TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B 2612 – Elektrotechnika a informatika

Studijní obor: 2612R011 – Elektronické informační a řídicí systémy

Vizualizace a postprocessing výsledků konečnoprvkových simulací

Vizualization and postprocessing of finite element simulation results

Bakalářská práce

Autor: **Jiří Staněk**

Vedoucí práce: Ing. Petr Šidlof, Ph.D.

V Liberci 28. 5. 2009

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé BP a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

V Liberci 28. 5. 2009

Podpis

Poděkování

Děkuji především vedoucímu mé bakalářské práce Ing. Petru Šidlofovi, Ph.D. za ochotu a vstřícnost při vzniku této bakalářské práce a s tím spjatém řešení problémů.

Práce byla částečně podpořena z projektu GAAV KJB200760801 "Matematické modelování a experimentální výzkum interakce proudění s pružnou strukturou v lidských hlasivkách".

Abstrakt

V této bakalářské práci je řešen problém vizualizace a postprocessing výsledků konečnoprvkových simulací. K zobrazení a vyhodnocení výsledků bylo použito programu Paraview.

V práci naleznete také vysvětlení principů konečnoprvkových simulací a to zejména tří jejich hlavních částí (preprocessing, vlastní výpočet a postprocessing). Bude zde popsán i princip metody konečných prvků.

Dozvíte se zde o nutnosti převodu souborů s popisem sítě a příslušnými hodnotami vektorů, či skaláru v příslušných místech sítě s příponami *.mesh a *.bb na formát kompatibilní s programem Praview (*.vtk). Tudíž bude představen konvertor těchto souborů, který bylo nutné naprogramovat.

Program Paraview bude představen na základě několika ukázkových souborů, které jsou k dispozici na oficiálních stránkách tohoto programu a také na našich konkrétních výsledcích konečnoprvkových simulací proudění v lidských hlasivkách.

Klíčová slova: Paraview, VTK, konvertor, konečnoprvkové simulace, lidské hlasivky

Abstract

In this bachelor thesis the problem of visualization and postprocessing of finite element simulations results is dealt with. The Paraview program was used for result projection and interpretation.

In the work you can also find the explanation of finite element simulation principles, especially their three main parts (preprocessing, the calculation itself and postprocessing). The principle of the finite element method will be described here, too.

You will learn about the need of conversion of files which contain mesh description as well as relevant vector or scalar values at mesh nodes with filename extensions *.mesh and *.bb to a format compatible with the Paraview software (*.vtk). Hence the converter of these files will be presented, which needed to be programmed.

The Paraview software will be presented on the basis of several sample files which are available on official websites of this program and also on our concrete results of the finite element simulation model airflow in human vocal folds.

Keywords: Paraview, VTK, convertor, finite element simulation, vocal folds

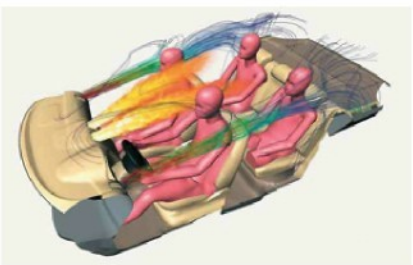
Obsah

Prohlášení ... 3
Poděkování ... 4
Abstrakt ... 5
Obsah ... 6
1 Úvod ... 7
2 Metoda konečných prvků ... 8
2.1 Historie ... 8
2.2 Úvod a použití ... 8
2.3 Modelování ... 10
3 Vizualizace ... 11
3.1 Úvod ... 11
3.2 Historie ... 11
4 VTK ... 12
4.1 Úvod a historie ... 12
4.2 Formát souborů VTK ... 14
5 Konvertor ... 16
5.1 Úvod ... 16
5.2 Soubor s příponou BB ... 17
5.3 Soubor s příponou MESH ... 17
5.4 Soubor s příponou VTK ... 19
5.5 Konvertor MESH_BB_TO_VTK ... 21
6 ParaView ... 22
6.1 Úvod ... 22
6.2 Uživatelské rozhraní ... 24
6.3 Zdroje dat ... 25
6.4 Filtry ... 26
6.5 Časově proměnné vizualizace ... 28
6.6 Problémy ... 31
7 Závěr ... 32
Použitá literatura ... 34
Přílohy ... 35
Konvertor souborů MESH a BB do souboru VTK ... 35

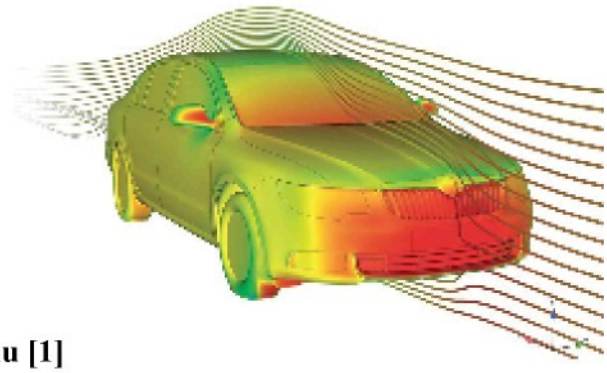
1 Úvod

Začátek bakalářské práce je věnován úvodu do problematiky modelování. Při modelování dochází ke zjednodušení reálného problému v důsledku použití numerických metod při jeho řešení. Jednou z nejpoužívanějších numerických metod pro diskretizaci původně spojitého problému a následné výpočty hodnot zkoumané veličiny je metoda konečných prvků. Fáze popisu geometrie a jeho diskretizace se nazývá preprocessing. Za ním následuje fáze vlastního výpočtu hodnot a poslední částí je postprocessing, kde zjišťujeme podrobnější informace na vizuálně zobrazených datech, které jsme získali v předchozích fázích. Numerické simulace jsou stále více používány ve vědě i průmyslu. Dochází i ke zdokonalování modelů a s tím roste množství zpracovávaných dat. Pracovat s rozsáhlými daty zabere mnoho výpočetního času, proto se neustále požaduje zvyšování výkonnosti hardwaru, na kterém jsou výpočty a vizualizace spuštěny. Samozřejmě je důležité vyvíjet i software, který by měl umět stabilně pracovat s takto rozsáhlými daty.

V dnešní době je možné dělit software do několika základních skupin: komerční software, akademický software a freeware. Akademické softwary mají výhodu, že jsou často ve formě opensource balíků a jsou dostupné zcela zdarma. Opensource znamená, že je volně ke stažení i zdrojový kód softwaru. Tudíž je možné software dle potřeb vyvíjet nebo upravovat. Jak již z názvu plyne, tak velké oblibě se tento druh softwaru těší v akademické vrstvě. Jednomu z opensource balíků se budu věnovat v této bakalářské práci podrobněji. Jedná se o software Paraview, pomocí kterého budeme vizualizovat konkrétní simulaci proudění v lidských hlasivkách. Dostupná data jsme měli v jiném formátu, než je Paraview schopen zpracovat, tudíž bylo nutné vytvořit konvertor do příslušného formátu. Simulace proudění je velmi často používaná například v automobilovém či leteckém průmyslu, kde se dolaďuje aerodynamika vozidel či letadel. Ovšem nemusí se jednat pouze o proudící vzduch, ale také o vodu a jiná média.



Obr. 1: Proudění v kabině automobilu [1]



Obr. 2: Proudění kolem exteriéru automobilu [1]

2 Metoda konečných prvků

2.1 Historie

Je velmi těžké napsat přesné datum vzniku jakékoliv metody, jelikož se ve většině případů jedná o postupný vývoj a může trvat desetiletí, než se objeví první ucelená práce na dané téma. Rychlejší vývoj této metody nastal se vznikem digitálních počítačů. Bylo to ze dvou prostých důvodů, kterými jsou efektivnost a aplikovatelnost. Tyto dvě vlastnosti jsou základem použitelnosti spousty teorií. Samozřejmě, že z počátku se vyskytly různé teorie pojednávající o téže věci. Lišily se například tvarem elementů použitých na diskretizaci tělesa. V šedesátých letech minulého století se vědci na celém světě začali touto metodou zabývat a začalo vycházet spousta publikací. Vývoj je rozdělil na tři základní skupiny: aplikované matematiky, fyziky a konstruktéry. První zaměření bylo na řešení fyzikálních problémů v mechanice. Jednalo se o deformace, průhyby a jiné vlastnosti materiálu při působení sil. Rychlé rozšíření do dalších oblastí nastalo po objevení univerzálnosti metody konečných prvků. [2]

2.2 Úvod a použití

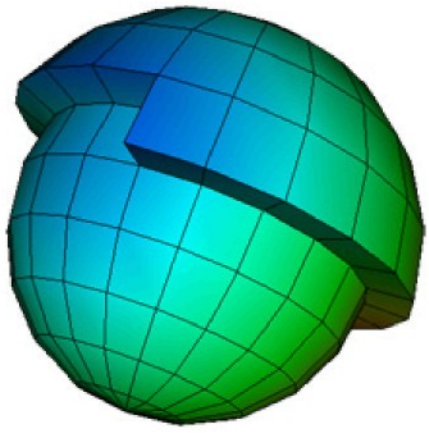
Metoda konečných prvků se používá pro řešení fyzikálních problémů v konstrukční analýze a návrhu nových zařízení. Hlavní myšlenkou je studovanou oblast rozdělit na drobné elementy (síť) a převést řešení parciální diferenciální rovnice na diskrétní problém (řešení soustavy lineárních rovnic). Při reprezentaci fyzikálního problému musíme brát v úvahu jisté předpoklady, které výsledný model sice znepřesní, ale umožní nám ho vyřešit. Jakmile zidealizujeme fyzikální problém, dostaneme matematický model skládající se z diferenciálních rovnic, jejichž řešení se provádí jednou z mnoha numerických metod, jako je Euler, Runge-Kutta, atd.. Jak jsem již výše zmiňoval, není možné dostat výsledek na sto procent odpovídající realitě, jelikož nedokážeme plně popsat přírodní děj a také při řešení matematického modelu dochází k jistým chybám. Výsledek se dá zpřesnit mnoha způsoby. Jedním z nich je upravení samotného řešení metody konečných prvků (zjemnění sítě). Další možností je upravit matematický model (geometrii, kinematiku, vlastnosti materiálů, okrajové podmínky).

Klíčovým krokem konstrukční analýzy je volba matematického modelu. Vybíráme ho na základě zkoumaných vlastností objektu. Potřebujeme získat co nejpřesnější a nejefektivnější matematický model. Abychom poznali, do jaké míry

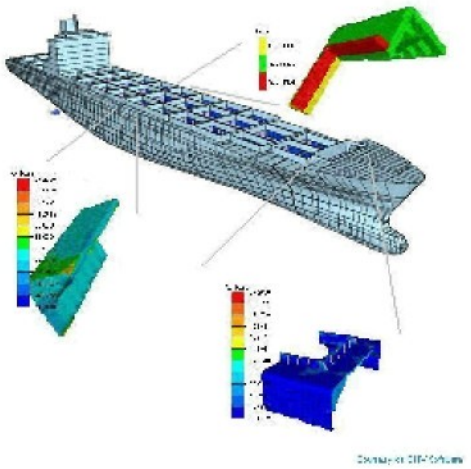
máme model vypovídající, porovnáme ho s úplným matematickým modelem (plně popsaný 3D model). Pod pojmem efektivní si představme model, který popisuje přesně to, co chceme s co nejmenšími náklady na čas i finance. Důležitá je také volba počtu elementů tvořících diskretizační síť. Při velmi hrubé síti nedosáhneme dostatečně přesného výsledku. Také volba velmi jemné sítě nemusí být nejlepší, jelikož nám přináší vysoké výkonové nároky s neodpovídajícím nárůstem kvality výsledku.

Metoda konečných prvků je obsažena v projektování pomocí počítačů, které se nazývá CAD (Computer-aided design). Metoda musí být velmi robustní, jelikož by měla být nezávislá na vstupních datech, jako jsou například počáteční podmínky. Bez robustnosti bychom určitě nedocílili požadované přesnosti. Pokud by metoda konečných prvků byla zaměřena pouze na pár konkrétních druhů materiálu a problémů, vznikalo by spousta chyb, jelikož je potřeba pracovat s obrovským spektrem různých vlastností a materiálů, z nichž každý má specifické vlastnosti. Jestliže to shrnu, tak metoda musí být spolehlivá, robustní a efektivní, jelikož konstruktér nemá čas řešit vzniklé chyby a ve většině případů ani pořádně nerozumí problematice, která se v programu skrývá pod jedním tlačítkem.

Tato metoda má široké spektrum využití a lez očekávat, že se i v nadcházejících letech bude stále více a více používat. Pokud bychom se zamysleli nad tím, co bude v budoucnosti, tak nejspíš dospějeme k názoru, že dojde k jisté integraci a automatizaci, kde zadáme, co chceme vyřešit a program se o to sám postará (vyřeší případné nepřesnosti, atd.). Dnes metodu konečných prvků používáme na řešení mnoha různých problémů, jako je mechanické namáhání, proudění tepla, či tekutiny atd.. Pokud bychom ale nahlédli do minulosti, zjistíme, že průkopníkem a počátečním motorem pro vývoj metody konečných prvků byl letecký průmysl. [2]



Obr. 3: Diskretizace kulové plochy [3]

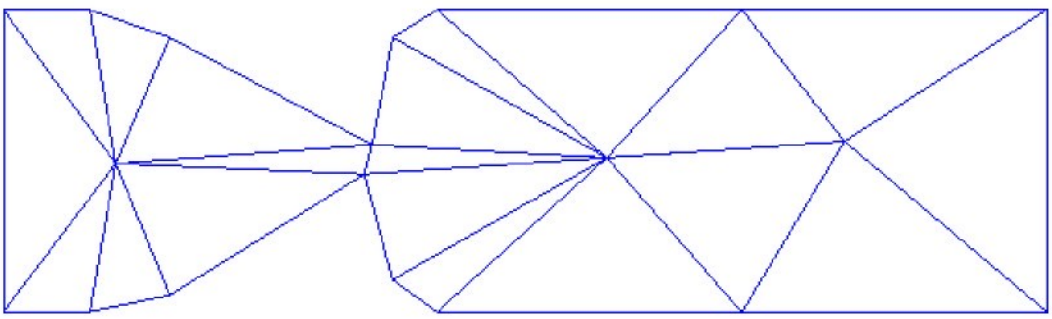


Obr. 4: Diskretizace složitější oblasti [4]

2.3 Modelování

Jak jsem již naznačil výše, tak v dnešní době zažívá modelování obrovský rozvoj. Ulehčuje nám práci, šetří čas i peníze a to je v dnešní uspěchané době to nejdůležitější. Často se říká, že čas jsou peníze. Schopnost virtuálně si vytvořit reálný problém se nazývá modelování. Je to široký pojem, který zasahuje do mnoha vědeckých i průmyslových oblastí. Například automobilový průmysl je na modelování velmi závislý. Automobil vzniká jako stavebnice malých modelů, ať už fyzických nebo virtuálních. Na modelech se podrobně testují úskalí a výdrže jednotlivých částí před finálním uvedením výrobku do prodeje.

Naše problematika se bude zabývat počítačovým modelováním využívajícím právě metodu konečných prvků. Jeho základní části jsou preprocessing, vlastní výpočet a postprocessing. V první části, která nese název preprocessing, se zabýváme popisem geometrie vlastního modelovaného objektu. Existuje na to i řada freewarových programových balíků. Geometrii se snažíme zachytit sítí, která se skládá z množiny elementů. V našem případě se jedná o trojúhelníky. Ovšem toto není pravidlo. Síť se může skládat z elementů různých tvarů. Pokud potřebujeme podrobné informace o všech místech modelovaného objektu, použijeme velmi jemnou síť s velkým množstvím elementů. Samozřejmě vše je poté více náročné na hardware i na kapacitu úložných prostorů počítače. Obrázek číslo pět je ilustrační znázornění, jak by mohla vypadat síť modelu lidských hlasivek složená z trojúhelníků. Pro vlastní výpočet je síť díky své hrubosti absolutně nepoužitelná.



Obr. 5: Velmi hrubá síť modelu lidských hlasivek

Pokud máme vytvořenou síť popisující modelovaný objekt, dostaneme soubor s tímto popisem a v další fázi se snažíme v jednotlivých uzlech sítě vypočítat konkrétní hodnoty vektoru (rychlosti) nebo skaláru (tlaku). To vše na základním principu metody konečných prvků. Nyní máme textový soubor se sítí popisující geometrii a soubor

s hodnotami konkrétních veličin. Aby to vše k něčemu bylo, musíme dosažené výsledky zobrazit. I na to v dnešní počítačové době existuje řada volně dostupných programů. Tyto programy jsou velmi propracované a mají nesmírnou výhodu, že jsou zdarma. Pokud již vidíme v grafické formě podobu našich výsledků, je nám jasné, kde je součást více namáhána nebo jakou rychlosti proudí vzduch kolem nějaké překážky atd... Modelování jako celek skýtá celou řadu různých úskalí. Já se v této práci zaměřím na problémy spjaté s vizualizací. Jako příklad uvedu nekompatibilitu textových souborů, což byl asi hlavní problém, který jsme museli vyřešit.

3 Vizualizace

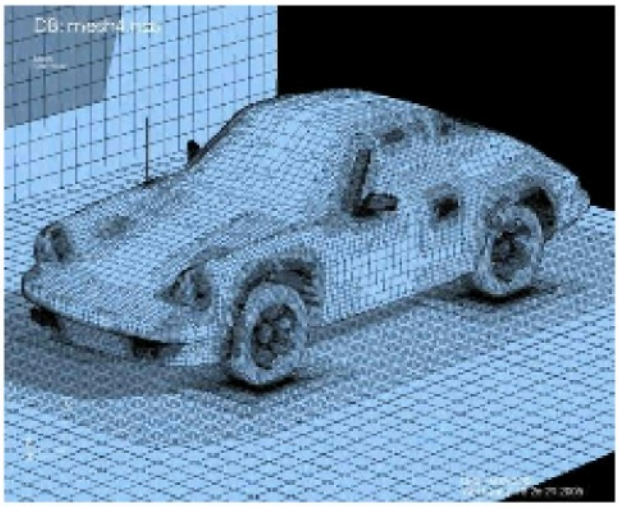
3.1 Úvod

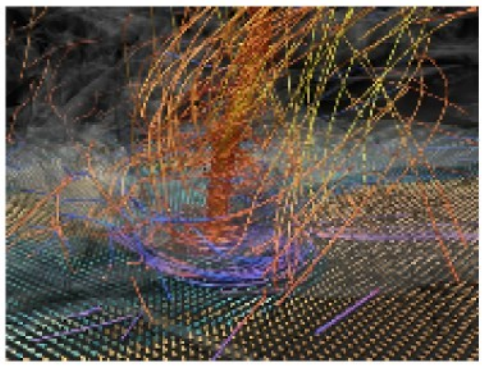
Vizualizace je využívána v mnoha vědeckých i průmyslových odvětvích. Hlavní důraz je kladen na zobrazování třídimenzionálních prvků, které se týkají například architektury, biomedicíny, atd... Soustřeďuje se na reálné vykreslení povrchů, objemů, samozřejmě se zachycením přírodních dějů, proudění větru, vody, atd... Ke zobrazení výsledků se využívá počítačové grafiky. Zobrazené výsledky nám umožňují porozumět dané problematice. Výsledky jsou reprezentací velkého množství numerických dat, které jsou získány buď měřením, nebo výpočtem z předchozích fází modelování. [5]

3.2 Historie

Obecně bychom mohli říci, že vizualizace je stará jako věda sama. Archimédes byl zabit, když vizuálně zobrazoval geometrické tvary do písku. Ve středověku již existovaly jednoduché astronomické vizuální modely, jako například proudění větru nad oceánem. Co se týče grafické vizualizace na počítačích, tak počátek byl v osmdesátých letech minulého století. Zpočátku byl vývoj velmi drahý a náročný. Náročný na software, který si každý dle svých potřeb musel vymyslet sám doma a drahý co se týče techniky. Kolem roku 1985 byl průlom z hlediska spojení dvou důležitých částí. Jedna skupina lidí se starala o algoritmizaci a druhá o příjemné uživatelské prostředí. Počítače otevřely nové možnosti výzkumu. Již byla možnost zpracovávat větší množství dat za krátkou časovou jednotku. V roce 1987 shrnul podstatu vizualizace McCormick. Ve zkratce to znamenalo, že pohled na obrázky je jasnější, jelikož 50% mozkových neuronů se soustředí na vidění. Kdežto pohled na hromadu vypočtených dat je velmi složitý na pochopení. Jednodušší je i následná výměna výsledků mezi různými pracovišti. Tato zpráva ujasnila, že ve vizualizaci je budoucnost

a dopomohla k jednotnému vývoji počítačové grafiky. V devadesátých letech docházelo k vývoji počítačů a nástrojů pro vizualizaci. Byly zaznamenány obrovské pokroky. Počátkem devadesátých let Daniel Thalmann poukázal na nový směr v numerické simulaci, který se soustředil na základní geometrii, animaci a vykreslování, stejně tak jako na konkrétní aplikace ve vědě. V nedávné době se uskutečnila konference SIGGRAPH (Special Interest Group on GRAPHics and Interactive Techniques), speciálně zaměřená skupina na grafické a interaktivní techniky. Tato konference se koná každý rok. Jsou zde představeny různé programové balíky na vizualizaci. Objasňují se zde základní problémy a pojmy zobrazování ve 2D a 3D (například barevná transformace, isočáry, atd…). Nyní se začíná odlišovat také vědecká a informační vizualizace. Vědecká se zabývá spojitými problémy v čase a prostoru, kdežto informační je čistě diskrétní. [5]





Obr. 6: Vizualizace tornáda [6]

Obr. 7: Vizualizace automobilu Porsche 911 [5]

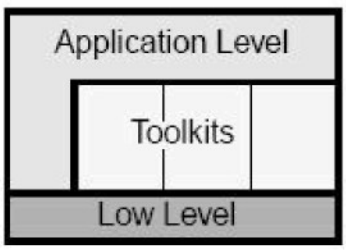
4 VTK

4.1 Úvod a historie

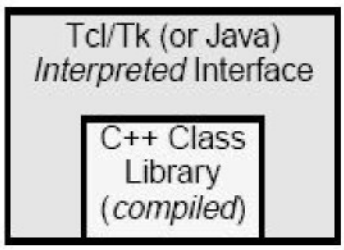
VTK (Visualization Toolkit) je volně dostupný open source systém pro 3D počítačovou grafiku a modelování, objemové vykreslování, získávání informací z vizualizací, atd… Jedná se o objektový toolkit, jako je například C++. Tudíž je potřeba dalších uživatelů, či vývojářů, kteří VTK využijí ve svých aplikacích. VTK je celosvětově využíváno ve spoustě aplikacích zaměřených na vizualizaci, kterými jsou například ParaView, VisIt, Slicer, OsiriX, atd… VTK bylo původně popsáno v knize od Willa Schroedera, Kena Martina a Billa Lorensena [9]. Tito tři vývojáři a grafici začali psát knihu a doprovodný software ve svém volném čase

v prosinci 1993. Hlavní motivací pro vznik této knihy byla spolupráce s ostatními vývojáři a konečné vytvoření otevřeného balíku, který by tvořil hlavní základ pro grafické a vizualizační aplikace. Poté, co bylo napsáno jádro VTK, začali uživatelé i vývojáři s vylepšováním a vytvářením aplikací. Také uváděli VTK do reálné problematiky. Stovky vývojářů považují VTK za přední systém pro vizualizaci ve světě.

