Scalability of parallel shared-memory isogeometric alternating-direction solver for wildfires simulations

Juliusz Wasieleski[1], Andres Medina[2], Maciej Woźniak[1][0000-0002-5576-5671], Marcin Łoś[1][0000-0002-8426-6345], Paulina Sepulveda[2][0000-0002-7146-2240], Anna Paszyńska[3][0000-0002-0716-0619], Maciej Paszyński[1][0000-0001-7766-6052]

[1]AGH University of Krakow, Poland
[2]Pontificia Universidad Catolica de Valparaiso, Chile
[3]Jagiellonian University, Krakow, Poland
juliuszwas@student.agh.edu.pl, anmedinas@gmail.com, macwozni@agh.edu.pl, los@agh.edu.pl, paulina.sepulveda@pucv.cl, anna.paszynska@uj.edu.pl, paszynsk@agh.edu.pl

Abstract. The problem of wildfires is very common in many countries around the world, such as Canada and Chile. One example is the wildfire that occurred in Grasslands National Park in Canada in April 2013. Another example is a series of wildfires between February 1 and 5, 2024, in the Valparaiso region of Chile. Computer simulations of fires allow us to predict their future evolution, which, in turn, can enable better control of firefighters' work. For this to be possible, the simulators developed must be efficient and effective. In our article, we analyze the parallel scalability of the WILDFIRES-IGA-ADS simulator on a concurrent server with shared memory. Our simulator employs the isogeometric finite element method with tensor product B-spline basis functions, explicit time integration schemes, and a linear computational cost solver resulting from the Kronecker product structure of the mass matrix. The parallel code reaches speedup up to 10 on 12 physical cores.

Keywords: Isogeometric L^2 projections · Explicit solver · Wildfire model · Parallel computing · Code installation

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_6

1 Introduction

The frequency, severity, and duration of forest fires are increasing worldwide. The risk of these phenomena increases in extremely dry conditions. Forest fires are becoming more extreme. They affect larger areas, have a longer duration, and exhibit stronger intensity. Classic finite element solvers used for forest fire simulations involve high computational costs and require careful adjustment of model parameters [6]. To overcome this problem, we apply a parallel shared-memory isogeometric finite element method solver. The main advantage is not only the Galerkin formulation, but the tensor product structure of the B-spline basis. It results in the Kronecker product structure of the mass matrix. This matrix can be inverted in linear computational cost with respect to the number of degrees of freedom. The cost also grows with the polynomial order, because the support of B-splines becomes wider for larger p. Thus, it allows for higher-order and continuity isogeometric analysis simulations [7] of the wildfires using a sequence of L2 projections. Our solver is an extension of the IGA-ADS code [14] for wildfire simulations.

2 Wildfire model

The partial differential model of wildfires employed in our paper is based on [6], where Physics Informed Neural Networks [11,2,3] were utilized for the simulations. The PDEs concern the temperature distribution T,

$$\rho \frac{\partial}{\partial t}\left(c_p T\right) = R_C + Q_W - \nabla \cdot \left(q_C + q_D + q_r\right) + Q_{conv} + Q_{rz}, \tag{1}$$

where c_p is the heat capacity coefficient, ρ is the density of the gas mixture. The right-hand side consists of multiple sinks and sources that are summarized below. The first term describes R_C, the combustion energy source

$$R_c = \rho c_h h_c \frac{M}{M_1} r. \tag{2}$$

It depends on the combustion enthalpy h_c, the enthalpy correction coefficient c_h, the reaction rate r, the local temperature T, the mole fraction of the fuel X_1, and the mole fraction of the oxidizer X_2, where M represents the molar mass of the gas mixture, M_i represents the molar mass of the i-th chemical component. The second term introduces Q_W, describing the spread of wildfires due to wind,

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_6

which represents the source modeled as an advection field b,

$$Q_W = -\rho c_W b \cdot \nabla (c_p T) . \tag{3}$$

Here C_w is the wind reduction coefficient. The next term describes the q_C, the conductive heat flux,

$$q_C = -\kappa \nabla T, \tag{4}$$

where κ is the thermal conductivity coefficient. The following term describes q_D, which is the interdiffusion enthalpy flux based on the Fickian approximation of the diffusive mass flux J_i.

