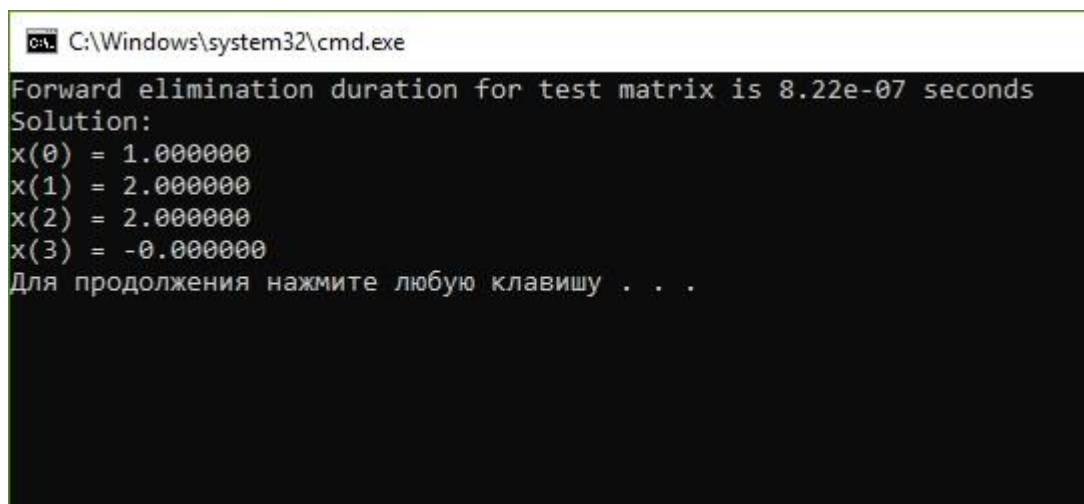


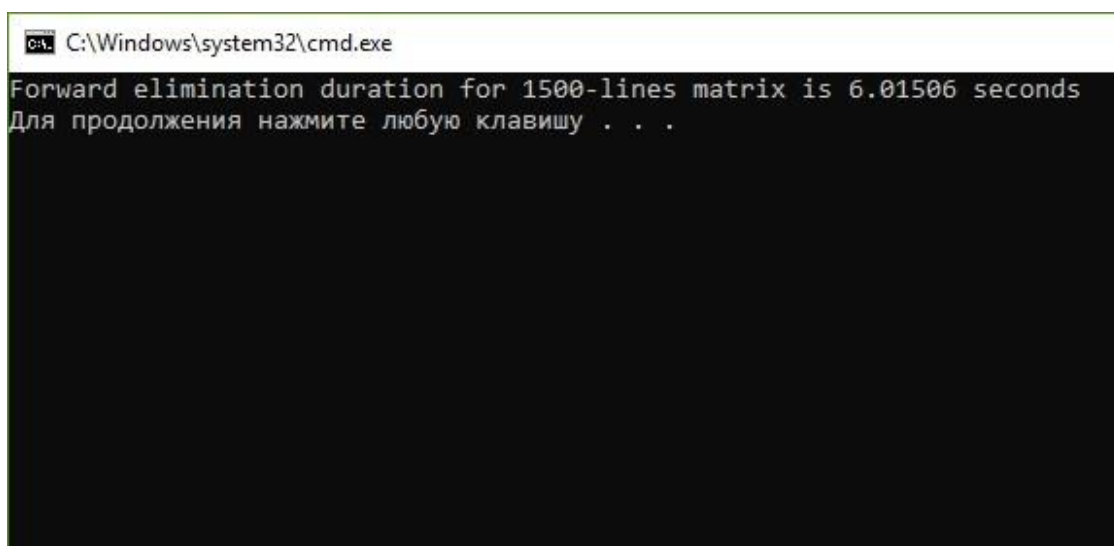
Проанализировав код программы, запустил её первоначальную версию и получил решение для тестовой матрицы *test_matrix*. Добавил строки кода для измерения времени выполнения прямого хода метода Гаусса в функции *SerialGaussMethod()*. Результат представлен на рис. 1. Программный код находится в файле *item2_1.cpp*.



```
C:\Windows\system32\cmd.exe
Forward elimination duration for test matrix is 8.22e-07 seconds
Solution:
x(0) = 1.000000
x(1) = 2.000000
x(2) = 2.000000
x(3) = -0.000000
Для продолжения нажмите любую клавишу . . .
```

