Mixed precision numerical methods for solving large systems of ordinary differential equations

M. Al Sayed Ali, S. Bernard, A. Marzorati, J. Rouzaud Cornabas

INRIA - Lyon

CANUM, Le-Bois-Plage-en-Ré, Ile de Ré, May 27-31, 2024

Floating-point precisions

Comparison of floating-point precisions	
Single precision (FP32)	**Double precision (FP64)**
· 32 bit memory to represent a value.	· 64 bit memory to represent a value.
· 7 decimal digits of precision	· 16 decimal digits of precision
· Machine precision $\approx 10^{-8}$	· Machine precision $\approx 10^{-16}$

Floating-point precisions

Comparison of floating-point precisions	
Single precision (FP32)	**Double precision (FP64)**
· 32 bit memory to represent a value.	· 64 bit memory to represent a value.
· 7 decimal digits of precision	· 16 decimal digits of precision
· Machine precision $\approx 10^{-8}$	· Machine precision $\approx 10^{-16}$

- Mixed precision = Single precision + Double precision

Floating-point precisions

Comparison of floating-point precisions	
Single precision (FP32)	**Double precision (FP64)**
· 32 bit memory to represent a value.	· 64 bit memory to represent a value.
· 7 decimal digits of precision	· 16 decimal digits of precision
· Machine precision $\approx 10^{-8}$	· Machine precision $\approx 10^{-16}$

- Mixed precision = Single precision + Double precision
- Numerical methods with mixed precision run faster and use less memory but with lost of accuracy

Floating-point precisions

Comparison of floating-point precisions	
Single precision (FP32)	**Double precision (FP64)**
· 32 bit memory to represent a value.	· 64 bit memory to represent a value.
· 7 decimal digits of precision	· 16 decimal digits of precision
· Machine precision $\approx 10^{-8}$	· Machine precision $\approx 10^{-16}$

- Mixed precision = Single precision + Double precision
- Numerical methods with mixed precision run faster and use less memory but with lost of accuracy
- Domain of applications : AI, Climate/weather modeling, Computational fluid dynamics, ...
- Our application : Biology and Health

- Consider the following ordinary differential equation (ODE) :

$$\begin{cases} \dot{y}(t) = f(t, y(t)) \ \forall \ t \in [t_0, T] \\ y(t_0) = y^{(0)} \end{cases} \tag{1}$$

where

- $y^{(0)}$ is a vector of $\mathbb{R}^n$.
- $f : [t_0, T] \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is a smooth function.

Numerical methods

- Discretization in time $\Longrightarrow$

$$y_{i+1} = F(y_{i-q}, y_{i-(q-1)}, \ldots, y_i, f) \qquad i = q, \ldots, N-1.$$

where

- $y_i \approx y(t_i)$,
- $t_i = t_0 + ih$ where h is the stepsize.

Numerical methods

- Discretization in time $\Longrightarrow$

$$y_{i+1} = F(y_{i-q}, y_{i-(q-1)}, \dots, y_i, f) \qquad i = q, \dots, N-1.$$

where

- $y_i \approx y(t_i)$,
- $t_i = t_0 + ih$ where h is the stepsize.

Class of explicit numerical methods :

$$y_{i+1} = y_i + h \sum_{l=1}^{q} a_l k_{l,i} \tag{2}$$

where $k_{l,i} = f(t_{l,i}, y_{l,i})$.

Mixed precision explicit methods

- Compute one/more stages $k_{\ell_0,i}$ in a lower precision.

Method	Mixed method
Explicit method	$\tilde{y}_{i+1} = y_i + h \sum_{\substack{l=1 \\ l \neq \ell_0}}^{q} a_l k_{l,i} + h a_{\ell_0} \tilde{k}_{\ell_0,i}$

where $\tilde{k}_{\ell_0,i} = f(t_{\ell_0,i}, y^{(L)}_{\ell_0,i})$ with $y^{(L)}_{\ell_0,i} = Low(y_{\ell_0,i})$.

- What about the error $\max_{0 \leq i \leq N} \|y_i - \tilde{y}_i\|$?

