Parallel Implicit Runge-Kutta Methods for Stiff ODEs

Bashir M. S. Khalaf  Abdulhabib A. A. Murshid

College Of Education
University of Mosul

ABSTRACT

The main objective of this paper is to develop and construct numerical algorithms for solving stiff system of ordinary differential equations (ODEs) which are suitable for running on parallel computers (MIMD computers). Semi-parallel implicit Runge-Kutta methods have been derived and parallel predictor - corrector methods are developed.
1- Arithmetic Mean and Geometric Mean Runge-Kutta Methods:

The general form of an r-step arithmetic mean (AM) Runge-Kutta method is:

\[ K_1 = f(y_n), \quad K_i = (y_n + h \sum_{j=1}^{i} b_{ij} K_j) \]

and

\[ y_{n+1} = y_n + \sum_{i=1}^{r} w_i K_i, \quad \sum_{i=1}^{r} w_i = 1 \]

with appropriate values of the b’s and w’s. If instead of the weighted arithmetic mean in eq. (1) we use the geometric mean, then the geometric mean (GM) Runge-Kutta formula is [2]:

\[ y_{n+1} = y_n + \prod_{i=1}^{r} K_1^{P_i}, \quad \sum_{i=1}^{r} P_i = 1 \quad \text{(1d)} \]

An example of the AM- Runge-Kutta formula is the 4th-order formula generated by the symbol matrix [2]:

\[
\begin{bmatrix}
0 & 0 & 0 & 0 \\
1/2 & 0 & 0 & 0 \\
0 & 1/2 & 0 & 0 \\
0 & 1/2 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1/6 & 1/3 & 1/6
\end{bmatrix}_{AM}
\]

And an example of the GM-Runge-Kutta formula is the 4th-order formula generated by the symbol matrix
If, while maintaining the order of accuracy of the formula, we can choose some of the w's, e.g. w, to vanish and the subsequent k's do not depend on k;, then an economical formula is obtained. An example of such formula is the 4\textsuperscript{th}-order formula generated by the symbol matrix

\[
\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\frac{1}{2} & 0 & 0 & 0 \\
\frac{1}{24} & 0 & 0 & 0 \\
\frac{7}{24} & \frac{5}{24} & 0 & 0 \\
\frac{-1}{4} & -\frac{59}{20} & \frac{21}{5} & 0 \\
\frac{1}{6} & -\frac{1}{3} & 1 & \frac{1}{6} \\
\end{bmatrix}_{AM}
\]

We can get the formula:
\[k_1 = f(x_n, y_n), \quad k_2 = f(x_n + h/2, y_n + h/2k_1), \quad k_4 = f(x_n + h, y_n + hk_1) \quad (2a)\]
and
\[y_{n+1} = y_n + l/6h (k_1 + 4k_2 + k_4) \quad (2b)\]

In this formula, we notice that while it is effectively a 3-stage formula generated by the symbol matrix.
The quantity $K_4$ does not depend on $k_2$ and thus allows us to compute $k_2$ and $k_4$ simultaneously on a parallel computer. By replacing $h$ by $-h$ in the formula (2) we obtain the corresponding implicit formula as:

$$L_1 = f(x_{n+1}, y_{n+1}),$$
$$L_2 = f(x_{n+1} - 1/2h, y_{n+1} + h/2 L_1),$$
$$L_4 = f(X_{n+1} - h, y_{n+1} - h, y_{n+1} - h L_1),$$

and

$$y^*_{n+1} = Y_n + h/6 (L_1 + 4L_2 + L_4) .$$

Now we will try to find other RK formulas of the type given by eqs (2) and (3) and other forms:

2- Semi-Parallel Implicit 3-Stage AM-Runge-Kutta Method:

The 3-stage AM-Runge-Kutta formula is of the form:

$$k_1 = f(x_n, y_n) \quad (4a)$$
$$k_2 = f(x_n + a_1 h, y_n + a_1 h k_1) \quad (4b)$$
$$k_3 = f (x_n, + h, y_n + a_2 h k_1 + a_3 h k_2), \quad (4c)$$

and

$$y_{n+1} = y_n + h (w_1 k_1 + w_2 k_2 + w_3 k_3) \quad (4d)$$

By setting $a_2 + a_3 = 1$ and by comparing the r.h.s. of equation (4d) with the Taylor series expansion for $y(X_{n+1})$, the following equations of conditions were obtained:

$$h^2 f.fy : w_2 B_1 + w_3 (a_2 + a_3) = 1/2 ,$$
$$h^3 f^2 fyy : 1/2 W2ai2 + 1/2 W3 (a2 + a3)2 = 1/6 ,$$

82
Parallel Implicit Runge-Kutta…

$h^3 f^2 y : w_3 a_1 a_3 = 1/6$

For the purpose of parallel computation we require that $a_3 = 0$, therefore, by solving this system, for $w_2$ and $w_3$ in terms of $a_1$ and $a_2$ we get:

$$W_2 = \frac{-3a_2 + 2}{6a_1(a_1 - a_2)} \quad \text{and} \quad W_3 = \frac{-3a_1 - 2}{6a_2(a_1 - a_2)}$$

Since $W_1 + W_2 + W_3 = 1$ and by setting $a_1 = V_i$, we find that the solution is:

$$a_1 = 1/2, \quad a_2 = 1, \quad w_1 = 1/6, \quad w_2 = 2/3, \quad w_3 = 1/6$$

The resulting formula is thus;

$$k_1 = f(x_n, y_n), \quad k_2 = f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1), \quad k_3 = f(x_n + h, y_n + hk_1)$$

and

$$y_{n+1} = y_n + 1/6h(k_1 + 4k_2 + k_3) \quad (6)$$

which is a semi-parallel 3-stage AM-Runge-Kutta method, because evaluations of $k_2$ and $k_4$ are independent, so they can be computed simultaneously, but sequentially to $k_1$, so that the formula is semiparallel explicit Runge-Kutta (SPERK) method.

Since we are solving stiff ODEs[8], we require implicit forms. By replacing $h$ by $-h$ (i.e. backward integration) in the formula (6) we obtain the corresponding implicit formula as:

$$L_1 = f(x_n + h, y_n + hL_1), \quad L_2 = f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hL_1),$$

$$L_4 = f(x_n - h, y_n - hL_1) \quad \begin{cases} L_1 = f(x_n + h, y_n + hL_1), \quad L_2 = f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hL_1), \quad L_4 = f(x_n - h, y_n - hL_1) \end{cases} \quad (7)$$

And

$$y_{n+1} = y_n + \frac{h}{6}(L_1 + 4h_2 + L_4),$$

83
In formula (7) we see that computing $y_{n+1}$ is depending on itself so that the method is Implicit Runge-Kutta (IRK). Also, we note that $L_4$ does not depend on $L_2$, so they can be computed in parallel (i.e. simultaneously) using two different processors. But simultaneous evaluation of $L_2$ and $L_4$ is sequential to evaluation of $L_1$, therefore, the method is a semi-parallel IRK (SPIRK) method.