V dnešní době se objevily dva nové směry v počítačovém průmyslu, a to objektově orientované programování a užití více komplexních metod v uživatelském rozhraní, jako je počítačová grafika. Objektově orientované systémy se lépe udržují a je zde možnost znovupoužití některých softwarových komponent. Počítačová grafika nám vytvoří okno, ve kterém si můžeme tvořit vlastní virtuální svět a rychleji porozumíme dané problematice. Pokud tyto dva směry spojíme, dostaneme hlavní motor počítačového průmyslu ve dvacátém prvním století. Počítačová 3D grafika se již dostala do hlavního proudu dnešních PC. Klasickým případem jsou počítačové hry, se kterými se setkal snad každý. V ideálním případě by se mělo přistoupit k objektově orientovanému programování a s ním spjatém vytváření toolkitů, na které by poté vývojář snadno postavil komplexní aplikaci pouhým poskládáním částí, které jsou již naprogramovány v toolkitu. Architektura toolkitu a aplikací, které ji využívají, je vidět na obrázcích osm a devět. I zde je vidět, že ukázat něco na jednoduchém obrázku je vypovídající jako stránka textu. [7] [8]



Obr. 8: Postavení aplikace pomocí toolkitu [8]



Obr. 9: Architektura toolkitu [8]

Pro tuto bakalářskou práci nebudeme programovat vlastní toolkit, nebo svou aplikaci využívající prvky z toolkitu. V úvodu jsem to zmínil pouze pro to, abychom dostali rámcový přehled, co je to VTK. Nám plně stačilo detailnější prozkoumání formátu souborů VTK, což popíši v kapitole 4.2.

4.2 Formát souborů VTK

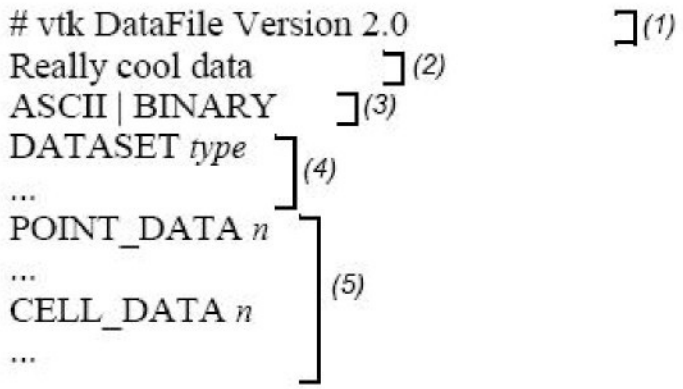
Hlavním důvodem vzniku vlastního formátu souboru *.vtk bylo vytvoření jednotného schéma pro různě popsaná data. Vznikla tím nesporná výhoda jednoduché metody pro komunikaci mezi daty a softwarem. Je dobré používat formát souboru, který je široce využitelný, tudíž vzniká obrovská možnost využití dat.

Existují dva základní typy souboru. Základní, který je jednoduchý pro ruční úpravy, čtení, zapisování a programování. Pokud si nevystačíme s tímto jednoduchým typem, tak existuje druhý XML typ, který poskytuje více možností komunikace, ale je také složitější. My si bohatě vystačíme se standardním formátem.

Jednoduchý VTK formát má pět základních částí:

1. Typ souboru a verze VTK, která musí být přesně definována pro komunikaci a kompatibilitu se softwarem.
2. Hlavičku, kam můžeme napsat co chceme. Slouží pro naši orientaci, co následující data popisují. Její délka musí být maximálně 256 znaků a je typu string.
3. Zde popisujeme formát souboru. Máme dvě možnosti ASCII nebo binární. Tento řádek musí obsahovat jednu z možností.
4. V další části následuje popis geometrie objektu. Nejprve je nutné definovat datový set (strukturovaná, nestrukturovaná síť, atd..). Poté následují souřadnice bodů, spojované body a tvar, jež spojované body představují.
5. Poslední část popisuje konkrétní datové hodnoty pro jednotlivé body, či buňky. Může se jednat o vektory, skaláry, tensory, atd…

První tři části jsou povinné, zbylé dvě jsou volitelné a je z nich možné tvořit různé kombinace. Klíčová slova jsou citlivá na velká, či malá písmena a měla by být oddělena mezerou. Na obrázku 10 je vidět popsaná základní struktura souboru vtk.

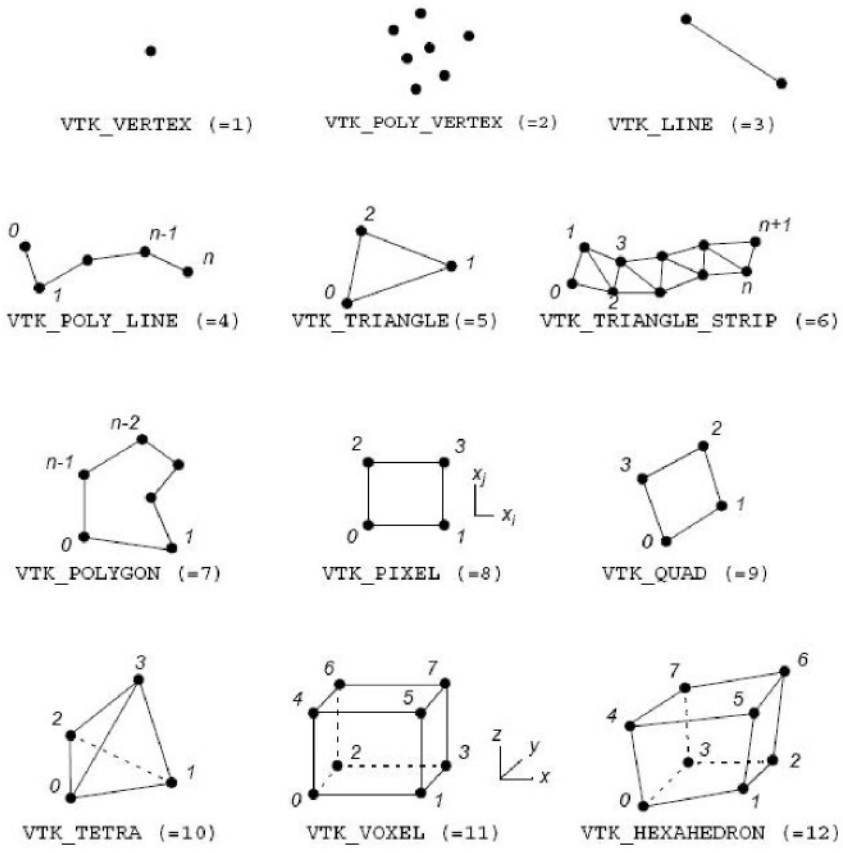


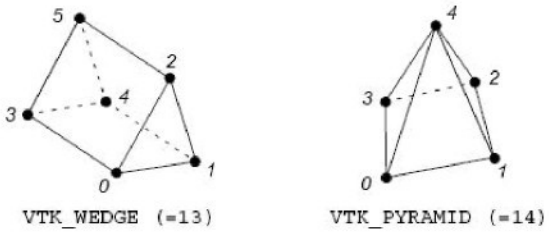
Obr. 10: Základní části VTK souboru [10]

Pár důležitých poznámek:

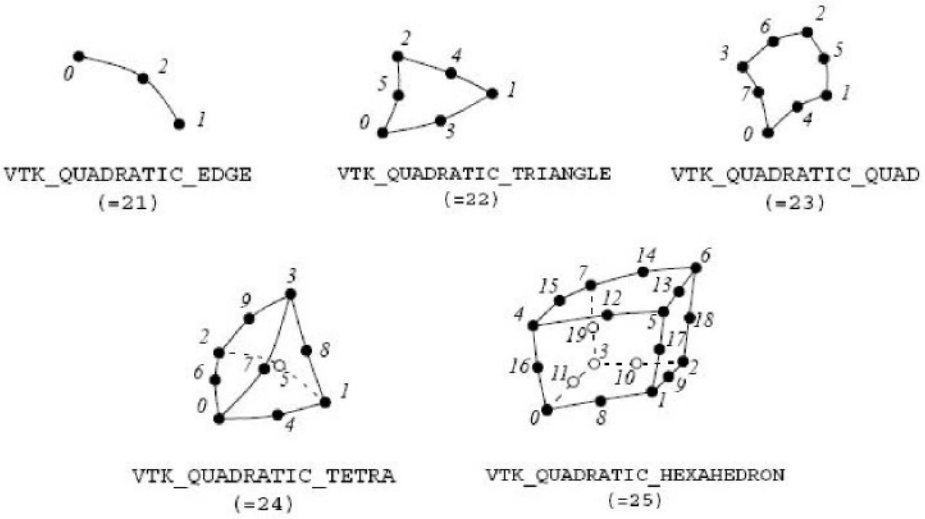
- Všechna klíčová slova nebo fráze se píší v ASCII, i když se může jednat o binární soubor. V tomto případě se píší binárně pouze data.
- Indexace bodů začíná od nuly.
- Pokud se v souboru dohromady objeví část popisující geometrii i data veličin bodů, či buněk, tak se musí rovnat počet bodů a počet hodnot veličin v příslušných bodech.
- Typy buněk a ukazatelé jsou typu integer.
- Binární data musí následovat bezprostředně na novém řádku po klíčových slovech.
- Popis geometrie musí být před hodnotami veličin v příslušných bodech, či buňkách.

Na následujících dvou obrázcích jsou všechny typy buněk, které soubor VTK podporuje. Jednotlivá čísla reprezentují typ buňky v souboru. V našem případě máme trojúhelníkovou síť, tudíž se v části typu buněk vyskytuje číslo pět. [10]





Obr. 11: Lineární typy buněk [10]



Obr. 12: Nelineární typy buněk [10]

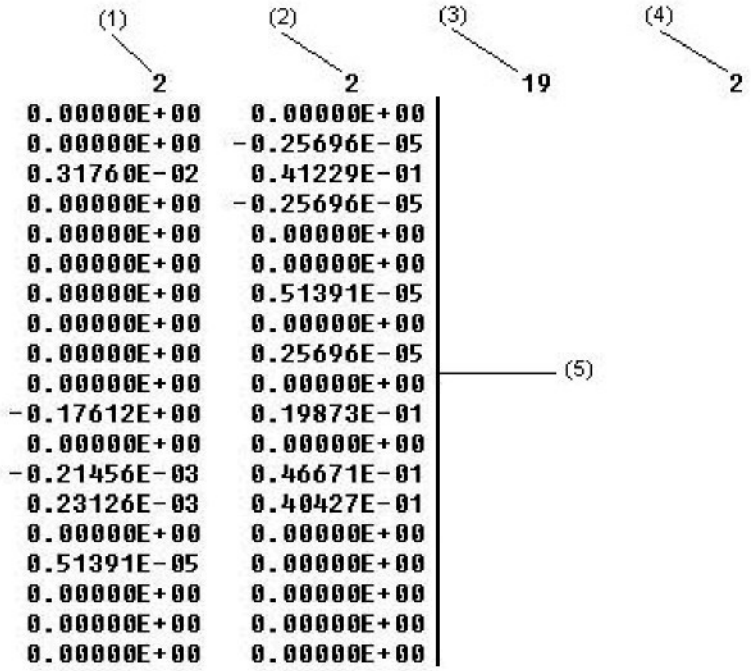
5 Konvertor

5.1 Úvod

V této kapitole se budu věnovat významné části mé bakalářské práce a to tvorbě konvertoru. Jedná se o konvertor ze dvou typů souborů s příponami BB a MESH do jednotného standardního formátu VTK. Důvod pro vytvoření konvertoru byl jednoduchý. VTK je univerzální a hojně využívaný formát, tudíž pro zpracování tohoto formátu je řada propracovaných softwarů, které nám umožní vyhodnotit data. Jeden z freewarových programových balíků je Paraview, který popíši v kapitole 6. Soubory s příponou BB a MESH jsou vytvářeny programem Medit, který není moc rozšířen a používá ho pouze pár vědeckých skupin. Dokumentace k tomuto programu je dostupná pouze ve francouzštině.

5.2 Soubor s příponou BB

Tento typ souborů popisuje hodnoty sledované veličiny v příslušných bodech sítě. V tomto konkrétním případě se jedná o vektory rychlosti proudění vzduchu v lidských hlasivkách.



Obr. 13: Soubor *.bb

Základní části formátu souborů BB:

1. Dimenze sítě
2. Dimenze řešených dat
3. Počet vrcholů sítě
4. Typ asociace
5. Hodnoty zkoumané veličiny

Na obr. 13 je popsáno devatenáct hodnot dvourozměrného vektoru rychlosti proudění vzduchu.

5.3 Soubor s příponou MESH

V souborech s příponou MESH se nachází popis geometrie konkrétního modelu. My si opět ukážeme formát souboru na případě modelu lidských hlasivek.

Obr.14 zachycuje popsanou geometrii dvourozměrného modelu lidských hlasivek, která se skládá z devatenácti vrcholů sítě a dvaceti trojúhelníkových elementů. Jedná

se o velmi hrubou síť, sloužící pouze pro popsání formátu souboru. Jednotlivé vrcholy mají indexy, které začínají od jedničky na rozdíl od formátu VTK, kde začínají indexem nula. Můžeme si všimnout, že pomocí formátu MESH lze vytvořit trojrozměrný model.

```
MeshVersionFormatted 1                                  (1)
Dimension            2                                  (2)
Vertices            19                                  (3)
 -0.25000E+00  -0.43482E+00  1
  0.00000E+00  -0.43482E+00  1
  0.79091E+00  -0.38474E-01  1
  0.10000E+01  -0.43482E+00  1
  0.27500E+01  -0.43482E+00  1
  0.27500E+01   0.43482E+00  1
  0.10000E+01   0.43482E+00  1
  0.81287E+00   0.46393E-01  1
  0.00000E+00   0.43482E+00  1                         (4)
 -0.25000E+00   0.43482E+00  1
  0.74117E-01  -0.78670E-02  1
  0.14874E+01   0.88437E-02  1
  0.23017E+00  -0.38704E+00  1
  0.87210E+00  -0.34262E+00  1
  0.22940E+00   0.35542E+00  1
  0.87287E+00   0.35542E+00  1
  0.18750E+01  -0.43482E+00  1
  0.21694E+01   0.55720E-01  1
  0.18750E+01   0.43482E+00  1
Triangles            20                                 (5)
      1       2       11       1
      2      13       11       1
      1      11       10       1
     13       3       11       1
      9      10       11       1
      3       8       11       1
     15       9       11       1
      3      12        8       1
      8      15       11       1                        (6)
      3      14       12       1
     16       8       12       1
     14       4       12       1
      7      16       12       1
      4      17       12       1
     19       7       12       1
     17      18       12       1
     18      19       12       1
     17       5       18       1
      6      19       18       1
     18       5        6       1
End
```

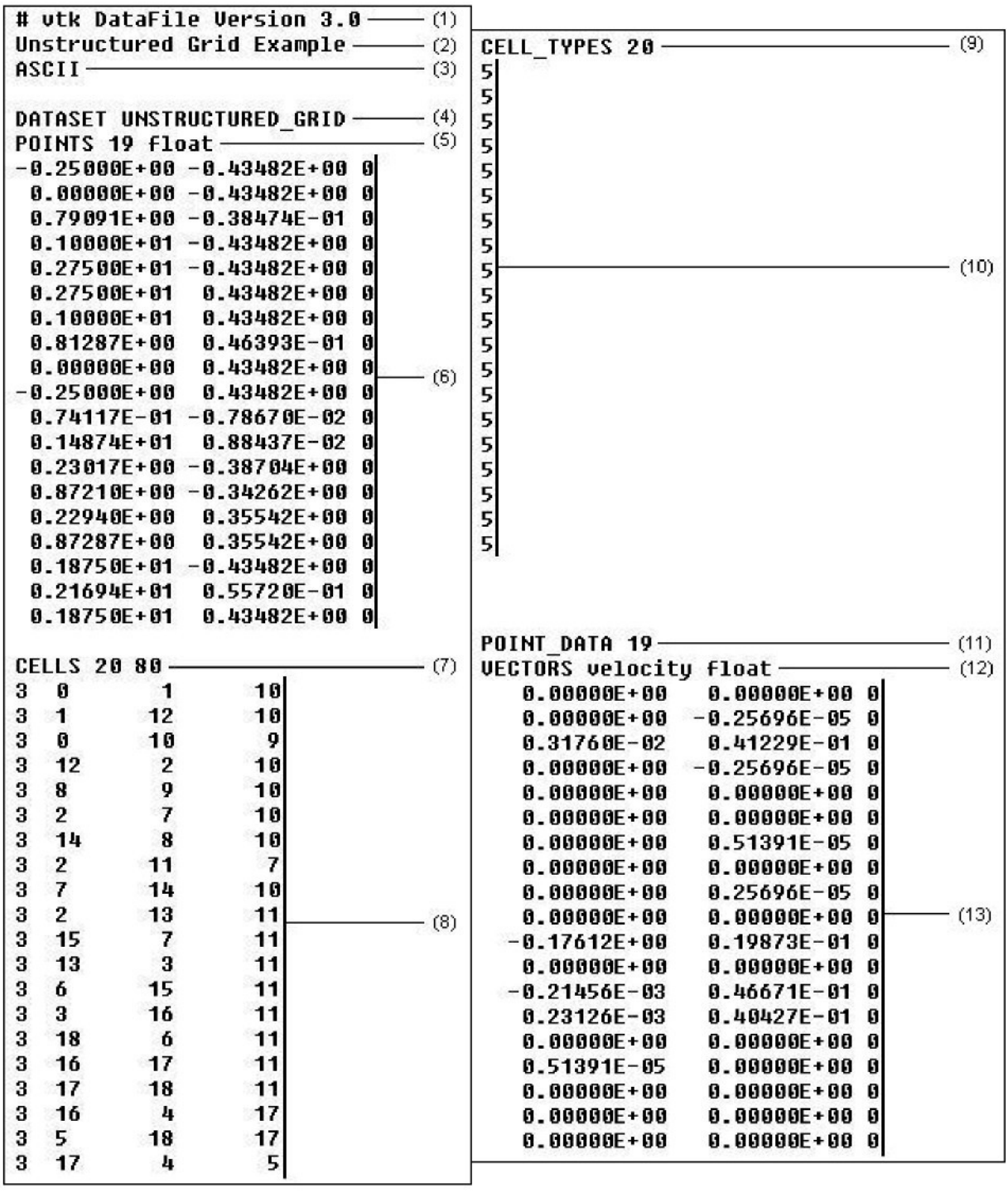
Obr. 14: Soubor *.mesh

Základní části formátu souboru MESH:

1. Verze souboru
2. Dimenze sítě

3. Počet vrcholů sítě
4. Souřadnice vrcholů sítě
5. Typ elementů sítě a jejich počet
6. Spojované vrcholy sítě nutné k vytvoření elementů

5.4 Soubor s příponou VTK




```
# vtk DataFile Version 3.0
Unstructured Grid Example
ASCII

DATASET UNSTRUCTURED_GRID
POINTS 19 float
-0.25000E+00 -0.43482E+00 0
 0.00000E+00 -0.43482E+00 0
 0.79091E+00 -0.38474E-01 0
 0.10000E+01 -0.43482E+00 0
 0.27500E+01 -0.43482E+00 0
 0.27500E+01  0.43482E+00 0
 0.10000E+01  0.43482E+00 0
 0.81287E+00  0.46393E-01 0
 0.00000E+00  0.43482E+00 0
-0.25000E+00  0.43482E+00 0
 0.74117E-01 -0.78670E-02 0
 0.14874E+01  0.88437E-02 0
 0.23017E+00 -0.38704E+00 0
 0.87210E+00 -0.34262E+00 0
 0.22940E+00  0.35542E+00 0
 0.87287E+00  0.35542E+00 0
 0.18750E+01 -0.43482E+00 0
 0.21694E+01  0.55720E-01 0
 0.18750E+01  0.43482E+00 0

CELLS 20 80
3  0      1      10
3  1      12     10
3  0      10      9
3  12     2      10
3  8      9      10
3  2      7      10
3  14     8      10
3  2      11      7
3  7      14     10
3  2      13     11
3  15     7      11
3  13     3      11
3  6      15     11
3  3      16     11
3  18     6      11
3  16     17     11
3  17     18     11
3  16     4      17
3  5      18     17
3  17     4       5

CELL_TYPES 20
5
5
5
5
5
5
5
5
5
5
5
5
5
5
5
5
5
5
5
5

POINT_DATA 19
VECTORS velocity float
   0.00000E+00   0.00000E+00 0
   0.00000E+00  -0.25696E-05 0
   0.31760E-02   0.41229E-01 0
   0.00000E+00  -0.25696E-05 0
   0.00000E+00   0.00000E+00 0
   0.00000E+00   0.00000E+00 0
   0.00000E+00   0.51391E-05 0
   0.00000E+00   0.00000E+00 0
   0.00000E+00   0.25696E-05 0
   0.00000E+00   0.00000E+00 0
  -0.17612E+00   0.19873E-01 0
   0.00000E+00   0.00000E+00 0
  -0.21456E-03   0.46671E-01 0
   0.23126E-03   0.40427E-01 0
   0.00000E+00   0.00000E+00 0
   0.51391E-05   0.00000E+00 0
   0.00000E+00   0.00000E+00 0
   0.00000E+00   0.00000E+00 0
   0.00000E+00   0.00000E+00 0
```

Obr. 15: Soubor *.vtk

V kapitole 4 jsme se věnovali formátu souboru VTK podrobně. Nyní pouze zobrazíme náš konkrétní výsledek, který vznikl konverzí souborů příponami MESH a BB do formátu VTK. Data, která jsou vidět v kapitolách 5.2 a 5.3, by měla odpovídat datům na obr. 15.