Рис. 1 – Вид консоли после выполнения программы с тестовой матрицей

После заполнения матрицы с количеством строк *MATRIX_SIZE* (1500 строк) случайными значениями получил следующий результат выполнения программы (рис. 2). Программный код представлен в файле *item2_2.cpp*.



```
C:\Windows\system32\cmd.exe
Forward elimination duration for 1500-lines matrix is 6.01506 seconds
Для продолжения нажмите любую клавишу . . .
```

Рис. 2 – Вид консоли после выполнения программы решения матрицы, содержащей 1500 строк

Проанализировал последовательный программный код с помощью инструмента *INTEL VTUNE AMPLIFIER 2019*, получил следующие данные (рис. 3–5).

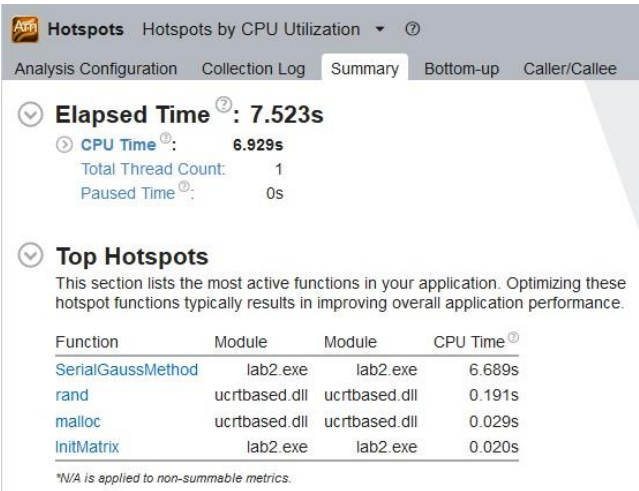


Рис. 3 – Время, потраченное на выполнение последовательного программного кода

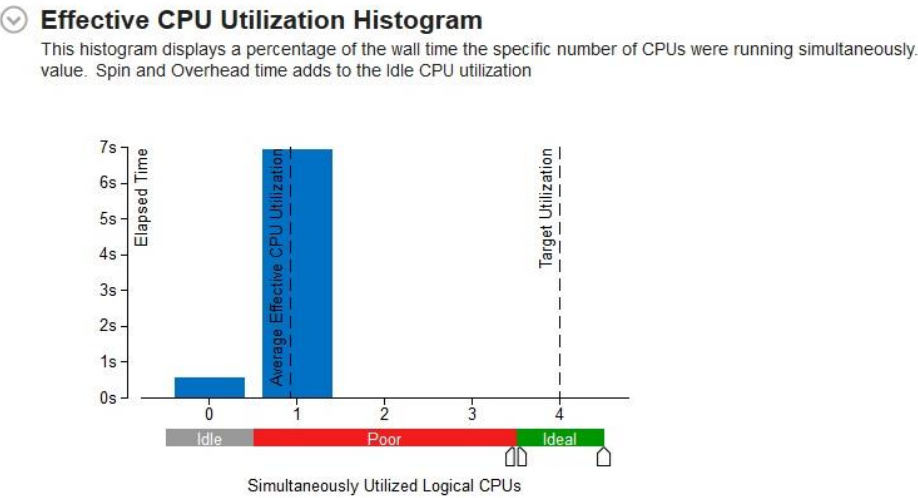


Рис. 4 – Зависимость времени выполнения программы от количества используемых процессов