The parallel predictor-corrector form of (6) and (7) is:

$$y_{n+1}^p = y_n^c + \frac{h}{6}(K_1^c + K_2^c + K_4^c)$$

$$y_{n+1}^c = y_n^p + \frac{h}{6}(L_1^p + L_2^p + L_4^p)$$

Where

$$K_1^c = f(x_n, y_n^c), \quad L_1^p = f(x_{n+1}, y_{n+1}^p)$$

$$K_2^c = f(x_n + \frac{h}{2}, y_n^c + \frac{h}{2}K_1^c), \quad L_2^p = f(x_{n+1} - \frac{h}{2}, y_{n+1}^p - \frac{h}{2}L_1^p)$$

$$K_4^c = f(x_n + h, y_n^c + hK_1^c), \quad L_4^p = f(x_{n+1} - h, y_{n+1}^p - hL_1^p)$$

(8)
The computation diagram is:

Fig. (1) The computation diagram of (8).
3- Semi-Parallel Implicit Geometric Mean Runge-Kutta (SPIGMRK) Method:

Now we repeat the analysis of equations (4) with (4d) replaced by

$$Y_{n+1} = y_n + hK_1^{(1-p-q)}K_2^pK_3^q, \ 0<p, q<l$$

(9)

to get a 2-stage SPIGMRK formula.

The Taylor series for $y(x_{n+1})$ are:

$$h^2f_{yy}: pa_1 + q(a_2 + a_3) = 1/2$$

(10a)

$$h^3f_{yy}: l/2p a_1^2 + l/2q(a_2 + a_3)^2 = 1/6$$

(10b)

$$h^3f_{y^2}: Pqa_1(a_2 + a_3) + l/2p(p - l) a_1^2 + qa_1 a_3$$

+ $l/2q(q - 1)(a_2 + a_3)^2 = 1/6$

By putting $a_3 = 0$, we get:

$$P a_1 + q a_2 = l/2$$

(11a)

$$P a_1^2 + q a_2^2 = l/3$$

(11b)

$$P q a_1 a_2 l/2P(P - l) a_1^2 = l/2q(q - 1) a_1^2 = l/6$$

(11c)

Unlike the linear case, (10c) is not immediately violated with the choice $a_3 = 0$. Solving this system for $p$ and $q$ from (11a) and (11b) in terms of $a_1$ and $a_2$ we have:

$$P = \frac{-3a_2 + 2}{6a_1(a_1 - a_2)} \quad \text{and} \quad q = \frac{3a_1 - 2}{6a_2(a_1 - a_2)}$$

(12)

Therefore with $a_1$ and $a_2$ chosen to satisfy (11a) and (11b) result in (11c) is not being satisfied. The resulting formula is thus order 2. As an example of the solution, setting $a_1 = 2/3$ and $a_2 = 1$ we obtain $p=3/4$ and $q=0$. The resulting formula is given by:

$$k_1 = f(x_n, y_n), \ k_2 = f(x_n + 2/3h, y_n + 2/3hk_1), \ k_3 = f(x_n + h, y_n + hk_1)$$

and

$$y_{n+1} = y_n + h k_1^{1/4} k_2^{1/4}$$

This is a parallel 2-stage formula since $k_3$ does not need to be evaluated. The corresponding implicit formula is:

$$L_1 = f(x_{n+1}, y_{n+1}), \ L_2 = f(x_{n+1} - 2/3h, y_{n+1} - 2/3hL_1), \ L_3 = f(x_{n+1} - h, y_{n+1} - hL_1$$

86
and
\[ y_{n+1}^* = y_n + hL_1^{1/4}L_2^{3/4} \]

4. Another semi–parallel implicit 2-stage GM–Runge–Kutta Formula:

Now, we try to find the solution to satisfy equations (11a) and (11c) but not (11b), we have:
\[ q a_2 = \frac{1}{2} - p a_1 \]
and substituting into (11c) we get:
\[ P a_1(1/2-pa_1)+1/2p(p-1)a_1^2+1/2(1/2-pa_1)(1/2-pa_1-a_2)=1/6 \]
Putting \( P a_1 = 0 \)
-eq.n. (11a) becomes \( q a_2=1/2 \),  -eqn (15) becomes \( a_2=-1/6 \)
a n example of the solution is:
\[ P=0 , a_1=0, a_2=-1/6 , q=-3 \]
which leads to the following formula:
\[ K_1 = f(x_n+1/6h , y_n-l/6hk_1) \]
And
\[ y_{n+1} = y_n + hk_1^{1/4}k_3^{-3} \]
which is a semi-parallel 2-stage GM-Runge-Kutta method. The corresponding implicit formula for this method is:
\[ L_1=f(x_n,y_n), L_3=f(x_n-1/6h, y_n+1/6hL_1), \]
And
\[ y_{n+1}^* = y_n + hL_1^{1/4}L_3^{-3} \]
Form (17) is considered as a parallel mode because the calculation of \( L_2 \) is not needed.

5-semi-parallel implicit 3rd order Runge-Kutta method:
In similar way as described previously, we can obtain another (SPIRK) of order three in the following form:
\[ k_1= f(x_n, y_n), k_2 = (x_n+1/4h, y_{n+1}-1/4hL_1), \]
\[ L_3=f (x_{n+1}-h, y_{n+1}-hL_1) \]
And
The corresponding implicit formula is:

\[ L_1 = (x_{n+1}, y_{n+1}), \quad L_2 = (x_{n+1} - 1/4h, y_{n+1} - 1/4hL_1), \quad L_3 = (x_{n+1} - h, y_{n+1} - hL_1), \]

And

\[ y_{n+1} = y_n + \frac{h}{12} (3L_1 + 4L_2 + 5L_1) \]  

6-Fourth order parallel predictor-corrector RK (PPCRK) method:

The general form of fourth order Runge-Kutta method is:

\[
\begin{align*}
  k_1 &= f(x_n, y_n), \\
  k_2 &= f(x_n + c_2 h, y_n + c_2 k_1), \\
  k_3 &= f(x_n + c_3 h, y_n + h(a_3 k_2 + (c_3 - a_3) k_1)), \\
  k_4 &= f(x_n + c_4 h, y_n + h(a_4 k_2 + a_4 k_3 + (c_4 - a_4 + a_4 k_1))), \\
  y_{n+1} &= y_n + h(w_1 k_1 + w_2 k_2 + w_3 k_3 + w_4 k_4)
\end{align*}
\]