Theorem

Let's assume that the function $f(t,y)$ is Lipschitz with respect to the variable y with constant c_0. Then :

- *If, for all iterations i, we have computed an approximation $\tilde{k}_{\ell_0,i}$ of $k_{\ell_0,i}$ in a lower precision such that $\|y^{(L)}_{\ell_0,i} - y_{\ell_0,i}\| \leq \varepsilon$. Then we have*

$$\max_{0\leq i\leq N} \|y_i - \tilde{y}_i\| \leq (T - t_0)|a_{\ell_0} c_0|\varepsilon + \mathcal{O}(h\varepsilon).$$

- *If we compute the numerical method in single precision then we get*

$$\max_{0\leq i\leq N} \|y_i - \tilde{y}_i\| = \mathcal{O}\left(\frac{\varepsilon}{h}\right). \blacksquare$$

Explicit Runge-Kutta 4

Runge-Kutta 4 (RK4) :

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

with

$$\begin{aligned} k_1 &= f(t_i, y_i), \\ k_2 &= f(t_i + \frac{h}{2}, y_i + hk_1/2), \\ k_3 &= f(t_i + \frac{h}{2}, y_i + hk_2/2), \\ k_4 &= f(t_{i+1}, y_i + hk_3). \end{aligned}$$

Mixed precision explicit numerical methods

- Compute one/more stages k_i in a lower precision.

Method	Expression	Mixed method
RK4	$\tilde{y}_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$	$A_1 A_2 A_3 A_4$

Table: $A_i \in \{S, D\}$ the precision of the stage k_i

Mixed precision explicit numerical methods

- Compute one/more stages k_i in a lower precision.

Method	Expression	Mixed method
RK4	$\tilde{y}_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$	$A_1A_2A_3A_4$

Table: $A_i \in \{S, D\}$ the precision of the stage k_i

For example, *SSDD* means that we have computed k_1, k_2 in single precision, k_3, k_4 in double precision and all other operations are performed in double precision.

Computational environment

- Compiler : Fortran
- Parallel computing : MPI, Ethernet, 25 Gbps
- Cluster : Gros on Grid5000, Intel Xeon Gold 5220, 2.20GHz
- # processors : 250

Numerical Experiments

Let's consider the following benchmark model, a large nonlinear densely coupled ODEs[1], $(S_i)_{i=1,\dots,N_h}$, where (S_i) is given by:

$$
\begin{aligned}
\frac{dy_1^{(i)}}{dt} &= \frac{1}{\tau}\left(\frac{\nu_{1b}(y_7^{(i)} + \psi_{1i})}{k_{1b}\left(1 + \left(\frac{y_3^{(i)}}{k_{1i}}\right)^{p_0}\right) + y_7^{(i)} + \psi_{1i}} - k_{1d}\,y_1^{(i)}\right) \\
\frac{dy_2^{(i)}}{dt} &= \frac{1}{\tau}\left(k_{2b}(y_1^{(i)})^q - k_{2d}\,y_2^{(i)} - k_{2t}\,y_2^{(i)} + k_{3t}\,y_3^{(i)}\right) \\
\frac{dy_3^{(i)}}{dt} &= \frac{1}{\tau}\left(k_{2t}\,y_2^{(i)} - k_{3t}\,y_3^{(i)} - k_{3d}\,y_3^{(i)}\right)
\end{aligned}
$$

[1] R. El cheikh, S. Bernard, N. El Khatib, *A multiscale modelling approach for the regulation of the cell cycle by the circadian clock*, Journal of Theoretical Biology 426 (2017) 117-125

$$
\begin{aligned}
\frac{dy_4^{(i)}}{dt} &= \frac{1}{\tau}\left(\frac{\nu_{4b}(y_3^{r_0})^{(i)}}{k_{4b}^{r_0}+(y_3^{r_0})^{(i)}} - k_{4d}y_4^{(i)}\right) \\
\frac{dy_5^{(i)}}{dt} &= \frac{1}{\tau}\left(k_{5b}y_4^{(i)} - k_{5d}y_5^{(i)} - k_{5t}y_5^{(i)} + k_{6t}y_6^{(i)}\right) \\
\frac{dy_6^{(i)}}{dt} &= \frac{1}{\tau}\left(k_{5t}y_5^{(i)} - k_{6t}y_6^{(i)} - k_{6d}y_6^{(i)} + k_{7a}y_7^{(i)} - k_{6a}y_6^{(i)}\right) \\
\frac{dy_7^{(i)}}{dt} &= \frac{1}{\tau}\left(k_{6a}y_6^{(i)} - k_{7a}y_7^{(i)} - k_{7d}y_7^{(i)}\right) \\
\frac{dy_8^{(i)}}{dt} &= \lambda\left(\frac{(k_{impf}+k_{0mpf}\exp(-\eta\psi_2))k_{1mpf}^{n_0}}{(k_{1mpf}^{n_0}+(y_8^{(i)})^{n_0}+sy_{10}^{n_0})(1-y_8^{(i)})} - d_{wee1}y_9^{(i)}y_8^{(i)}\right)
\end{aligned}
$$

