High-performance computing for the analysis of future power systems

Zhaoran Wang

14 11 2022

Table of Contents

Table of Contents Summaries

OMIB Model

Regarding the OMIB Model as follows:

$$\frac{\partial \delta}{\partial t} = \Omega_b(\omega - \omega_s) \tag{1}$$
$$\frac{\partial \omega}{\partial t} = \frac{1}{2H}(p_m - \frac{ev}{x_{eq}} sin\delta) \tag{2}$$

we substitute all physical parameters by C_k for later intuitive derivation and observation. It becomes:

$$\frac{\partial \delta}{\partial t} = C_1(\omega - C_2) = C_a\omega \tag{3}$$
$$\frac{\partial \omega}{\partial t} = C_3(C_4 - C_5 sin\delta) = C_b sin\delta \tag{4}$$

Forward Euler's Method

$$\frac{\partial \delta}{\partial t} = C_a \omega \tag{5}$$

$$\frac{\partial \omega}{\partial t} = C_b sin\delta \tag{6}$$

after Forward Euler's Method is applied to:

$$(\frac{\partial \delta}{\partial t})_{\omega_{i+1}} = C_a \omega_{i+1} \tag{7}$$

$$\omega_{i+1} = \omega_i + (\frac{\partial \omega_i}{\partial t})\Delta t = \omega_i + C_b sin\delta_i \tag{8}$$

$$(\frac{\partial \omega}{\partial t})_{\delta_{i+1}} = C_b sin\delta_{i+1} \tag{9}$$

$$\delta_{i+1} = \delta_i + (\frac{\partial \delta_i}{\partial t})\Delta t = \delta_i + C_a \omega_i \tag{10}$$

Next point = last point + last point numerical derivative * step size

Backward Euler's Method

$$\frac{\partial \delta}{\partial t} = C_a \omega \tag{11}$$

$$\frac{\partial \omega}{\partial t} = C_b sin\delta \tag{12}$$

after Backward Euler's Method is applied to:

$$(\frac{\partial \delta}{\partial t})_{\omega_{i+1}} = C_a \omega_{i+1} \tag{13}$$

$$\omega_{i+1} = \omega_i + (\frac{\partial \omega_{i+1}}{\partial t})\Delta t = \omega_i + C_b sin\delta_{i+1} \tag{14}$$

$$(\frac{\partial \omega}{\partial t})_{\delta_{i+1}} = C_b sin\delta_{i+1} \tag{15}$$

$$\delta_{i+1} = \delta_i + (\frac{\partial \delta_{i+1}}{\partial t})\Delta t = \delta_i + C_a \omega_{i+1} \tag{16}$$

Next point = last point + next point numerical derivative * step size

Backward Euler's Method

$$\omega_{i+1} = \omega_i + (\frac{\partial \omega_{i+1}}{\partial t})\Delta t = \omega_i + C_b sin\delta_{i+1} \tag{17}$$

$$\delta_{i+1} = \delta_i + (\frac{\partial \delta_{i+1}}{\partial t})\Delta t = \delta_i + C_a\omega_{i+1} \tag{18}$$

$$\omega_{i+1} - C_b sin\delta_{i+1} = \omega_i \tag{19}$$

$$\delta_{i+1} - C_a\omega_{i+1} = \delta_i \tag{20}$$

$$A * x = b \tag{21}$$

where $A = \begin{bmatrix} 1 & -C_b \\ -C_a & 1 \end{bmatrix}$ $x = \begin{bmatrix} \omega_{i+1} \\ sin\delta_{i+1} \end{bmatrix}$ $b = \begin{bmatrix} \omega_i \\ \delta_i \end{bmatrix}$
According to the properties of matrix A, there are lots of numerical methods.

4th order Runge-Kutta Method

$$f_0 = f(x(t)) \tag{22}$$
$$f_1 = f(x(t) + 0.5 * f_0 \Delta t) \tag{23}$$
$$f_2 = f(x(t) + 0.5 * t f_1 \Delta) \tag{24}$$
$$f_3 = f(x(t) + t f_2 \Delta) \tag{25}$$
$$x(t + \Delta t) = x(t) + \frac{(f_0 + 2f_1 + 2f_2 + f_3)\Delta t}{6} \tag{26}$$

The basic RK4 format is above, regarding the RK4 applied to OMIB, please see details in article and code.

Table of Contents Summaries

Personal Understanding of OMIB Simulation

Target 1. To analyze if the ω and δ would diverge or converge with the increasing of time?

Target 2. Example (three-phase fault) in book is that, at some moment we disconnect the faulted transmission line to obverse the target 1? In numerical experiment it is after some seconds we change the value of x_{eq}?

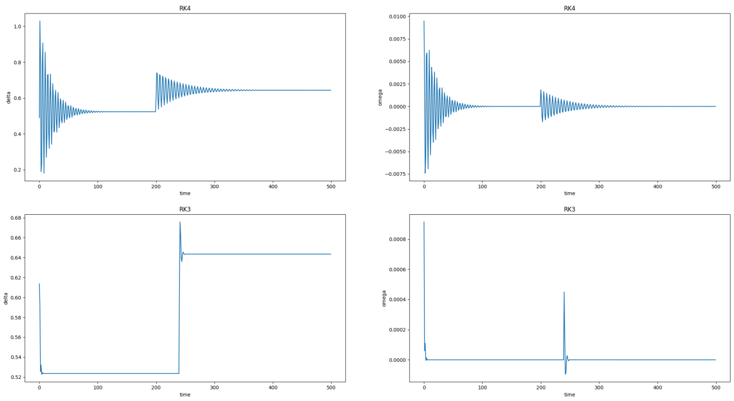
So the interesting and amazing truth is that, when we disconnect it at different moments, the result is different? If we need a converge series, it is really important and necessary to know when we should or should not disconnect?

Jupyter Notebook

Please see the details in jupyter notebook if you are interested in the implementation.

Table of Contents Summaries

Python Implementation



Please see the details in PyCharm if you are interested in the implementation.

Table of Contents Summaries

Future Improvement

1. Performance and Precision Requirement

Parallel Programming (OpenMP, CUDA, ...)

2. Proper and accurate Method Selection

Analyze the properties of model equations and related matrices, utilize or develop the best suitable method.

3. Monte Carlo Numerical Method

Especially apply to high dimension (dimension independence method) and recursive problem (e.g. ray-tracing rendering equation).