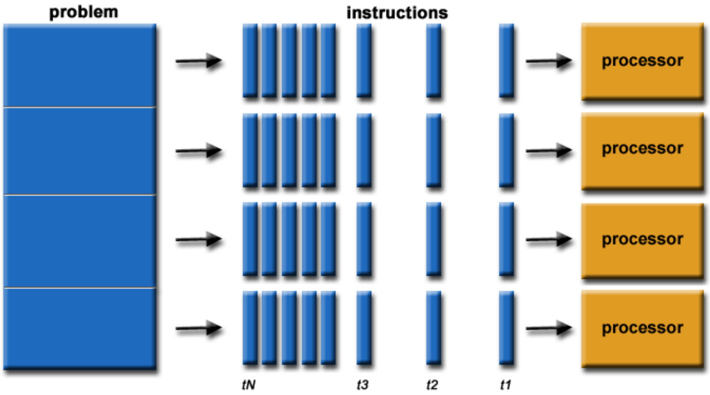
Software within building physics and ground heat storage

HEAT3 version 7

A PC-program for heat transfer in three dimensions

Update manual



June 15, 2015

BLOCON

www.buildingphysics.com

Contents

1. What's new in HEAT3 version 7

Blocon is proud to present a new version of HEAT3. Many new important features have been added. Up-to-date information is given on www.buildingphysics.com.

This update manual covers the new features that have been added since version 6.0. New users should also read the update manuals for version 6, 5, and the full manual for version 4 at

http://www.buildingphysics.com/manuals/HEAT3_6_update.pdf

http://www.buildingphysics.com/manuals/HEAT3 5 update manual.pdf

http://www.buildingphysics.com/manuals/HEAT3_4.pdf

Tips for reading for beginners: For a quick start read Chapter 4 in Manual HEAT3 4.0. The examples in Chapter 5 would also be useful. After this, look at Update manual for HEAT3 5.0 and Update manual for HEAT3 6.0. Also see the examples for the test cases: ISO 10211 validation test cases.

1.1 Parallel computation

HEAT3 now solves larger problems much faster using multiple cores (CPU:s) for both steady-state and transient cases. For large problems there may be up to 4-5 times of speed increase. E.g. a problem that took 1h to solve with version 6 (single core) may take about 15 min to be solved using version 7 (multi core) on a PC with 4 cores.

1.2 Miscellaneous

Many minor improvements are made since version 6.0. The most important are shown below:

- Several instances of HEAT3 can now be started and active at the same time. This makes it possible to work on a new project while another one is simulated.
- Earlier versions were minimized when focus was set to another application in Windows. This version does not have this behavior.
- The number of colors (materials) and legends shown in the post-processor was limited to 15 in old versions. This is now increased to 30.
- Better adaption to Windows Vista, Windows 7 and Windows 8.
- The pre-processor is now unlocked after every simulation (option "Always unlock preprocessor after simulation" in the settings menu).
- New improved license management system.

1.3 Requirements

HEAT3 runs on both 32-bit and 64-bit Windows 8, 7, Vista, and compatibles. Only about 15 MB hard disk space is needed.

HEAT3 needs about 1.7 GB RAM allocated space when all nodes (50 million) are used. A PC with 3 GB RAM is recommended in that case.

Earlier versions had two different exe-files: one with 2.2 million and one with 50 million nodes, respectively. Version 7 has only one exe-file that allows up to 50 million nodes:

Version	Max number of nodes	Required memory RAM	Recommended RAM for PC
HEAT3_v7.exe	50,653,000 (370·370·370)	1.7 GB	3 GB

A good tip is to close down other applications to increase the available memory (if insufficient). If HEAT3 is swapping data to/from the hard disk it is a sign that virtual memory beyond the installed RAM is used. This will happen if you try to run large cases on a PC with too little memory. This can cause very long calculation time since the hard disk will be used for some of the memory for the allocated matrices. It should not be any problem unless the PC has less than 3GB RAM.

1.4 Light version

The light (demo) version has the following restrictions:

- Data cannot be saved.
- Materials cannot be added or changed.
- Max 1,000,000 (100·100·100) nodes.
- Results for heat flows are hidden for larger problems with more than 25,000 nodes and with more than 30 nodes in each direction.
- Thermal coupling coefficients (L2D) will not be calculated.
- Report preview is disabled.
- Some other restrictions apply.