By using series expansion, one obtains after rather complicated calculations, the following system [35]

\[
\begin{align*}
  w_1 + w_2 + w_3 + w_4 &= 1, \\
  w_3 c_2 a_{31} + w_4 (c_3 a_{42} + c_2 a_{41}) &= 1/6, \\
  w_2 c_2 + w_3 c_3 + w_4 c_4 &= 1/2, \\
  w_3 c_2 a_{31} + w_4 c_4 (c_3 a_{42} + c_2 a_{41}) &= 1/8, \\
  w_2 c_2^2 + w_3 c_3^2 + w_4 c_4^2 &= 1/3, \\
  w_3 c_2 a_{31} + w_4 c_4 (c_3^2 a_{42} + c_2 a_{41}) &= 1/12, \\
  w_2 c_2^3 + w_3 c_3^3 + w_4 c_4^3 &= 1/4, \\
  w_4 c_2 a_{42} &= 1/24.
\end{align*}
\]

We have eight equation in ten unknowns, and hence we can choose two quantities arbitrarily. If we assume that \( c_2 = c_3 \), which seems rather natural, we find:

\[
\begin{align*}
  c_2 = c_3 = 1/2, \quad w_1 = w_4 = 1/6, \quad w_2 + w_3 = 2/3, \quad a_{41} + a_{42} = 1, \quad w_3 a_{31} = 1/6
\end{align*}
\]

If we further choose \( w_2 = w_3 \) and for the purpose of parallel computation we require \( a_{42} = 0 \) we get:
Thus we have the final formula:
\[ k_1 = f(x_n, y_n), \]
\[ k_2 = f(x_n + 1/2h, y_n + 1/2hk_1), \]
\[ k_3 = f(x_n + 1/2h, y_n + 1/2hk_2), \]
\[ k_4 = f(x_n + h, y_n + hk_2), \]
\[ y_{n+1} = y_n + h/6 (k_1 + 2k_2 + 2k_3 + k_4) \]

This is a (SPERK) method of order four:

From (20) we can get a semi-parallel implicit RK (SPIRK) method of order four by backward integration process as follows:

\[ L_1 = f(x_{n+1}, y_{n+1}), \]
\[ L_2 = f(x_{n+1} - \frac{1}{2}h, y_{n+1} - \frac{1}{2}hL_1), \]
\[ L_3 = f(x_{n+1} - \frac{1}{2}h, y_{n+1} - \frac{1}{2}hL_2), \]
\[ L_4 = f(x_{n+1} - h, y_{n+1} - hL_2), \]
\[ y_n = y_{n+1} - \frac{h}{6} (L_1 + 2L_2 + 2L_3 + L_4), \]

Rearranging this formula we get:
\[ y_{n+1} = y_n + \frac{h}{6} (L_1 + 2L_2 + 2L_3 + L_4) \]

In this implicit method \( L_3 \) and \( L_4 \) are independent and can be computed in parallel. To get a parallel predictor-corrector RK (PPCRK) mode of the methods (20) and (21), we can write them in the following form:
\[ y_{n+2}^p = y_{n+1}^p + \frac{h}{6} (K_1^p + 2K_2^p + 2K_3^p + K_4^p), \text{ predictor from} \]
\[ y_{n+1}^c = y_n^c + \frac{h}{6} (L_1^p + 2L_2^p + 2L_3^p + L_4^p), \text{ corrector from} \]
Where
\[ K_1 = f(x_{n+1}, y_{n+1}^p), \]
\[ L_1 = f(x_{n+1}, y_{n+1}^p), \]
\[ K_2 = f(x_{n+1} + \frac{1}{2}h, y_{n+1}^p + \frac{1}{2}hK_1^p), \]
\[ L_2 = f(x_{n+1} - \frac{1}{2}h, y_{n+1}^p - \frac{1}{2}hL_1^p), \]
\[ K_3 = f(x_{n+1} + \frac{1}{2}h, y_{n+1}^p + \frac{1}{2}hK_2^p), \]
\[ L_3 = f(x_{n+1} - \frac{1}{2}h, y_{n+1}^p - \frac{1}{2}hL_2^p), \]
\[ K_4 = f(x_{n+1} + h, y_{n+1}^p + hK_2^p), \]
\[ L_4 = f(x_{n+1} - h, y_{n+1}^p - hL_2^p), \]

Where \( y_{1}^p \) is computed in advance

**Fig. (2): Information flow in the PPCRK scheme.**

The problem with this PPCRK mode is that: the predictor formula always compute \( y_{n+2}^p \) depending on predicted values and because we are solving stiff ODEs this means that the predictor formula will be unstable. Hence, we require a modification of the PPCRK mode, so that the predictor formula depends on corrected values, we can develop a formula which repeats the
correction twice: \((y_1, y_2\) and \(y_3\) can be computed first sequentially).

First calculation stage:
\[
y_{n+1}^c = y_n^{cc} + \frac{h}{6} (L_1^p + 2L_2^p + 2L_3^p + L_4^p)
\]

Second calculation stage:
\[
y_{n+2}^p = y_{n+1}^c + \frac{h}{6} (K_1^c + 2K_2^c + 2K_3^c + K_4^c)
\]
\[
y_{n+1}^{cc} = y_n^{cc} + \frac{h}{6} (L_1^c + 2L_2^c + 2L_3^c + L_4^c)
\]

7-PPCRK Method for a system of first order ODEs:
Runge-Kutta method can be applied directly to differential equations of higher order, taking, for example, the equation \(y''=f(x,y,y')\), we put \(y'=z\) and obtain the following system of first-order equation:
\[
y' = z, \quad z' = f(x,y,z)
\]
This is a special case of:
\[
y' = F(x,y,z), \quad z' = G(x,y,z),
\]
(22)

which can be integrated (using method 6):
\[ K_1 = F(x_n, y_n, z_n), \quad L_1 = G(x_n, y_n, z_n), \]

\[ K_2 = F(x_n + \frac{1}{2} h, y_n + \frac{1}{2} hK_1, z_n + \frac{1}{2} hL_1), \]

\[ L_2 = G(x_n + \frac{1}{2} h, y_n + \frac{1}{2} hK_1, z_n + \frac{1}{2} hL_1), \]

\[ K_3 = F(x_n + \frac{1}{2} h, y_n + \frac{1}{2} hK_2, z_n + \frac{1}{2} hL_2), \]

\[ L_3 = G(x_n + \frac{1}{2} h, y_n + \frac{1}{2} hK_2, z_n + \frac{1}{2} hL_2), \]

\[ K_4 = F(x_n + h, y_n + hK_2, z_n + hL_2), \]

\[ L_4 = G(x_n + h, y_n + hK_2, z_n + hL_2), \]

\[ y_{n+1} = y_n + \frac{h}{6} (K_1 + 2K_2 + 2K_3 + K_4), \]

\[ z_{n+1} = z_n + \frac{h}{6} (L_1 + 2L_2 + 2L_3 + L_4), \]

