x_i, y_i, z_i, X_i, Y_i, Z_i$ for $i = 0, ..., N-1$. The procedure is actually the so called automatic differentiation, or sometimes called algorithmic differentiation.

Computing the Taylor coefficients (the normalized derivatives)

The procedure: for $i = 0, ..., N-1$ compute:

$$x_{i+1} = \frac{1}{i+1} a(y_i - x_i)$$

$$y_{i+1} = \frac{1}{i+1}(r_s x_i - y_i - \sum_{j=0}^{i} x_{i-j} z_j - \varepsilon_s \sum_{j=0}^{i} X_{i-j} Y_j)$$

$$z_{i+1} = \frac{1}{i+1}(\sum_{j=0}^{i} x_{i-j} y_j - b z_i)$$

$$X_{i+1} = \frac{1}{i+1} ca(Y_i - X_i)$$

$$Y_{i+1} = \frac{1}{i+1}(c(r_f X_i - Y_i - \sum_{j=0}^{i} X_{i-j} Z_j) + \varepsilon_f \sum_{j=0}^{i} X_{i-j} y_j)$$

$$Z_{i+1} = \frac{1}{i+1} c(\sum_{j=0}^{i} X_{i-j} Y_j - b Z_i).$$

(2)

Pseudocode of Taylor series method for coupled Lorenz system


```
while (time < T)
{
   // Computing derivatives
   // N -order of the method
   for (i = 0; i<N; i++)
   {
      s1=s2=s3=s4=s5=0.0;
      for (j=0; j<=i; j++)
      {
         s1+=x[i-j]*z[j];
         s2+=x[i-j]*y[j];
         s3+=X[i-j]*Y[j];
         s4+=X[i-j]*Z[j];
         s5+=X[i-j]*y[j];
      }
      .....
      // Computing x[i+1],y[i+1],z[i+1],
      // X[i+1],Y[i+1],Z[i+1] from formulas (2)
      // by using s1,s2,s3,s4,s5
      .....
   }
   .....
   //Computing the optimal stepsize tau
   //from N-th and N-1-th derivatives
   .....
   //  One step forward with Horner's rule
   //  for the new x[0],y[0],z[0],
   //  X[0],Y[0],Z[0]
   .....
   time+=tau;
}
```

Why OpenMP parallel technology?

OpenMP has its own importance for the above algorithm, because:

1. OpenMP is simpler than MPI, since the communication between threads is realized by shared memory and we do not need to learn special libraries for packaging and unpackaging of multiple precision numbers.

2. OpenMP is slightly faster than pure MPI.

3. OpenMP uses less memory, since the algorithm does not allow domain decomposition and the computational domain has to be multiplied by the number of processes, when MPI is used.

The sketch of OpenMP code in terms of GMP library


```
#pragma omp parallel private(i,j,tid)
{
   tid = omp_get_thread_num();
   for (i = 0; i<N; i++) //N - the order of the method
   {
      # pragma omp for schedule(static)
      for (j=0; j<=i; j++)
      {
         mpf_mul(tempv[pad*tid],x[i-j],z[j]);
         mpf_add(sum[pad*tid],sum[pad*tid],tempv[pad*tid]);
         .....
         // The same computations for the other 4 sums
      }
      // Explicit tree based parallel Reduction
      .....
      #pragma omp sections
      {
      // Computing x[i+1],y[i+1],z[i+1],X[i+1],Y[i+1],Z[i+1]
      // independently in 6 parallel sections
      }
      .....
      // Setting elements of the array "sum" to zero
   }
   #pragma omp single
   {
   //Computing the variable stepsize
   }
   #pragma omp sections
   {
   // One step forward with the Horner's rule
   // independently for each 6 components
   }
}
```

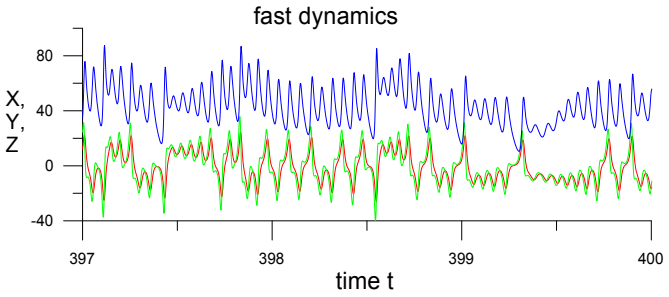
Numerical and performance results

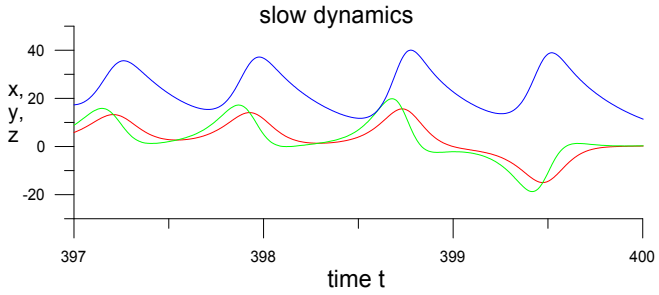
The preparation of the parallel program and the many tests are performed in the **HybriLIT** Platform at MLIT, JINR and in the **Nestum** Cluster, Sofia, Bulgaria.

- Following Shijun Liao we first computed a priori estimations for the needed order of the method and the needed precision and then computed a reference solution in the rather long time interval $\mathbf{[0, 400]}$**.** We took as initial conditions those from paper [2] in order to compare with the benchmark table up to time=100.
- We performed two large computations, each using one CPU-node (32 cores) in Nestum cluster. One computation with decimal digits of precision K=2158 and N=2480. And second computation for verification with K=2254 and N=2580.
- The needed time for the second (larger) computation using one node (32 cores) in Nestum cluster is $\mathbf{6.3}$ days. The parallel speedup when using these $\mathbf{32}$ cores is $\mathbf{23.1}$ with parallel efficiency $\mathbf{72.1\%}$**.**

[2] *Wang, P. et al. (2014). Clean numerical simulation for some chaotic systems using the parallel multiple-precision Taylor scheme. Chinese science bulletin, 59(33), 4465-4472*

Numerical and performance results





Reference solution at $t = 400$ with 60 correct digits

x=0.191694920337222404144404571185494721981620909648958369199 47
y=0.294822227157236985567028530513747928619544694688111752213113
z=11.444656446248354208274706070234976393205687332689206785061 2

X=-16.458240499211209486747598587152707492776465527484698855041 3
Y=-15.030542931420567594853310603065759778516919869290052951724 2
Z=54.374493561553596045355265999831749122934512315564306087507 7

ACKNOWLEDGEMENTS

We thank the Meshcheryakov Laboratory of Information Technologies of JINR, Dubna, Russia and the HybriLIT team, for the opportunity to use the computational resources of the HybriLIT Platform. We also appreciate the opportunity to use the computational resources of the Nestum cluster, Sofia, Bulgaria.

The work is supported by a grant of the Plenipotentiary Representative of the Republic of Bulgaria at JINR, Dubna, Russia.

THANK YOU FOR YOUR ATTENTION!