$$\frac{dy_9^{(i)}}{dt} = \lambda\left(\frac{k_{actw}}{k_{actw}+dw_1}(cw + C(y_7 - bbmal_0) + bbmal_0) + \ldots \right.$$
$$\left. (\frac{k_{actw}}{k_{actw}+dw_1} - 1)\frac{kin_{actw}(y_8^{(i)})^{n_0} y_9^{(i)}}{k^{n_0}{}_{1wee_1} + (y_8^{(i)})^{n_0}} - dw_2 y_9^{(i)}\right)$$
$$\frac{dy_{10}^{(i)}}{dt} = \lambda\left(kk_{act}(y_8^{(i)} - y_{10}^{(i)})\right)$$

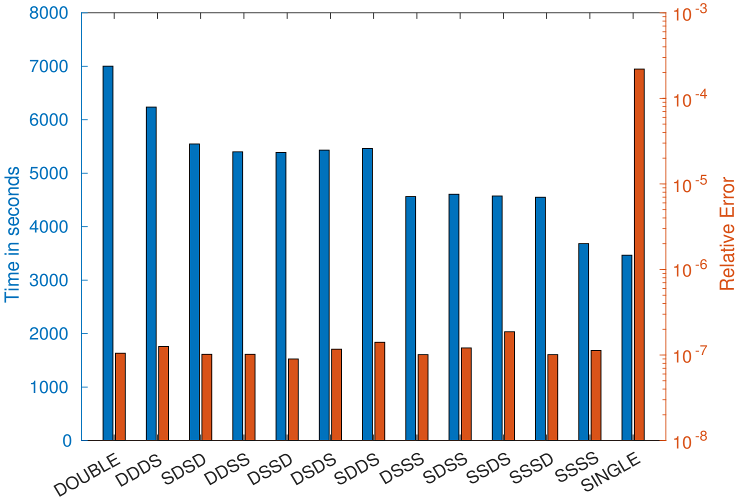
where $t \in [0, 120]$ and ψ_{1i}, ψ_{2i}, for $i = 1, \ldots, N_h$, are given by

$$\psi_{1i} = \frac{ks}{N_h}\sum_{j=1}^{N_h} \arctan(y_2^{(j)} - y_2^{(i)}) + \frac{\pi}{2}ks$$
$$\psi_{2i} = N_h = 10^4.$$

- Let $y_{ref} \approx y(120)$ be the solution, in double precision, computed with the step size $h_{ref} = \frac{120}{5\times 10^5}$.
- The step size h in the RK4 method is $120/10^5$.

Method	Time	$\frac{\|y-y_{ref}\|_\infty}{\|y_{ref}\|_\infty}$	$\frac{Time(Double)}{Time(Mixed)}$
DOUBLE	7 002	1.05×10^{-7}	1
DDDS	6 237	1.26×10^{-7}	1.12
SDSD	5 547	1.02×10^{-7}	1.26
DDSS	5 400	1.02×10^{-7}	1.29
DSSD	5 389	8.99×10^{-8}	1.3
DSDS	5 432	1.17×10^{-7}	1.28
SDDS	5 464	1.41×10^{-7}	1.28
DSSS	4 563	1.01×10^{-7}	1.53
SDSS	4 607	1.21×10^{-7}	1.51
SSDS	4 574	1.87×10^{-7}	1.53
SSSD	4 551	1.01×10^{-7}	1.53
SSSS	3 683	1.13×10^{-7}	1.9
SINGLE	**3 467**	2.2×10^{-4}	2.01