The corresponding implicit formula is:

\[ m_1 = F(x_{n+1}, y_{n+1}, z_{n+1}), \quad S_1 = G(x_{n+1}, y_{n+1}, z_{n+1}), \]

\[ m_2 = F(x_{n+1} - \frac{1}{2} h, y_{n+1} - \frac{1}{2} hS_1, z_{n+1} - \frac{1}{2} hS_1), \]

\[ S_2 = G(x_{n+1} - \frac{1}{2} h, y_{n+1} - \frac{1}{2} hS_1, z_{n+1} - \frac{1}{2} hS_1), \]

\[ m_3 = F(x_{n+1} - \frac{1}{2} h, y_{n+1} - \frac{1}{2} hS_2, z_{n+1} - \frac{1}{2} hS_2), \]

\[ S_3 = G(x_{n+1} - \frac{1}{2} h, y_{n+1} - \frac{1}{2} hS_2, z_{n+1} - \frac{1}{2} hS_2), \]

\[ m_4 = F(x_{n+1} - h, y_{n+1} - hS_2, z_{n+1} - hS_2), \]

\[ S_4 = G(x_{n+1} - h, y_{n+1} - hS_2, z_{n+1} - hS_2), \]
We can modify the algorithms (23) and (24) to parallel predictor-corrector RK (PPCRK) version as follows:

1) We compute $y_1$ and $z_1$ by using (13) in advance, then
2) We compute the following in parallel:

$$y_{n+1}^p = y_n + \frac{h}{6}(m_1 + 2m_2 + 2m_3 + m_4),$$
$$z_{n+1}^p = z_n + \frac{h}{6}(S_1 + 2S_2 + 2S_3 + S_4),$$

$$y_{n+1}^c = y_n + \frac{h}{6}(m_1 + 2m_2 + 2m_3 + m_4),$$
$$z_{n+1}^c = z_n + \frac{h}{6}(S_1 + 2S_2 + 2S_3 + S_4),$$

Where $p$ means using predicted values and $c$ denotes the using of corrected values.

**Fig. (5): Information flow in the PPCRK scheme (25).**

Processor $y^p$ compute values of $y_{n+2}^p$, $n=0,1,\ldots$
Processor $z^p$ compute values of $z_{n+2}^p$, $n=0,1,\ldots$
Processor $y^c$ compute values of $y_{n+2}^c$, $n=0,1,\ldots$
Processor $z^c$ compute values of $z_{n+2}^c$, $n=0,1,\ldots$
The message passing between the processors will be as follows:
Processor $y^p$ computes values of $y^{p}_{n+2}$, where the values of $y^{p}_{n+1}$ and $z^{p}_{n+1}$ are ready at processor $y^p$ from last step, at the same time processor $z^p$ computes the value of $z^{p}_{n+2}$ as the values of $y^{p}_{n+1}$ and $z^{p}_{n+1}$ are ready from last step. At the same time, processor $y^c$ computes the value of $y^{c}_{n+1}$, as the values of $y^{c}_{n}$, $z^{c}_{n}$, $y^{p}_{n+1}$ and $z^{p}_{n+1}$ ready from last step. At the same time, processor $z^c$ computes values of $z^{c}_{n+1}$ as the values of $y^{c}_{n}$, $z^{c}_{n}$, $y^{p}_{n+1}$ and $y^{p}_{n+1}$ are known from last step, now,

1- Processor $y^p$ sends the values of $y^{p}_{n+2}$ to the processors $z^p$, $y^c$ and $z^c$ and they wait until the message will be received.

2- Processor $z^p$ sends the values of $z^{p}_{n+2}$ to the processors $y^p$, $y^c$ and $z^c$ and they wait until the message will be received.

3- Processor $y^c$ sends the values of $y^{c}_{n+1}$ to the other processors and the process will wait for the receiving the value.

4- Processor $z^c$ sends the values of $z^{c}_{n+1}$ to the other processors.

Now all processors $y^p$, $z^p$, $y^c$ and $z^c$ have the necessary values to start the calculation of the next step. Since we are dealing with only stiff ODEs, the predictor mode of the method should depend on the corrector mode, hence the formula can be adapted to the following form:

First stage calculation:

\[
y^{c}_{n+1} = y^{cc}_{n} + \frac{h}{6} \left( m^p_1 + 2m^p_2 + 2m^p_3 + m^p_4 \right)
\]

\[
z^{c}_{n+1} = z^{cc}_{n} + \frac{h}{6} \left( S^p_1 + 2S^p_2 + 2S^p_3 + S^p_4 \right)
\]

Where
\[m_1^p = F(x_{n+1}, y_{n+1}, z_{n+1}^p),
S_1^p = G(x_{n+1}, y_{n+1}, z_{n+1}^p),\]
\[m_2^p = F(x_{n+1} - \frac{h}{2}, y_{n+1}^p - \frac{h}{2} hm_1^p, z_{n+1}^p - \frac{h}{2} S_1^p),\]
\[S_2^p = G(x_{n+1} - \frac{h}{2}, y_{n+1}^p - \frac{h}{2} hm_1^p, z_{n+1}^p - \frac{h}{2} S_1^p),\]
\[m_3^p = F(x_{n+1} - \frac{h}{2}, y_{n+1}^p - \frac{h}{2} hm_2^p, z_{n+1}^p - \frac{h}{2} S_2^p),\]
\[S_3^p = G(x_{n+1} - \frac{h}{2}, y_{n+1}^p - \frac{h}{2} hm_2^p, z_{n+1}^p - \frac{h}{2} S_2^p),\]
\[m_4^p = F(x_{n+1} - h, y_{n+1}^p - hhm_2^p, z_{n+1}^p - hS_2^p),\]
\[S_4^p = G(x_{n+1} - h, y_{n+1}^p - hm_2^p, z_{n+1}^p - hS_2^p),\]

Second stage calculation:
\[y_{n+2}^p = y_{n+1}^p + \frac{h}{6} (K_1^c + 2K_2^c + 2K_3^c + K_4^c)\]
\[z_{n+2}^p = z_{n+1}^p + \frac{h}{6} (L_1^c + 2L_2^c + 2L_3^c + L_4^c)\]
\[y_{n+1}^{cc} = y_{n+1}^{cc} + \frac{h}{6} (m_1^c + 2m_2^c + 2m_3^c + m_4^c)\]
\[z_{n+1}^{cc} = z_{n+1}^{cc} + \frac{h}{6} (S_1^c + 2S_2^c + 2S_3^c + S_4^c)\]