2. HEAT3 goes parallel

2.1 Introduction

HEAT3 now solves large problems much faster using multiple cores for both steady-state and transient cases. There may be up to 4-5 times of speed increase. E.g. a problem that takes 1h to solve with version 6 (single core) may take 15 min to solve using version 7 (multi core) on the same PC. The enhancement in speed is due to parallel calculations.

According to Wikipedia, a parallel algorithm, as opposed to a traditional serial algorithm, is an algorithm which can be executed a piece at a time on many different processing devices, and then combined together again at the end to get the correct result.

Parallel algorithms on individual devices have become more common since the early 2000s because of substantial improvements in multiprocessing systems and the rise of multi-core processors. Up until the end of 2004, single-core processor performance rapidly increased via frequency scaling, and thus it was easier to construct a computer with a single fast core than one with many slower cores with the same throughput, so multicore systems were of more limited use. Since 2004 however, frequency scaling hit a wall, and thus multicore systems have become more widespread, making parallel algorithms of more general use.

Algorithms vary significantly in how parallelizable they are, ranging from easily parallelizable to completely unparallelizable. HEAT3 uses a robust numerical method with explicit forward finite differences. This method makes it quite suitable to split up the geometrical area into parallel portions to be solved quite efficiently using a PC with multi-core processors.

Multi-core processors may have two cores (dual-core CPUs, for example, AMD Phenom II X2 and Intel Core Duo), four cores (quad-core CPUs, for example, AMD Phenom II X4, Intel's i5 and i7 processors), six cores (hexa-core CPUs, for example, AMD Phenom II X6 and Intel Core i7 Extreme Edition 980X), eight cores (octo-core CPUs, for example, Intel Xeon E7-2820 and AMD FX-8350), ten cores (for example, Intel Xeon E7-2850), or more. In fact, the general trend in processor development has moved from dual-, tri-, quad-, hex-, oct-core chips to ones with tens or even thousands of cores.

2.2 Physical cores vs logical cores

The more logical cores available, the quicker the calculations will be by HEAT3 using parallel execution for large cases. There is however a higher limit for this, see https://en.wikipedia.org/wiki/Amdahl's_law

As an example, a calculation might be 3-4 faster to solve using a CPU with 4 cores in parallel mode vs single mode for a certain problem. The same problem might be 2-3 faster to solve using a CPU with 2 cores in parallel mode vs single mode.

A **physical core** is what it sounds like: an actual physical processor core in your CPU. Each physical core has its own circuitry and its own L1 (and usually L2) cache that can read and execute instructions separately (for the most part) from the other physical cores on the chip.

A **logical core** is more of a programming abstraction than an actual physical entity. A simple definition of a logical core is that it is a processing unit that is capable of executing its own thread in parallel with other logical cores. In fact you could say that a logical core is the same as a **thread**.

There can be multiple logical cores per physical core (this is often the case with modern PC:s). However logical cores share resources with other logical cores operating on the same physical core, so having more logical cores will not necessarily get you the same performance increase as having more physical cores.

In the case of intel hyper-threading (HT), you have two logical cores per physical core, so a quad-(physical) core i7 processor will have eight logical cores. However the two logical cores within one physical core cannot truly operate in parallel with respect to each other. This is because HT works by having one logical core operate while the other logical core is waiting and has nothing to do (for example when it is waiting on a cache or memory fetch).

Then how can these logical cores be considered in parallel? Well most of the time they can be because during typical CPU operation you will almost never see continuous execution of a single thread on every clock cycle - there are always gaps when one logical core is waiting for something and the second logical core can kick in and do its job.