Table: $N = 100\ 000$, $\|y_{ref}\|_\infty = 2.5923$

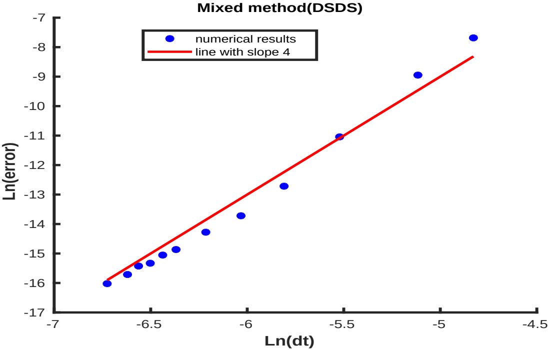

Time in seconds
Relative Error
0
1000
2000
3000
4000
5000
6000
7000
8000
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
DOUBLE
DDDS
SDSD
DDSS
DSSD
DSDS
SDDS
DSSS
SDSS
SSDS
SSSD
SSSS
SINGLE

- We have implemented, using mixed precisions arithmetic, numerical methods (explicit and implicit), on CPU cluster with MPI, for solving large systems of ODE's.
- These methods **run faster and use less memory with the same quality** as methods running in "full precision".
- The numerical results show the **parallel methods running in mixed precision are up to** $1.5 - 2$ **times faster** than those running in double precision with sufficient accuracy.

- Implemented Mixed precision's numerical methods on a Tensor Processing Unit (matrix processor).
- Tested our approach on another numerical methods.
- Use other type of precisions.

Thank you for your attention.

Order of the Mixed method DSDS



Method	Time	$\|y_D - y_M\|_\infty$	$\|y_S - y_M\|_\infty$	$\|\frac{y_D - y_M}{y_D}\|_\infty$
DOUBLE	7 002	0	5.8441×10^{-4}	0
DDDS	6 237	5.9679×10^{-8}	5.8441×10^{-4}	2.5640×10^{-7}
DDSS	5 400	1.4211×10^{-7}	5.8441×10^{-4}	4.9659×10^{-7}
DSDS	5 432	1.1512×10^{-7}	5.8441×10^{-4}	4.8019×10^{-7}
DSSD	5 389	2.1592×10^{-7}	5.8441×10^{-4}	9.0710×10^{-7}
SDDS	5 464	1.1089×10^{-7}	5.8441×10^{-4}	4.9863×10^{-7}
SDSD	5 547	1.4250×10^{-7}	5.8441×10^{-4}	4.9758×10^{-7}
DSSS	4 563	2.3106×10^{-7}	5.8441×10^{-4}	9.3677×10^{-7}
SDSS	4 607	1.7184×10^{-7}	5.8441×10^{-4}	5.9969×10^{-7}
SSDS	4 574	2.7871×10^{-7}	5.8441×10^{-4}	1.0509×10^{-6}
SSSD	4 551	2.3202×10^{-7}	5.8441×10^{-4}	9.3439×10^{-7}
SSSS	3 683	2.3963×10^{-7}	5.8441×10^{-4}	9.2155×10^{-7}
SINGLE	3 467	5.8441×10^{-4}	0	2.0789×10^{-3}

Method	Time	$\frac{\lVert y-y_{ref}\rVert_\infty}{\lVert y_{ref}\rVert_\infty}$	$\frac{Time(Double)}{Time(Mixed)}$
DOUBLE	8 334	1.28×10^{-5}	1
DS	4 563	1.28×10^{-5}	1.8
SD	4 453	1.28×10^{-5}	1.87
SS	3 827	1.28×10^{-5}	2.1
SINGLE	**3 621**	2.39×10^{-4}	2.3

Table: **Runge-Kutta** 2, Nova, #procs = 128

type	Time	$\|\|y_D - y_M\|\|_\infty$	$\|\|y_S - y_M\|\|_\infty$	$\|\|\frac{y_D - y_M}{y_D}\|\|_\infty$
DOUBLE	8 334	0	6.0503×10^{-4}	0
DS	4 563	2.6678×10^{-7}	6.0513×10^{-4}	9.9485×10^{-7}
SD	4 453	1.7214×10^{-7}	6.0503×10^{-4}	5.8557×10^{-7}
SS	3 827	3.3097×10^{-7}	6.0509×10^{-4}	1.1138×10^{-6}
SINGLE	3 621	6.0503×10^{-4}	0	2.1676×10^{-3}

Table: **Runge-Kutta** 2, Nova, $\#\text{procs} = 128$

.

