УДК 519.683.8

РАСЧЕТ УСТАНОВИВШЕГОСЯ РЕЖИМА ЭЛЕКТРИЧЕСКОЙ СЕТИ СРЕДСТВАМИ GPU

Ф.Н. ЯСИНСКИЙ, Н.Б. ИЛЬИЧЕВ, А.И. КУЛЕШОВ, Д.П ХАРИТОНОВ
ФГБОУВПО «Ивановский государственный энергетический университет им. В.И. Ленина»,
Иваново, Российская Федерация
E-mail: kharitonov.dennis@gmail.com

Авторское резюме

Состояние вопроса: На данный момент задача получения установившегося режима является решенной и результат получается за конечное время. Однако при больших объемах сети и массовых расчетах установившегося режима необходимо сократить время вычисления.
Материалы и методы: Для расчета установившегося режима используется метод узловых потенциалов. При решении системы нелинейных уравнений методом Ньютона выполняется линеаризация уравнений посредством определения матрицы Якоби и решается система линеаризованных уравнений при помощи GPU.
Результаты: Рассматривается алгоритм расчета установившегося режима электросети. Описывается общая структура видеокарты. Приводится анализ распараллеливания приведенного алгоритма.
Выводы: Показано, что можно получить ускорение расчета установившегося режима электросети.

Ключевые слова: установившийся режим, GPU, параллельный алгоритм.

CALCULATION OF STEADY MODE OF ELECTRICAL NETWORK WITH MEANS OF GPU

F.N. YASINSKIY, N.B. IL'ICHEV, A.I. KULESHOV, D.P. KHARITONOV
Ivanovo State Power Engineering University, Ivanovo, Russian Federation
E-mail: kharitonov.dennis@gmail.com

Abstract

Background: At present the problem of getting the steady mode is solved, and the result can be received for the short period of time. However, for big volumes of networks and mass calculation of steady mode it is necessary to decrease the calculation time.
Materials and methods: The nodal-voltage method is used for calculation of steady mode. The authors carry out the linearization of equations by means of finding the matrix of Jacobi while solving the system of nonlineal equations with the Newton method. The linearization of equations helps to solve linearized equations with GPU.
Results: The article considers the calculation algorithm of steady mode of electrical network. The description of the graphical card general structure is given. The analysis of the given algorithm paralleling is carried out.
Conclusions: It is shown that it is possible to get calculation acceleration of steady mode of electrical network.

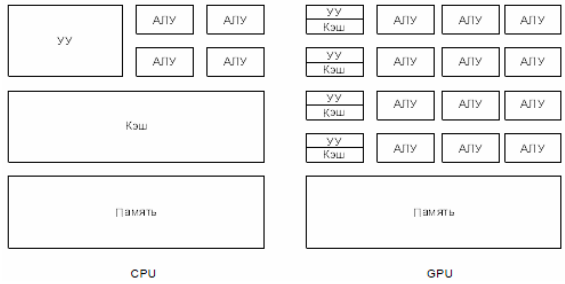
Key words: steady mode, GPU, parallel algorithm.

Расчеты установившихся режимов в электроэнергетических системах являются наиболее массовыми расчетами. Усложнение расчетных моделей, связанное с более детальным представлением сети, значительно увеличивает размерность задачи и затраты времени на получение решения. В то же время современная вычислительная техника имеет неиспользованный в настоящее время потенциал производительности вычислительных систем, связанный с применением многоядерных процессоров и наличием матричных процессоров в составе видеокарт. То, что данный ресурс пока не используется, обусловлено тем, что распараллеливание процесса расчета установившегося режима связано с разработкой специальных вычислительных алгоритмов. Известно, что расчет установившегося режима, как правило, состоит в решении системы нелинейных уравнений узловых напряжений, которые могут быть записаны в форме баланса мощностей или в форме баланса токов. При этом матрица коэффициентов левой части уравнения является слабозаполненной. Таким образом, речь идет о распараллеливании процесса решения системы нелинейных уравнений.

Архитектура GPU и особенности работы с устройством. Для расчета установившегося режима (УР) в качестве входных данных используется матрица собственных и взаимных проводимостей. Эта матрица является разреженной (слабозаполненной), что накладывает определенные ограничения на выбор математических методов, применяемых для расчета УР. При составлении матрицы собственных и взаимных проводимостей учитываются различные типы ветвей: трансформаторы, линии, генераторы и пр.

Для начала рассмотрим оборудование, с которым придется работать, учтем особенности его работы, которые могут пригодиться нам для эффективного расчета УР.