Základní části formátu souboru VTK:

1. Verze souboru
2. Popis souboru
3. Typ souboru
4. Datový set a jeho typ
5. Počet vrcholů a jejich typ
6. Souřadnice vrcholů
7. Počet elementů a celkový počet číselných hodnot, kterými jsou definovány
8. Počet číselných hodnot nutných pro vytvoření příslušného elementu a spojované vrcholy sítě nutné k vytvoření elemetů
9. Počet elementů
10. Druh elementů
11. Počet hodnot zkoumané veličiny
12. Druh, název a typ zkoumané veličiny
13. Hodnoty zkoumané veličiny

Pokud slovně popíšeme to, co jsme získali pomocí našeho konvertoru v tomto případě, mohlo by to znít následovně. Geometrii modelu lidských hlasivek nám popisuje nestrukturovaná síť tvořená dvanácti vrcholy typu float. Jako element pro tvorbu sítě jsme zvolili trojúhelník, což odpovídá hodnotě pět. Pro naši síť jich potřebujeme dvacet a jak známo, k vytvoření trojúhelníku nám stačí tři body. Indexace vrcholů je zde od nuly. Ve vrcholech sítě nalezneme příslušné hodnoty dvourozměrného vektoru rychlosti typu float. V našem případě ještě budeme zkoumat skalární hodnoty tlaku. Tam je popis ještě jednodušší, jelikož nám stačí pouze jediná hodnota. Opět vidíme, že i formát VTK je kompatibilní s trojrozměrným prostorem. Samozřejmě obr.15 je orientační a je složen ze dvou částí, abychom uspořili nějaké místo. V reálném souboru jsou tyto části pod sebou.

5.5 Konvertor MESH_BB_TO_VTK

Celý program je napsán v programovacím jazyce FreePascal pomocí vývojového prostředí Lazarus, které je volně dostupné na internetu. Jedná se o objektově orientovaný jazyk. Lazarus je velmi podobná aplikace, jako je Delphi. Jedinou výhradu bych měl k nápovědě, jelikož Lazarus má k dispozici pouze online nápovědu a ve srovnání s Delphi není tak propracovaná.

Náš program lze spustit jak pod systémem Linux, tak pod Windows. Jedná se o konzolovou aplikaci, kterou je možno spustit z příkazového řádku.

Krátce popíši funkci programu. V první řadě musíme nalézt adresář obsahující soubory s příponami MESH a BB. Jakmile máme v příkazové řádce aktivní příslušný adresář, stačí program spustit pomocí příkazu meshbb2vtk. Po spuštění se v aktivním adresáři, ze kterého jsme program spustili, vytvoří automaticky adresář s názvem VTK, do kterého se kopírují výsledné soubory.

Program je rozdělen do procedur, které obsluhují jednotlivé významné části konvertoru. Důvodem k rozčlenění programu do procedur byla přehlednost. Hlavní tělo programu obsahuje pár řádků kódu, pomocí kterého voláme příslušné procedury. Dále jsme hodnoty, které se za běhu programu nemění, definovali jako konstanty a to především kvůli univerzálnosti programu. Pokud potřebujeme některou z konstant změnit, jednoduše ji přepíšeme na jednom místě v programu a už nemusíme řešit problém, jestli jsme ji přepsali ve všech místech programu, kde s ní pracujeme. Přesně to bývá častým zdrojem chyb.

Nyní se budu věnovat samotné konstrukci programu. V hlavním těle se řeší cesta do aktivního adresáře, vytvoření adresáře VTK a také se zde získávají názvy souborů s příponami MESH a BB. Program je univerzálně řešen tak, aby soubory mohly být v adresáři umístěny libovolně. Pokud program nenalezne oba požadované soubory (BB i MESH), nic nedělá a načítá další soubor BB, ke kterému se opět snaží nalézt soubor se stejným názvem, ale s příponou MESH. Pokud je vše splněno, je zavolána procedura `Konvertor`, které se předává jako parametr název souboru BB a MESH. V této proceduře dochází k definování konkrétních souborů. Připravují se zde pro čtení, či zápis. Pomocí chráněného bloku `try` spouštíme další tři procedury, které zapisují příslušná data do nově vytvořeného souboru VTK. V následujícím bloku `finally` zavíráme načtené soubory. Použití chráněného bloku `try` má tu výhodu, že pokud dojde k chybě při běhu programu, tak se program dokončí za použití definovaných

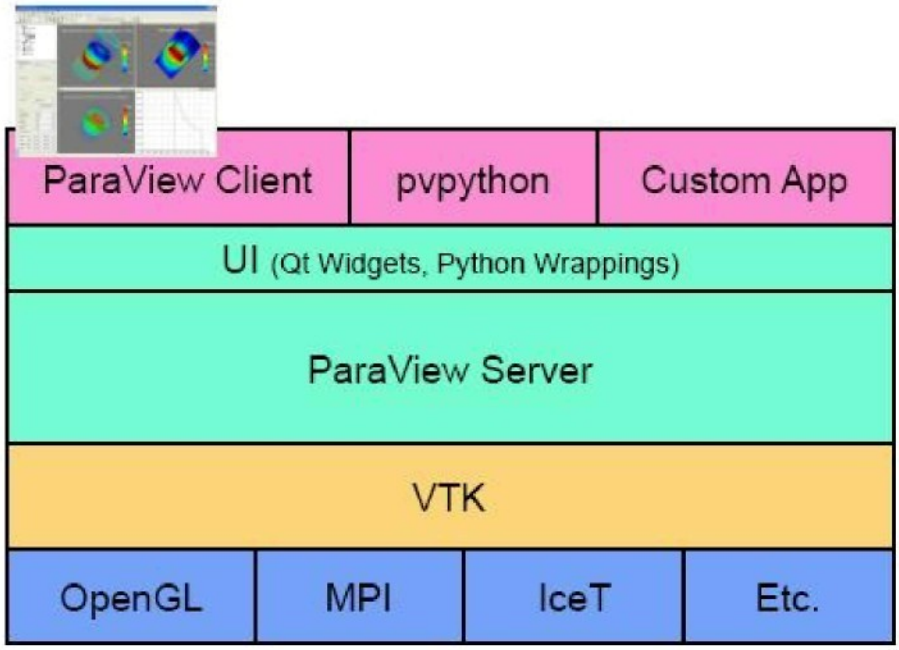
pravidel, které nalezneme v bloku *finally*. První spouštěnou procedurou je *hlavicka*, která nám zapíše do souboru VTK pouze úvodní hlavičku. V kapitole 5.4 jsou to body 1, 2, 3. Druhou spouštěnou procedurou je *zapis_vrcholy*. Tato procedura nedělá nic jiného, než že přepíše souřadnice vrcholů sítě ze souboru MESH do souboru VTK. Zde se řešil pouze problém s třetí souřadnicí, kterou jsme do souboru VTK zadávali nulovou, místo načtené jedničkové. Poslední volaná procedura se jmenuje *zapis_elementy*. Tato procedura je nejrozsáhlejší a vytvoří nám zbytek souboru. Definuje nám příslušné elementy, jejich typ a data vektoru rychlosti nebo skaláru tlaku v příslušných vrcholech sítě. Bylo nutné ošetřit indexaci spojovaných vrcholů, která je v souboru VTK od nuly a ne od jedničky jako v souboru MESH. Dopočítali jsme počet všech hodnot, kterými jsme definovali elementy sítě a před indexy spojovaných bodů každého elementu jsme doplnili jejich počet. V poslední části této procedury jsme museli definovat, jestli se jedná o vektorovou rychlost nebo skalární tlak. Jako rozhodovací parametr nám posloužila hodnota dimenze řešení, kterou jsme načetli ze souboru BB. Pokud se jednalo o vektor, tak jsme ho museli doplnit na trojrozměrný pomocí nulové třetí souřadnice.

6 ParaView

6.1 Úvod

ParaView je open-source aplikace pro vizualizaci 2D a 3D dat. ParaView umí zpracovávat široké spektrum dat. Od menších dat po velké detailní sítě. Většinou to nezáleží na softwaru, ale na hardwaru, který je schopen zpracovat tak velký datový soubor. Aplikace podporuje jednoprocesorové i víceprocesorové stanice či výpočetní clustery. Umožňuje také paralelní přístup k datům, což je důležité právě u rozsáhlých souborů. Pro představu pomocí ParaView byla zpracovávána síť o velikosti šesti biliónů strukturovaných buněk, dvě stě padesáti milionů nestrukturovaných buněk a jednoho bilionu buněk strukturované AMR (Adaptive Mesh Refinemet – přizpůsobivá jemnost) sítě. Mezi další výhody patří komerční podpora vývoje tohoto open-source programu. Jeho stavba je založena na modulární architektuře, která se velmi snadno rozšiřuje. PraView je možné spustit na platformě Linux i Windows. Mezi výhody patří bezesporu i příjemné uživatelské rozhraní. ParaView využívá mnoho akademických, vládních nebo komerčních institucí. Je to vidět i na počtu stažení, které činí kolem tří tisíc za měsíc. Na obrázku 16 je vidět základní architektura, na které je ParaView postaveno.

Jedná se pouze o drobnou aplikaci, která využívá mnoho funkcí z různých knihoven, které jsou naprogramovány ve vrstvách pod tímto programem. V podstatě se jedná ografické prostředí soustřeďující možnosti, které nám knihovny nabízejí do jednoho místa a to je pro uživatele příjemné.



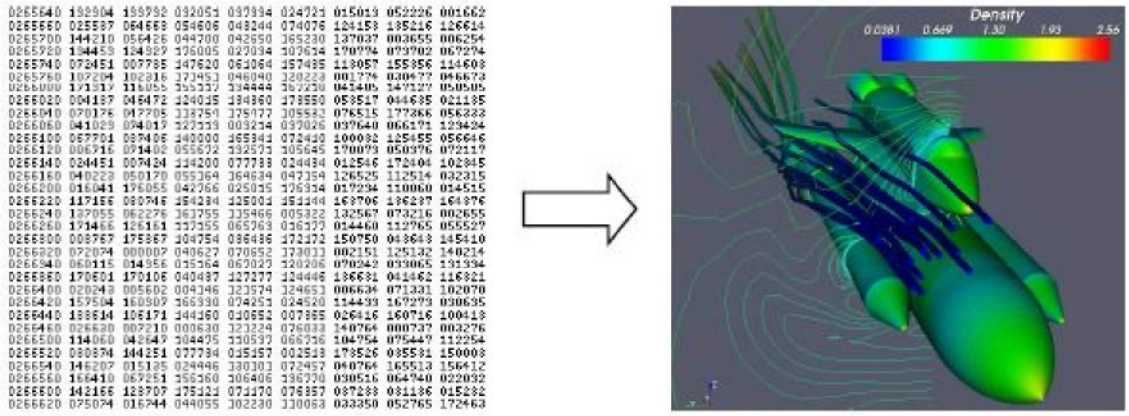
Obr. 16: Architektura, na níž je postaveno ParaView [11]

Například ParaView Server se stará především o paralelní zpracování dat. Bez této vrstvy by aplikace neměla možnost zpracovávat rozsáhlejší data. Vrstvu VTK jsem popsal v kapitole 4. Ve zkratce řečeno obsahuje algoritmy pro vykreslování a vizualizaci dat.

Jednoduše řečeno tak ParaView zajišťuje vizuální podobu dat. Názorně to je vidět na obrázku 17. Můžeme mít 2D nebo 3D data, které mohou vzniknout jako výsledky různých simulací. ParaView pracuje ve třech základních krocích.

- Načtení dat
- Filtrování dat
- Vykreslení dat

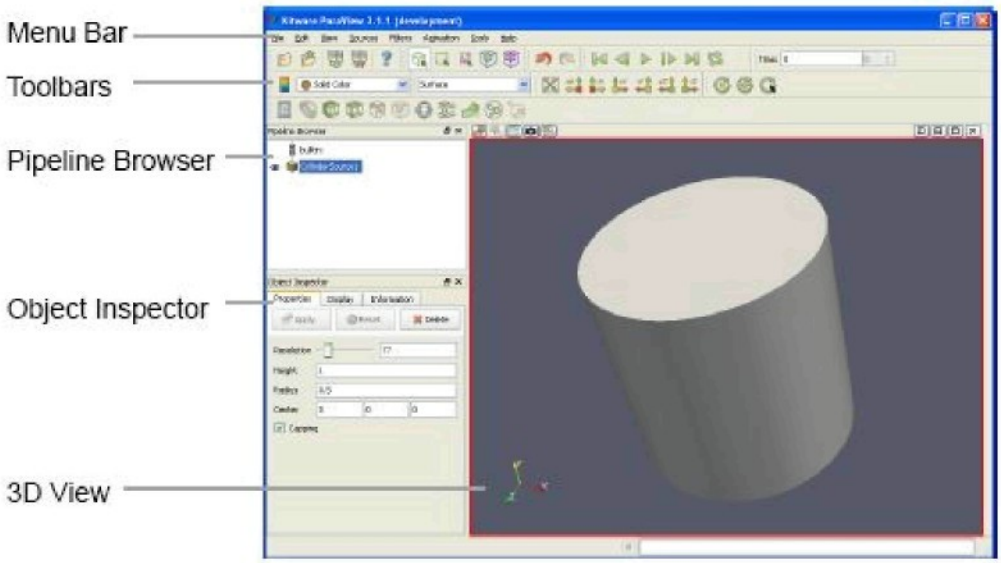
Nejprve potřebujeme data načíst, poté z nich pomocí filtrů získat určité vlastnosti a v konečné řadě je vykreslit. Základním datovým typem, se kterým ParaView pracuje, jsou sítě. [11]



Obr. 17: Hlavní princip vizualizace [11]

6.2 Uživatelské rozhraní

Vzhled uživatelského rozhraní se může drobně lišit pro různé systémy, na kterých je ParaView spuštěno.



Obr. 18: Grafické uživatelské rozhraní programu ParaView [11]

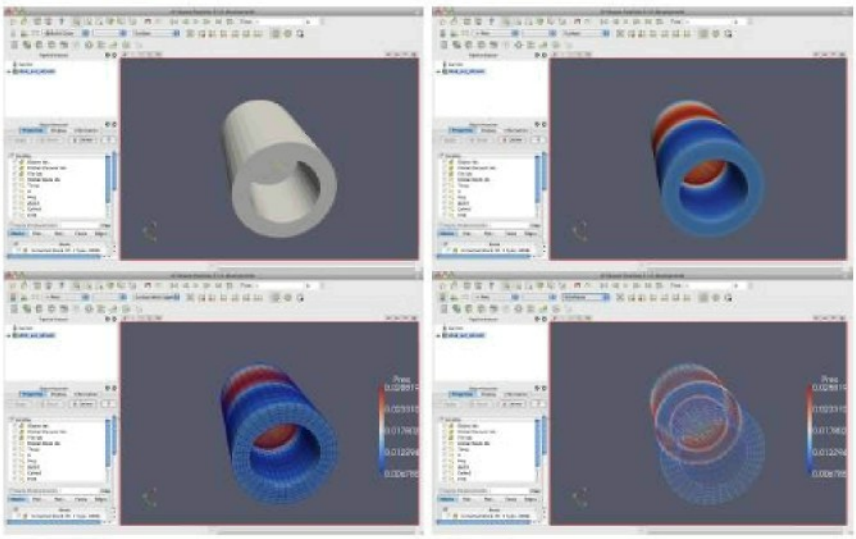
Na obr. 18 je vidět, že se uživatelské rozhraní skládá z pěti základních částí. Pomocí **menu** se dostaneme k většině funkcí, které nám aplikace poskytuje. **Toolbar** zajišťuje rychlý přístup k nejpoužívanějším funkcím. V **Pipeline Browseru** vidíme otevřené zdroje a použité filtry. Je velmi jednoduché změnit nastavení jednotlivých filtrů. Můžeme je zakázat nebo zjistit, co konkrétně daný filtr zobrazuje. Jednoduše zjistíme, na která data je filtr použit. Další částí je **Object Inspector**, kterým můžeme upravovat zvolenou položku z Pipeline Browseru. Obsahuje tři záložky

(Display, Properties, Information). Pomocí properties nastavujeme vlastnosti objektu. Display zajišťuje vzhledovou stránku při vykreslování a information nám poskytuje souhrnné informace o zvoleném objektu. Poslední položkou je okno pro samotné vykreslení zvoleného objektu (**3D View**). Vzhled ParaView je možné jednoduše měnit. Pipeline Browser i Object Inspector jsou dokovací okna, kterými je možné libovolně pohybovat. [11]

6.3 Zdroje dat

Do ParaView je možné nahrát data dvěma způsoby. Prvním z nich je načíst předem připravená data ze souboru. Druhou možností je použít předdefinovaná data v softwaru. Pomocí nich je možné jednoduše vysvětlit ovládání ParaView. Po načtení válce není vidět krásný hladký válec, ale válec, který se skládá z plošek (defaultně šest). Pokud zvětšíme rozlišovací schopnost na více plošek, ze kterých se válec skládá, tak můžeme dostat krásný válec, na první pohled hladký. Pomocí myši nebo klávesnice je možné s válcem pohybovat, různě ho otáčet a přibližovat ho. Je možné se jednoduše vracet o krok zpět a to jak v akci, které provedeme, tak i v pohledu kamery.