Method	Time	$\frac{\|y - y_{ref}\|_\infty}{\|y_{ref}\|_\infty}$	$\frac{Time(Double)}{Time(Mixed)}$
DOUBLE	3 509	3.4×10^{-5}	1
SS	1 942	3.4×10^{-5}	1.8
SINGLE	**1 188**	2.3×10^{-4}	2.3

Table: **Adams Bashforth** 2, Nova, $\#procs = 128$

Method	Time	$\|y_D - y_M\|_\infty$	$\|y_S - y_M\|_\infty$	$\|\frac{y_D - y_M}{y_D}\|_\infty$
DOUBLE	3 509	0	5.6×10^{-4}	0
SS	1 942	3.4×10^{-7}	5.6×10^{-4}	1.331×10^{-6}
SINGLE	1 188	5.6×10^{-4}	0	2.026×10^{-3}

Table: **Adams Bashforth** 2, Nova, $\#\text{procs} = 128$

Thank you for your attention.

Implicit methods for solving the ODE

We have

$$y_{i+1} = a_i + h\beta_0 f(t_{i+1}, y_{i+1}) \qquad i = q, \ldots, N-1,$$

where $a_i = F(y_i, y_{i-1}, \ldots, y_{i-q+1})$.
Let $x_i = \frac{y_{i+1} - a_i}{\beta_0}$ then the numerical method becomes

$$G_i(x_i) = 0, \tag{3}$$

where

$$G_i(x) = x - hf(t_{i+1}, a_i + \beta_0 x).$$

Newton method

1. The nonlinear system $G_i(x_i) = 0$ is solved by the Newton method.

Newton method

1. The nonlinear system $G_i(x_i) = 0$ is solved by the Newton method.
2. The Newton method computes a sequence $(x_i^{(m)})$ such that $\lim\limits_{m \to +\infty} x_i^{(m)} = x_i$ when $x_i^{(0)} \approx x_i$.

Newton method

1. The nonlinear system $G_i(x_i) = 0$ is solved by the Newton method.
2. The Newton method computes a sequence $(x_i^{(m)})$ such that $\lim_{m \to +\infty} x_i^{(m)} = x_i$ when $x_i^{(0)} \approx x_i$.
3. This sequence is given by $x_i^{(m+1)} = x_i^{(m)} + z_i^{(m)}$ where $z_i^{(m)}$ is the solution of

$$G_i'(x_i^{(m)}) z_i^{(m)} = -G_i(x_i^{(m)}).$$

Newton method

1. The nonlinear system $G_i(x_i) = 0$ is solved by the Newton method.
2. The Newton method computes a sequence $(x_i^{(m)})$ such that $\lim_{m\to+\infty} x_i^{(m)} = x_i$ when $x_i^{(0)} \approx x_i$.
3. This sequence is given by $x_i^{(m+1)} = x_i^{(m)} + z_i^{(m)}$ where $z_i^{(m)}$ is the solution of

$$G_i'(x_i^{(m)})z_i^{(m)} = -G_i(x_i^{(m)}).$$

4. To compute good initial guess $x_i^{(0)}$ we use the Line-search algorithm.

Line-search algorithm for computing $x_i^{(0)}$

- The goal of this algorithm is to compute a local minimum of the function H_i defined by $H_i(x) = \frac{1}{2}\|G_i(x)\|^2$.

Line-search algorithm for computing $x_i^{(0)}$

- The goal of this algorithm is to compute a local minimum of the function H_i defined by $H_i(x) = \frac{1}{2}\|G_i(x)\|^2$.
- It computes a sequence $(u_i^{(k)})_{k\geq 0}$ s.t. $\nabla H_i(u_i^{(k)}) \underset{k\to+\infty}{\to} 0$.