Тенденция последних лет к увеличению числа полупроводниковых элементов на устройстве, которые можно было бы использовать в вычислениях, привела к созданию GPGPU (General-Purpose Graphics Processing Units) – техники использования графического процессора видеокарты для выполнения расчётов в приложениях для общих вычислений, которые обычно проводит центральный процессор. На рис. 1 представлены схематичные изображения архитектуры центрального и графического процессоров, демонстрирующие разницу этих двух устройств.



а) б)

Рис. 1. Схематичное изображение архитектуры процессоров: а – центрального; б – графического

Графический процессор обладает большим количеством ядер, но следует отметить, что ядра GPU предназначены только для вычислений и имеют меньшее количество инструкций по сравнению с CPU. Соответственно, исходя из различной организации архитектур устройств логично предположить, что наличие меньшего числа ядер с бóльшим количеством инструкций рационально для использования параллелизма по задачам (MIMT – Multiple Instructions Multiple Threads), а использование устройства с большим количеством параллельно работающих ядер оптимально для распараллеливания по данным (SIMT – Single Instruction Multiple Threads).

Для понимания особенностей реализации алгоритмов для GPU необходимо иметь представление об архитектуре устройства с точки зрения логической компоновки вычислительных элементов. На рис. 2 представлена архитектура GPU в терминах технологии CUDA (альтернатива CUDA – OpenCL – имеет очень похожую архитектуру, только используется другая терминология, но сути программирования под GPU с использованием OpenCL это не меняет).

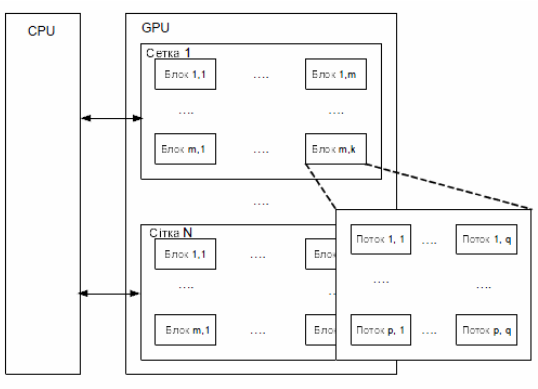


Рис. 2. Архитектура GPU CUDA

Также стоит отметить наличие различных типов памяти в GPU-устройстве (рис. 3).

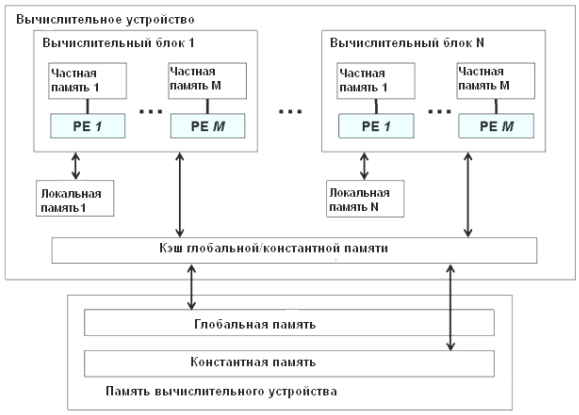


Рис. 3. Модель памяти GPU

Различные типы памяти имеют различные скорости доступа. Так, например, самая медленная память – глобальная (global), далее идут локальная (local) и частная (private). Но и объемы у различных типов памяти разные. Так, глобальная память – это вся оперативная память компьютера, как правило, достигает объема в несколько гигабайт. А локальная и частная память намного меньше – как правило, несколько десятков килобайт.

Алгоритм расчета УР. Теперь рассмотрим основные этапы расчета установившегося режима:

1) получение матрицы узловых проводимостей на вход (процесс формирования этой матрицы можно найти в [1]);

2) задание начальных приближений напряжений узлов (искомых переменных) (в качестве первого приближения берутся номинальные напряжения узлов, которые в дальнейшем корректируются на последующих итерациях выбранного математического метода);

3) определение величины небалансов мощностей;

4) формирование матрицы Якоби;

5) решение системы линейных уравнений;

6) определение новых значений напряжений и фазовых углов напряжений;

7) проверка условий окончания расчета и возврат на п.3, если это необходимо.

Остановимся поподробнее на некоторых из перечисленных выше пунктах.