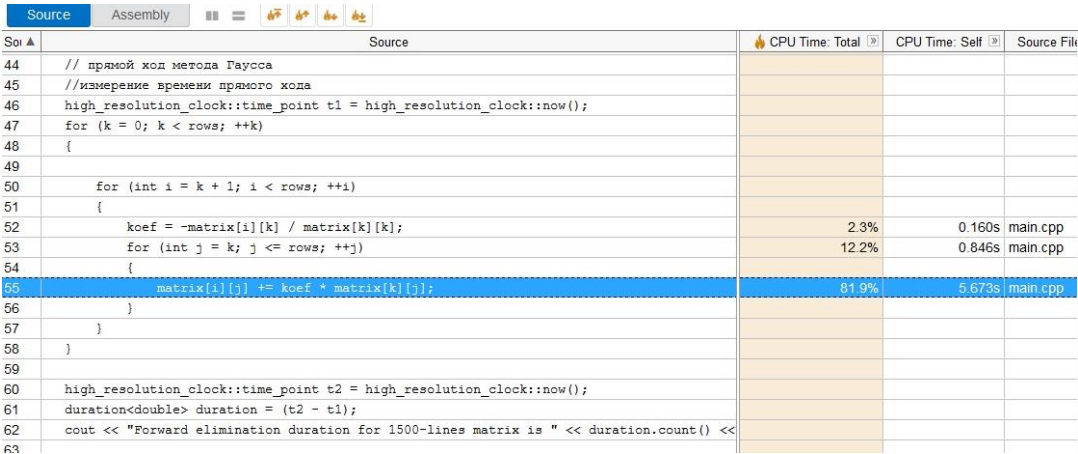


Рис. 5 – Окно с изображением наиболее часто используемых участков кода

На основании полученной информации можно сделать вывод о возможности распараллеливания стандартного цикла *for* с помощью *cilk_for*. Данную конструкцию применил ко второму вложенному циклу прямого хода метода Гаусса,

несмотря на то, что наиболее используемый участок – это цикл выделенный синим на рис. 5. Зависимость по данным ограничивает использование параллелизации в нём.

Программный код с использованием конструкции *cilk_for* представлен в файле *item3.cpp*. Результат выполнения программы приведён на рис. 6.

```

C:\Windows\system32\cmd.exe
Forward elimination duration for 1500-lines matrix is 4.03975 seconds
Для продолжения нажмите любую клавишу . . .
  
```

Рис. 6 – Результат выполнения программы после применения конструкции *cilk_for* в прямом ходе метода Гаусса

Последним этапом в анализе программного кода стало использование инструмента *INTEL INSPECTOR 2019*. С помощью данного инструмента я проанализировал программу на наличие различных ошибок, таких как, например, гонка данных. Результат анализа представлен на рис. 7.

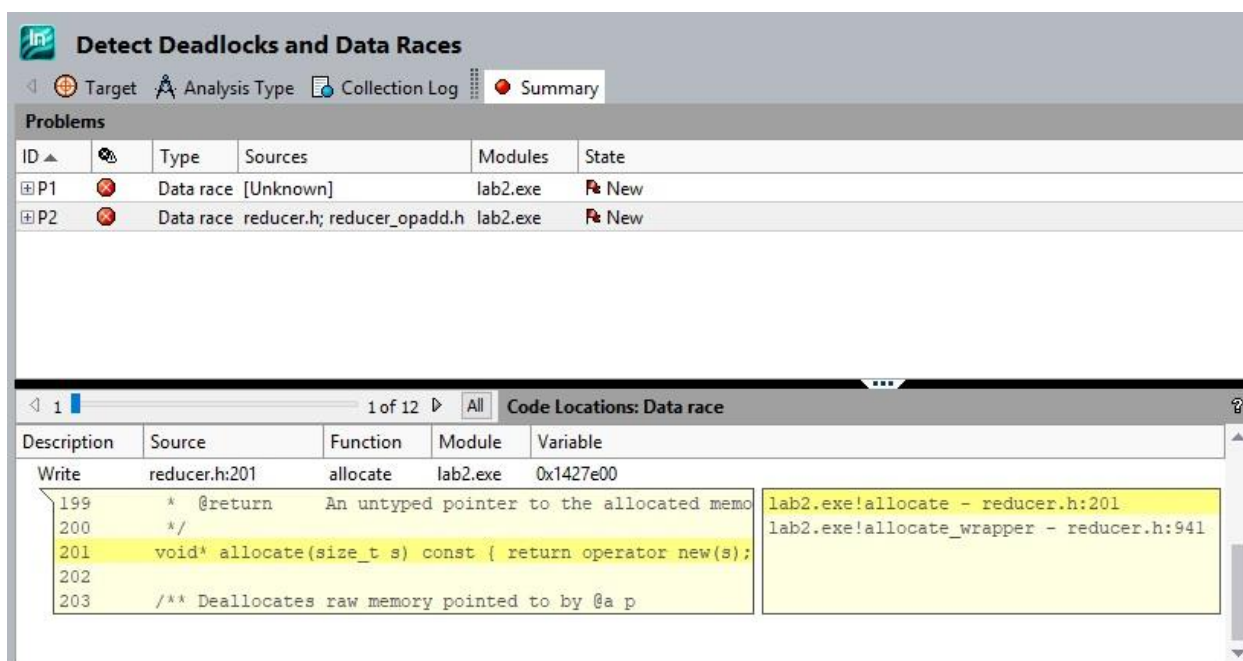


Рис. 7 – Окно ошибок инструментария *INTEL INSPECTOR 2019*

Решением ошибок, связанных с гонкой данных может быть добавление конструкции *reducer_opadd*. После этого ошибок быть не должно (рис. 8).