$$q_D = \sum_i h_i J_i. \tag{5}$$

The next term describes the wildfire propagation due to q_r the radiative heat flux,

$$q_r = -4\sigma\epsilon\delta_x T^3 \nabla T, \tag{6}$$

where σ is the Boltzmann constant, ϵ is the emittance, and δ_x is the absorption length. The next term on the right-hand side describes Q_{conv}, representing the convection heat loss

$$Q_{conv} = \chi(T_{amb} - T), \tag{7}$$

with χ being the convective heat transfer coefficient, and T_{amb} the ambient temperature of the atmospheric domain above the vegetation. The last term Q_{rz} describes the vertical radiation heat loss

$$Q_{rz} = \sigma\epsilon\delta_z^{-1}(T_{amb}^4 - T^4) \tag{8}$$

where δ_z is the vertical emission coefficient. These equations are based on 20 parameters described in [6] and [12]. We adopted the parameter values from these references, and used them in our simulations. This resulted in stable numerical simulations.

We solve the problem on a two-dimensional horizontal domain. The visualizations shown later present this plane. At the domain boundaries, we use Neumann boundary conditions for the temperature field. The temperature T is measured in K. The other physical coefficients follow the units given in [6,12].

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_6

3 Wildfire simulations

The Canada wildfire simulation In this section, we present the simulation of the wildfire that occurred in the West Block of Grasslands National Park, Canada, as described in [8]. The fuel map has been generated using the notebook[1]. It has been obtained based on the satellite image of the wildfire terrain. We map the terrain color to the fuel amount using the following equation, with $n = 2.5$ chosen empirically:

$$fuel = (1 - green)^n \tag{9}$$

The intuition of this formula is that we relate the amount of fuel by the intensity of green color in the satellite image. The mapping has been implemented using the following python code:

```
POWER_EXPONENT = 2.5
green_channel = satellite_array[:, :, 1].astype(float) # take
    only green from RGB
green_normalized = green_channel / 255.0 #normalize green to
    0-1
fuel_nonlinear = np.power(1.0 - green_normalized,
    POWER_EXPONENT) #map accorging to the formula
```

Additionally, we have added one of the highways that pass through the national park. The output is presented in Figure 1.

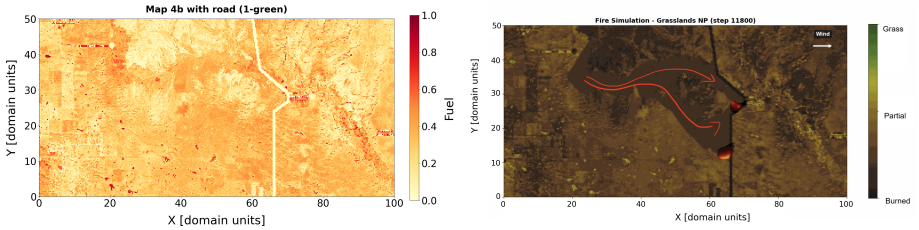


Fig. 1: Result of the fuel map generation. Frame from the simulation, from the moment just before fire stops. The arrows denote the real wildfire propagation.

We have executed the IGA-ADS simulation with the following arguments:

[1] https://github.com/puszekjuliuszek/fire-simulations/blob/main/grasslands_fuel_map.ipynb

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_6

```
N_STEPS = 14000 # number of simulation step
DT = 2.5e-4 # time step, bigger were crashing the simulation
IGNITION_X = 20 # x cooridinate of the fire ignition
IGNITION_Y = 35 # y cooridinate of fire ignition
WIND_BX = 15.0 # x value of the wind
WIND_BY = 0.0 #y value of the wind
```

The C++ code that had been used to generate the simulation can be found at the reference links[2][3]. The .gif file with the resulting simulation can be find at the link [4]here. The moment just before the fire stops is presented in Figure 1.

The Valparaiso area in Chile wildfire simulation From February 1 to 5, 2024, there was a series of catastrophic wildfires in different districts of Chile, such as Valparaiso, O'Higgins, Maule, Biobío, and Los Lagos. In our simulation, we focus on a localized part of the wildfire concerning the highway area. The fuel map is based on the satellite images from the CNN article[5]. The snapshots of the simulation are presented in Figure 2.

4 Scalability of the shared-memory parallel code

The computations have been executed on a Ryzen 9 3900X processor with 12 physical cores and 64 GB of RAM in DDR dual channel mode. The time is measured in seconds. The parallel implementation uses the Galois framework. In all experiments, we used different number of software threads, and measured the total wall-clock time.