Так, уравнения небалансов мощностей имеют следующий вид:

$$\varphi_i(U,\delta) = U_i^2 g_{ii} + \sum_{j=1}^{m} U_i U_j \times \times\left[g_{ij}\cos(\delta_i - \delta_j) + b_{ij}\sin(\delta_i - \delta_j) \right] - P_i = 0; \quad (1)$$

$$\psi_i(U,\delta) = -U_i^2 b_{ii} + \sum_{j=1}^{m} U_i U_j \times \times\left[g_{ij}\sin(\delta_i - \delta_j) - b_{ij}\cos(\delta_i - \delta_j) \right] - Q_i = 0, \quad (2)$$

где k – число узлов в сети без учета балансирующего узла и узлов с заданным модулем напряжения.

Формирование матрицы Якоби производится по следующим формулам:

$$\frac{\partial P_i}{\partial \delta_i} = \sum_{j=1}^{m} U_i U_j \left[b_{ij}\cos(\delta_i - \delta_j) - g_{ij}\sin(\delta_i - \delta_j) \right]; \quad (3)$$

$$\frac{\partial P_i}{\partial \delta_j} = -U_i U_j \left[b_{ij}\cos(\delta_i - \delta_j) - g_{ij}\sin(\delta_i - \delta_j) \right]; \quad (4)$$

$$\frac{\partial P_i}{\partial U_i} = 2U_i g_{ii} + \sum_{j=1}^{m} U_j \left[g_{ij}\cos(\delta_i - \delta_j) + b_{ij}\sin(\delta_i - \delta_j) \right]; \quad (5)$$

$$\frac{\partial P_i}{\partial U_j} = U_i \left[g_{ij}\cos(\delta_i - \delta_j) + b_{ij}\sin(\delta_i - \delta_j) \right]; \quad (6)$$

$$\frac{\partial Q_i}{\partial \delta_i} = \sum_{j=1}^{m} U_i U_j \left[g_{ij}\cos(\delta_i - \delta_j) + b_{ij}\sin(\delta_i - \delta_j) \right]; \quad (7)$$

$$\frac{\partial Q_i}{\partial \delta_j} = -\sum_{j=1}^{m} U_i U_j \left[g_{ij}\cos(\delta_i - \delta_j) + b_{ij}\sin(\delta_i - \delta_j) \right]; \quad (8)$$

$$\frac{\partial Q_i}{\partial U_i} = 2U_i b_{ii} - \sum_{j=1}^{m} U_j \left[b_{ij}\cos(\delta_i - \delta_j) - g_{ij}\sin(\delta_i - \delta_j) \right]; \quad (9)$$

$$\frac{\partial Q_i}{\partial U_j} = -\sum_{j=1}^{m} U_j \left[b_{ij}\cos(\delta_i - \delta_j) - g_{ij}\sin(\delta_i - \delta_j) \right]. \quad (10)$$

Объединяя элементы вида $\frac{\partial P_i}{\partial \delta_i}$; $\frac{\partial P_i}{\partial \delta_j}$ в матрицу $\frac{\partial P}{\partial \delta}$, элементы вида $\frac{\partial P_i}{\partial U_i}$; $\frac{\partial P_i}{\partial U_j}$ в матрицу $\frac{\partial P}{\partial U}$ и аналогичным образом формируя матрицы $\frac{\partial Q}{\partial \delta}$ и $\frac{\partial Q}{\partial U}$, уравнение баланса мощностей в блочно-матричной форме будет выглядеть следующим образом:

$$\left(\frac{\partial P}{\partial \delta}\right)\Delta\delta^k + \left(\frac{\partial P}{\partial U}\right)\Delta U^k = -\Delta P^k - 1; \quad (11)$$

$$\left(\frac{\partial Q}{\partial \delta}\right)\Delta\delta^k + \left(\frac{\partial Q}{\partial U}\right)\Delta U^k = -\Delta Q^{k-1}. \quad (12)$$

Следующим шагом является решение системы линейных уравнений, полученных в результате выполнения предыдущего шага. Решив полученную систему уравнений, получаем новые значения приращения напряжения ΔU и фазовых углов $\Delta\delta$. После этого можно получить очередное приближение этих двух параметров, воспользовавшись следующими формулами:

$$U_i^k = U_i^{k-1} + \Delta U_i^{k-1};$$

$$\delta_i^k = \delta_i^{k-1} + \Delta\delta_i^{k-1}.$$

Получив новые значения, проверяем на сходимость (проверка на сходимость включает в себя больше, чем просто проверку на достижение значений небаланса желаемой точности. Более подробно этот шаг описан в [1, с. 70]) и в случае, если желаемая точность не достигнута, переходим к следующей итерации, повторив вышеупомянутые действия, начиная с шага пересчета небаланса мощностей.