Pokud se rozhodneme načíst data ze souboru, tak je dobré vědět, které typy souborů ParaView podporuje (*.pvd, *.vtp, *.vtu, *.vti, *.vts, *.vtr, *.vtm, *.vtmb, *.vtmg, *.vthd, *.vthb, *.pvtu, *.pvti, *.pvts, *.pvtr, *.vtk, *.xmf, *.xdmf, *.case, *.sos, *.g, *.pdb, *.dem, *.wrl, *.stl, *.cube, *.inp, *.mhd, *.mha, *.pht, *.csv, *.ex2). Další funkce programu ukážu na demonstrativním souboru, který je dostupný na stránkách ParaView. Jedná se o dutý válec, který ovšem není dutý v celé své délce. Uvnitř válce rotuje disk a tře se o dno a stěny. Po načtení dat nám ParaView nabízí několik možností vykreslení. Na obr. 19 vidíme základní možnosti vykreslení dat. Na prvním je geometrie válce v jednolité barvě. Toto slouží pro dokonalejší představu tvaru zkoumaného objektu. Pokud chceme vidět průběh nějaké veličiny, tak si ji musíme vybrat. Na druhém obrázku je zvolen tlak. Stále jsme nechali pohled neprůhledné plné geometrie (**Surface**). Červená barva na obrázku představuje nejvyšší hodnoty tlaku. Na dalším obrázku vidíme pohled na plnou geometrii se zobrazenou sítí (**Surface With Edges**). Poslední obrázek znázorňuje zobrazení geometrie pomocí sítě, kterou je popsána (**Wireframe**). Dobrou pomůckou je to, že zbarvení, které odpovídá jednotlivým tlakům, zůstává a usnadňuje nám orientaci v síti. [11]



Obr. 19: Základní zobrazení geometrie dutého válce s rotujícím diskem uvnitř [11]

6.4 Filtry

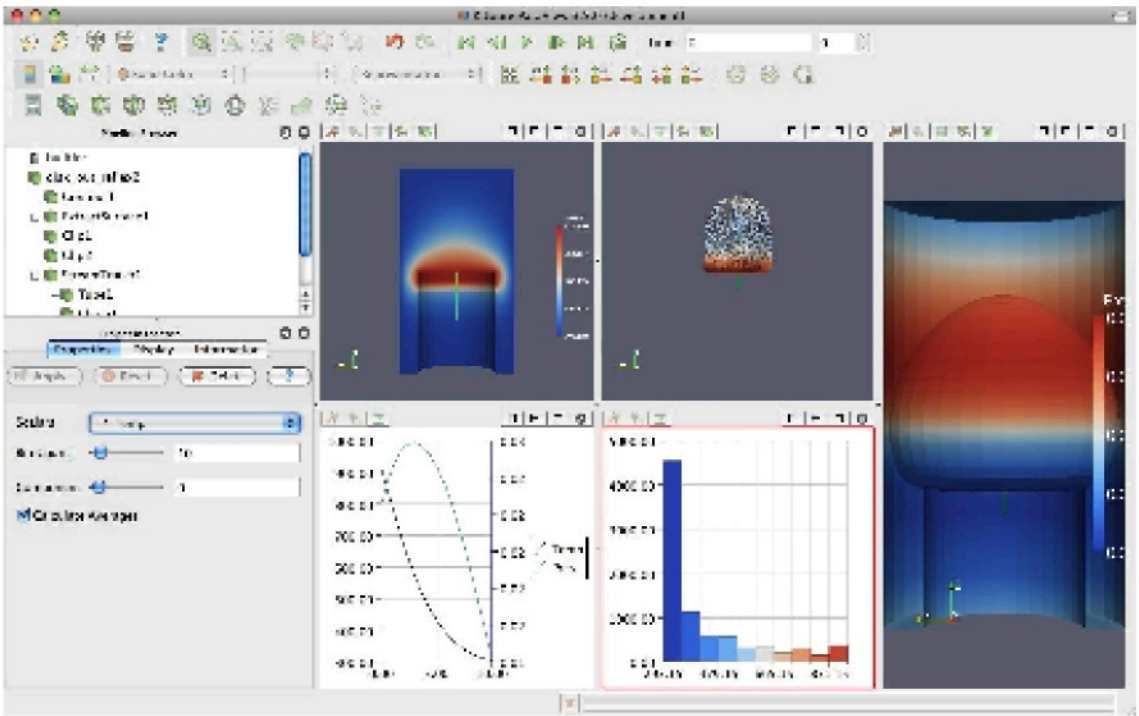
Nyní již umíme data zobrazit a vyčíst z nich základní informace. To ovšem ve většině případů nestačí, proto zde máme spousty filtrů, které nám umožňují získat podrobnější informace. Pomocí filtrů můžeme určit vlastnosti uvnitř materiálu, které bývají důležitější než povrchové informace. Nyní popíši několik základních filtrů, které naleznete v toolbaru. Filtr **Contour** nám vyřízne plochu s požadovanou hodnotou. **Clip** umožňuje rozříznutí objektu s tím, že jedna část před ořezávací rovinou zůstane a druhá zmizí. **Slice** má podobnou funkci jako předchozí filtr, pouze s tím rozdílem, že po aplikaci filtru zůstane zobrazena pouze ořezávací rovina. **Treshold** zanechá pouze buňky s námi definovanou hodnotou. Aplikací filtru **Glyph** dojde ke zobrazení jednoduchého tvaru v každém bodě sítě. Velikost tvaru poukazuje na velikost hodnoty veličiny. Směr, kterým jednoduchý tvar ukazuje, odpovídá směru působení veličiny. Pokud se jedná o vektorovou veličinu. **Stream Tracer** nahradí vektorové pole body, které následně spojí křivkou. Po aplikaci filtru Stream Tracer dostaneme 1D křivky, ale pro lepší přehled doporučujeme aplikovat ještě filtr **Tube**, který použijeme na předchozí filtr. Tím získáme 2D křivky. Samozřejmě toto je jen zlomek filtrů, se kterými ParaView dokáže pracovat. Dalším příkladem jsou filtry pracující s časem, zobrazování průběhu zvolené veličiny v daném bodě, či podél zvolené přímky.

Velmi důležitou částí postprocessingu je vyhodnocení dat formou grafů. Ukazují nám přehledně průběh námi zvolené veličiny. V ParaView na to samozřejmě existuje řada filtrů. Pro ukázku vybereme filtr Plot Data Over Line. Jak již název napovídá, zobrazí nám data podél úsečky, kterou si můžeme zvolit podle našich potřeb. Filtr

nám v grafu zobrazí všechny veličiny, které jsou v daných datech dostupné, ale není problém si jednoduše vybrat pouze ty, které nás zajímají.

ParaView také umožňuje otevřít více nezávislých pohledů v jedné aplikaci. Mezi jednotlivými okny je možné provázat kameru tak, že při pohybu s jedním objektem se ostatní budou natáčet stejně a my je tak budeme moci lépe porovnávat. Další možnost, která nám zjednoduší práci, jsou tlačítka, která zarovnají zvolený objekt podle jednotlivých rovin, nebo ho roztáhnou na maximální užitečnou plochu okna.

Na obr. 20 vidíme konkrétní aplikaci popsaných filtrů. Ve spojnicovém grafu jsme si zvolili průběh teploty a tlaku podél dané úsečky. Na ose y jsou hodnoty tlaku a teploty. V případě, že bychom nechali hodnoty na jedné ose, tak bychom z tlaku příliš nevyčetli, jelikož jeho hodnoty se pohybují v setinách. Kdežto u teploty jsou to stovky kelvinů. Teplotu jsme ponechali na ose y vlevo, ale tlak jsme posunuli na druhou stranu grafu. Nyní jsou obě osy nezávislé a grafy se dají krásně porovnat i v případě, že se rozmezí hodnot velmi liší. Na vrchní části obrázku vidíme aplikaci filtru Clip, který nám válec rozřízl napůl. Na pravém obrázku je použit filtr Contour, který nám vyřízl oblast odpovídající teplotě 400K. Uprostřed je aplikován filtr Stream Tracer včetně rozšíření o filtr Tube a Glyph. Filtr Glyph nám ukazuje vektorové pole rychlosti. Nyní můžeme říci více podrobností o našich datech. Například jakým směrem se disk točí, nebo kde je rychlost proudícího vzduchu největší. [11]

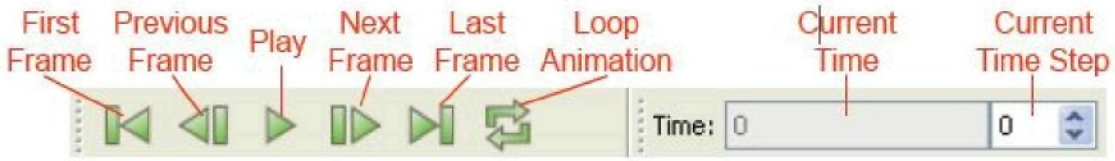


Obr. 20: Grafy a rozčlenění aplikace ParaView [11]

6.5 Časově proměnné vizualizace

Do teď jsme se zabývali pouze statickým modelem, ale to ve většině případů nestačí. ParaView je na toto připraveno a umožňuje pracovat s modely, které se v čase mění. Veškeré výsledky je možné ukládat buď jako sérii obrázků nebo přímo jako video. Pokud chceme získat lepší kvalitu, je lepší uložit si výsledky jako sérii obrázků a poté z nich vytvořit video.

Práce s časově proměnnými vizualizacemi je velmi jednoduchá. Nalezneme zde standardní ovládací prvky, na které narazíme v každém přehrávači videa, tudíž je ovládání velmi intuitivní a je zbytečné ho více popisovat.



Obr. 21: Ovládací panel pro práci s časově proměnnými vizualizacemi [11]

Při práci s časem bychom si měli dávat pozor na barevnou škálu u legendy popisující, která barva odpovídá určité hodnotě veličiny. Při spuštění vizualizace se objekt různě deformuje a s tím se mění i hodnoty zkoumaných veličin. Po skončení vizualizace je možné, že barevná škála v legendě nebude vůbec vypovídající o zobrazených konečných datech. Na první pohled to můžeme považovat za chybu, ale není tomu tak. Při zobrazení legendy se barevná škála načte pro maximální a minimální hodnotu zobrazené veličiny v příslušném čase. Pokud bychom měli větší data a měli bychom barevnou škálu přepočítávat pro každý krok v simulaci, bylo by to velmi náročné na hardware a opět by to zpomalovalo jiné podstatnější výpočty. Proto zde existuje funkce **Rescale Data Range**, která přepočítá odpovídající barevnou škálu pro legendu v zobrazeném čase. Rozsah barevné škály lze nastavit i ručně.

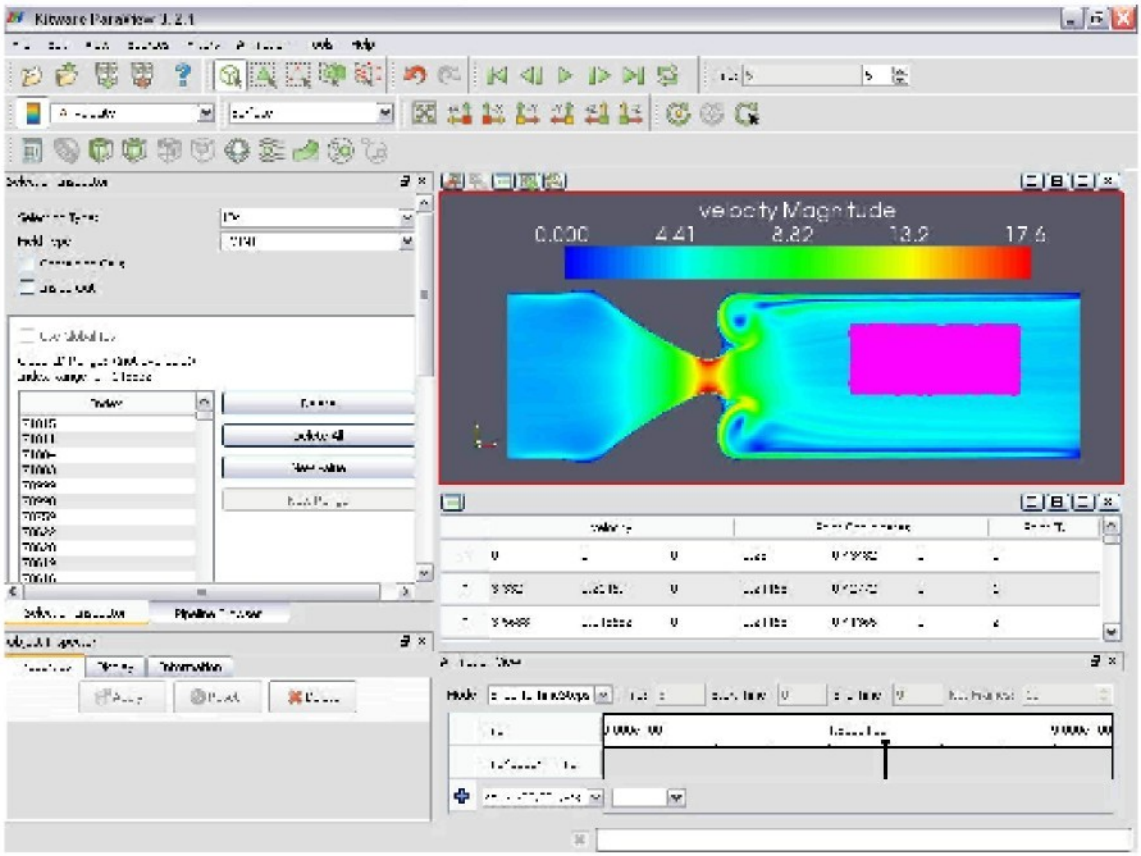
Další důležitou funkcí, kterou ParaView umí, je výběr konkrétní oblasti, kde nás zajímají podrobnosti. Existuje zde mnoho variant, jak vybrat oblast. Nejjednodušší je výběr pomocí myši. V ovládacím panelu si zvolíme požadovanou funkci výběru a poté jen myší označíme oblast, která nás zajímá. Nyní popíši základní typy výběru oblasti. Prvním je **Select Cells On**, který vybírá označené buňky (elementy) na povrchu modelu. Další velmi podobný druh výběru je **Select Points On**. Ten vybere označené body spojující jednotlivé elementy na povrchu modelu. Toto byly typy výběru na aktuálně viditelném povrchu. Další skupinou jsou typy výběru elementů

nebo bodů, které vyberou pod označenou oblastí veškeré prvky. To znamená i ty, které nejsou vidět. Toto se dá využít u 3D objektů, kdy nevidím skrz, ale pomocí této funkce mohu jednoduše vybrat všechna data, která se nacházejí pod označenou oblastí. V ParaView tyto funkce nalezneme pod názvy **Select Cells Through** a **Select Points Through**. Tyto funkce se dají opět najít v podobě ikon v ovládacích panelech pro rychlé spuštění. Další možností jak vybrat nebo upravovat vybraná data je otevření **Selection Inspectoru**. Nalezneme jej jako záložku pod Pipeline Browserem. Pomocí něj můžeme invertovat výběr dat (Invert Selection). Pokud chceme zobrazit název jednotlivých bodů nebo buněk pro lepší přehlednost, nalezneme to právě zde. Umožňuje nám i změnu viditelných vlastností vybrané oblasti (barva, tloušťka čar). Vidíme zde také seznam indexů vybraných bodů nebo elementů. Jsou zde dva základní druhy výběru. Pomocí komolého kužele (**Frustum**) je vybraná oblast na stejném místě v prostoru a její obsah se mění podle toho, které elementy nebo body právě vstupují nebo vystupují. Druhou možností výběru je pomocí **IDs**, tudíž pomocí indexů. Zde se vybraná oblast pohybuje s označenými body. Obsah vybrané oblasti se nemění. Významným ovládacím a informativním prostředkem je **spreadsheet view**. Je možné jej zobrazit v novém okně, nebo ho změnit za stávající 3D pohled na model. Jedná se o tabulku, která nám poskytuje kompletní a detailní informace o jednotlivých bodech sítě. Označíme-li příslušný řádek v tabulce, tak se nám pozice bodu zvýrazní na zobrazeném modelu. V kombinaci se Selection Inspectorem je možné získat ucelený přehled o námi zvolené oblasti. Všechny body ve výběrovém inspektoru jsou zvýrazněné řádky v tabulce.

V poslední řadě bych popsal okno **Animation View**. Pomocí něj ParaView řídí průběh časově proměnných simulací. Existuje zde několik režimů, které si můžeme vybrat. Při otevření dat a zobrazení animation view se jako defaultní režim zobrazí **Snap To TimeSteps**, který nám přehraje přesně to, co je v datech definováno. V datech máme například definovanou délku simulace. Dalším režimem je **Real Time**, u kterého si ručně nastavíme čas simulace. Při zvolení delšího času, než je čas definovaný v datech, se bude obraz více zasekávat, jelikož časové mezery mezi jednotlivými snímky budou velké. Pokud bychom chtěli mít plynulou simulaci, doporučuji použít na data velmi důležitý filtr **Temporal Interpolator**, který nám dopočítá hodnoty, které nemáme v datech definovány. Tudíž nám zajistí plynulost simulace. Posledním režimem simulace je **Sequence**. Tento režim nám dovoluje nastavovat počet snímků pro defaultní čas.

ParaView nám pro přehlednost umožňuje komentovat simulace. Tuto funkci nalezneme pod názvem **Text**. Pokud bychom potřebovali mít viditelné informace o čase simulace, umožní nám to funkce **Annotation Time**. Jsou dvě možnosti, jak ji získat a každá z nich se liší. Pokud ji vezmeme z menu Source, tak získáváme informace o čase, ve kterém se nachází naše simulace. Ale pokud ji vezmeme z menu Filters, dává nám informace o aktuálním čase, který je definovaný v datech.

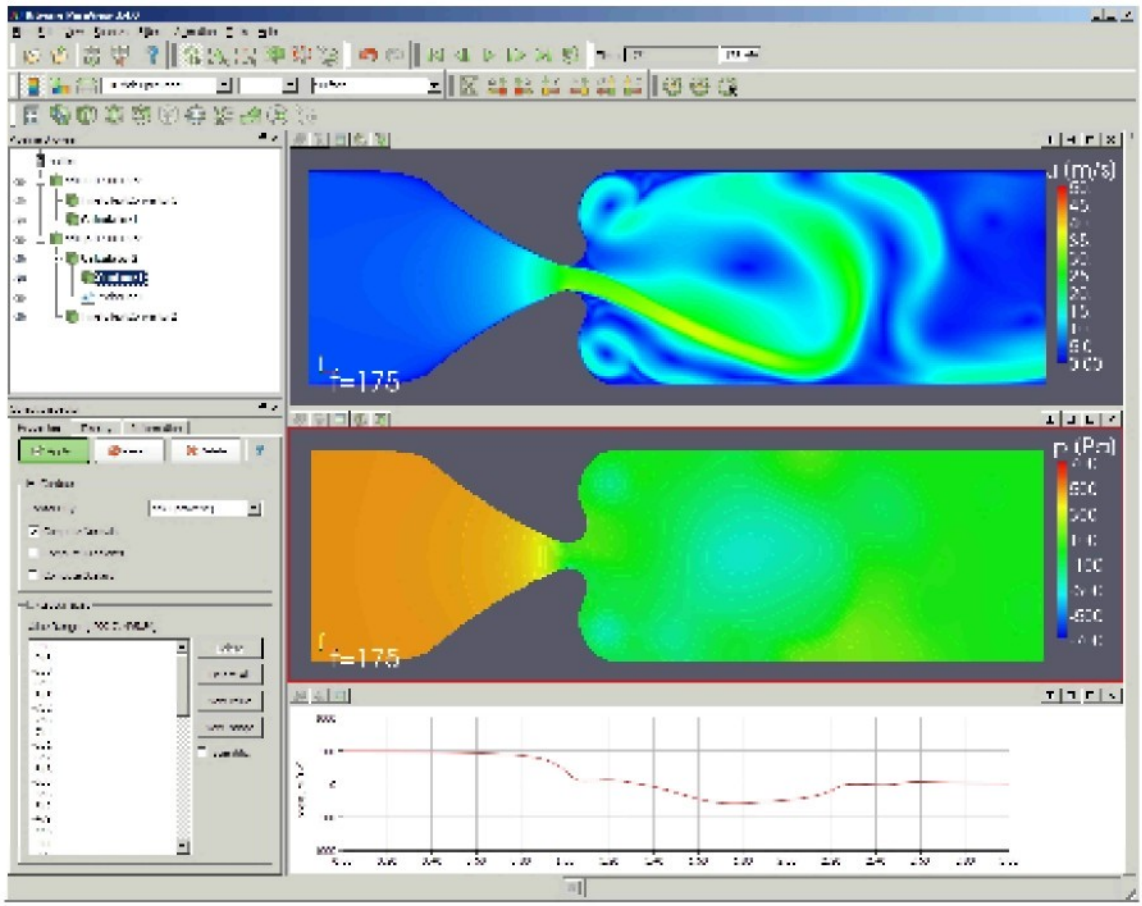
Na následujícím obrázku číslo dvacet dva vidíme simulaci konkrétních výsledků modelu lidských hlasivek. Nejdůležitější ale je, že zde vidíme všechny části, které jsem výše popisoval (Selection Inspector, Spredsheet View, Object Inspector a Animation View). Na modelu vidíme simulaci rychlosti proudícího vzduchu v lidských hlasivkách. Jedná se o simulaci kolem kmitajícího profilu. V našem případě kmitá zúžený střed. Další zkoumanou veličinou může být například tlak. Růžově zvýrazněná část představuje výběr elementů, jejichž seznam je v Selection Inspectoru. [11]



Obr. 22: Pohled na základní výběrové části simulace v ParaView

Na závěr bych chtěl ukázat konkrétní výsledek simulace na modelu lidských hlasivek (obr. 23). Vrchní část zobrazuje rychlost, v prostřední části nalezneme tlak a v dolní části vidím průběh tlaku podél přímky vedoucí středem lidských hlasivek. Obrázky jsou umístěné pod sebou pro snadnější orientaci a větší vypovídavost. Vidíme,

že v první části hlasivek je v tomto případě zvolen tlak pět set Pascalů, který neprostupuje zúžením a téměř okamžitě klesá na nulu. Za zúženým místem se tvoří víry, ve kterých tlak klesá k nejnižším hodnotám. Pokud by platila Bernoulliho rovnice, tak by tlak za zúžením vzrostl na výchozí hodnotu, ale u viskózního proudění Bernoulliho rovnice neplatí. Na konci hlasivek tlak klesá na nulu, což je dáno námi zvolenou okrajovou podmínkou. U rychlosti vidíme tendenci tvořit víry po výstupu ze zúžení. Můžeme snadno odečíst, že maximální rychlost odpovídá červené barvě a padesáti metrům za sekundu a minimální rychlost nula metrů za sekundu odpovídající barvě modré. Podobně můžeme hodnoty odečíst i u tlaku. Izočáry, které jsou vykresleny filtrem Contour, znázorňují místo s velkým gradientem tlaku pro lepší orientaci.