Each reported timing comes from one execution. In future work, we plan to repeat the runs and report averages and standard deviations.

Benchmark setup We measure the execution time of multiple consecutive time steps. After spatial discretization, the temperature equation can be written in abstract form

$$T_{n+1} = \underbrace{\tau T_n + \tau \mathcal{L}(T_n) + \tau F(T_n)}_{\text{RHS term } B}, \tag{10}$$

where $\mathcal{L}$ denotes the linear transport–diffusion operator and F collects all nonlinear source and loss terms, including combustion heat release, convective exchange

[2] https://github.com/puszekjuliuszek/fire-simulations/blob/main/fire_grasslands.cpp
[3] https://github.com/puszekjuliuszek/fire-simulations/blob/main/fire_grasslands.hpp
[4] https://github.com/puszekjuliuszek/fire-simulations/blob/main/fire_burn_ash_animation.gif
[5] https://www.cnnbrasil.com.br/internacional/veja-imagens-de-satelite-que-mostram-o-chile-antes-e-depois-dos-incendios/

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_6

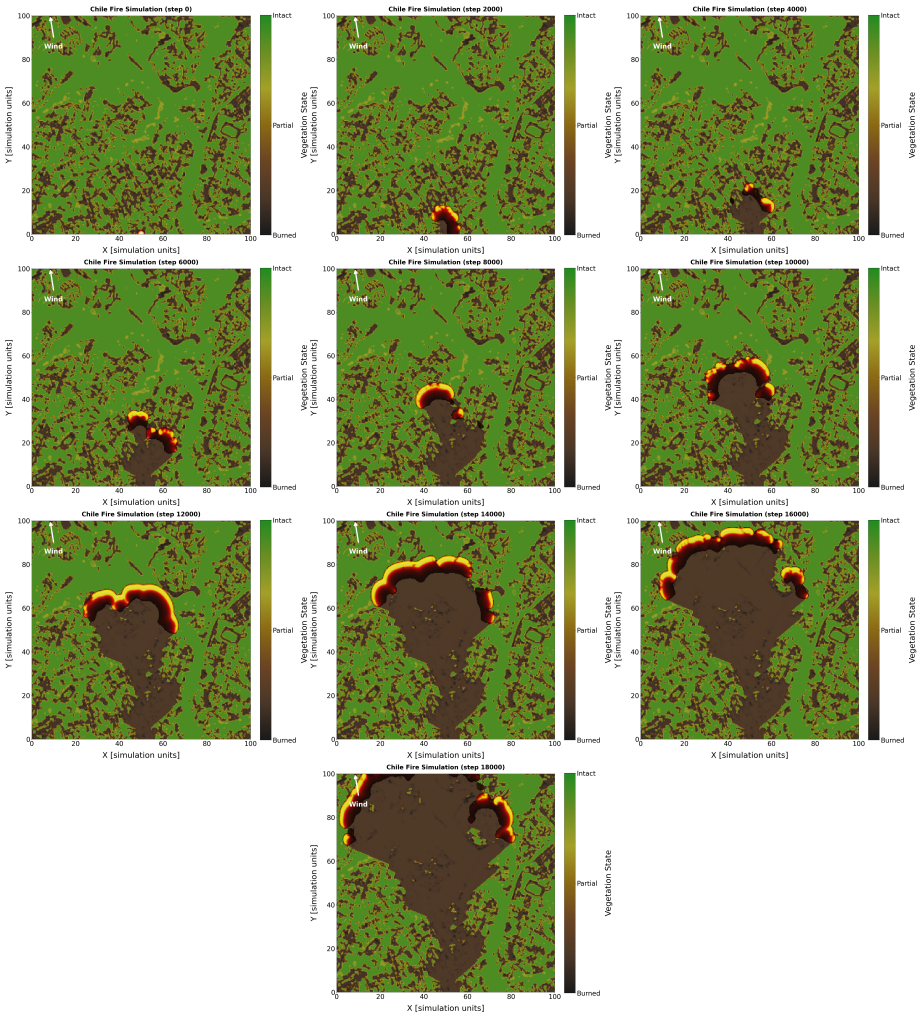


Fig. 2: Snapshot from the simulation of the wildfire at Valparaiso region in Chile. The labels in each panel show the simulation step. In future versions, we plan to replace them with physical time.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_6