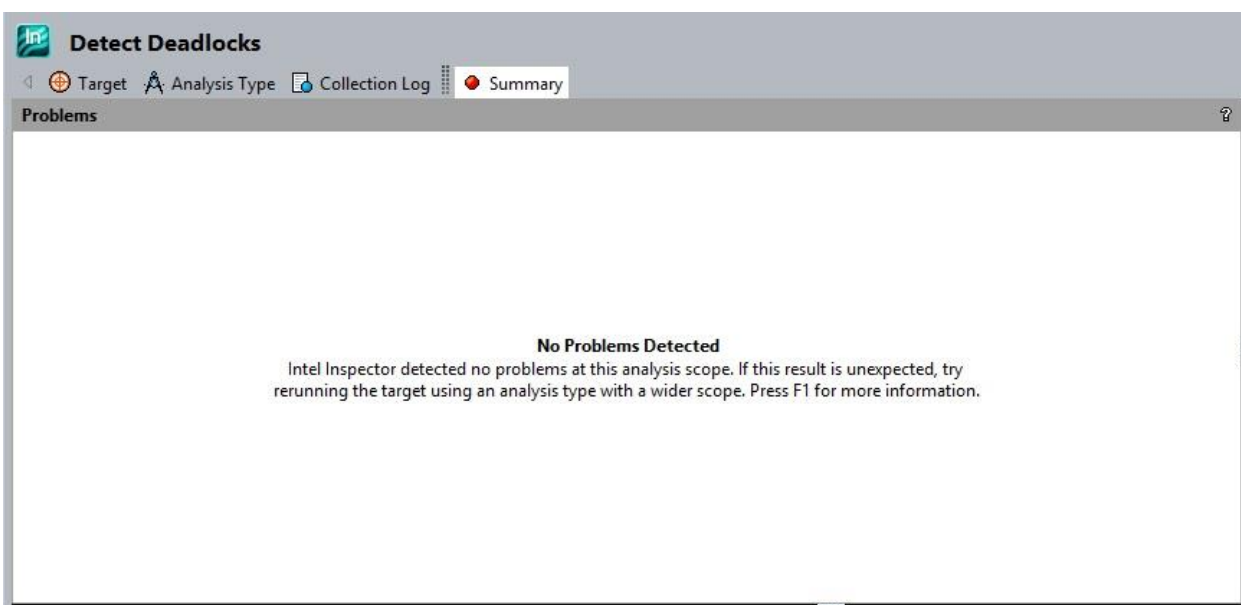


Рис. 8 Окно ошибок инструментария *INTEL INSPECTOR 2019* после добавления конструкции *reducer_opadd*

После учёта всех предложений, сделанных инструментами *INTEL VTUNE AMPLIFIER 2019* и *INTEL INSPECTOR 2019*, я отредактировал программный код (файл *item5.cpp*). Результат сравнения прямого хода метода Гаусса в последовательном и параллельном исполнении представлен на рис. 9.

```
C:\Windows\system32\cmd.exe
Consistent realization
Forward elimination duration for 1500-lines matrix is 2.11125 seconds
Parallel realization
Forward elimination duration for 1500-lines matrix is 2.14754 seconds
Time ratio is 0.983102
Для продолжения нажмите любую клавишу . . .
```

Рис. 9 – Время выполнения последовательного и параллельного участков прямого хода метода Гаусса для матрицы размера 1500 строк, заполненной случайными числами

Полученные данные отличаются от предыдущих версий программы по причине того, что проект запущен в режиме *Release*, и для выполнения цикла *cilk_for* требуется некоторое время для распределения данных по потокам.

Ссылка на GitHub: <https://github.com/alexleo20/lab2>