2.3 What speed increase to expect using parallel calculations

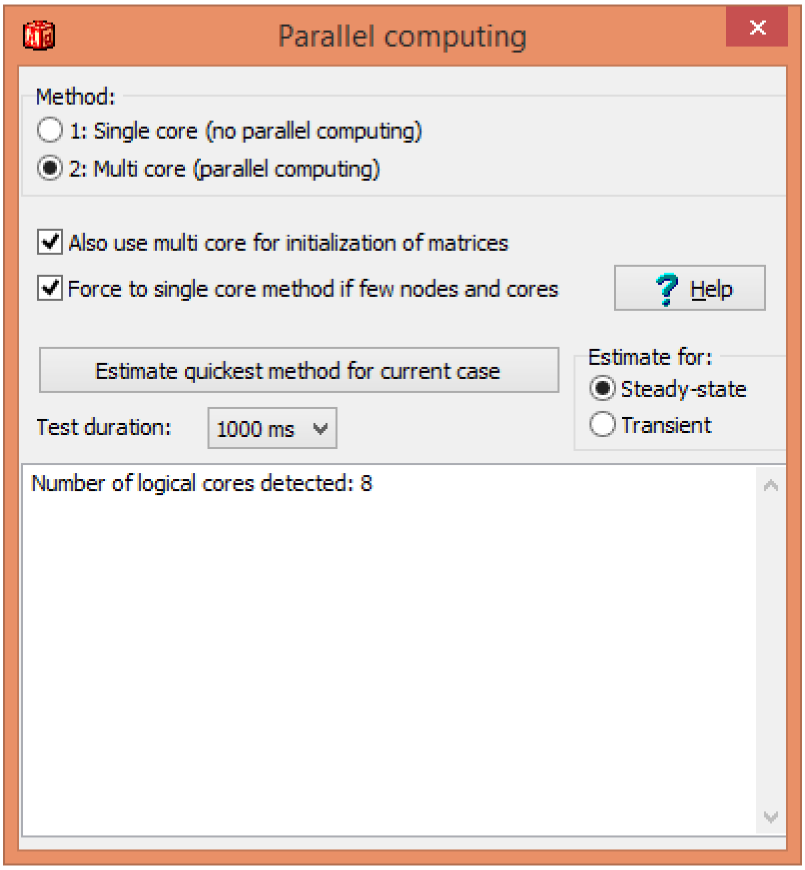
The more nodes (computational cells) the higher the speed increase will be to solve a problem using parallel execution. This is good because large problems may take a lot of time to solve and it is those problems we want to solve quicker. (Small problems often takes a few seconds to solve so it really does not matter in these cases.)

The number of logical cores will also affect the speed increase. Some benchmarks are shown in the next section.

For cases with fewer nodes (typically less than about 100,000-1,000,000) the parallel execution is often slower compared to the single thread execution due to internal overhead costs to supervise the threads.

2.4 Options for parallel execution

When HEAT3 is started, the option for parallel computing is on by default. The figure below shows the options for parallel computing (menu item Solve/Parallel computing).



Method:
○ 1: Single core (no parallel computing)
● 2: Multi core (parallel computing)

☑ Also use multi core for initialization of matrices
☑ Force to single core method if few nodes and cores

Test duration: 1000 ms

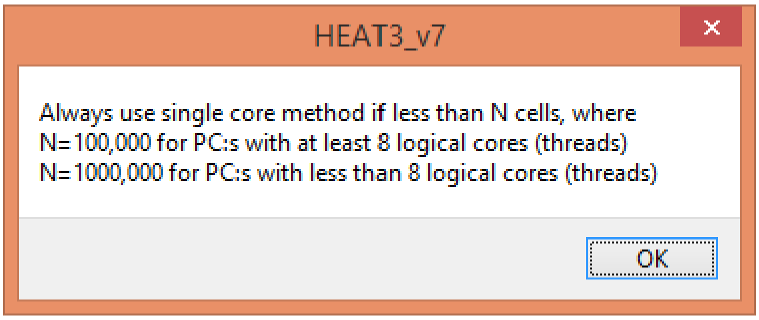
Estimate for:
● Steady-state
○ Transient

```
Number of logical cores detected: 8
```

Method for numerical solution
The multi-core method for the numerical solution is checked by default.

Also use multi core for initialization of matrices
For large cases (i.e. many nodes), the initialization can take a few seconds and it is possible to use multi core speed for this as well. This option only applies if the multi core method is checked.