Where
\[K_1^c = F(x_{n+1}, y_{n+1}^c, z_{n+1}^c), \quad L_1^c = G(x_{n+1}, y_{n+1}^c, z_{n+1}^c),\]
\[K_2^c = F(x_{n+1} - \frac{1}{2} h, y_{n+1}^c - \frac{1}{2} hK_1^c, z_{n+1}^c - \frac{1}{2} hL_1^c),\]
\[L_2^c = G(x_{n+1} - \frac{1}{2} h, y_{n+1}^c - \frac{1}{2} hK_1^c, z_{n+1}^c - \frac{1}{2} hL_1^c),\]
\[
K_2^c = F(x_{n+1} - \frac{1}{2}h, y_{n+1}^c - \frac{1}{2}hK_1^c, z_{n+1}^c - \frac{1}{2}hL_1^c),
\]
\[
L_2^c = G(x_{n+1} - \frac{1}{2}h, y_{n+1}^p - \frac{1}{2}hK_1^c, z_{n+1}^c - \frac{1}{2}hL_1^p),
\]
\[
K_3^c = F(x_{n+1} - \frac{1}{2}h, y_{n+1}^c - \frac{1}{2}hK_2^c, z_{n+1}^c - \frac{1}{2}hL_2^c),
\]
\[
L_3^c = G(x_{n+1} - \frac{1}{2}h, y_{n+1}^p - \frac{1}{2}hK_2^c, z_{n+1}^c - \frac{1}{2}hL_2^p),
\]
\[
K_4^c = F(x_{n+1} - h, y_{n+1}^c - hK_2^c, z_{n+1}^c - hL_2^c),
\]
\[
L_4^c = G(x_{n+1} - h, y_{n+1}^p - hK_2^c, z_{n+1}^c - hL_2^p),
\]
\[
m_1^c = F(x_{n+1}, y_{n+1}^c, z_{n+1}^c), S_1^c = G(x_{n+1}, y_{n+1}^c, z_{n+1}^c),
\]
\[
m_2^c = F(x_{n+1} - \frac{h}{2}, y_{n+1}^c - \frac{h}{2}m_1^c, z_{n+1}^c - \frac{h}{2}S_1^c),
\]
\[
S_2^c = G(x_{n+1} - \frac{h}{2}, y_{n+1}^c - \frac{h}{2}m_1^c, z_{n+1}^c - \frac{h}{2}S_1^c),
\]
\[
m_3^c = F(x_{n+1} - \frac{h}{2}, y_{n+1}^c - \frac{h}{2}m_1^c, z_{n+1}^c - \frac{h}{2}S_2^c),
\]
\[
S_3^c = G(x_{n+1} - \frac{h}{2}, y_{n+1}^c - \frac{h}{2}m_1^c, z_{n+1}^c - \frac{h}{2}S_2^c),
\]
\[
m_4^c = F(x_{n+1} - h, y_{n+1}^c - hm_1^c, z_{n+1}^c - hS_2^c),
\]
\[
S_4^c = G(x_{n+1} - h, y_{n+1}^c - hm_1^c, z_{n+1}^c - hS_2^c),
\]

In two processors, we can assign the calculations of first stage as follows: \(y_{n+1}^c\) to first processor and \(z_{n+1}^c\) to second processor, and the calculation of second stage as follows: \(y_{n+2}^c\) and \(z_{n+2}^c\) first processor and \(y_{n+1}^{cc}\) and \(z_{n+1}^{cc}\) to second processor.
Fig. (6): The mapping of the calculations for the two processors is given below:

<table>
<thead>
<tr>
<th>$P_1$</th>
<th>$P_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1\textsuperscript{st} stage calculations:</strong></td>
<td><strong>1\textsuperscript{st} stage calculations:</strong></td>
</tr>
<tr>
<td>Calculates</td>
<td>Calculates</td>
</tr>
<tr>
<td>1. $m_1^p$</td>
<td>1. $S_1^p$</td>
</tr>
<tr>
<td>sends $m_1^p$</td>
<td>receives $m_1^p$</td>
</tr>
<tr>
<td>receive $S_1^p$</td>
<td>sends $S_1^p$</td>
</tr>
<tr>
<td>2. $m_2^p$</td>
<td>2. $S_2^p$</td>
</tr>
<tr>
<td>sends $m_2^p$</td>
<td>receives $m_2^p$</td>
</tr>
<tr>
<td>receive $S_2^p$</td>
<td>sends $S_2^p$</td>
</tr>
<tr>
<td>3. $m_3^p$</td>
<td>3. $S_3^p$</td>
</tr>
<tr>
<td>sends $m_3^p$</td>
<td>receives $m_3^p$</td>
</tr>
<tr>
<td>receive $S_3^p$</td>
<td>sends $S_3^p$</td>
</tr>
<tr>
<td>4. $m_4^p$</td>
<td>4. $S_4^p$</td>
</tr>
<tr>
<td>sends $m_4^p$</td>
<td>receives $m_4^p$</td>
</tr>
<tr>
<td>receive $S_4^p$</td>
<td>sends $S_4^p$</td>
</tr>
<tr>
<td>5. $y_{n+1}^o$</td>
<td>5. $z_{n+1}^o$</td>
</tr>
<tr>
<td>sends $y_{n+1}^o$</td>
<td>receives $y_{n+1}^o$</td>
</tr>
<tr>
<td>receive $z_{n+1}^o$</td>
<td>sends $z_{n+1}^o$</td>
</tr>
</tbody>
</table>

| **2\textsuperscript{nd} stage calculations:** | **2\textsuperscript{nd} stage calculations:** |
| Calculates | Calculates |
| 1. $K_1^c, L_1^c, K_2^c, L_2^c, K_3^c, L_3^c, K_4^c, L_4^c$ | $m_1^o, S_1^o, m_2^o, S_2^o, m_3^o, S_3^o, m_4^o, S_4^o$ |
| 2. $y_{n+2}^p$ and $z_{n+2}^p$ | 2. $y_{n+1}^{o.o}$ and $z_{n+1}^{o.o}$ |
| sends $y_{n+2}^p$ and $z_{n+2}^p$ | sends $y_{n+1}^{o.o}$ and $z_{n+1}^{o.o}$ |
8. PPCGMRK Method of Order Four:

A new fourth order GM-Runge-Kutta method is of the form [81]:

\[ K_1 = f(x_n, y_n) \]
\[ K_2 = f(x_n + c_2h, y_n + c_2hk_1), \]
\[ K_3 = f(x_n + c_3h, y_n + h(a_31k_1 + a_32k_2)), \]
\[ K_4 = f(x_n + c_4h, y_n + h(a_41k_1 + a_42k_2 + a_43k_3)), \]

And

\[ y_{n+1} = y_n + \frac{h}{3} (\sqrt{k_1k_2} + \sqrt{k_2k_3} + \sqrt{k_3k_4}) \]  