На рис. 4 приводится схема сети, для которой выполняется расчет установившегося режима. Подробное описание формирования исходных данных для этой сети и пошаговый алгоритм расчета для однопроцессорного варианта можно найти в [1].

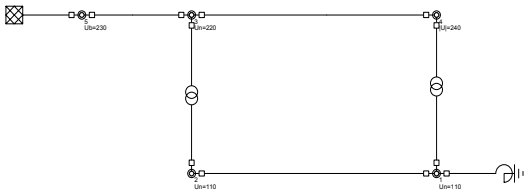


Рис. 4. Примерная схема для расчета установившегося режима

Распараллеливание алгоритма расчета УР. Выше был приведен алгоритм выполнения расчета УР электроэнергетической системы, представляющий собой алгоритм решения системы нелинейных уравнений. Согласно этому алгоритму, линеаризация уравнений выполняется посредством получения матрицы Якоби. Таким образом, для решения системы нелинейных уравнений необходимо решить систему линейных уравнений.

В качестве алгоритма решения системы нелинейных уравнений будем использовать метод Ньютона [7, 8].

Ниже рассмотрим, как можно распараллелить расчет матрицы Якоби и подробно остановимся на решении системы линейных уравнений.

Согласно формулам (3)–(10), можно выделить 8 значений, которые необходимо получить. Причем каждое из этих значений, во-первых, не зависит от других значений, во-вторых, если оперировать матричной терминологией, не зависит от значений по столбцам, «накапливая» данные по строкам. Таким образом, само собой напрашивается распараллеливание данных по строкам, предоставив каж-

дому процессу (нити, потоку – в различных русскоязычных источниках используется разная терминология для термина thread) считать свою строку, разграничивая пересчет диагональных и недиагональных элементов. То есть каждый процесс будет считать одну строку, во время которой будет пересчитано 8 значений. Посчитаем теоретическое ускорение, которое можно получить на данном этапе. Для простоты вычислений примем, что не используются структуры данных для хранения разреженной матрицы (т.е. хранятся и нулевые элементы). Тогда, если в строке содержится N значений, то пересчет одной строки матрицы Якоби составит $8 \cdot N$. А пересчет всей матрицы Якоби при однопроцессорной обработке составит $8 \cdot N \cdot N$. При использовании N процессов получаем, что каждый процесс считает $8 \cdot N$ значений. Теоретически получаемое ускорение по количеству выполняемых операций будет рассчитываться как

$$\frac{8 \cdot N^2}{8 \cdot N} = N.$$

Это довольно грубый расчет, не учитывающий время на передачу между различными типами памяти (см. рис. 3.) и различное время обращения к памяти на CPU и GPU (в зависимости от типа памяти). Целью подсчета ускорения по операциям было показать наличие ускорения как такового на данном этапе.

Однако вся выгода от ускорения на этапе получения матрицы Якоби может быть «съедена» неправильным выбором метода расчета системы линейных уравнений и частыми передачами данных с ядра видеокарты в оперативную память (об этом будет сказано ниже).

Ключевым моментом, таким образом, становится выбор расчета системы линейных уравнений. Существует множество способов решения системы линейных уравнений, которые можно подразделить на 2 больших класса – прямые и итерационные математические методы. К прямым методам относятся:

- метод Гаусса;
- метод Жордана-Гаусса;
- метод Крамера;
- разложение Холецкого;
- метод Томаса.

К итерационным методам относятся:

- метод Якоби (метод простой итерации);
- метод Гаусса – Зейделя;
- метод релаксации;
- многосеточный метод;
- метод Монтанте;
- метод Абрамова;
- метод обобщенных минимальных невязок;
- метод бисопряженных градиентов;
- стабилизированный метод бисопряженных градиентов;
- квадратичный метод сопряженных градиентов;
- метод квази-минимальных невязок.

Обоснование использования того или иного метода представляет собой отдельную задачу, на решении которой мы останавливаться не будем. Описание некоторых этих методов можно найти в [3, 4, 5].