Force to single core method if few nodes and cores
The gain of speed will be greater the more nodes that are used. With few cells, the multi core method often is *slower* than the single core method due to overhead "costs" of management and synchronization of all cores. Therefore, the single core method will be used (if this option is checked) for cases with less than 100,000 nodes for PC:s with at least 8 logical cores (normally 4 physical cores) and for cases with less than 1000,000 nodes for PC:s with less than 8 logical cores (e.g. for a PC with two physical cores). The Help button shows the following message:



Always use single core method if less than N cells, where
N=100,000 for PC:s with at least 8 logical cores (threads)
N=1000,000 for PC:s with less than 8 logical cores (threads)

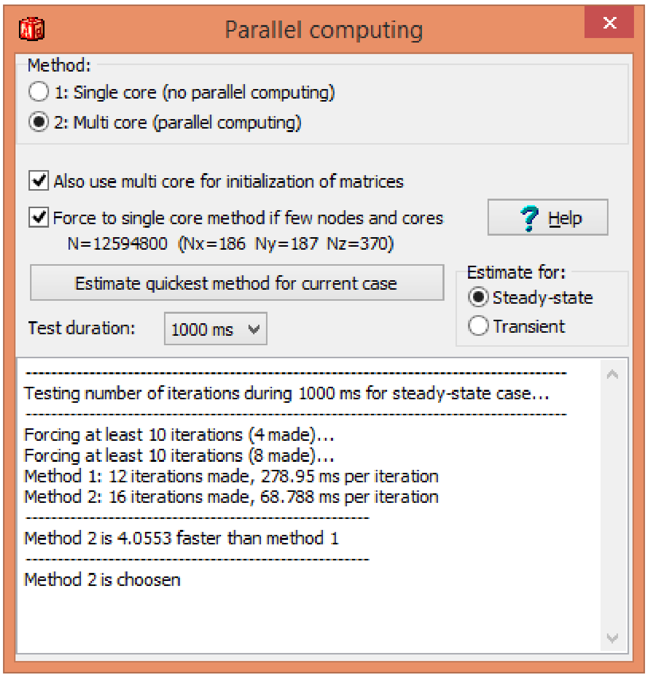
If you find out that a case still is quicker to solve (see next option regarding estimation) using the multi core method you can turn this option off.

Estimate quickest method for current case
Press this button to determine how much faster (or slower) the multi core method will be for the *current case with the current mesh*. Two calculations will be made counting the number of iterations achieved during a certain lapsed time duration (by default 1000 milliseconds) using the single core and multi core method, respectively. Estimations can be made for both **steady-state** and **transient** cases. The quickest method obtained will be chosen (checked).

There is no need to give boundary conditions in order to start the estimation.

It might be good to know that the estimation will perform actual calculations of both the temperatures and the heat flows. This means that you should not start an estimation if you just made a transient calculation (and need those correct temperatures and heat flows).

Below is an estimation showing that the multi core method is about four times faster than the single core method. Note that at least 10 iterations is needed. For the chosen test duration (here 1000 ms) only 4 iteration was made in single core mode, so more iterations are made automatically.



Method:
○ 1: Single core (no parallel computing)
● 2: Multi core (parallel computing)

☑ Also use multi core for initialization of matrices
☑ Force to single core method if few nodes and cores
N=12594800 (Nx=186 Ny=187 Nz=370)

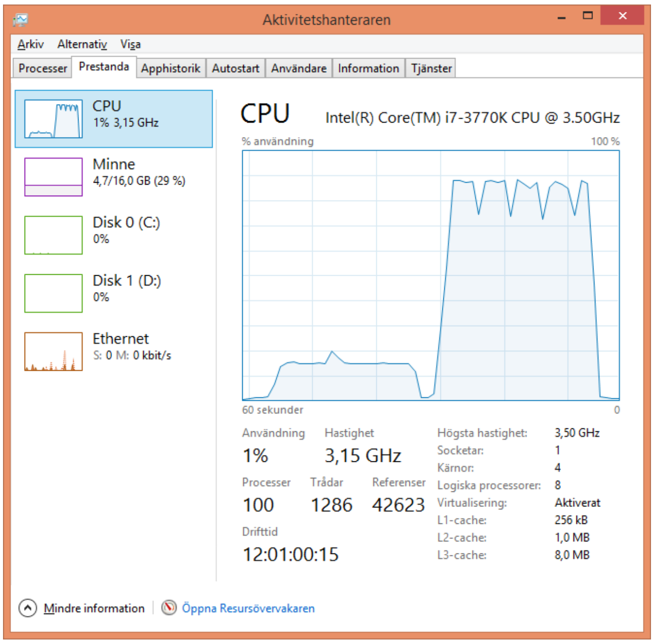
Test duration: 1000 ms

Estimate for:
● Steady-state
○ Transient

```
-----------------------------------------------------------------------
Testing number of iterations during 1000 ms for steady-state case...
-----------------------------------------------------------------------
Forcing at least 10 iterations (4 made)...
Forcing at least 10 iterations (8 made)...
Method 1: 12 iterations made, 278.95 ms per iteration
Method 2: 16 iterations made, 68.788 ms per iteration
--------------------------------------------------------
Method 2 is 4.0553 faster than method 1
--------------------------------------------------------
Method 2 is choosen
```