Obr. 23: Průběh tlaku a rychlosti v lidských hlasivkách

6.6 Problémy

Jedná se o propracovaný program, který nám nabízí mnoh možností vyhodnocení konkrétních simulací. Nicméně není dokonalý. Při zkoumání jednotlivých funkcí došlo několikrát k nekontrolovanému pádu softwaru, což určitě není vhodné pro průmyslové využití, ale pro studijní účely bohatě stačí.

7 Závěr

Na začátku této bakalářské práce jsem se zaměřil na jednoduchý popis metody konečných prvků, která původně spojitý problém rozčlení na síť tvořenou základními elementy. Jelikož se jedná o numerickou metodu, řešící diskrétní problém, tak zde vzniká chyba oproti spojitému řešení, která je způsobena zaokrouhlováním neceločíselných hodnot v počítači. Další chyba vzniká při zanedbání některých fyzikálních zákonitostí (zjednodušením modelu). Při zadávání úlohy je potřeba uvážit, které jevy můžeme zanedbat, abychom dostali dostatečně vypovídající informace v rozumném čase a s rozumným úsilím. Neměla by nastat situace, že výpočty trvají neúměrně víc času s minimální změnou výstupní vypovídavosti. Touto metodou se převážně řídí výpočty v simulačních softwarech. Samozřejmě, že existuje i řada jiných metod, ale ty nejsou tak často používány.

Následující kapitolu jsem věnoval modelování. Modelovat můžeme konkrétní výrobky, ale i mnoho vědeckých problémů, které se nám snáze chápou na modelech, o kterých můžeme zjistit pohodlně velké množství informací.

Další část byla věnována vizualizaci jako takové. Pokud jsem namodeloval objekt, rád bych viděl nějaké výsledky v přijatelné podobě, kterým snáze porozumím. V této partii jsem také popsal Vizualization ToolKit, jelikož jsem se potřeboval seznámit s formátem souborů *.vtk. Poté jsem mohl vytvořit konvertor výsledků a architektury sítě do jednoho souboru kompatibilního s programem ParaView. Konvertor byl napsán ve vývojovém prostředí Lazarus (obdoba Delphi) pomocí programovacího jazyka FreePascal. Zdrojový kód konvertoru naleznete v příloze. Program je rozdělen přehledně do procedur a funguje bez problémů pod platformou Windows i Linux. Po spuštění programu dojde k automatické konverzi dat ze souborů BB a MESH do jednoho souboru VTK. Názvy vytvořených VTK souborů odpovídají názvu souborů MESH a BB. Adresář pro ukládání souborů je také vytvořen automaticky.

V poslední části jsem se zabýval prací s výše uvedeným programem ParaView. Jedná se o opensource aplikaci využívající VTK. Popsal jsem uživatelské rozhraní, základní práci s programem a několik užitečných filtrů pro přehledné zobrazení výsledků. K dispozici jsem měl statická data s konkrétními hodnotami v jednom čase, nebo data, která jsou v čase proměnná. Na každé části jsem ukázal základní vlastnosti aplikace. Jak pomocí programu zpracujeme časově proměnná data, jsem ukázal na proudění vzduchu kolem modelu lidských hlasivek.

Celou práci jsem se snažil udělat srozumitelnou, proto jsem používal demonstrativní obrázky, na kterých se snáze věci pochopí a představí. Práce byla také pojata jako základní návod pro práci s programem ParaView.

Použitá literatura

[1] Počítačové ofukovanie
URL: <http://new.skoda-auto.sk/skoda_magazin/?skoda_magazin-zima2008_s16> [cit. 2009-03-20]

[2] Bathe, Klaus-Jürgen. *Finite Element Procedures.* Prentice Hall, Pearson Education, Inc., 2006. 1037 stran.

[3] https://wci.llnl.gov/codes/visit/media/onion3_S.jpg [cit. 2009-04-02]

[4] The Finite Element Method
URL:
<http://illustrations.marin.ntnu.no/structures/analysis/FEM/index.html>
[cit. 2009-04-10]

[5] Scientific visualization
URL: <http://en.wikipedia.org/wiki/Scientific_visualization>
[cit. 2009-04-25]

[6] URL:
<http://www.nsf.gov/news/overviews/computer/assets/tornado2.jpg>
[cit. 2009-05-15]

[7] VTK – Vizualization Toolkit
URL: <http://www.vtk.org/VTK/project/about.html> [cit. 2009-05-15]

[8] Schroeder, William J.; Martin, Kenneth M.; Lorensen, William E.. *The Design and Implementation Of An Object-Oriented Toolkit For 3D Graphics And Visualization* [online]. [cit. 2008-2-12]. URL: <http://www.vtk.org/VTK/img/dioot.pdf>

[9] Schroeder, Will; Martin, Ken; Bill, Lorensen. *The Visualization Toolkit An Object-Oriented Approach to 3D Graphics.* Kitware, Inc., 2003. 496 stran.

[10] Kitware, Inc.. *VTK User's Guide*. Kitware Inc., 2006. 382 stran. [část online]. [cit. 2008-2-12]. URL: <http://www.vtk.org/VTK/img/file-formats.pdf>

[11] Moreland, Kenneth; Greenfield, John; Scott, W. Alan; Ayachit, Utkarsh; Geveci, Berk; DeMarle, David. *Large Scale Visualization with ParaView Supercomputing 2008 Tutorial.* [online]. [cit. 2008-10-11]. URL: <http://paraview.org/Wiki/images/6/62/SC08_tut121_ParaView_Handouts.pdf >

Přílohy

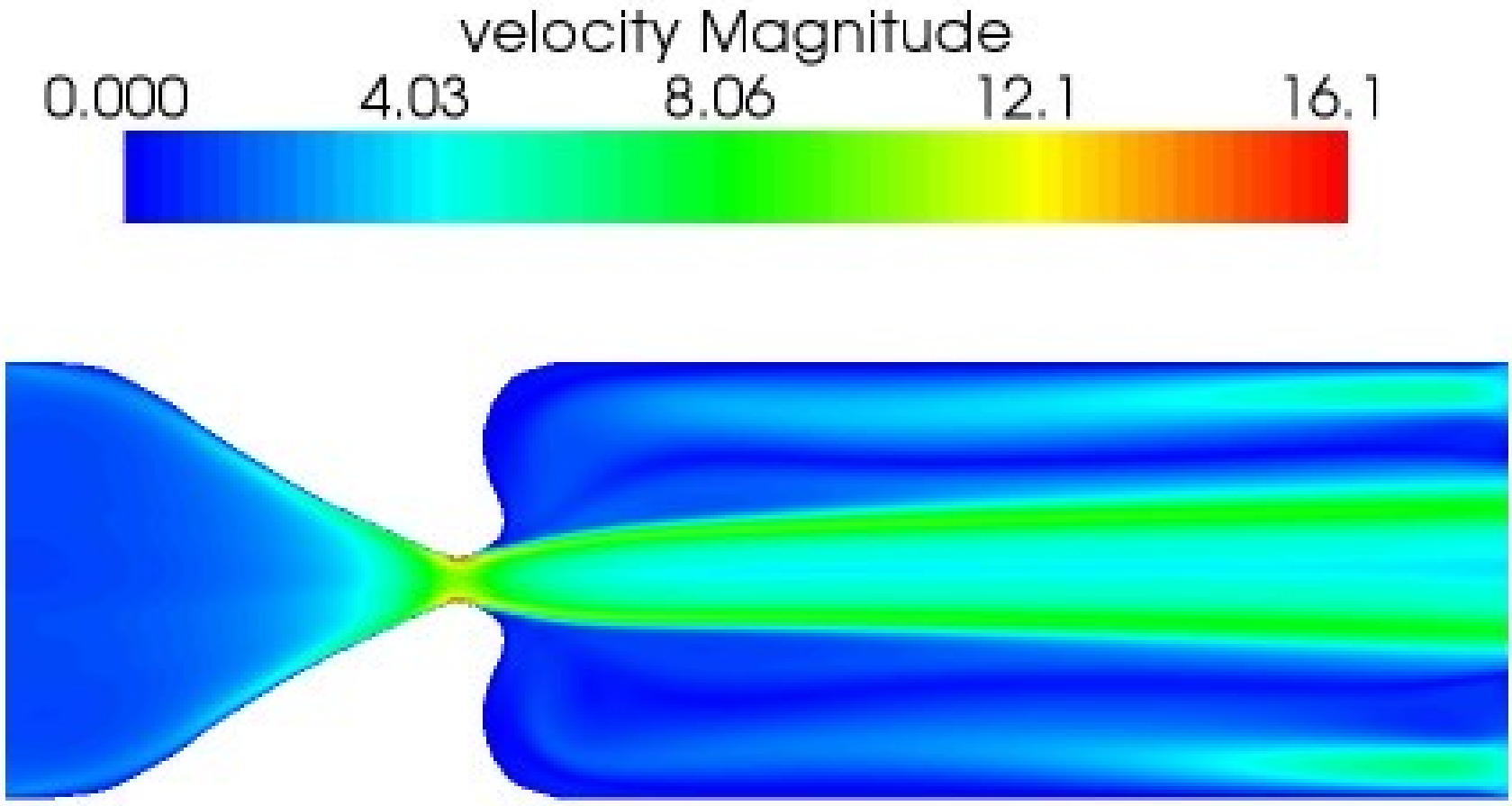
Konvertor souborů MESH a BB do souboru VTK

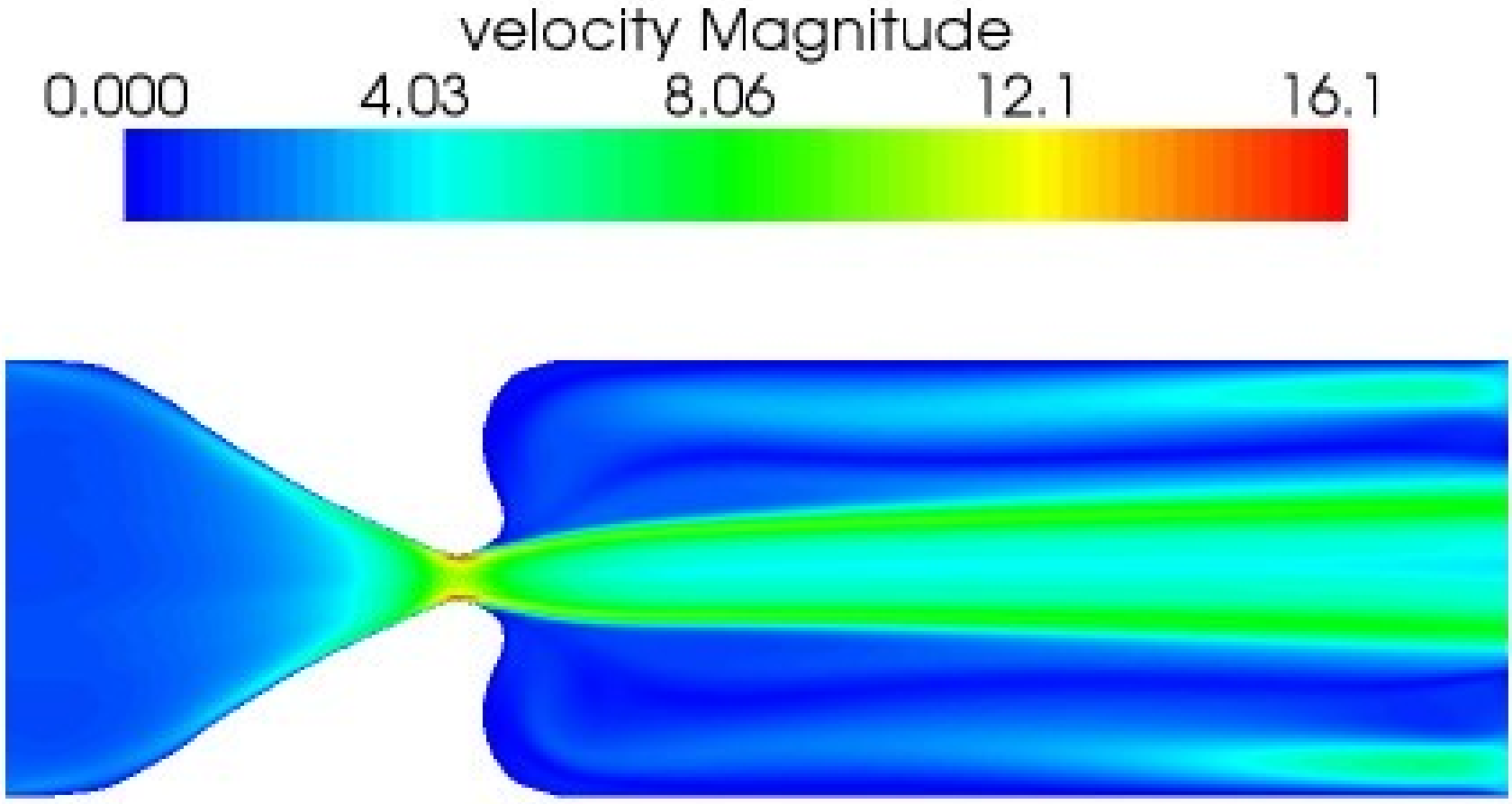
```
program meshbb2vtk;

{$mode objfpc}{$H+}

uses
  {$IFDEF UNIX}{$IFDEF UseCThreads}
  cthreads,
  {$ENDIF}{$ENDIF}
  Classes, SysUtils, CustApp, DateUtils
  { you can add units after this };

// GLOBALNI KONSTANTY
const VERZE_VTK = '# vtk DataFile Version 3.0';
      HLAVICKA_TYP ='ASCII';
      HLAVICKA_TYP_SITE ='DATASET UNSTRUCTURED_GRID';
      VRCHOLY_NADPIS = 'POINTS ';
      REALNA_CISLA_TYP = ' float';
      ELEMENTY_NADPIS =  'CELLS ';
      VRCHOLU_NA_ELEMENT = 3 ;
      POCET_SPOJOVANYCH_VRCHOLU = ' 3 ';
      ELEMENTY_TYP_NADPIS = 'CELL_TYPES ';
      ELEMENT_TYP_TROJUHELNIK = ' 5';                    //trojuhelniky
      VRCHOLY_DATA_NADPIS = 'POINT_DATA ';
      SKALAR_NAZEV = 'SCALARS pressure ';
      INDEXOVACI_TABULKA_NAZEV = 'LOOKUP_TABLE default';
      VEKTOR_NAZEV = 'VECTORS velocity ';
      PRIPONA_VTK = '.vtk';
      CILOVY_ADRESAR = 'VTK' + PathDelim;

type

  { Tmeshbb2vtk }

  Tmeshbb2vtk = class(TCustomApplication)
  protected
    procedure DoRun; override;
  public
    constructor Create(TheOwner: TComponent); override;
    destructor Destroy; override;
    procedure WriteHelp; virtual;
  private
(* telo programu pro prehlednost rozdeleno do tri procedur *)
    procedure zapis_hlavicka(var f_vtk : textfile);
    procedure zapis_vrcholy(var f_vtk,f_mesh,f_bb:textfile);
    procedure zapis_elementy(var f_vtk,f_mesh,f_bb:textfile);
(* konverze jednoho souboru .bb a .mesh -> .vtk*)
    procedure Konvertor(jmeno_bb,jmeno_mesh:TFileName);

    cesta : string; // adresar, ve kterem se bude konvertovat
    sr : TSearchRec; // record pro FindFirst/FindNext
    file_mesh,file_bb,file_vtk : textfile; // 2 vstupni a 1 vystupni(vtk) soubor
    jmeno_mesh,jmeno_bb,jmeno_vtk:TFileName;  // jmena souboru
    dimenze_sit,dimenze_reseni,pocet_vrcholu,pocet_elementu : longint;
  end;

{ Tmeshbb2vtk }
```

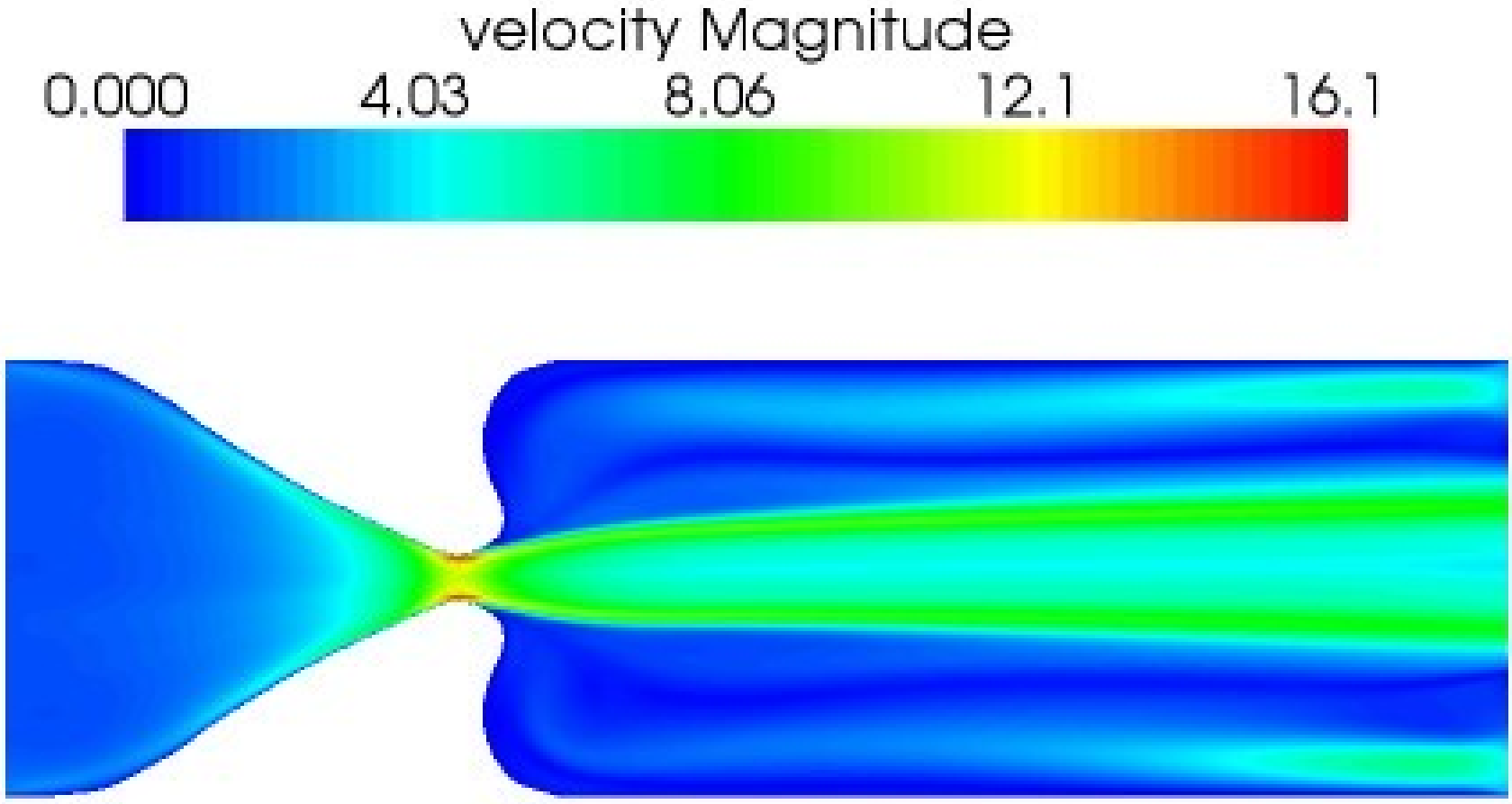
```
(* zapise textovou hlavicku .vtk souboru*)
procedure Tmeshbb2vtk.zapis_hlavicka(var f_vtk: textfile);
begin
  writeln(f_vtk,VERZE_VTK);      // W: # vtk DataFile Version 3.0
  writeln(f_vtk,jmeno_vtk);      // W: solu-u-xxx.vtk
  writeln(f_vtk,HLAVICKA_TYP);   // W: ASCII
  writeln(f_vtk);
end;

(* zapise sekci s vrcholy *)
procedure Tmeshbb2vtk.zapis_vrcholy(var f_vtk,f_mesh,f_bb:textfile);
var radek:string;
    i:longint;
    typ_asociace:integer;
begin
  readln(f_bb,dimenze_sit,dimenze_reseni,pocet_vrcholu,typ_asociace);

  Readln(f_mesh);                      // R: MeshVersionFormatted 1
  Readln(f_mesh);                      // R: Dimension 2
  Readln(f_mesh,radek);                //  R: Vertices            19
  writeln(f_vtk,HLAVICKA_TYP_SITE);  // W: DATASET UNSTRUCTURED_GRID
  writeln(f_vtk,VRCHOLY_NADPIS,' ',IntToStr(pocet_vrcholu),REALNA_CISLA_TYP);
                                       // W: POINTS  19 float
  for i:=1 to pocet_vrcholu do
    begin
      readln(f_mesh,radek);            // R:   -0.25000E+00  -0.43482E+00  1
      radek[length(radek)]:='0';
      writeln(f_vtk,radek);            // W:   -0.25000E+00  -0.43482E+00  0
    end;
  writeln(f_vtk);
end;

(* zapise sekci s elementy *)
procedure Tmeshbb2vtk.zapis_elementy(var f_vtk,f_mesh,f_bb:textfile);
var radek:string;
    i,j,index:longint;
    str:string[12];

begin
  Readln(f_mesh,str,pocet_elementu);          //  R: Triangles             20
  writeln(f_vtk,ELEMENTY_NADPIS,IntToStr(pocet_elementu),' ',IntToStr(
   pocet_elementu*(VRCHOLU_NA_ELEMENT+1)));   // W: CELLS 20 80

  for i:=1 to pocet_elementu do               // R: 7 13 11  -> W: 3 6 12 10
    begin
      write(f_vtk,POCET_SPOJOVANYCH_VRCHOLU);
      for j:=1 to VRCHOLU_NA_ELEMENT do
      begin
        read(f_mesh, index);
        write(f_vtk, index-1);
        write(f_vtk,' ');
      end;
      writeln(f_vtk);
      readln(f_mesh);
    end;

  writeln(f_vtk);
  writeln(f_vtk,ELEMENTY_TYP_NADPIS,pocet_elementu);// W: CELL_TYPES 20
  for i:=1 to pocet_elementu do
   writeln(f_vtk,ELEMENT_TYP_TROJUHELNIK);          // W: 5
  writeln(f_vtk);
```