Line-search algorithm for computing $x_i^{(0)}$

- The goal of this algorithm is to compute a local minimum of the function H_i defined by $H_i(x) = \frac{1}{2}\|G_i(x)\|^2$.
- It computes a sequence $(u_i^{(k)})_{k\geq 0}$ s.t. $\nabla H_i(u_i^{(k)}) \underset{k\to+\infty}{\longrightarrow} 0$.
- $u_i^{(k+1)} = u_i^{(k)} + \lambda_k p_i^{(k)}$ where $p_i^{(k)}$ solves

$$(G_i'(u_i^{(k)}))p_i^{(k)} = -G_i(u_i^{(k)}).$$

Line-search algorithm for computing $x_i^{(0)}$

- The goal of this algorithm is to compute a local minimum of the function H_i defined by $H_i(x) = \frac{1}{2}\|G_i(x)\|^2$.
- It computes a sequence $(u_i^{(k)})_{k\geq 0}$ s.t. $\nabla H_i(u_i^{(k)}) \underset{k\to+\infty}{\rightarrow} 0$.
- $u_i^{(k+1)} = u_i^{(k)} + \lambda_k p_i^{(k)}$ where $p_i^{(k)}$ solves

$$(G_i'(u_i^{(k)}))p_i^{(k)} = -G_i(u_i^{(k)}).$$

- λ_k is a positive scalar chosen to satisfy the Goldstein-Armijo condition:

$$h_i(u_i^{(k+1)}) \leq h_i(u_i^{(k)}) + \alpha\lambda_k \nabla H_i(u_i^{(k)})^T p_i^{(k)} \tag{4}$$

where $\alpha \in (0, 1/2)$ is a parameter typically set to 10^{-4}.

Solution of the linear systems $G_i'(x_i^{(m)})z_i^{(m)} = -G_i(x_i^{(m)})$ and $(G_i'(u_i^{(k)}))p_i^{(k)} = -G_i(u_i^{(k)})$.

Solution of the linear systems $G_i'(x_i^{(m)})z_i^{(m)} = -G_i(x_i^{(m)})$ and $(G_i'(u_i^{(k)}))p_i^{(k)} = -G_i(u_i^{(k)})$.

- These linear systems are solved by an iterative method.

Solution of the linear systems $G_i'(x_i^{(m)})z_i^{(m)} = -G_i(x_i^{(m)})$ and $(G_i'(u_i^{(k)}))p_i^{(k)} = -G_i(u_i^{(k)})$.

- These linear systems are solved by an iterative method.

- This method computes an approximation $\tilde{x}$ of a linear system, of the form $Ax = b$, such that

$$\|A\tilde{x} - b\| < \varepsilon_1 \|b\|, \tag{5}$$

where $\varepsilon_1 > 0$ is some tolerance threshold.

Solution of the linear systems $G_i'(x_i^{(m)})z_i^{(m)} = -G_i(x_i^{(m)})$ and $(G_i'(u_i^{(k)}))p_i^{(k)} = -G_i(u_i^{(k)})$.

- These linear systems are solved by an iterative method.

- This method computes an approximation $\tilde{x}$ of a linear system, of the form $Ax = b$, such that

$$\|A\tilde{x} - b\| < \varepsilon_1 \|b\|, \tag{5}$$

where $\varepsilon_1 > 0$ is some tolerance threshold.

- We will use the following mixed precision iterative method.

Algorithm 1 Mixed Precision GMRES (Generalized Minimal RESidual) for $Az = b$

Input: z_0 **initial guess**

1: for $i = 0, ...imax$ or until converged
2: Compute $r_i = b - Az_i$ in **double** precision.
3: Solve $Au = r_i$ in **single** precision.
4: Update $z_{i+1} = z_i + u$ in **double** precision.
5: end

Theorem

Let's suppose that for any vector v, we have $\|v - v^{(L)}\| = \mathcal{O}(\varepsilon)$. Let's $x^{(M)}$ and $x^{(D)}$ denote, respectively, the solution computed by Mixed precision GMRES and double precision GMRES. Then we have:

$$\|b - Ax^{(M)}\| = \|b - Ax^{(D)}\| + \mathcal{O}(\varepsilon).$$

Remark

If ε is small enough then the mixed solution $x^{(M)}$ satisfies

$$\|b - Ax^{(M)}\| \leq c\|b\|$$

where $c < 1$, and therefore $\tilde{p}_i^{(k)}$ is a descent direction and the Newton method will be locally convergent.