2.5 CPU loads

A parallel execution requires more system resources compared to a single thread execution. Consider the figure below regarding the CPU load. The first part shows a calculation using the single core method (up to about 20% of "CPU" is used). The second part shows the calculation using the multi core mode (up to 90% is used) for the same case.



CPU 1% 3,15 GHz
Minne 4,7/16,0 GB (29 %)
Disk 0 (C:) 0%
Disk 1 (D:) 0%
Ethernet S: 0 M: 0 kbit/s

CPU Intel(R) Core(TM) i7-3770K CPU @ 3.50GHz

Användning 1%
Hastighet 3,15 GHz
Processer 100
Trådar 1286
Referenser 42623
Drifttid 12:01:00:15

Högsta hastighet:	3,50 GHz
Socketar:	1
Kärnor:	4
Logiska processorer:	8
Virtualisering:	Aktiverat
L1-cache:	256 kB
L2-cache:	1,0 MB
L3-cache:	8,0 MB

To view this chart in Windows 8, do as follows:

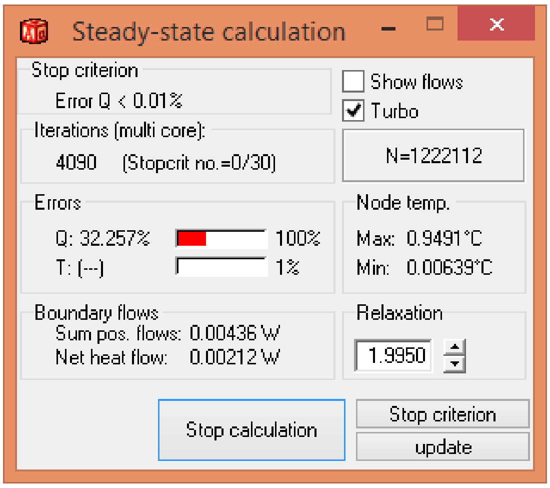
1. Open Task Manager by right-clicking the taskbar, and then clicking Task Manager.
2. Click the Performance tab.

A high percentage means that the programs or processes that are running require a lot of CPU resources, which can slow your computer in regards to using other programs.

To determine what kind of CPU is installed on your computer click the Start button, right-click Computer, and then click Properties. In the System section, next to Processor, view your computer's CPU speed and type. You can also go to the menu Info/System info in HEAT3 to get the number of logical cores. It is also shown in the window for “Parallel computing” in HEAT3.

2.6 So what do I need to do to use multi core computing

Basically nothing. When HEAT3 is started, the option for parallel computing is on by default. If there are few nodes, the single core method will be used. If you have interest in knowing the speed increase (or decrease) just use the estimation button (see above). The solve window will indicate if multi core computing is used (see “Iterations (multi core)” below).



Stop criterion
Error Q < 0.01%

Iterations (multi core):
4090 (Stopcrit no.=0/30)