with the ambient atmosphere, and vertical radiative losses. Since an explicit time integration scheme is employed, the time step Δt and the number of time steps are selected to satisfy the Courant–Friedrichs–Lewy (CFL) condition for each mesh resolution. After B-spline basis discretization, we obtain

$$\mathcal{M}_x \otimes \mathcal{M}_y T_{n+1} = \mathcal{B}(T_n) \tag{11}$$

The dominant cost per time step is the numerical integration over the computational domain. For each element Ω_e and the associated tensor-product B-spline basis functions N_i, we evaluate T and ∇T at Gaussian quadrature points. Next, we compute local contributions and assemble the elemental right-hand side. For the RHS integrand $g(\mathbf{x}, t)$, the elemental contributions take the classical form

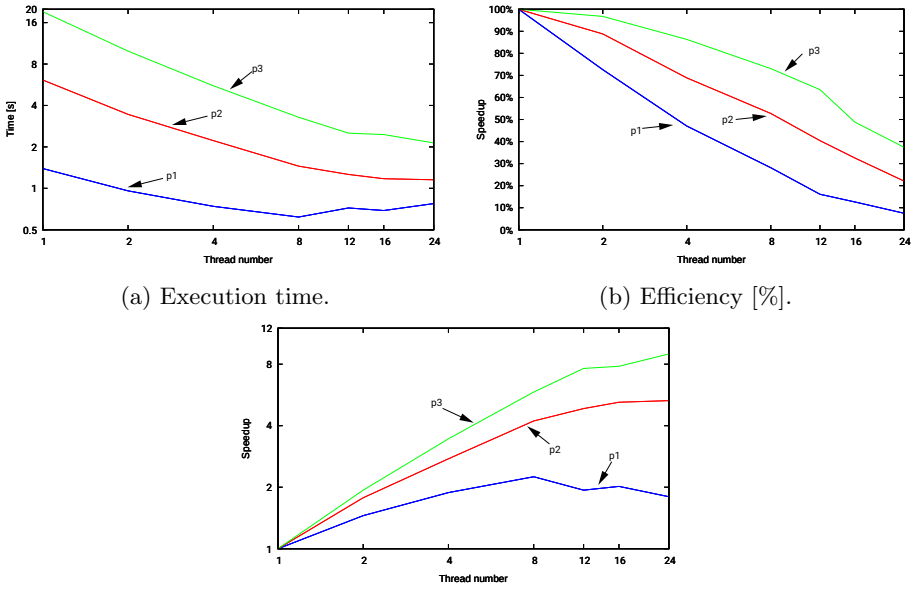
$$b_i^{(e)} = \int_{\Omega_e} g(\mathbf{x}, t)\, N_i(\mathbf{x})\, \mathrm{d}\Omega, \tag{12}$$

evaluated by a tensor-product Gauss rule. The update can be interpreted as a sequence of L^2 projections, i.e., solving at each step a mass-matrix system of the form (11). In our implementation, the application of $\mathcal{M}^{-1} = \mathcal{M}_x^{-1} \otimes \mathcal{M}_y^{-1}$ exploits the Kronecker-product structure of B-spline basis function, reducing the computations to linear time cost per time step.

Strong scaling We first tested the execution time of the wildfire simulation code on the computational mesh of size 100×100 using 120 time steps, varying the polynomial order of approximation from $p = 1, 2, 3$, as shown in Figure 3. As can be seen from Figure 3c, the speedup reaches 2 for linear B-splines, 5 for quadratic B-splines, and 9 for cubic B-splines. The drop after 12 cores is related to the fact that there are 12 physical cores, and the additional cores are simulated through multi-threading. A similar behavior can be observed in the efficiency estimates shown in Figure 3b. From the time measurements in Figure 3a, we see that we can solve the problem over 100×100 mesh in less than 1 second for linear B-splines, around 1.2 seconds using quadratic B-splines, and 2 seconds using cubic B-splines.

Next, we test the scalability of the wildfire simulations on the computational mesh of size 600×600 using 720 time steps for linear, quadratic, and cubic B-splines. These experiments are summarized in Figures 4. This is the number of time steps for which the explicit simulation behaves stably, and the total physical simulation time is the same as for the 100×100 mesh case. We compare wall-clock time needed to compute this simulation. We can see from Figure 4c

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_6



(a) Execution time.

(b) Efficiency [%].