(27)

We know that \( c_3 = a_31 + a_32 \) and \( c_4 = a_41 + a_42 + a_43 \), in particular by setting \( a_31 + a_32 = \frac{1}{2} \), \( a_41 + a_42 + a_43 = 1 \) and by comparing the r.h.s. of equation (27) with the Taylor series expansion for \( y(x_{n+1}) \), the following six equation of conditions were obtained:

\[ h^2 f_{2y}y: -192c_2 + 96 = 0 \]  

(28a)

\[ h^3 f_{2y}y: -108 - 48a_{43} - 24c_2^2 - 96c_2a_{32} + 48c_2^2 = 0 \]  

(28b)

\[ h^3 f_{2y}y: 24 - 96c_2^2 = 0 \]  

(28c)

\[ h^4 f_{3y}y: 18 + 12a_{43} + 3c_2 + 24c_2a_{42} - 96c_2a_{32}a_{43} + 6c_2^2 - 48c_2^2a_{32} - 246c_2^3 = 0 \]  

(28d)

\[ h^4 f_{3y}y: 108 - 60a_{43} - 6c_2 - 96c_2a_{32} - 12c_2^2 - 48c_2^2a_{42} - 96c_2^2a_{32} + 48c_2^3 = 0 \]  

(28e)

and

\[ h^4 f_{3y}y: 4 - 32c_2^3 = 0 \]  

(28f)

Now we shall combine equation (28) with the arithmetic mean (AM) formula of the form:

\[ y_{n+1} = y_n + h \left( w_1 k_1 + w_2 k_2 + w_3 k_3 + w_4 k_4 \right) \]  

(29)

where \( w_1 + w_2 + w_3 + w_4 = 1 \), which uses the similar function evaluation \( k_i \) (i = 1, 2, 3, 4) as given by equation (26)

It can be established that the equation of condition for the fourth order terms are given as follows:

\[ h^2 f_{2y}: w_2c_2 + w_3c_3 + w_4c_4 = 1/2 \]  

(30a)
Parallel Implicit Runge-Kutta...

\[ h^3 f^2 y : w_3 c_2 a_{32} + w_4 (a_{42} c_2 + a_{43} c_3) = 1/6 \]  \hspace{1cm} (30b)

\[ h^3 f^2 yy : \frac{w_3 c_2^2}{2} + \frac{w_3 c_3^2}{2} + \frac{w_4 c_4^2}{2} = 1/6 \]  \hspace{1cm} (30c)

\[ h^4 f^3 yyy : \frac{w_2 c_2^3}{6} + \frac{w_3 c_3^3}{6} + \frac{w_4 c_4^3}{6} = 1/24 \]  \hspace{1cm} (30d)

\[ h^4 f^3 y : w_4 a_{43} c_2 a_{32} = 1/24 \]  \hspace{1cm} (30e)

and

\[ h^4 f^2 yy : \]

\[ \frac{w_3 c_2^2 a_{32}}{2} + \frac{w_3 c_2 a_{32} c_3}{2} + \frac{w_4 a_{42} c_2^2}{2} + \frac{w_4 a_{43} c_3^2}{6} \]

\[ + w_4 c_4 (a_{42} c_2 + a_{43} c_3) = 1/6 \]  \hspace{1cm} (30f)

In obtaining equations (28a) – (28f) we have previously set \( a_{31} + a_{32} = \frac{1}{2} \) and \( a_{41} + a_{42} + a_{43} = 1 \), thus we still have one degree of freedom for the nine unknown variables, if we choose equation (30c) to be the ninth equation, then the resulting AM and GM formulae will be of third order when applied to a general problem of the form:

\[ y'(x) = f(x, y(x)) \]  \hspace{1cm} (31)

Thus, by substituting \( c_3 = 1/2 \) and \( c_4 = 1 \) where appropriate, the remaining seven equations of conditions for the AM and GM formulae are given as follows:

AM:

\[ h^2 fyy : 1/2w_2 + 1/2w_3 + w_4 = 1/2 \]  \hspace{1cm} (32a)

\[ h^3 f^2 y : 1/3w_2 a_{32} + w_4 (1/2a_{42} + 1/2a_{43}) = 1/6 \]  \hspace{1cm} (32b)

\[ h^4 f^3 y : 1/2w_4 a_{43} a_{32} = 1/24 \]  \hspace{1cm} (32c)

and

GM:

\[ h^2 fyy : -192 c_2 = 96 \]  \hspace{1cm} (32e)

\[ h^3 f^2 y : 8a_{32} + 4a_{42} + 4a_{43} = 9 \]  \hspace{1cm} (32f)

and

\[ h^4 f^3 y : 2a_{32} - 2a_{42} - 2a_{43} + 8a_{32} a_{43} = 3 \]  \hspace{1cm} (32g)
For the purpose of parallel computation we shall assume that $a_{43}=0$. So these equations are then solved simultaneously, we get:

$$
c_2=1/2, \ a_{31}=-3/4, \ a_{32}=5/4, \ a_{42}=-1/4, \ a_{43}=0, w_1=1/11, w_2=17/33, w_3=10/33, w_4=1/11
$$

Thus the method is given in its final form by:

$$
k_1 = f(x_n, y_n),
k_2 = f(x_n+1/2h, y_n+1/2hk_1),
k_3 = f(x_n+1/2h, y_n+h(-3/4k_1 + 5/4k_2)),
k_4 = f(x_n+h, y_n+h(5/4k_1-1/4k_2)),
$$

(33)

and

$$
y^*_n = y_n + \frac{h}{33}(3k_1 + 17k_2 + 10k_3 + 3k_4)(AM\text{-mode}) \quad (34)
y^{**}_n = y_n + \frac{h}{3}(\sqrt{k_1k_2} + \sqrt{k_2k_3} + \sqrt{k_3k_4})(GM\text{-mode}) \quad (35)
$$

Again $k_3$ and $k_4$ are independent tasks, hence they can be computed in parallel. Anyhow, evaluation of $K_1, K_2$ and (parallel evaluation of $K_3$ and $K_4$) are sequential processes which means that speed-up factor $S_p$ of the solution will remain small ($1 \leq S_p < 2$) we have called these type of methods as semi-parallel explicit methods.

Since we are dealing with stiff differential equations, we need that the integration method to be implicit, so we can convert formula (34) and (35) to implicit forms by backward integration (i.e. using $-h, h > 0$).

Again we see in the implicit forms that $K_3$ and $K_4$ are defined independently, so that they can be evaluated at the same time using two different processors but their parallel evaluation should be done sequentially with evaluation of $K_1$ and $K_2$, which means that the new implicit methods are semi-parallel. We can
produce parallel versions of the methods by arranging them in parallel PCRK forms.