□ Show flows
☑ Turbo

N=1222112

Errors
Q: 32.257% 100%
T: (---) 1%

Node temp.
Max: 0.9491°C
Min: 0.00639°C

Boundary flows
Sum pos. flows: 0.00436 W
Net heat flow: 0.00212 W

Relaxation
1.9950

2.7 Benchmarks

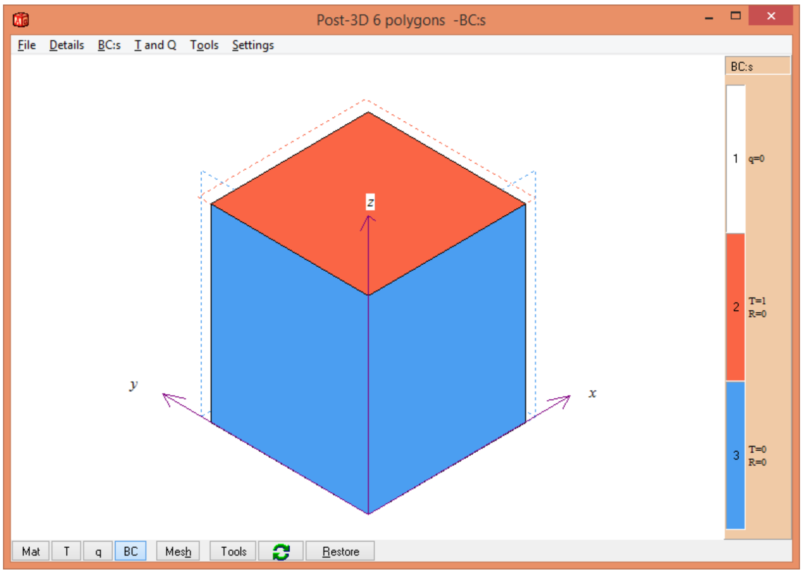
The benchmarks below are made on the following computers:

PC1: Stationary, Intel Core i7-3770K 3.5GHz (4 cores, 8 threads) / 16GB RAM / Windows 8.1
PC2: Laptop, Intel Core i7-3517U (2 cores, 4 threads) / 4 GB RAM / Windows 8.1

2.7.1 Benchmark 1: Heat flow through a cube

This case is described in "Theory of HEAT2/HEAT3, (Doctorate thesis, Blomberg), (PDF), 2.5 MB, 202 pages, 1996", see page 21-22 in
http://www.buildingphysics.com/manuals/avh_TB.pdf

We have a cube with T=1 on the upper side and T=0 on the other five sides. The exact heat flow through the bottom side is from analytical solution 0.068819 W.



1 million nodes (symmetry: 50x50x100 used):

PC1	**Iterations**	**Heat flow (W)**	**Cpu time**	**Quicker**
Single core	700	0.068848	4s	
Multi core	700	0.068848	2s	x2

PC2	**Iterations**	**Heat flow (W)**	**Cpu time**	**Quicker**
Single core	700	0.068848	6s	
Multi core	700	0.068848	22s	no

8 million nodes (symmetry: 100x100x200 used):

PC1	**Iterations**	**Heat flow (W)**	**Cpu time**	**Quicker**
Single core	1144	0.068824	55s	
Multi core	1084	0.068824	15s	x3,7

PC2	**Iterations**	**Heat flow (W)**	**Cpu time**	**Quicker**
Single core	1100	0.068824	70s	
Multi core	1100	0.068824	39s	1,8

50.7 million nodes (symmetry: 185x185x370 used):

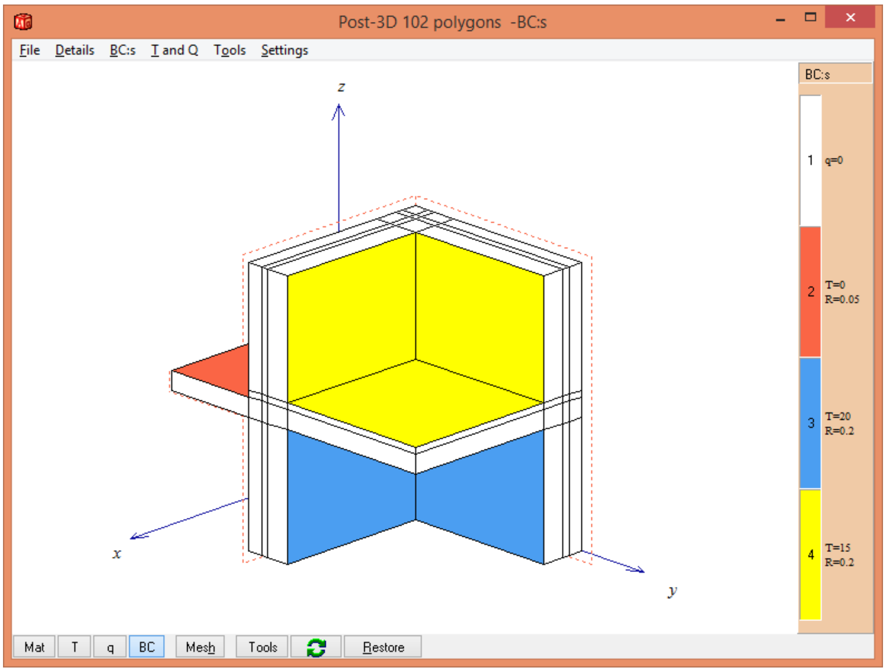
PC1	**Iterations**	**Heat flow (W)**	**Cpu time**	**Quicker**
Single core	3500	0.068820	17min	
Multi core	2400	0.068820	3min	x5,7