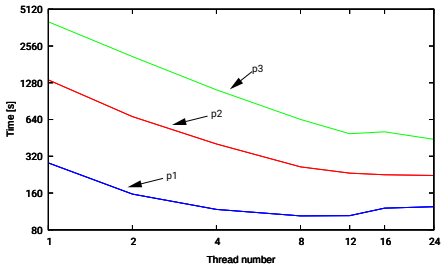
(c) Speedup.

Fig. 3: Strong scaling for 120 time steps of the wildfire simulation on a 100×100 mesh: (a) execution time, (b) efficiency, (c) speedup.

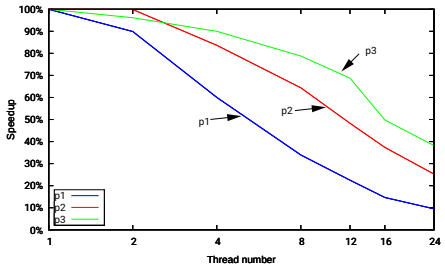
that the speedup reaches 3 for linear B-splines, 6 for quadratic B-splines, and 9 for cubic B-splines. The efficiency drop when we increase the number of cores is presented in Figure 4b. We can see that multi-threading is actually doing a good job for linear and quadratic B-splines (since the curve does not turn downward), but it is reducing the efficiency of the cubic B-splines. The drop after 12 cores is again related to the number of physical cores. The execution time for the problem size on the 600×600 mesh can be read from Figure 4a. We can solve the problem in around 100 seconds using linear B-splines, in around 230 seconds using quadratic B-splines, and in around 480 seconds using cubic B-splines.

Finally, we test the scalability of the wildfire simulator on 1500×1500 mesh. To have a stable numerical simulation, we use 1080 time steps. We vary the polynomial order of approximation from $p = 1$ through $p = 2$, up to $p = 3$, as shown in Figures 5. This time, the speedup reaches 3 for linear B-splines, 6 for quadratic B-splines, and 10 for cubic B-splines. The drop after 12 cores is again related to multi-threading. The efficiency is highest for cubic B-splines, moderate for quadratic B-splines, and lowest for linear B-splines; see Figure 5b. Execution time analysis of Figure 5a shows that we can solve the problem over 1500×1500

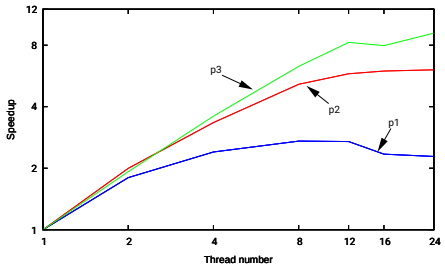
ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_6



(a) Execution time.



(b) Efficiency [%].



(c) Speedup.

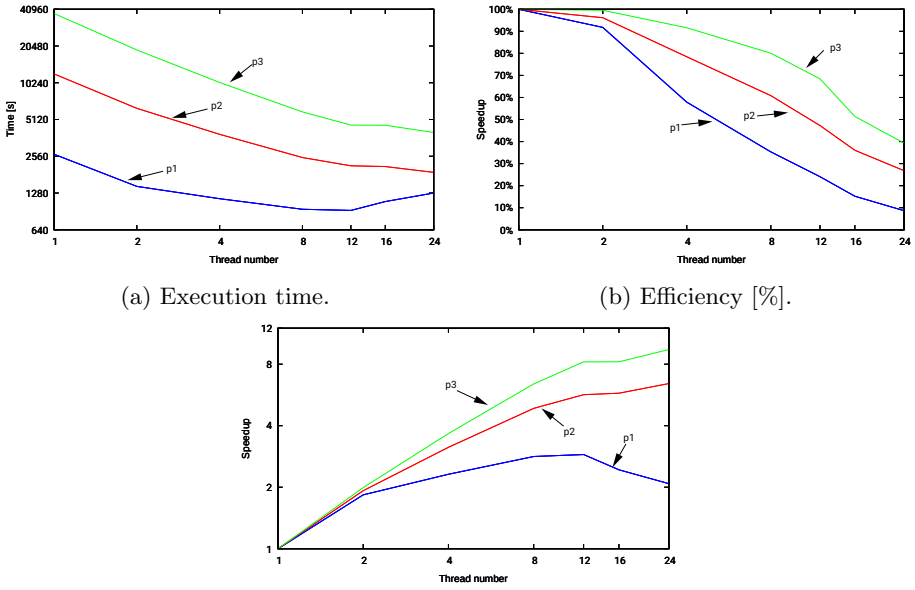
Fig. 4: Strong scaling for 720 time steps of the wildfire simulation on a 600×600 mesh: (a) execution time, (b) efficiency, (c) speedup.

mesh in around 1000 seconds with linear B-splines, around 2100 seconds using quadratic B-splines, and around 4000 seconds using cubic B-splines.