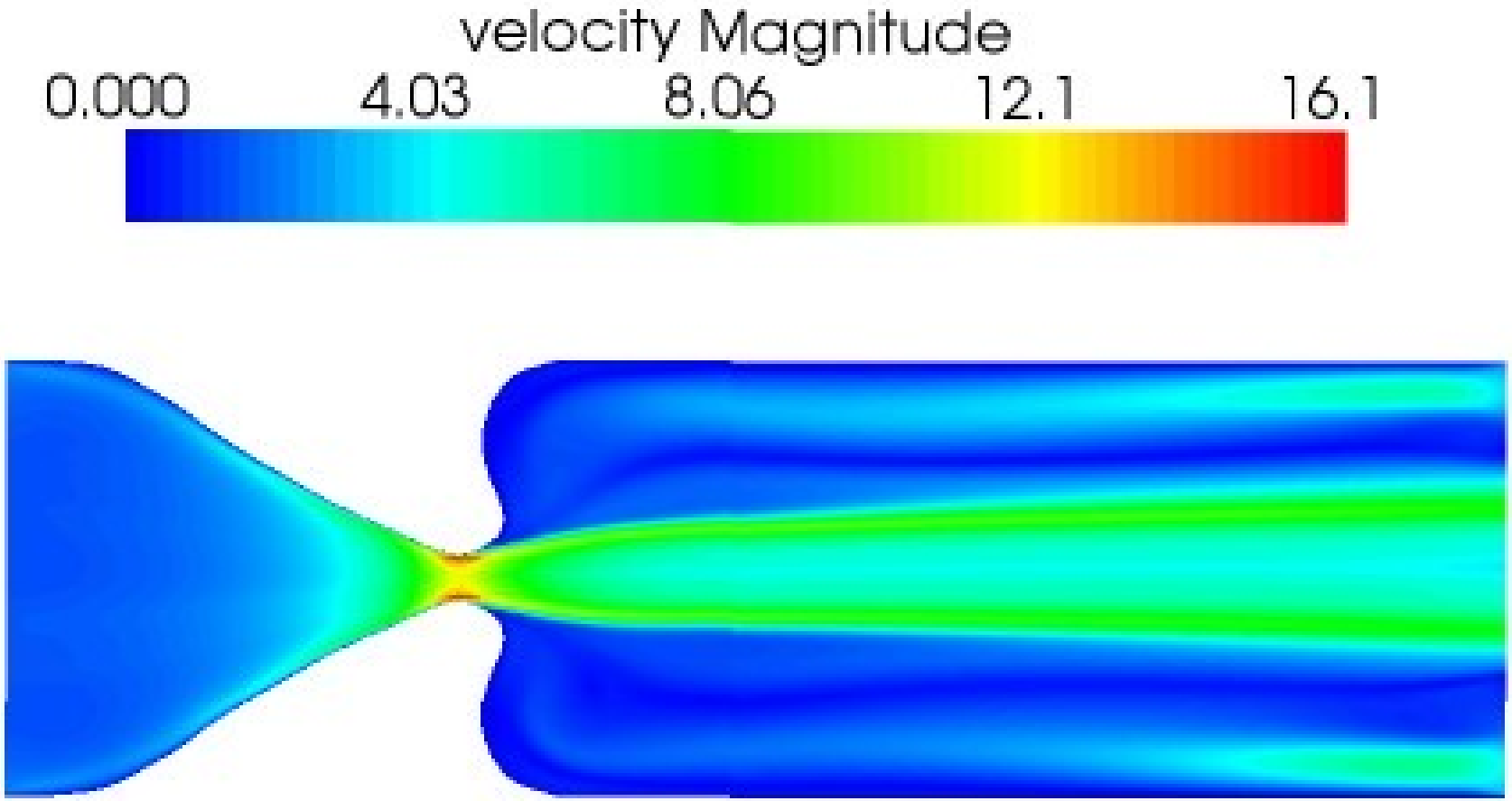
```
  writeln(f_vtk,VRCHOLY_DATA_NADPIS,pocet_vrcholu); // W: POINT_DATA 19
  if dimenze_reseni = 1 then begin;
   writeln(f_vtk,SKALAR_NAZEV+REALNA_CISLA_TYP);     // W: SCALARS pressure float
   writeln(f_vtk,INDEXOVACI_TABULKA_NAZEV);          // W: LOOKUP_TABLE default
  end
  else writeln(f_vtk,VEKTOR_NAZEV+REALNA_CISLA_TYP);// W: VECTORS velocity float

  for i:=1 to pocet_vrcholu do
  begin
    readln(f_bb, radek);                 // R:    0.31760E-02   0.41229E-01
    if dimenze_reseni = 1 then writeln(f_vtk,radek)
    else writeln(f_vtk,radek,'   0');    // W:    0.31760E-02   0.41229E-01   0
  end;
end;

(* zkonvertuje jednu dvojici souboru *.bb, *.mesh *)
procedure Tmeshbb2vtk.Konvertor(jmeno_bb,jmeno_mesh: TFileName);
begin
//prirazeni souboru
  System.Assign(file_bb,cesta+jmeno_bb);
  System.Assign(file_mesh,cesta+jmeno_mesh);
  jmeno_vtk:=ChangeFileExt(jmeno_bb, PRIPONA_VTK);
  System.Assign(file_vtk,cesta+CILOVY_ADRESAR+jmeno_vtk);
  write(jmeno_vtk);
// otevreni souboru
  Rewrite(file_vtk);
  Reset(file_mesh);
  Reset(file_bb);

  try
   zapis_hlavicka(file_vtk);
   zapis_vrcholy (file_vtk,file_mesh,file_bb);
   zapis_elementy(file_vtk,file_mesh,file_bb);
  finally
   Closefile(file_vtk);
   CloseFile(file_mesh);
   CloseFile(file_bb);
  end;
  Writeln('   OK!');
end;


procedure Tmeshbb2vtk.DoRun;
var errcode,n: integer;
    t_start,t_end:TDateTime;
begin
// pokud byl program spusten s parametrem -h nebo -help, vypise format parametru
  if HasOption('h','help') then begin
    WriteHelp;
    Halt;
  end
  else begin;
// cesta := prvni parametr prikazove radky nebo aktualni adresar
   if ParamCount <> 0 then cesta:=ParamStr(1) else cesta:='.' + PathDelim;
  end;

  SetCurrentDir(cesta);
  CreateDir('VTK');

  n:=0; t_start:=Now;
// FindFirst-FindNext vyhleda vsechny soubory s maskou *.bb v aktualnim adresari
  errcode:=FindFirst('*.bb',faAnyFile,sr);
```

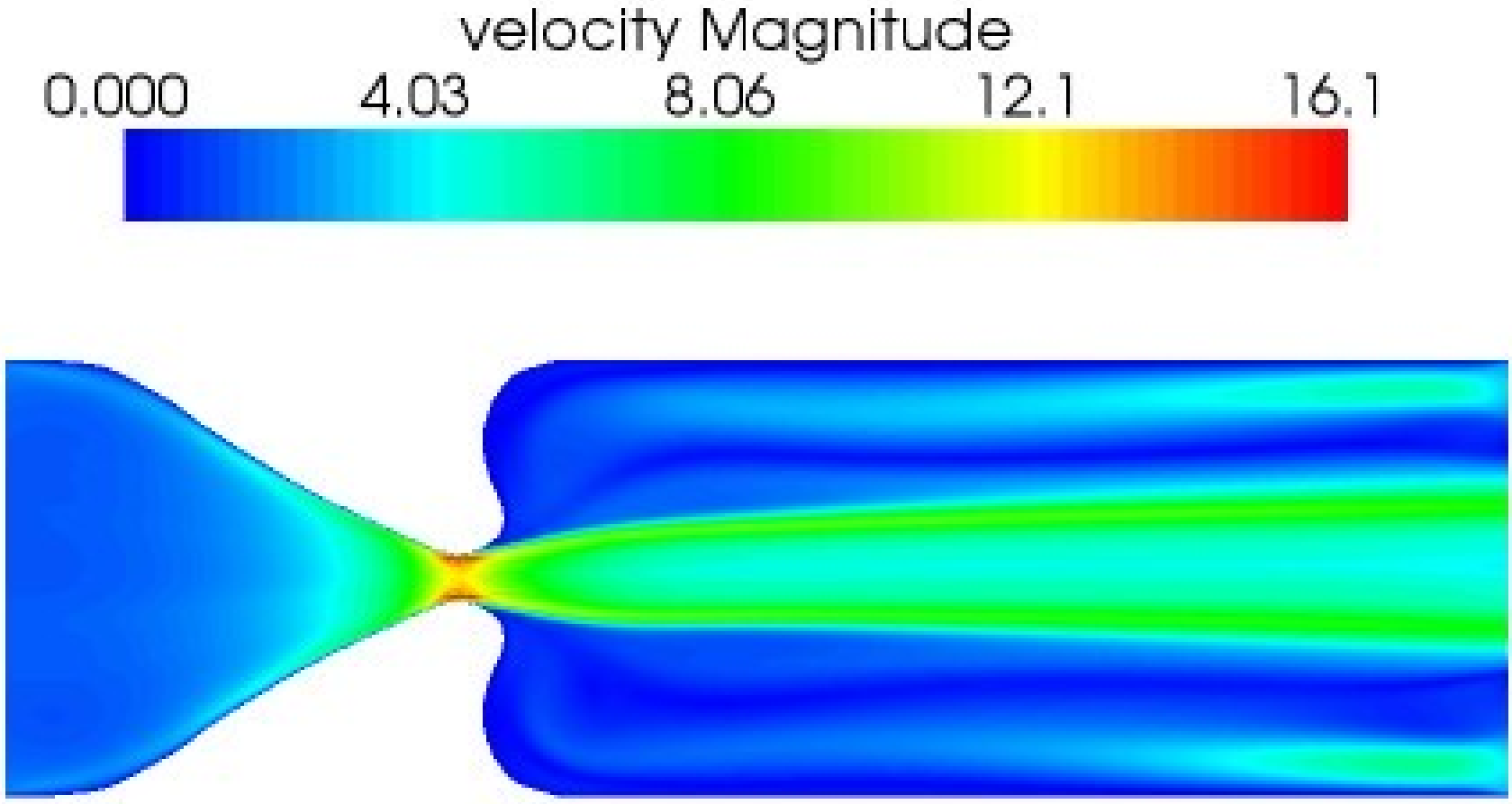
```
  while errcode = 0 do begin;
    inc(n);
    jmeno_bb:=sr.Name;
    jmeno_mesh:=ChangeFileExt(jmeno_bb,'.mesh');
    if FileExists(cesta+jmeno_mesh) then Konvertor(jmeno_bb,jmeno_mesh);
    errcode:=FindNext(sr);
  end;

  t_end:=Now; writeln;
  writeln(Format('%d files converted in %.2f seconds.',
   [n,MilliSecondSpan(t_start,t_end)/1000]));
  Terminate;  // set terminated flag
end;

constructor Tmeshbb2vtk.Create(TheOwner: TComponent);
begin
  inherited Create(TheOwner);
  StopOnException:=True;
end;

destructor Tmeshbb2vtk.Destroy;
begin
  inherited Destroy;
end;

procedure Tmeshbb2vtk.WriteHelp;
begin
  writeln('Usage: ',ExeName,' -h  to get help');
  writeln('       ',ExeName,
   '     to convert *.mesh *.bb in current directory to ./VTK/*.vtk');
  writeln('       ',ExeName,
   ' dir to convert dir/*.mesh dir/*.bb');
end;

var
  Application: Tmeshbb2vtk;

begin
  Application:=Tmeshbb2vtk.Create(nil);
  Application.Title:='meshbb2vtk';
  Application.Run;
  Application.Free;
end.
```