На данный момент в системе EnergyCS (работа выполняется для модификации расчетного блока программного комплекса) используется метод Гаусса, дающий точное решение за N шагов. Однако реализация этого метода на видеокарте является нетривиальной задачей, так как в процессе решения системы данным методом матрица коэффициентов раскладывается на верхнюю треугольную и нижнюю треугольную матрицы (LU-разложение, подробное описание этого метода см. в [4, 6]). При LU-разложении имеет место зависимость по данным, то есть расчет следующего значения разложенной матрицы невозможен до тех пор, пока не будут посчитаны предыдущие значения. Следовательно, толку от распараллеливания разложения не будет, так как каждый процесс будет ждать пересчета других значений, превратив параллельный расчет LU-разложения в последовательный. Одним из вариантов выхода из данной ситуации является использование особого алгоритма расчета LU-разложения, позволяющего параллельно проводить эту операцию. Информация о параллельном LU-разложении представлена в [6].

Другим вариантом выхода из этой ситуации является использование итерационного метода. Однако тут необходимо учитывать следующее. При использовании разреженных матриц используются особые структуры для их хранения, что ведет к тому, что, например, операция транспонирования матрицы представляет собой нетривиальную и довольно трудоемкую задачу. Другой момент – некоторые итерационные математические методы требуют предварительного расчета собственных чисел матрицы, что ведет к дополнительным затратам машинного времени. И наконец, итерационные методы могут быть нестабильны. Стабилизация того или иного метода также является большой областью исследований в численных методах и зачастую носит очень конкретный характер, варьируясь от задачи к задаче.

Ниже приведем два алгоритма итерационных методов: метода простых итераций (как самого простого и понятного) и метода бисопряженных градиентов стабилизированного (BiCGStab – метод, по которому в русскоязычной литературе мало информации). Недостатком этих методов является то, что они предназначены лишь для положительно определенных матриц и не всегда сходятся.

Основная формула для метода простых итераций (метода Якоби):

$$x_i = b_i - \sum_{j=1}^{i-1} \alpha_{ij} x_j - \sum_{j=i+1}^{m} \alpha_{ij} x_j, \qquad (13)$$

где $b_i = \frac{f_i}{a_{ii}}$; $\alpha_{ij} = \frac{a_{ij}}{a_{ii}}$.

Согласно (13), данный метод очень хорошо подходит под парадигму SIMT. Каждый процесс считает свою строку, собирая в самом конце невязку в общую переменную. Процесс идет до тех пор, пока отношение суммарной невязки к количеству задействованных процессов не достигнет необходимой точности.

Данный метод прост в реализации, прост в понимании, по нему довольно много информации в русскоязычной литературе, однако он довольно требователен к исходным данным и долго сходится.

Алгоритм метода BiCGStab следующий:

$k = -1,$
выбрать x_0 и $\tilde{r}_0,$
посчитать $r_0 = b - Ax_0,$
$u_{-1} = \tilde{u}_{-1} = 0, \quad \rho_{-1} = 1,$
повторять пока $\|r_{k+1}\|$ не достаточно мало
$k := k + 1,$
$\rho_k = (r_k, \tilde{r}_k), \quad \beta_k = -\rho_k / \rho_{k-1},$
$u_k = r_k - \beta_k u_{k-1}, \quad c_k = Au_k,$
$\tilde{u}_k = \tilde{r}_k - \beta_k \tilde{u}_{k-1},$
$\gamma_k = (c_k, \tilde{r}_k), \quad \alpha_k = \rho_k / \gamma_k,$
$x_{k+1} = x_k + \alpha_k u_k, \quad r_{k+1} = r_k - \alpha_k c_k, \quad \tilde{r}_{k+1} = \tilde{r}_k - \alpha_k A^T \tilde{u}_k$

Данный метод, являющийся модификацией метода сопряженных градиентов (CG – Conjugate Gradient), отличается повышенной стабильностью, однако все равно не всегда сходится. Способы повышения стабильности данного метода можно найти в [7]. В основе данного алгоритма лежит одна из базовых векторных операций – *saxpy*:

$saxpy(a, x, y) = a\,x + y,$

где a – число; x, y – векторы.

Распараллеливание данной операции очень просто и не требует особых трудозатрат. Однако, ввиду того что метод BiCGStab требует синхронизации во время расчета для получения промежуточных значений, это замедляет скорость его расчета.

Если сравнивать методы Якоби и BiCGStab, то второй метод сходится за меньшее количество итераций, однако выгода от его использования становится очевидна лишь при больших размерах матрицы, так как при малых размерах матрицы метод Якоби, не требующий синхронизации значений во время выполнения, оказывается быстрее.

Стоит обратить внимание на метод итераций Чебышева [2], не требующий межпроцессорной синхронизации во время выполнения. Однако недостатком данного метода является необходимость знать спектр матрицы (максимальное и минимальное собственные числа) перед началом расчета.