Weak scaling The weak scaling analysis is presented in Figures 6. First, we assign a patch of 100×100 elements to a single core and increase the number of cores from 1 to 24 while simultaneously increasing the mesh size. The mesh sizes for the increasing number of cores are summarized in Table 1. The weak scalability for 100×100 element patches is presented in Figure 6a. For linear B-splines, the execution time grows from less than 2 seconds to 6 seconds for 12 cores and 13 seconds for 24 cores. For quadratic B-splines, the time grows from 6 seconds for 1 patch to 12 seconds for 12 cores (mesh size 400×300) and up to 24 seconds for 24 cores (mesh size 600×400). For cubic B-splines, the single patch mesh takes 18 seconds, the 12 patch mesh takes 27 seconds, and the 24 patch mesh takes 49 seconds.

The weak scalability for 600×600 element patches is presented in Figure 6b. In this case, the single patch takes 277 seconds for linear B-splines, 1280 seconds for quadratic B-splines, and 3940 seconds for cubic B-splines. This execution time does not change much if we use two patches (1200×600 elements). Then,

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_6



(a) Execution time.

(b) Efficiency [%].

(c) Speedup.

Fig. 5: Strong scaling for 1080 time steps of the wildfire simulation on a 1500×1500 mesh: (a) execution time, (b) efficiency, (c) speedup.

it grows to around 1200 seconds for linear B-splines, 2600 seconds for quadratic B-splines, and 5600 seconds for cubic B-splines with 12 patches (mesh size of 2400×1800 elements). For multi-threading (more than 12 cores), the time further increases to around 2800 seconds for linear B-splines, 4700 seconds for quadratic B-splines, and 8500 seconds for cubic B-splines.

Finally, the weak scalability for 1500×1500 element patches is presented in Figure 6c. Here, single patch takes 2600 seconds for linear B-splines, 12000 seconds for quadratic B-splines, and 37000 seconds for cubic B-splines. The execution time decreases when we move to two patches (the mesh size 3000×1500 elements), down to around 2200 seconds for linear B-splines, 9200 seconds for quadratic B-splines, and 27000 seconds for cubic B-splines. We do not know how to explain this phenomenon. One possible reason is an effect related to cache usage or memory bandwidth. We leave a detailed study for future work. Next, the execution time increases to around 11000 seconds for linear B-splines, 21000 seconds for quadratic B-splines, and 4000 seconds for cubic B-splines when considering 12 cores and 6000×3000 elements. The final execution times for 24 cores

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_6

(9000×6000 elements) are 33000 seconds for linear B-splines, 48000 seconds for quadratic B-splines, and 82000 seconds for cubic B-splines.

Number of cores	Mesh for patch 100×100	Mesh for patch 600×600	Mesh for patch 1500×1500
1	100×100	600×600	1500×1500
2	200×100	1200×600	3000×1500
4	200×200	1200×1200	3000×3000
8	400×200	2400×1200	6000×3000
12	400×300	2400×1800	6000×4500
16	400×400	2400×2400	6000×6000
24	600×400	3600×2400	9000×6000

Table 1: Mesh sizes for increasing number of cores, for different patch sizes.

5 Compiling and running the wildfire simulator

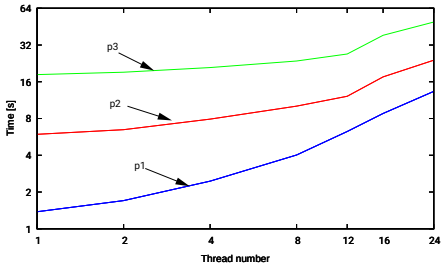
The source code for the wildfire simulation is available on Github: https://github.com/marcinlos/iga-ads. To download it, execute the following command in your terminal:

```
git clone https://github.com/marcinlos/iga-ads
```