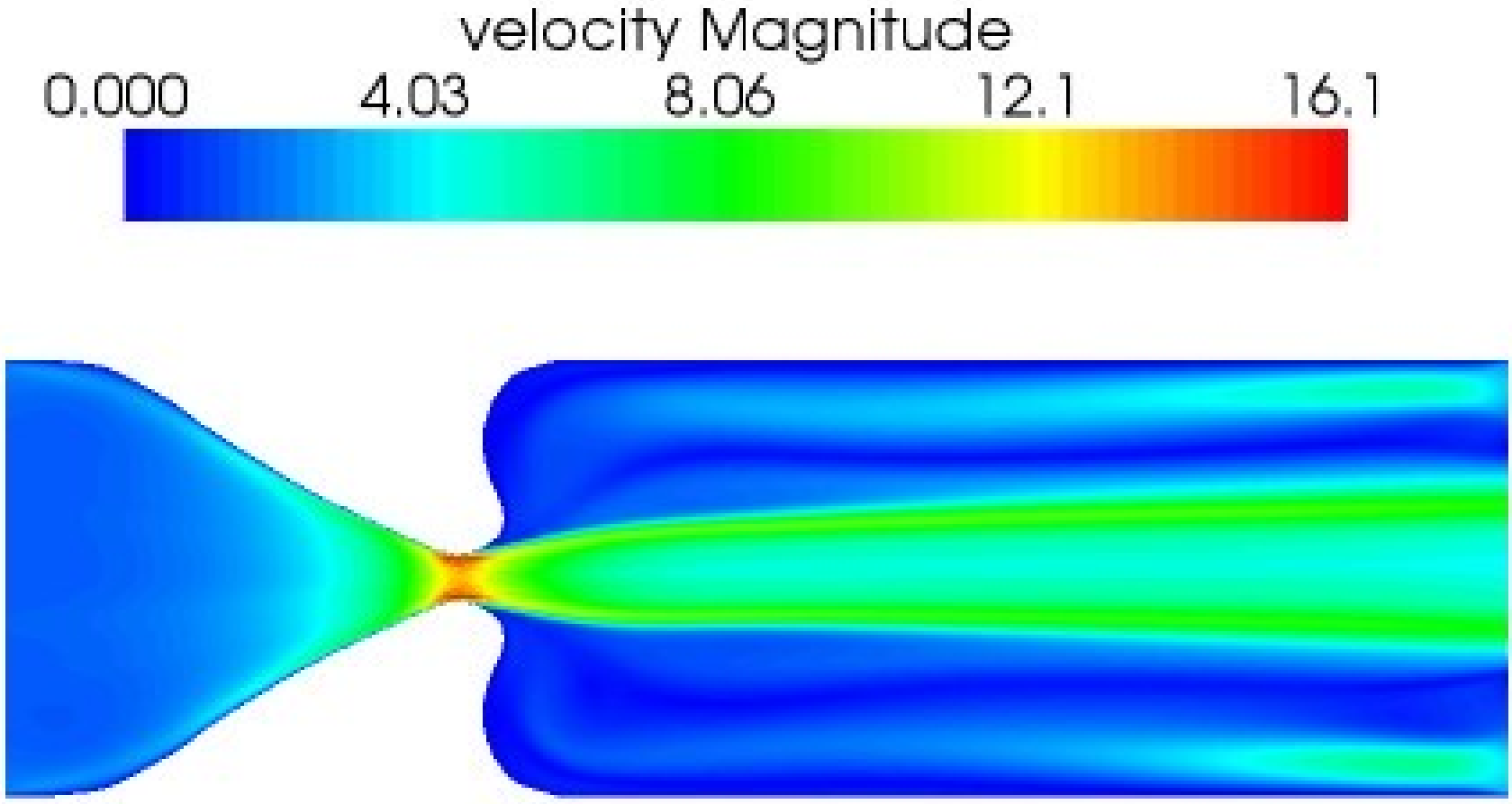

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

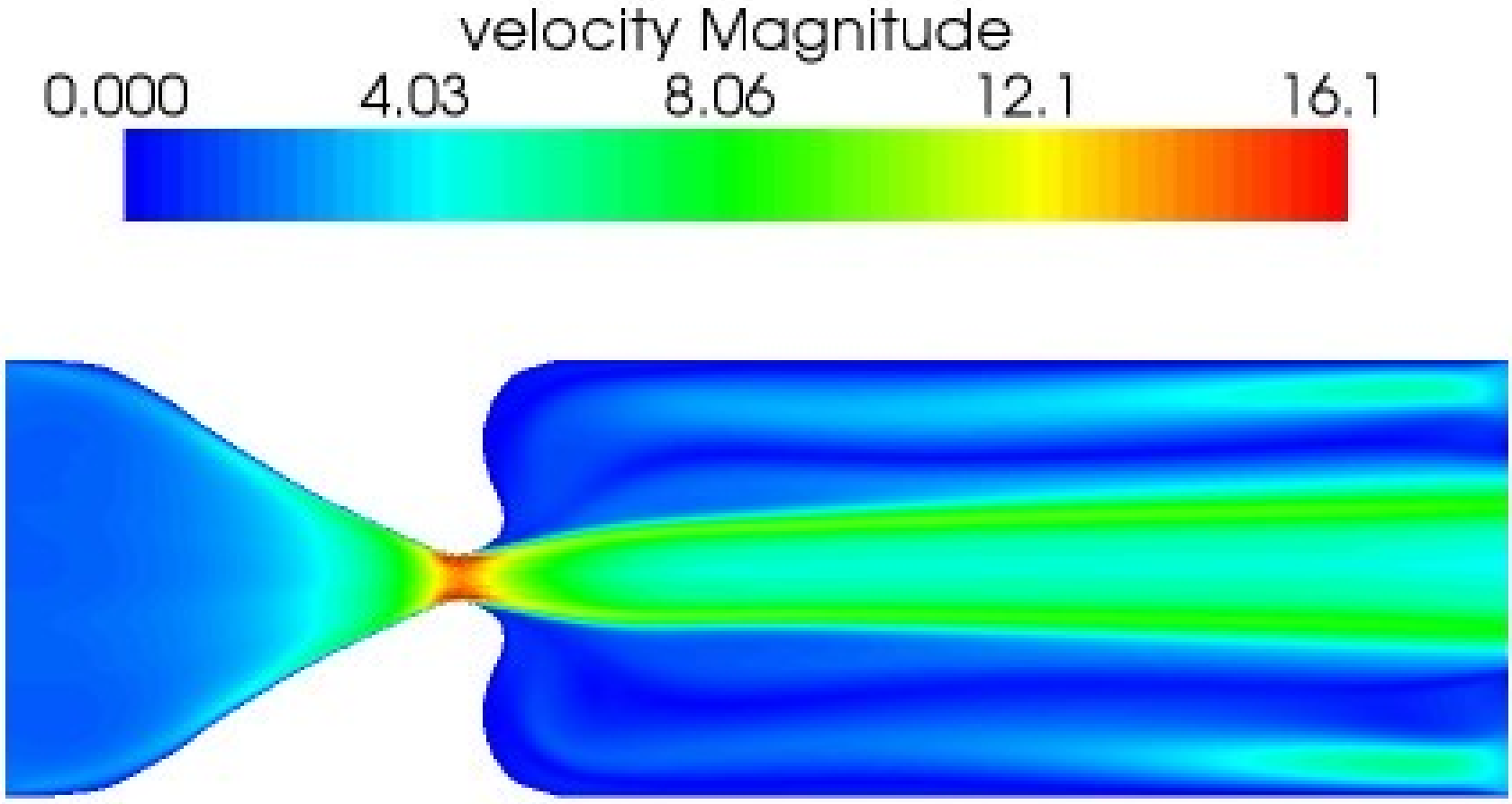

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

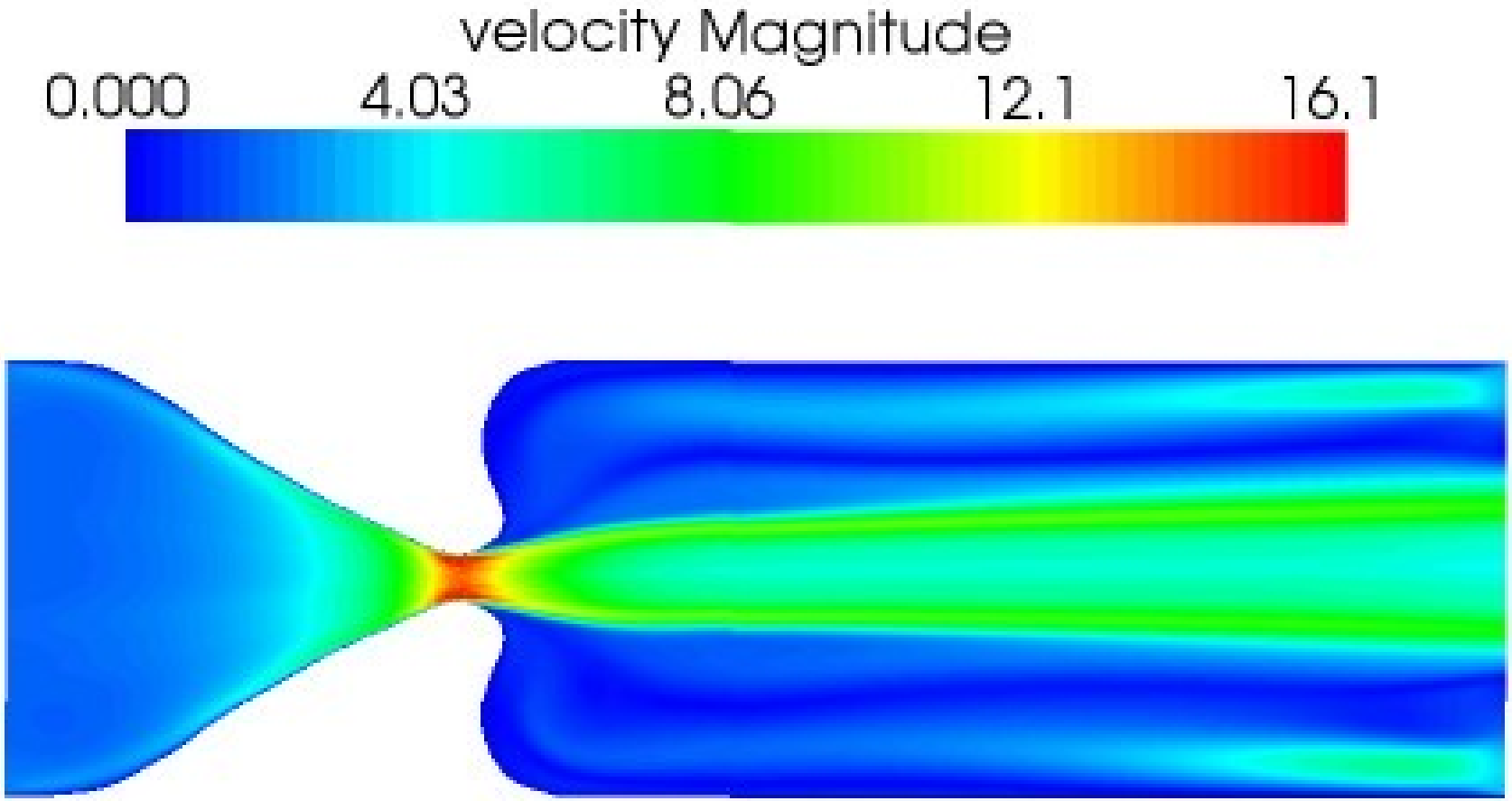

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

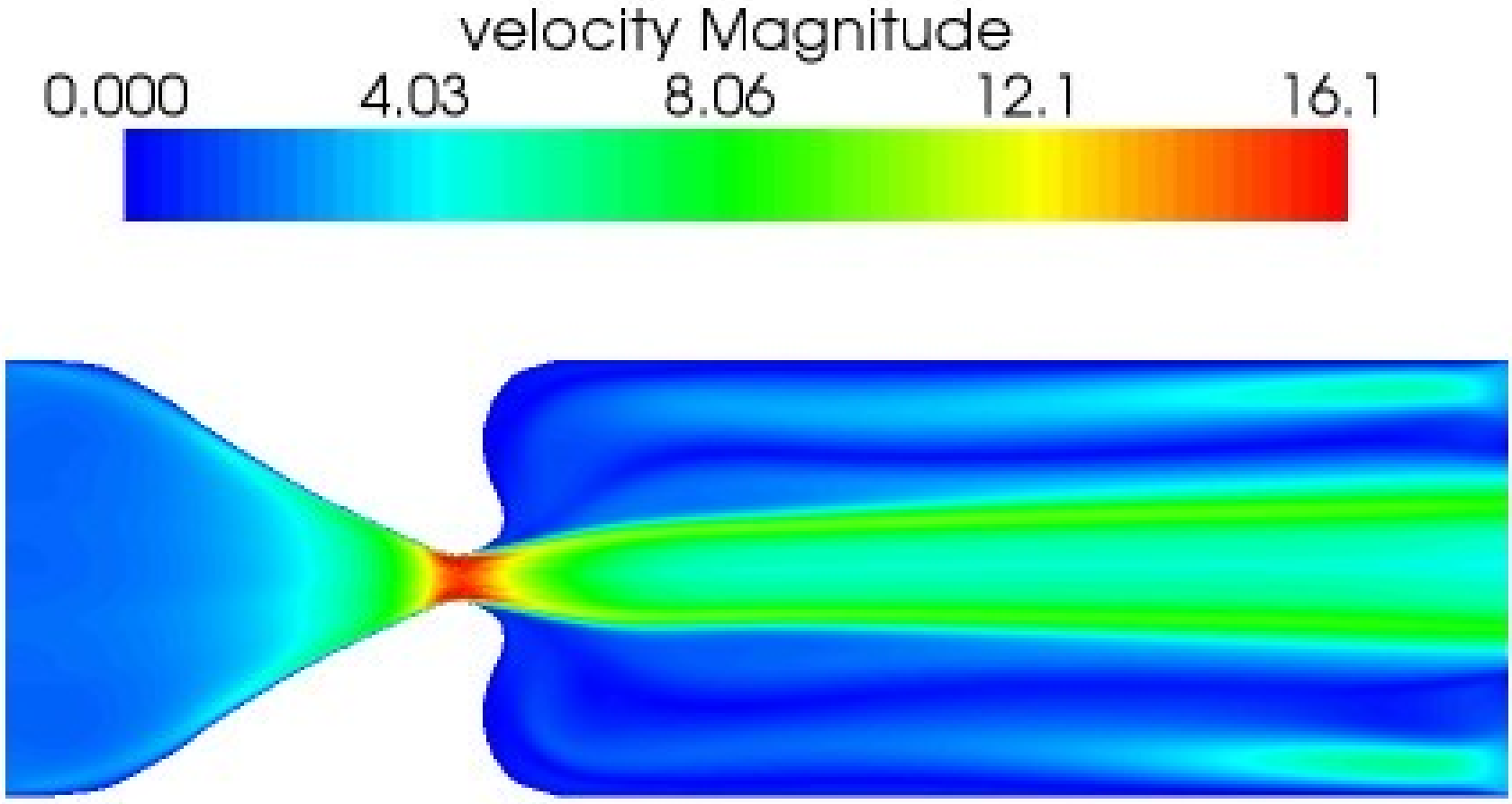

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

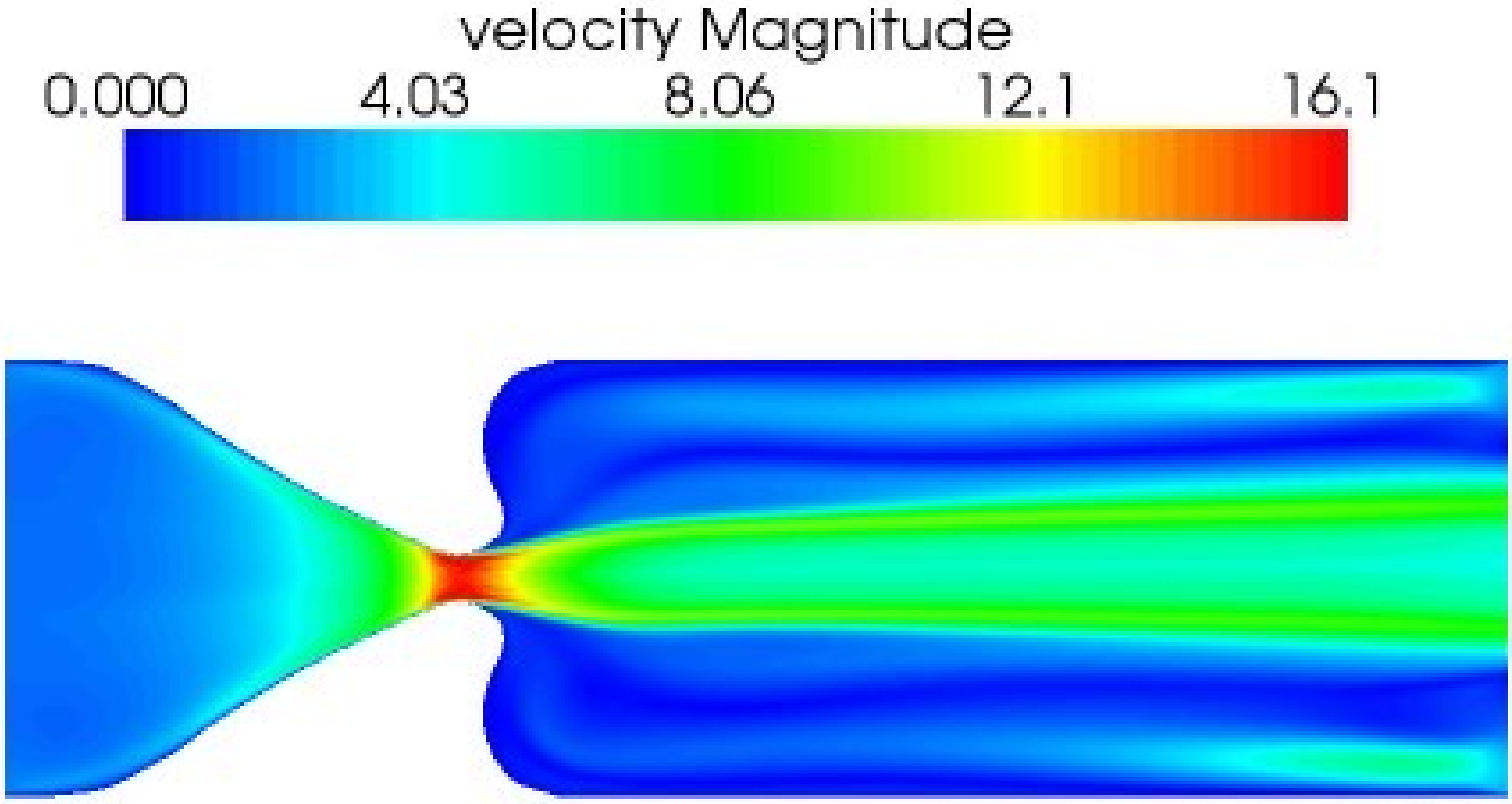

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

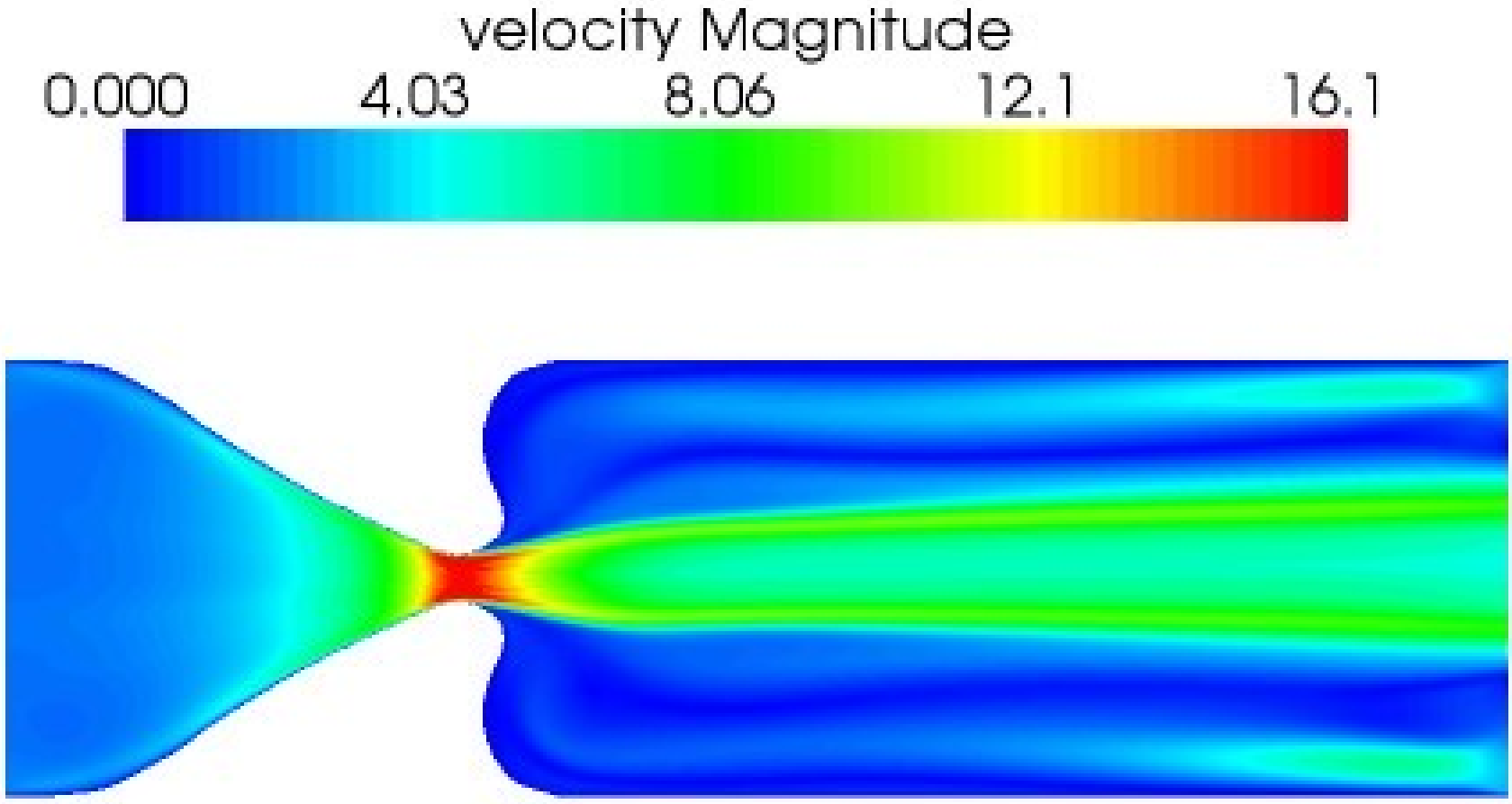

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

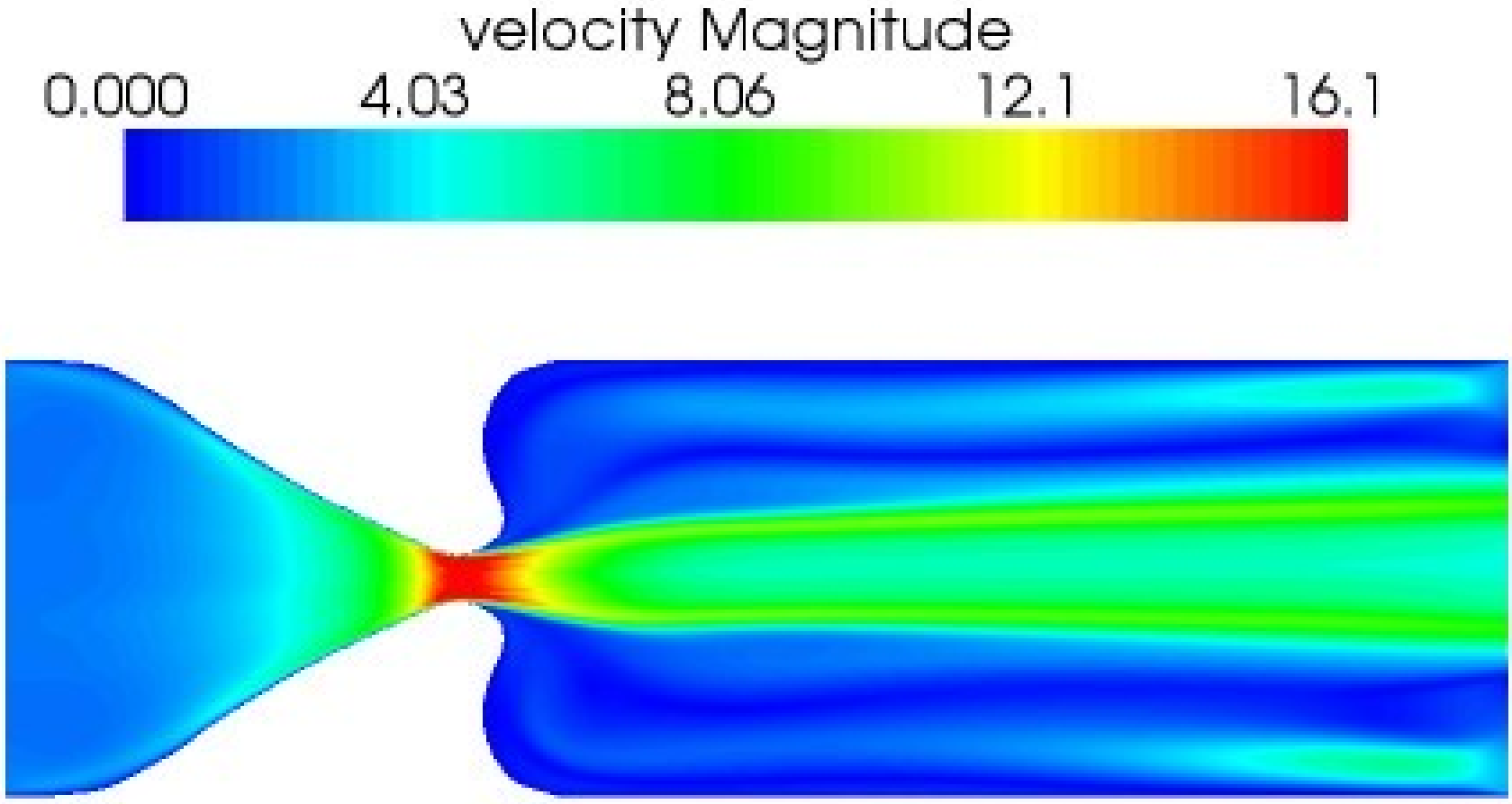