9-Numerical Example:

Consider the second order stiff equation:
\[
\frac{d^2 y}{dx^2} = 1001 \frac{dy}{dx} + 1000y = 0, \quad x \in (0,1)
\]
(40)

The general solution of (40) is
\[
Y(x) = Ae^{-x} + Be^{-1000x}
\]

If we impose the initial conditions \(Y(0) = 1\), \(y'(0) = -1\), the exact solution is:
\[
Y(x) = e^{-x}
\]

We now try to solve (40) with this initial condition using the 4th order Runge-Kutta method. The system can be rewritten as a first-order system[23]
\[
\frac{dy_1}{dx} = y_2, \quad y_1(0) = 1
\]
\[
\frac{dy_2}{dx} = -1001y_2 - 1000y_1, \quad y_2(0) = -1
\]
(41)

Results of applications of the methods of section (2), (4), (6) and (8) for the above problem are given in tables (1), (2), (3) and (4) respectively (h=0.1).
Table (1) Results of method of section (2)

<table>
<thead>
<tr>
<th>X</th>
<th>Corr. Y₁</th>
<th>Corr. Y₂</th>
<th>Exact Y₁</th>
<th>Exact Y₂</th>
<th>Error 1</th>
<th>Error 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.90502</td>
<td>-0.90502</td>
<td>0.90483</td>
<td>-0.90483</td>
<td>1.852×10⁻⁴</td>
<td>-1.852×10⁻⁴</td>
</tr>
<tr>
<td>0.2</td>
<td>0.81904</td>
<td>-0.81904</td>
<td>0.81873</td>
<td>-0.81873</td>
<td>3.147×10⁻⁴</td>
<td>-3.147×10⁻⁴</td>
</tr>
<tr>
<td>0.3</td>
<td>0.74123</td>
<td>-0.74123</td>
<td>0.74087</td>
<td>-0.74087</td>
<td>4.18×10⁻⁴</td>
<td>-4.18×10⁻⁴</td>
</tr>
<tr>
<td>0.4</td>
<td>0.67081</td>
<td>-0.67081</td>
<td>0.67032</td>
<td>-0.67032</td>
<td>4.987×10⁻⁴</td>
<td>-4.987×10⁻⁴</td>
</tr>
<tr>
<td>0.5</td>
<td>0.60709</td>
<td>-0.60709</td>
<td>0.60653</td>
<td>-0.60653</td>
<td>5.602×10⁻⁴</td>
<td>-5.602×10⁻⁴</td>
</tr>
<tr>
<td>0.6</td>
<td>0.54941</td>
<td>-0.54941</td>
<td>0.54881</td>
<td>-0.54881</td>
<td>6.057×10⁻⁴</td>
<td>-6.057×10⁻⁴</td>
</tr>
<tr>
<td>0.7</td>
<td>0.49722</td>
<td>-0.49722</td>
<td>0.49658</td>
<td>-0.49658</td>
<td>6.374×10⁻⁴</td>
<td>-6.374×10⁻⁴</td>
</tr>
<tr>
<td>0.8</td>
<td>0.44998</td>
<td>-0.44998</td>
<td>0.44932</td>
<td>-0.44932</td>
<td>6.575×10⁻⁴</td>
<td>-6.575×10⁻⁴</td>
</tr>
<tr>
<td>0.9</td>
<td>0.40723</td>
<td>-0.40723</td>
<td>0.40659</td>
<td>-0.40659</td>
<td>6.681×10⁻⁴</td>
<td>-6.681×10⁻⁴</td>
</tr>
<tr>
<td>1</td>
<td>0.36855</td>
<td>-0.36855</td>
<td>0.36787</td>
<td>-0.36787</td>
<td>6.708×10⁻⁴</td>
<td>-6.708×10⁻⁴</td>
</tr>
</tbody>
</table>
TABLE (2), Result of method of section (4)

<table>
<thead>
<tr>
<th>X</th>
<th>Corr.(Y_1)</th>
<th>Corr.(Y_2)</th>
<th>Exact (Y_1)</th>
<th>Exact (Y_2)</th>
<th>Error 1</th>
<th>Error 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.9048387</td>
<td>-0.9048387</td>
<td>0.9048374</td>
<td>-0.9048374</td>
<td>-1.3(\times10^{-6})</td>
<td>1.3(\times10^{-6})</td>
</tr>
<tr>
<td>0.2</td>
<td>0.8187323</td>
<td>-0.8187308</td>
<td>0.8187308</td>
<td>-0.8187308</td>
<td>-1.5(\times10^{-6})</td>
<td>1.5(\times10^{-6})</td>
</tr>
<tr>
<td>0.3</td>
<td>0.7408200</td>
<td>-0.7408182</td>
<td>0.7408182</td>
<td>-0.7408182</td>
<td>-1.8(\times10^{-6})</td>
<td>1.8(\times10^{-6})</td>
</tr>
<tr>
<td>0.4</td>
<td>0.6703219</td>
<td>-0.6703200</td>
<td>0.6703200</td>
<td>-0.6703200</td>
<td>-1.9(\times10^{-6})</td>
<td>1.9(\times10^{-6})</td>
</tr>
<tr>
<td>0.5</td>
<td>0.6065327</td>
<td>-0.6065307</td>
<td>0.6065307</td>
<td>-0.6005307</td>
<td>-2(\times10^{-6})</td>
<td>2(\times10^{-6})</td>
</tr>
<tr>
<td>0.6</td>
<td>0.5488137</td>
<td>-0.5488116</td>
<td>0.5488116</td>
<td>-0.4588116</td>
<td>-2.1(\times10^{-6})</td>
<td>2.1(\times10^{-6})</td>
</tr>
<tr>
<td>0.7</td>
<td>0.4965874</td>
<td>-0.4965853</td>
<td>0.4965853</td>
<td>-0.4965853</td>
<td>-2.1(\times10^{-6})</td>
<td>2.1(\times10^{-6})</td>
</tr>
<tr>
<td>0.8</td>
<td>0.4493311</td>
<td>-0.4493290</td>
<td>0.4493290</td>
<td>-0.4493290</td>
<td>-2.1(\times10^{-6})</td>
<td>2.1(\times10^{-6})</td>
</tr>
<tr>
<td>0.9</td>
<td>0.465718</td>
<td>-0.4065697</td>
<td>0.4065697</td>
<td>-0.4065697</td>
<td>-2.1(\times10^{-6})</td>
<td>2.1(\times10^{-6})</td>
</tr>
<tr>
<td>1</td>
<td>0.3678815</td>
<td>-0.3678794</td>
<td>0.3678794</td>
<td>-0.3678794</td>
<td>-2.1(\times10^{-6})</td>
<td>2.1(\times10^{-6})</td>
</tr>
</tbody>
</table>
TABLE (3) Results of method of section (6)