Заключение

Алгоритм расчета установившегося режима электрической сети и предложенные подходы к распараллеливанию этого расчета позволяют сделать вывод, что выбор численного метода остается за пользователем и осуществляется в зависимости от используемых структур для хранения данных и возможностей параллельной обработки данных на используемой машине. Рекомендации, касающиеся выбора математического метода с учетом особенностей работы с вычислительным блоком на видеокарте, даны с учетом того, что выбор видеокарты для расчетов обусловлен наличием большого количества процессоров (технология легких ядер) на борту.

Список литературы

1. **Кулешов А.И., Прахин Б.Я.** Расчет и анализ установившихся режимов электроэнергетических систем на персональных компьютерах: учеб. пособие / Иван. гос. энерг. ун-т. – Иваново, 2001. – 171 с.
2. **Чадов С.Н.** Некоторые вопросы численного моделирования динамических систем / ГОУВПО «Ивановский государственный энергетический университет им. В.И. Ленина». – Иваново, 2010. – 120 с.
3. **Баландин М.Ю., Шурина Э.П**. Методы решения СЛАУ большой размерности. – Новосибирск: Изд-во НГТУ, 2000. – 70 с.
4. **Голуб Дж., Ван Лоун Ч**. Матричные вычисления: пер. с англ. – М.: Мир, 1999. – 548 с.
5. **BiCGstab**(L) for linear equations involving unsymmetric matrices with complex spectrum, gerard l.g. sleijpen, diederik r. fokkema, Electronic Transactions on Numerical Analysis. – September 1993. – Vol. 1. – P. 11–32.
6. **Семушин И.В.** Численные методы алгебры: учебное пособие для вузов. –Ульяновск: УлГУ, 2008. 178 с.
7. **Radolph E. Bank, Tony F. Chan**. An analysis of the composite step biconjugate gradient method // Numerische mathematic. – Vol. 66. – № 1. – P. 295–319.
8. http://ru.wikipedia.org/wiki/Метод_Ньютона
9. http://en.wikipedia.org/wiki/Newton's_method

References

1. Kuleshov, A.I., Prakhin, B.Ya. *Raschet i analiz ustanovivshikhsya rezhimov elektroenergeticheskikh sistem na personal'nykh komp'yuterakh* [Calculation and Analysis of Steady Modes of Electric Power Systems with using of PCs], Ivanovo, 2001, 171 p.
2. Chadov, S.N. *Nekotorye voprosy chislennogo modelirovaniya dinamicheskikh sistem* [Specific Issues of Numerical Simulation in Dynamic Systems], Ivanovo, 2010, 120 p.
3. Balandin, M.Yu., Shurina, E.P. *Metody resheniya SLAU bol'shoy razmernosti* [Solving Methods of SLAU with great Dimention], Novosibirsk: Izd-vo NGTU, 2000, 70 p.
4. Golub, J., Van Loan, Ch. *Matrichnye vychisleniya* [Matrix Computations], Moscow: Mir, 1999, 548 p.
5. BiCGstab(L) for linear equations involving unsymmetric matrices with complex spectrum, gerard l.g. sleijpen, diederik r. fokkema, Electronic Transactions on Numerical Analysis, vol. 1, pp. 11–32, September 1993.
6. Semushin, I.V. *Chislennye metody algebry* [Computational Algebra Methods], Ul'yanovsk: UlGU, 2008, 178 p.
7. Radolph, E. Bank, Tony, F. Chan. An analysis of the composite step biconjugate gradient method. Numerische mathematic, vol. 66, 1, pp. 295–319.
8. http://ru.wikipedia.org/wiki/Метод_Ньютона
9. http://en.wikipedia.org/wiki/Newton's_method

Ясинский Федор Николаевич,
ФГБОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
доктор физико-математических наук, профессор кафедры высокопроизводительных вычислительных систем,
телефон (4932) 26-98-29.

Ильичев Николай Борисович,
ФГБОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
кандидат технических наук, доцент кафедры электрических станций, подстанций и диагностики электрооборудования,
e-mail: ilichevnb@rambler.ru

Кулешов Анатолий Иванович,
ФГБОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
кандидат технических наук, доцент кафедры электрических станций, подстанций и диагностики электрооборудования,
e-mail: aikuleshov@yandex.ru

Харитонов Денис Петрович,
ФГБОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
аспирант кафедры высокопроизводительных вычислительных систем,
телефон (4932) 26-98-29,
e-mail: kharitonov.dennis@gmail.com