Mixed precision implicit numerical method (MPIN)

(MPIN1) Set $i = 0$. For $i \leq NT$, compute the numerical method y_i as follows :

A) Line-search (LS) algorithm for computing $x_i^{(0)}$:

(LS1) Let $u_i^{(0)} = 0$. While $\|G_i(u_i^{(k)})\| > \varepsilon$, do:

(LS2) Compute an approximation $\tilde{p}_i^{(k)} \approx p_i^{(k)}$ by **MP GMRES.**

(LS3) Compute $u_i^{(k+1)} = u_i^{(k)} + \lambda_k \tilde{p}_i^{(k)}$.

(LS4) Set $x_i^{(0)} = u_i^{(k)}$.

B) Inexact Newton (IN):

(IN1) Let $m = 0$ and repeat until $\|G_i(x_i^{(m)})\| \leq \varepsilon$:

(IN2) Compute an approximation $\tilde{z}_i^{(m)} \approx z_i^{(m)}$ by **MP GMRES.**

(IN3) $x_i^{(m+1)} = x_i^{(m)} + \tilde{z}_i^{(m)}$.

(MPIN2) Set $x_i = x_i^{(m)}$.

(MPIN3) Compute $y_{i+1} = a_i + \beta_0 x_i$.

(MPIN4) $i = i + 1$.

(MPIN5) $y_M = y_{NT}$.

- Let T_M be the total time in seconds for computing (y_i) by the above algorithm.
- (T_D, Y_D) for Double precision algorithm.
- (T_S, Y_S) for Single precision algorithm.

In the table below, we present the numerical results when the circadian clock's problem is solved by the BDF method of order 2 that is given by

$$y_{i+2} = \frac{4}{3} y_{i+1} - \frac{1}{3} y_i + \frac{2}{3} h f(t_{i+2}, y_{i+2}).$$

NT	T_S	T_M	T_D	$\frac{\lVert y_{ref} - y_S \rVert_\infty}{\lVert y_{ref} \rVert_\infty}$	$\frac{\lVert y_{ref} - y_M \rVert_\infty}{\lVert y_{ref} \rVert_\infty}$	$\frac{\lVert y_{ref} - y_D \rVert_\infty}{\lVert y_{ref} \rVert_\infty}$
1000	2 394	2 900	4 125	8.3×10^{-3}	8.2×10^{-3}	8.3×10^{-3}
10000	5 565	8 387	10 051	1.3×10^{-3}	3.9×10^{-4}	3.9×10^{-4}
20000	15 190	22 943	26 773	6.4×10^{-3}	5.5×10^{-4}	5.5×10^{-4}

Table: **BDF2** scheme, with $n = 100\ 000$, cluster = nova, nbprocs = 128.

NT	$\|y_D - y_M\|_\infty$	$\|y_S - y_M\|_\infty$	$\|y_D - y_S\|_\infty$	$\|\frac{y_D - y_M}{y_D}\|_\infty$	$\|\frac{y_D - y_S}{y_D}\|_\infty$
1000	9.8×10^{-14}	5×10^{-4}	5×10^{-4}	5×10^{-13}	1.9×10^{-3}
10000	1.1×10^{-4}	3.2×10^{-3}	3.2×10^{-3}	3.4×10^{-4}	1.1×10^{-2}
20000	4.4×10^{-16}	1.7×10^{-2}	1.7×10^{-2}	6.2×10^{-16}	6.1×10^{-2}

Table: **BDF2** scheme, with $n = 100\ 000$, cluster = nova, nbprocs = 128.

NT	$\frac{1}{N_h}\sum_{l=1}^{N_h}\left\Vert\frac{y_D(l)-y_M(l)}{y_D(l)}\right\Vert_\infty$	$\frac{1}{N_h}\sum_{l=1}^{N_h}\left\Vert\frac{y_D(l)-y_S(l)}{y_D(l)}\right\Vert_\infty$
1000	1.0×10^{-13}	2.4×10^{-4}
10000	5.8×10^{-8}	1.5×10^{-3}
20000	4.4×10^{-17}	8.8×10^{-3}

Table: **BDF2** scheme, with $n = 100\ 000$, cluster $=$ nova, nbprocs $= 128$.

Thank you for your attention.