<table>
<thead>
<tr>
<th>X</th>
<th>Corr. Y₁</th>
<th>Corr. Y₂</th>
<th>Exact Y₁</th>
<th>Exact Y₂</th>
<th>Error 1</th>
<th>Error 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.9048258</td>
<td>-0.9048258</td>
<td>0.9048374</td>
<td>-0.9048374</td>
<td>-1.16×10⁻⁵</td>
<td>1.16×10⁻⁵</td>
</tr>
<tr>
<td>0.2</td>
<td>0.8187165</td>
<td>-0.8187165</td>
<td>0.8187308</td>
<td>-0.8187308</td>
<td>-1.43×10⁻⁵</td>
<td>1.43×10⁻⁵</td>
</tr>
<tr>
<td>0.3</td>
<td>0.7408020</td>
<td>-0.7408020</td>
<td>0.74087182</td>
<td>-0.7408182</td>
<td>-1.62×10⁻⁵</td>
<td>1.62×10⁻⁵</td>
</tr>
<tr>
<td>0.4</td>
<td>0.6703023</td>
<td>-0.6703023</td>
<td>0.6703200</td>
<td>-0.6703200</td>
<td>-1.77×10⁻⁵</td>
<td>1.77×10⁻⁵</td>
</tr>
<tr>
<td>0.5</td>
<td>0.6065119</td>
<td>-0.6065119</td>
<td>0.6065307</td>
<td>-0.6065307</td>
<td>-1.88×10⁻⁵</td>
<td>1.88×10⁻⁵</td>
</tr>
<tr>
<td>0.6</td>
<td>0.5487922</td>
<td>-0.5487922</td>
<td>0.5488116</td>
<td>-0.5488116</td>
<td>-1.94×10⁻⁵</td>
<td>1.94×10⁻⁵</td>
</tr>
<tr>
<td>0.7</td>
<td>0.4965655</td>
<td>-0.4965655</td>
<td>0.4965853</td>
<td>-0.4965853</td>
<td>-1.98×10⁻⁵</td>
<td>1.98×10⁻⁵</td>
</tr>
<tr>
<td>0.8</td>
<td>0.4493090</td>
<td>-0.4493090</td>
<td>0.4493290</td>
<td>-0.4493290</td>
<td>-2×10⁻⁵</td>
<td>2×10⁻⁵</td>
</tr>
<tr>
<td>0.9</td>
<td>0.4065497</td>
<td>-0.4065497</td>
<td>0.4065697</td>
<td>-0.4065697</td>
<td>-2×10⁻⁵</td>
<td>2×10⁻⁵</td>
</tr>
<tr>
<td>1</td>
<td>0.3678598</td>
<td>-0.3678598</td>
<td>0.3678794</td>
<td>-0.3678794</td>
<td>-1.96×10⁻⁵</td>
<td>1.96×10⁻⁵</td>
</tr>
</tbody>
</table>

Table (4) Results of method of section (8)

<table>
<thead>
<tr>
<th>X</th>
<th>Corr. Y₁</th>
<th>Corr. Y₂</th>
<th>Exact Y₁</th>
<th>Exact Y₂</th>
<th>Error 1</th>
<th>Error 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.9048123</td>
<td>-0.9048123</td>
<td>0.9048374</td>
<td>-0.9048374</td>
<td>-2.51×10⁻⁵</td>
<td>2.51×10⁻⁵</td>
</tr>
<tr>
<td>0.2</td>
<td>0.816941</td>
<td>-0.816941</td>
<td>0.8187308</td>
<td>-0.8187308</td>
<td>-3.67×10⁻⁵</td>
<td>3.67×10⁻⁵</td>
</tr>
<tr>
<td>0.3</td>
<td>0.7407724</td>
<td>-0.7407724</td>
<td>0.74087182</td>
<td>-0.7408182</td>
<td>-4.58×10⁻⁵</td>
<td>4.58×10⁻⁵</td>
</tr>
<tr>
<td>0.4</td>
<td>0.6702671</td>
<td>-0.6702671</td>
<td>0.6703200</td>
<td>-0.6703200</td>
<td>-5.29×10⁻⁵</td>
<td>5.29×10⁻⁵</td>
</tr>
<tr>
<td>0.5</td>
<td>0.6064724</td>
<td>-0.6064724</td>
<td>0.6065307</td>
<td>-0.6065307</td>
<td>-5.83×10⁻⁵</td>
<td>5.83×10⁻⁵</td>
</tr>
<tr>
<td>0.6</td>
<td>0.5487496</td>
<td>-0.5487496</td>
<td>0.5488116</td>
<td>-0.5488116</td>
<td>-6.2×10⁻⁵</td>
<td>6.2×10⁻⁵</td>
</tr>
<tr>
<td>0.7</td>
<td>0.4965207</td>
<td>-0.4965207</td>
<td>0.4965853</td>
<td>-0.4965853</td>
<td>-6.46×10⁻⁵</td>
<td>6.46×10⁻⁵</td>
</tr>
<tr>
<td>0.8</td>
<td>0.4492628</td>
<td>-0.4492628</td>
<td>0.4493290</td>
<td>-0.4493290</td>
<td>-6.62×10⁻⁵</td>
<td>6.62×10⁻⁵</td>
</tr>
<tr>
<td>0.9</td>
<td>0.4065029</td>
<td>-0.4065029</td>
<td>0.4065697</td>
<td>-0.4065697</td>
<td>-6.68×10⁻⁵</td>
<td>6.68×10⁻⁵</td>
</tr>
<tr>
<td>1</td>
<td>0.3678127</td>
<td>-0.3678127</td>
<td>0.3678794</td>
<td>-0.3678794</td>
<td>-6.67×10⁻⁵</td>
<td>6.67×10⁻⁵</td>
</tr>
</tbody>
</table>


104
In each table, the first column gives values of the independent variable \( x \), the second and the third columns give the corresponding corrected values of \( y_1 \) and \( y_2 \) respectively computed by implicit mode of the method being used, the fourth and the fifth columns give the corresponding values \( y_1 \) and \( y_2 \) computed from the exact solution of the problem, the sixth and seventh columns give the corresponding errors of the numerical solutions of \( y_1 \) and \( y_2 \) respectively. The two last columns of each table make clear the effectiveness of the newly developed methods in solving stiff ODEs since we are using fixed large stepsize and the integration method is of fixed order.

11- Conclusion
We have developed several semi-parallel explicit Runge-Kutta method for solving ODEs using AM and GM techniques. Then we converted these methods into SPIRK methods by converting the direction of integration. SPIRK methods are suitable for integrating stiff ODEs. Also we have developed PPCRK methods by advancing the predictor mode one-step, and we have modified the PPCRK methods for a system of stiff ODEs.
REFERENCES