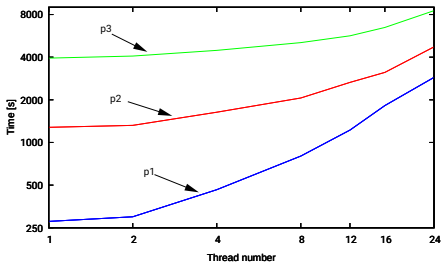
Installing dependencies Compiling the code and its dependencies requires installing a few libraries and tools. The following command installs the prerequisites on Ubuntu 24.04.

```
apt-get update && apt-get install \
    libzstd-dev         \
    curl                \
    gfortran            \
    g++                 \
    cmake               \
    ninja-build         \
    liblapack-dev       \
    libboost-all-dev    \
    llvm-dev            \
    libmumps-dev
```

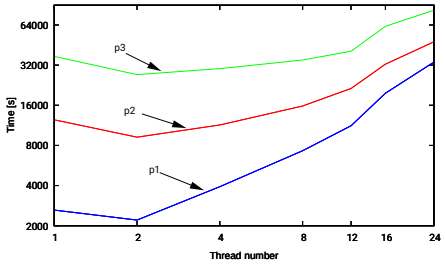
ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_6



(a) 100×100 patch.



(b) 600×600 patch.



(c) 1500×1500 patch.

Fig. 6: Weak scaling results for different patch sizes: (a) 100×100, (b) 600×600, (c) 1500×1500 elements.

Other distributions may require adjustments to this command. Alternatively, it is possible to skip this step and use the Docker/Podman image that contains the necessary packages (described later).

To compile the dependencies of the wildfire code, select some directories for source code and build artifacts of the dependencies:

```
DEPS=''/directory/for/sources''
DEPS_BUILD=''/directory/for/installing/libraries''
```

Important: these directories should not exist (they will be created by a script). Then, execute

```
iga-ads/scripts/install-dependencies.sh ''${DEPS}''
''${DEPS_DUILD}''
```

Compiling the wildfire simulation code

The IGA-ADS framework the wildfire code is a part of uses CMake build system. First, create a build directory, e.g.

```
cd iga-ads; mkdir build
```

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_6

and generate the build scripts by executing

```
cmake \
    -S . \
    -B build \
    -D CMAKE_BUILD_TYPE=Release \
    -D ADS_USE_GALOIS=ON \
    -D ADS_USE_MUMPS=OFF \
    -D CMAKE_PREFIX_PATH=''${DEPS}$''
```

Finally, to compile the code, execute

```
cmake --build build -j$(nproc) --target ads-example-fire
```

After that step, the compiled wildfire simulation is stored in `build/examples/fire`.

Running the simulation

The compiled simulation can be executed as

```
build/examples/fire <N> <p> <threads>
```

where `N` is the size of the mesh in each dimension, `p` is the B-spline degree, and `threads` is the number of threads to use.

6 Conclusions

Summing up, we can run the wildfire simulations on a workstation equipped with 24 cores (12 physical and 12 virtual) solving 120 time steps over 100×100 mesh of quadratic or cubic B-splines in less than 4 seconds. Increasing the resolution to 600×600 mesh and the number of time steps to 720 (due to the CFL condition), we need less than 8 minutes for higher-order cubic B-splines. Highest considered resolutions over 1500×1500 mesh requires between 30 and 60 minutes, using quadratic or cubic B-splines. These measurements imply high accuracy with reasonable timing on a single workstation without accessing supercomputing centers. From the presented simulations, we can conclude that both the amounts of fuel on the fuel map and the wind force and its direction had an influence on the simulation. The IGA-ADS is a good tool to simulate wildfires because we managed to simulate the real fire with good accuracy, as presented in Figure 1, where the arrows represent the real life wildfire phenomena. A full quantitative validation against field measurements is left for future work. The fire stops at the crossing road, both in the real wildfire case documented in Figure 1 and in our simulation. The future work may include comparison of the IGA

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_6

solvers to *hp*-FEM solvers [4,10,9]. The future work may also include the formulation and solution of an inverse problem related to fitting model parameters into measurement data from wildfire simulations. We may employ the inverse solvers described in [1,5]. We may also consider the development of distributed memory parallelization [13].

Acknowledgments

The authors are grateful for the support from the funds that the Polish Ministry of Science and Higher Education assigned to AGH University of Krakow. The work is supported by the "Excellence initiative - research university" for AGH University of Krakow.