50.7 million nodes (symmetry: 185x185x370 used):

PC2	**Iterations**	**Heat flow (W)**	**Cpu time**	**Quicker**
Single core	3500	0.068820	22min	
Multi core	2400	0.068820	7min	x3

2.7.2 Benchmark 2: EN ISO 10211 test case 3

This case is described in
http://www.buildingphysics.com/download/iso/ISO_10211_HEAT2_HEAT3.pdf


Post-3D 102 polygons -BC:s
File Details BC:s T and Q Tools Settings
z
x
y
BC:s
1 q=0
2 T=0 R=0.05
3 T=20 R=0.2
4 T=15 R=0.2
Mat T q BC Mesh Tools Restore

571 000 nodes:

PC1	Iterations	Heat flow (W)	Cpu time	Quicker
Single core	274	64.75	6s	
Multi core	274	64.75	3s	x2

PC2	Iterations	Heat flow (W)	Cpu time	Quicker
Single core	274	64.72	7s	
Multi core	274	64.72	9s	no

1,3 million nodes:

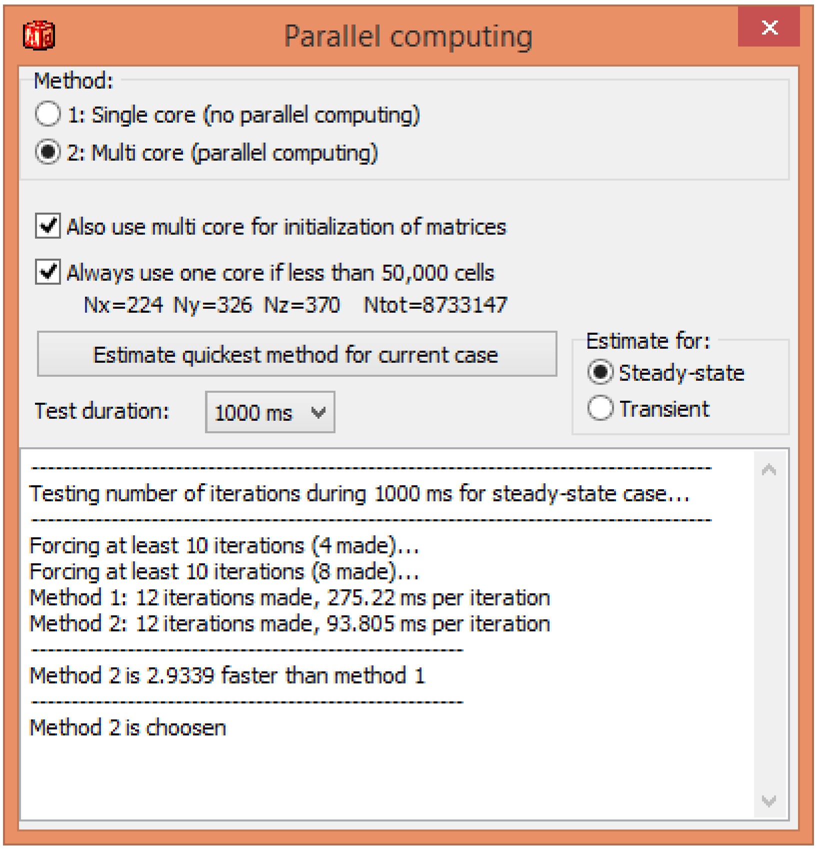
PC1	Iterations	Heat flow (W)	Cpu time	Quicker
Single core	414	64.75	20s	
Multi core	459	64.75	9s	x2,2

PC2	Iterations	Heat flow (W)	Cpu time	Quicker
Single core	414	64.75	24s	
Multi core	459	64.75	19s	x1,3

8,7 million nodes:

PC1	Iterations	Heat flow (W)	Cpu time	Quicker
Single core	750	64.79	225	
Multi core	689	64.79	74s	x3,0

PC2	Iterations	Heat flow (W)	Cpu time	Quicker
Single core	729	64.79	285s	
Multi core	793	64.79	162s	x2,2