References

1. Barabasz, B., Migórski, S., Schaefer, R., Paszyński, M.: Multi-deme, twin adaptive strategy hp-hgs. Inverse Problems in Science and Engineering **19**(1), 3–16 (2011). https://doi.org/10.1080/17415977.2010.531477
2. Cai, S., Mao, Z., Wang, Z., Yin, M., Karniadakis, G.E.: Physics-informed neural networks (PINNs) for fluid mechanics: A review. Acta Mechanica Sinica **37**(12), 1727–1738 (2021)
3. Chen, Y., Lu, L., Karniadakis, G.E., Dal Negro, L.: Physics-informed neural networks for inverse problems in nano-optics and metamaterials. Optics Express **28**(8), 11618–11633 (2020). https://doi.org/10.1364/OE.384875
4. Demkowicz, L., Gatto, P., Kurtz, J., Paszyński, M., Rachowicz, W., Bleszyński, E., Bleszyński, M., Hamilton, M., Champlin, C., Pardo, D.: Modeling of bone conduction of sound in the human head using hp-finite elements: Code design and verification. Computer Methods in Applied Mechanics and Engineering **200**(21), 1757–1773 (2011). https://doi.org/https://doi.org/10.1016/j.cma.2011.02.001, https://www.sciencedirect.com/science/article/pii/S0045782511000363
5. Gajda-Zagórska, E., Schaefer, R., Smołka, M., Paszyński, M., Pardo, D.: A hybrid method for inversion of 3d dc resistivity logging measurements. Natural computing **14**(3), 355–374 (2015)
6. Grasso, P., Innocente, M.S.: Physics-based model of wildfire propagation towards faster-than-real-time simulations. Computers & Mathematics with Applications **80**(5), 790–808 (2020). https://doi.org/https://doi.org/10.1016/j.camwa.2020.05.009, https://www.sciencedirect.com/science/article/pii/S0898122120302078

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_6

7. Hughes, T., Cottrell, J., Bazilevs, Y.: Isogeometric analysis: CAD, finite elements, nurbs, exact geometry and mesh refinement. Computer Methods in Applied Mechanics and Engineering **194**(39), 4135–4195 (2005). https://doi.org/https://doi.org/10.1016/j.cma.2004.10.008, https://www.sciencedirect.com/science/article/pii/S0045782504005171
8. Li, M., Guo, X.L.: Evaluating post-fire vegetation recovery in North American mixed prairie using remote sensing approaches. Open Journal of Ecology **8**, 646–680 (2018). https://doi.org/10.4236/oje.2018.812038
9. Pardo, D., Demkowicz, L., Torres-Verdín, C., Paszynski, M.: A self-adaptive goal-oriented hp-finite element method with electromagnetic applications. part ii: Electrodynamics. Computer Methods in Applied Mechanics and Engineering **196**(37), 3585–3597 (2007). https://doi.org/https://doi.org/10.1016/j.cma.2006.10.016, https://www.sciencedirect.com/science/article/pii/S0045782507000898, special Issue Honoring the 80th Birthday of Professor Ivo Babuška
10. Paszyński, M.: On the parallelization of self-adaptive hp-finite element methods part ii. partitioning communication agglomeration mapping (pcam) analysis. Fundamenta Informaticae **93**(4), 435–457 (2009)
11. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational physics **378**, 686–707 (2019)
12. Varadachari, C., Bhowmick, R.: Ab initio derivation of a dataset of real temperature thermodynamic properties: Case study with sic. Modelling and Simulation in Materials Science and Engineering **17**(7), 075006 (aug 2009). https://doi.org/10.1088/0965-0393/17/7/075006, https://dx.doi.org/10.1088/0965-0393/17/7/075006
13. Woźniak, M., Łoś, M., Paszyński, M., Dalcin, L., Calo, V.M.: Parallel fast isogeometric solvers for explicit dynamics. Computing and Informatics **36**(2), 423–448 (Jun 2017), https://www.cai.sk/ojs/index.php/cai/article/view/2017_2_423
14. Łoś, M.M., Woźniak, M., Paszyński, M., Lenharth, A., Hassaan, M.A., Pingali, K.: IGA-ADS: Isogeometric analysis fem using ads solver. Computer Physics Communications **217**, 99–116 (2017). https://doi.org/https://doi.org/10.1016/j.cpc.2017.02.023, https://www.sciencedirect.com/science/article/pii/S0010465517300759

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_6