Method:
○ 1: Single core (no parallel computing)
● 2: Multi core (parallel computing)

☑ Also use multi core for initialization of matrices

☑ Always use one core if less than 50,000 cells
Nx=224 Ny=326 Nz=370 Ntot=8733147

Test duration: 1000 ms

Estimate for:
● Steady-state
○ Transient

```
--------------------------------------------------------------------------------
Testing number of iterations during 1000 ms for steady-state case...
--------------------------------------------------------------------------------
Forcing at least 10 iterations (4 made)...
Forcing at least 10 iterations (8 made)...
Method 1: 12 iterations made, 275.22 ms per iteration
Method 2: 12 iterations made, 93.805 ms per iteration
---------------------------------------------------
Method 2 is 2.9339 faster than method 1
---------------------------------------------------
Method 2 is choosen
```

Estimation for PC1, 8,7 million nodes.

3. Miscellaneous

3.1 Adapted for Windows 7 and 8

HEAT3 is now fully adapted for windows Vista and Windows 7 and 8. HEAT3_v7.exe is placed in folder %ProgramFiles(x86)" during installation.

One important change from previous versions is that the startup folder now is "My Documents\Blocon\HEAT3 7". This means that the path should be "C:\Users\[user name]\Blocon\HEAT3 7". The materials files (*.mtl) should be placed in this folder. This set up is similar for HEAT2 v9.

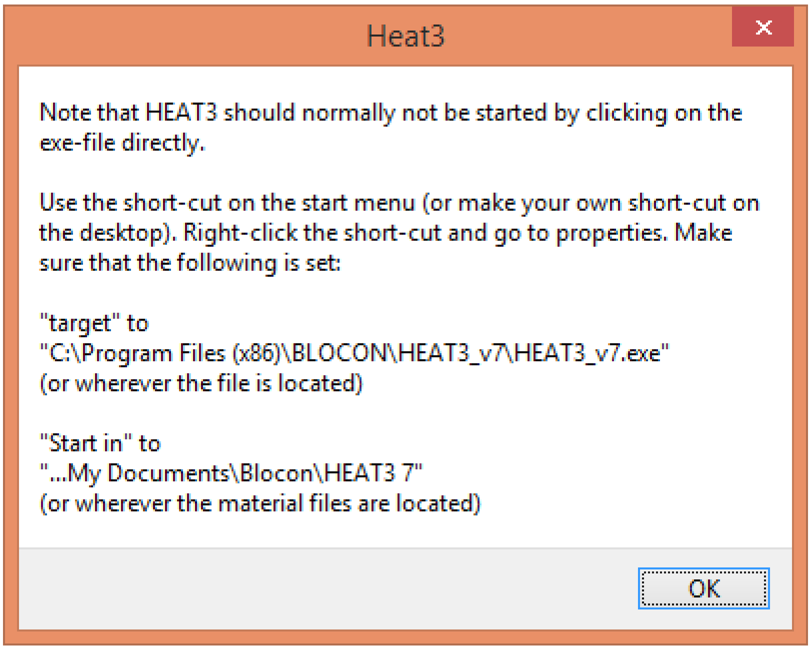
The program should normally be started using the short-cut on the start menu. To start HEAT3, press the windows key and write "HEAT3 v7". The icon should be displayed. Click this.

Do as follows if you want to create a short-cut on the desktop:

1. Right-click the desktop, choose New/Short-cut. Browse to "C:\Program Files (x86)\BLOCON\HEAT3_v7" and add "HEAT3_v7.exe".
2. Right-click on the icon ("HEAT3_v7.exe".) created on the desktopn and choose properties.

   Make sure that "target" is set to
   "C:\Program Files (x86)\BLOCON\HEAT2_v9\HEAT2_v9.exe"
   (or wherever the file is located)

   and "Start in" is set to
   "...My Documents\Blocon\HEAT2 9" (or wherever the files are located)



Note that HEAT3 should normally not be started by clicking on the exe-file directly.

Use the short-cut on the start menu (or make your own short-cut on the desktop). Right-click the short-cut and go to properties. Make sure that the following is set:

"target" to
"C:\Program Files (x86)\BLOCON\HEAT3_v7\HEAT3_v7.exe"
(or wherever the file is located)

"Start in" to
"...My Documents\Blocon\HEAT3 7"
(or wherever the material files are located)