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

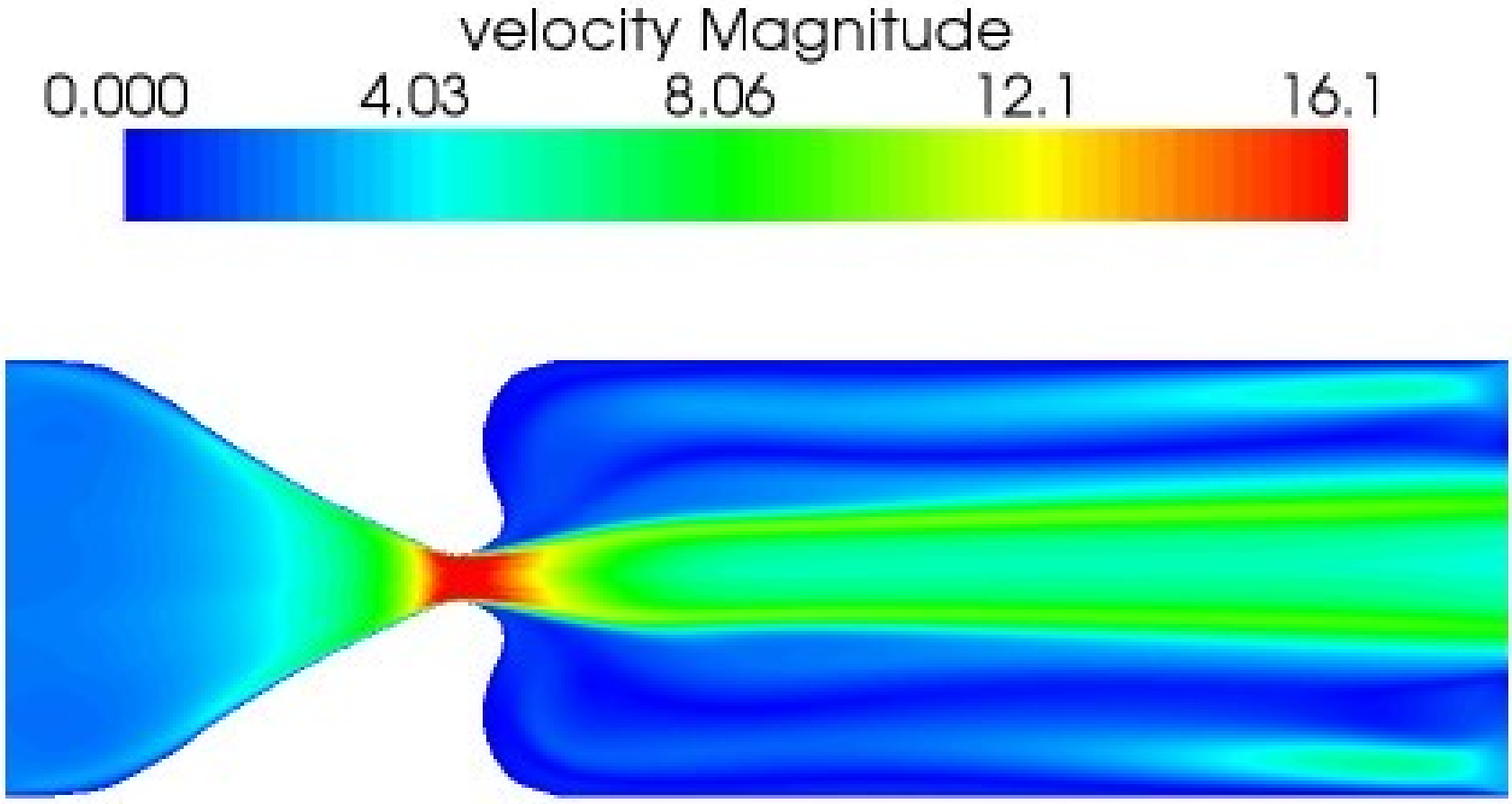

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

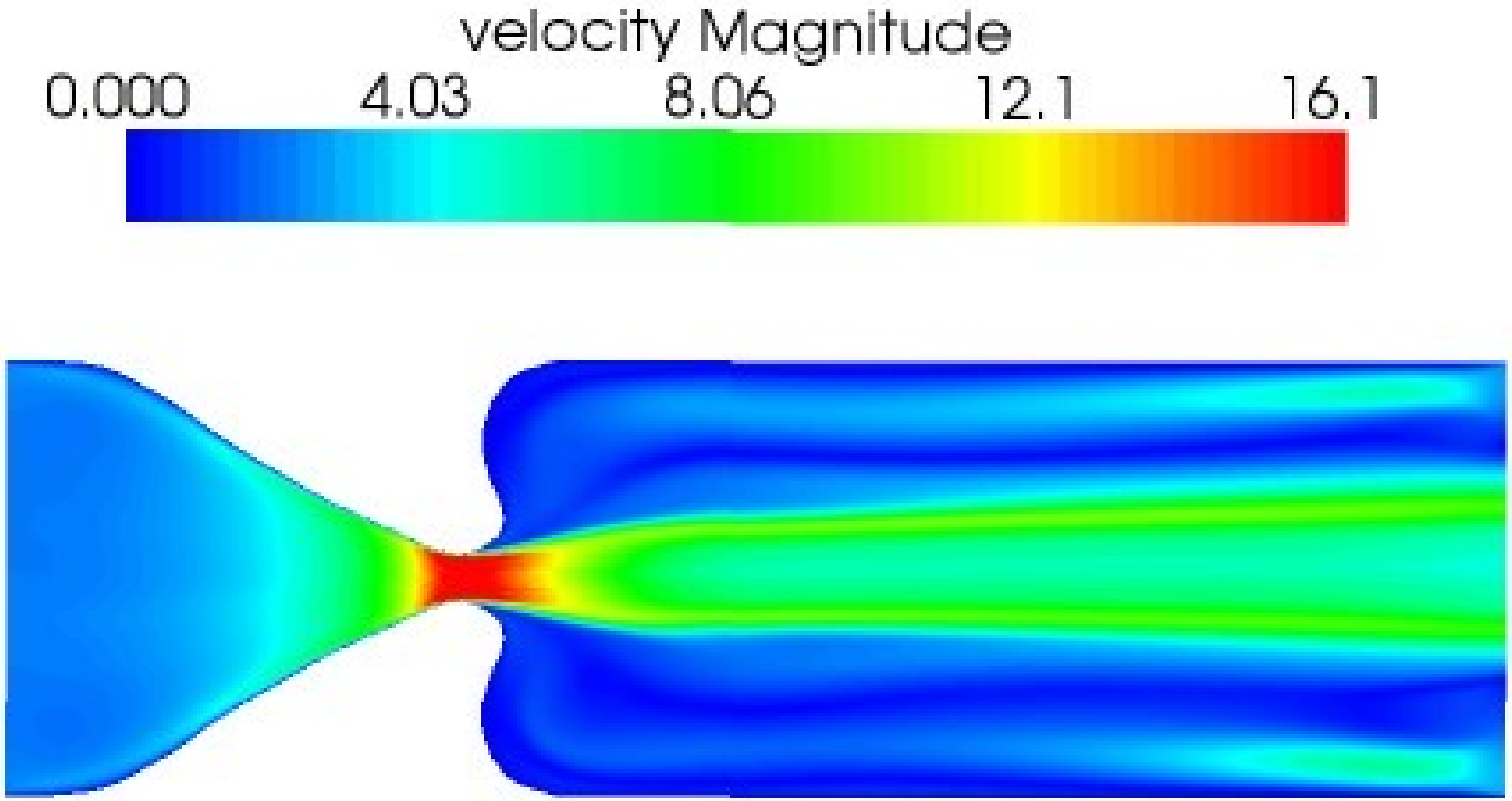

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

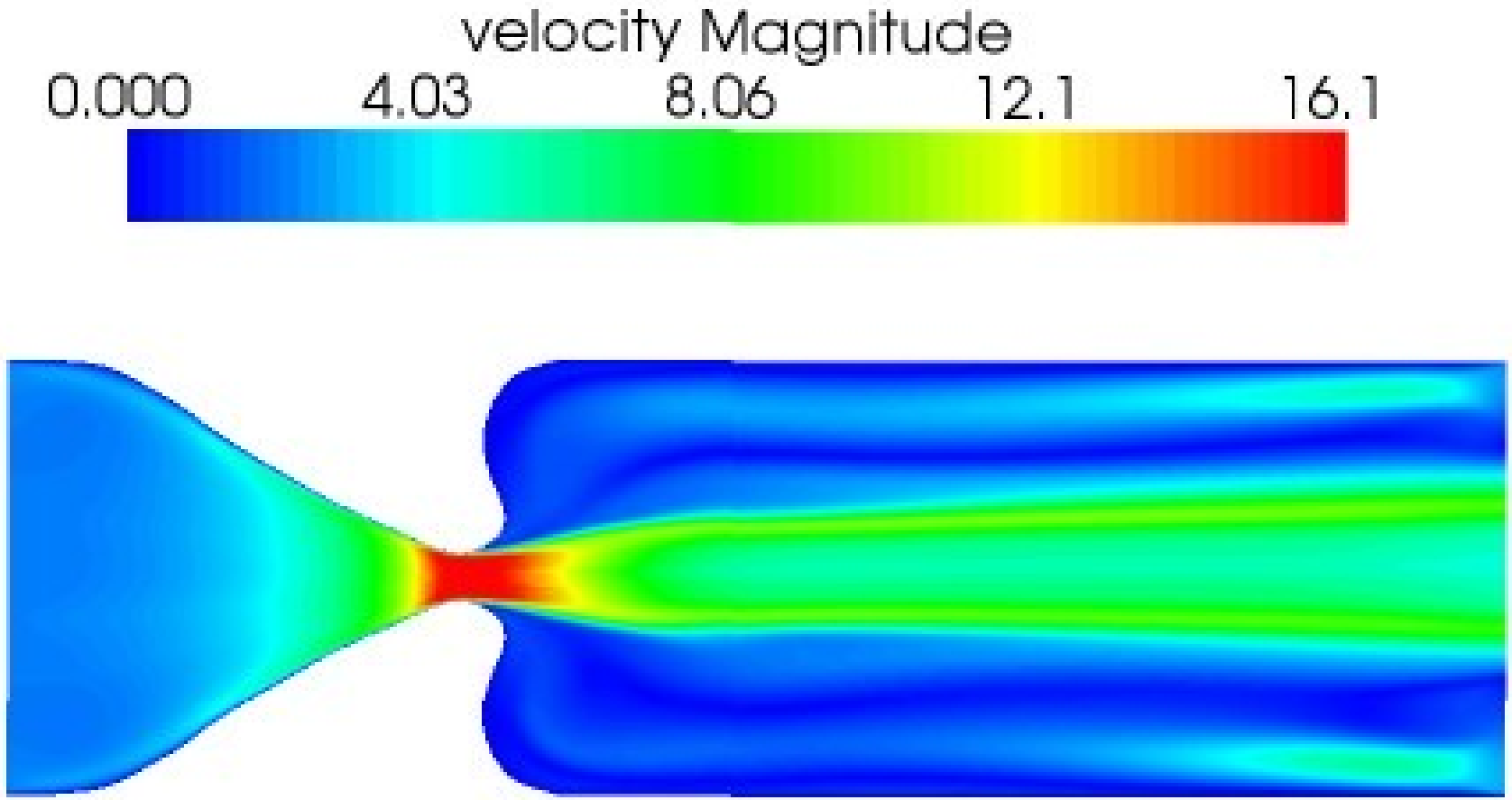

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

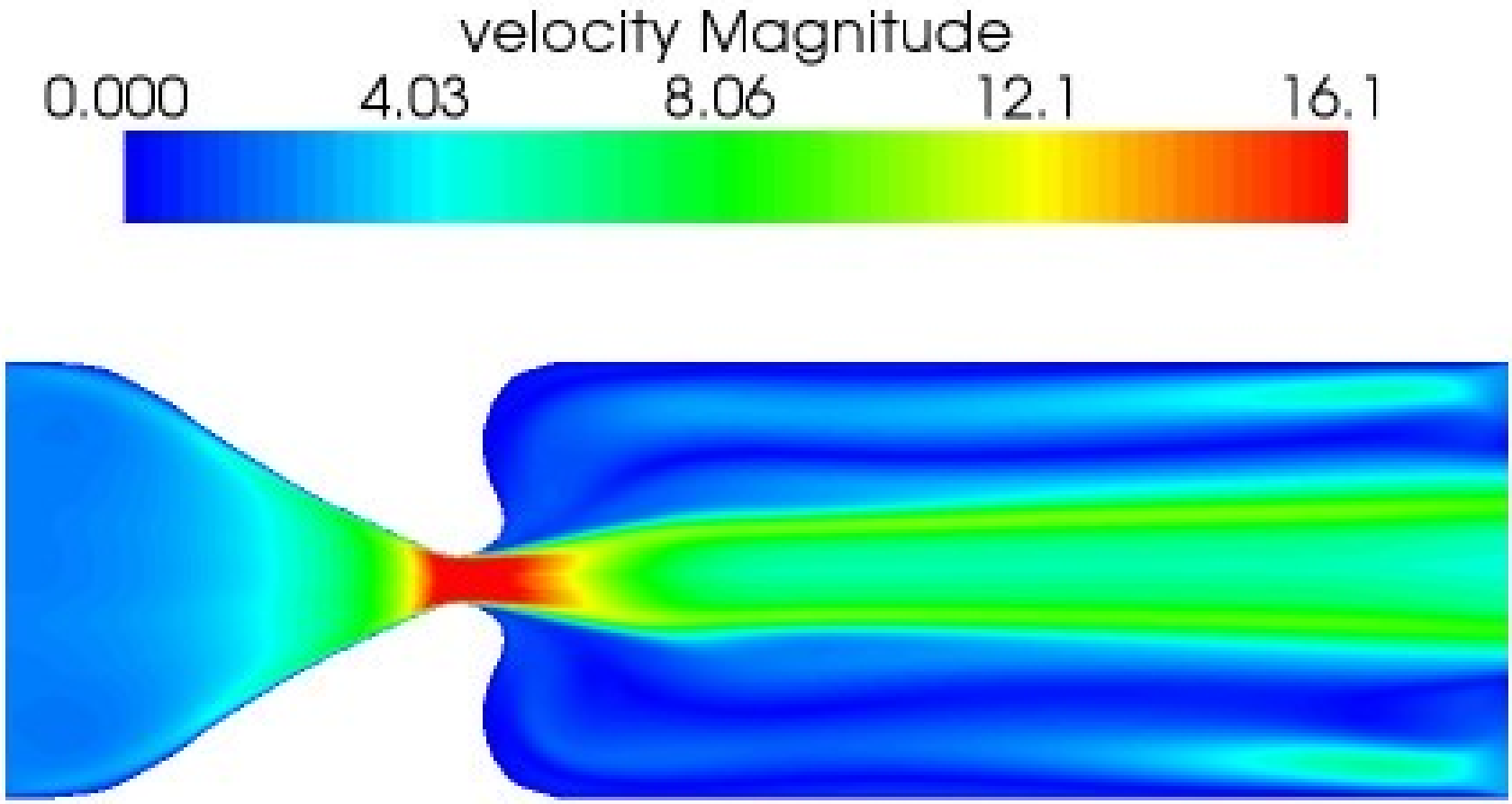

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

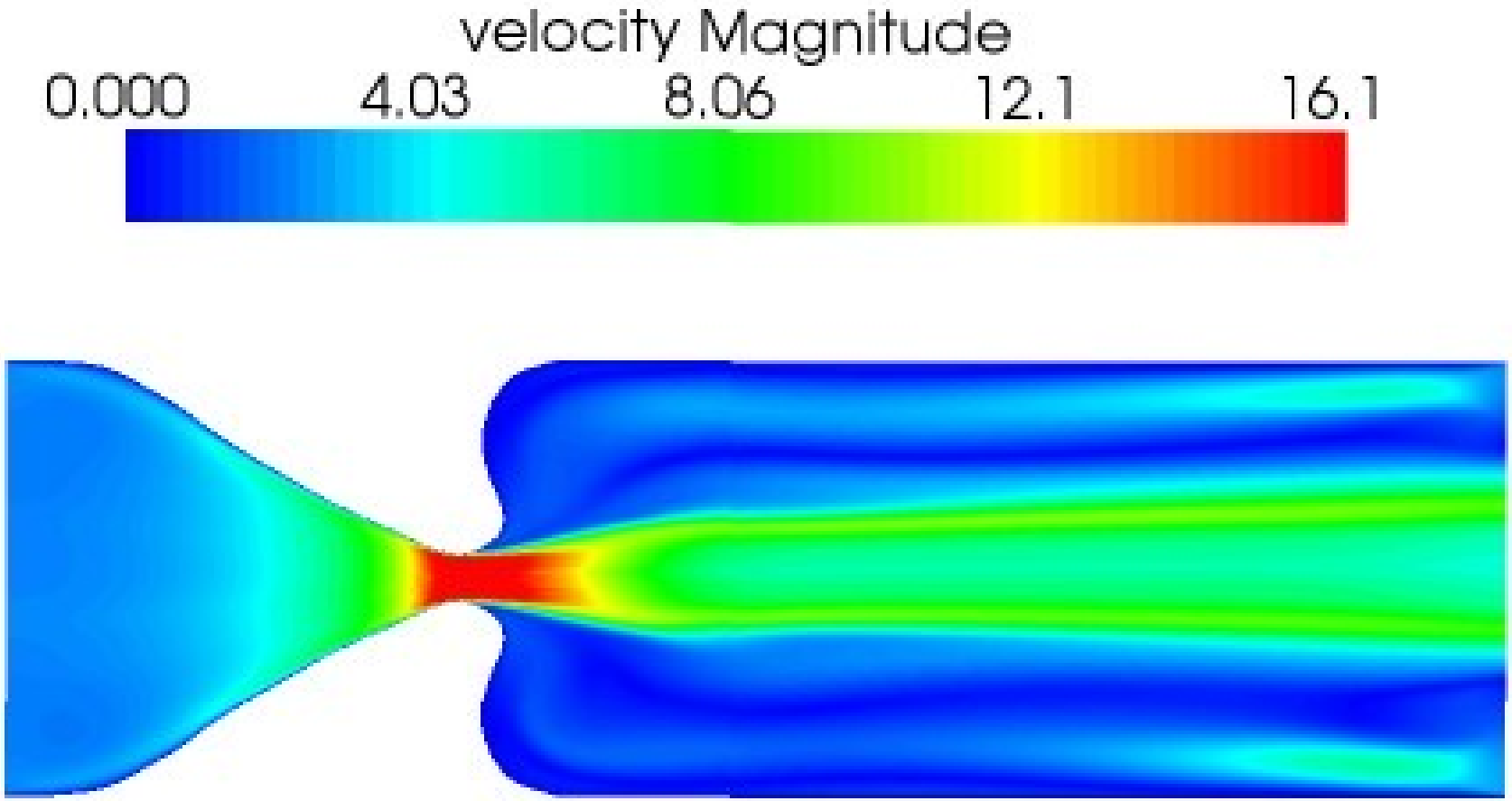

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

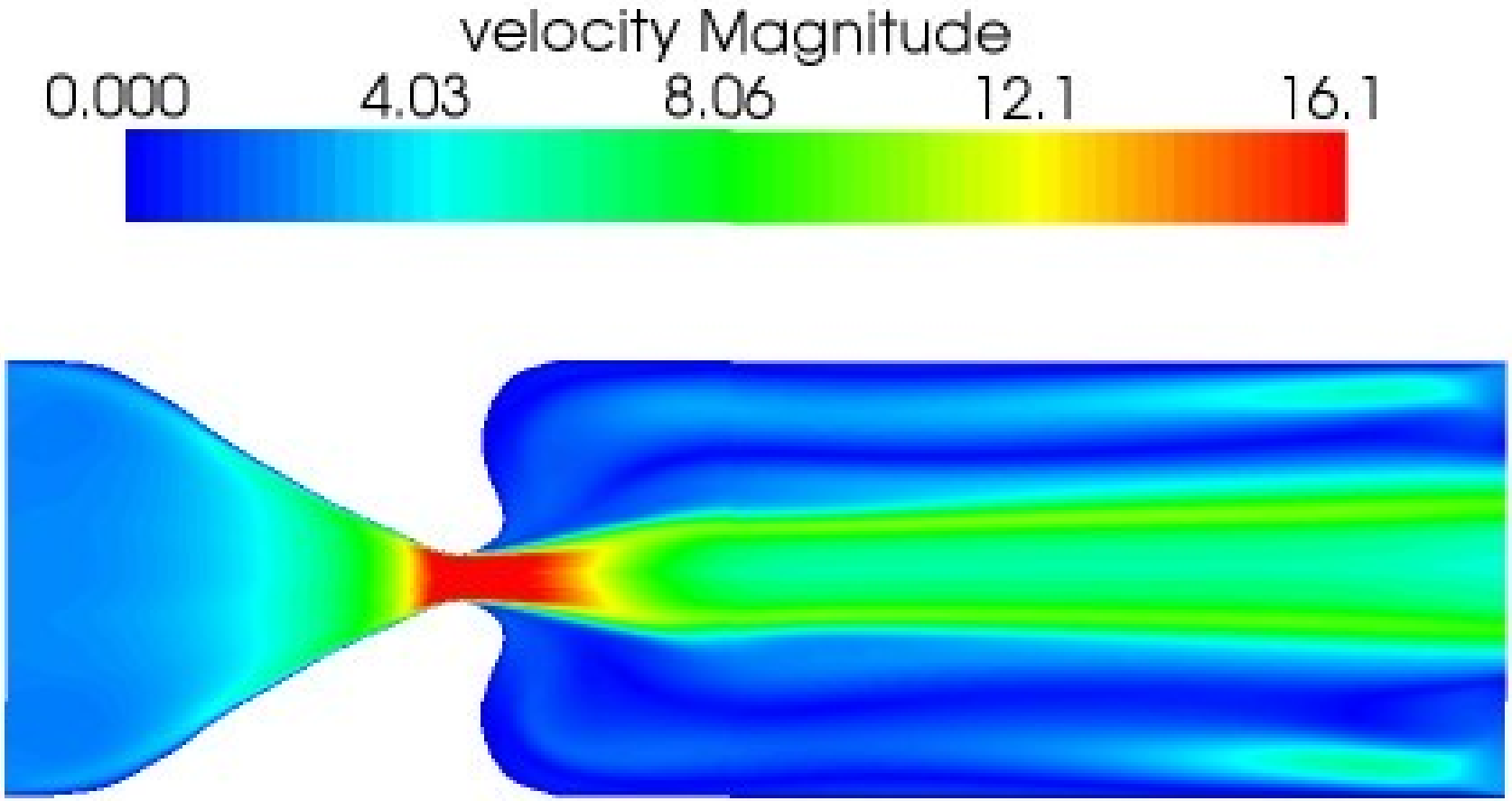

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

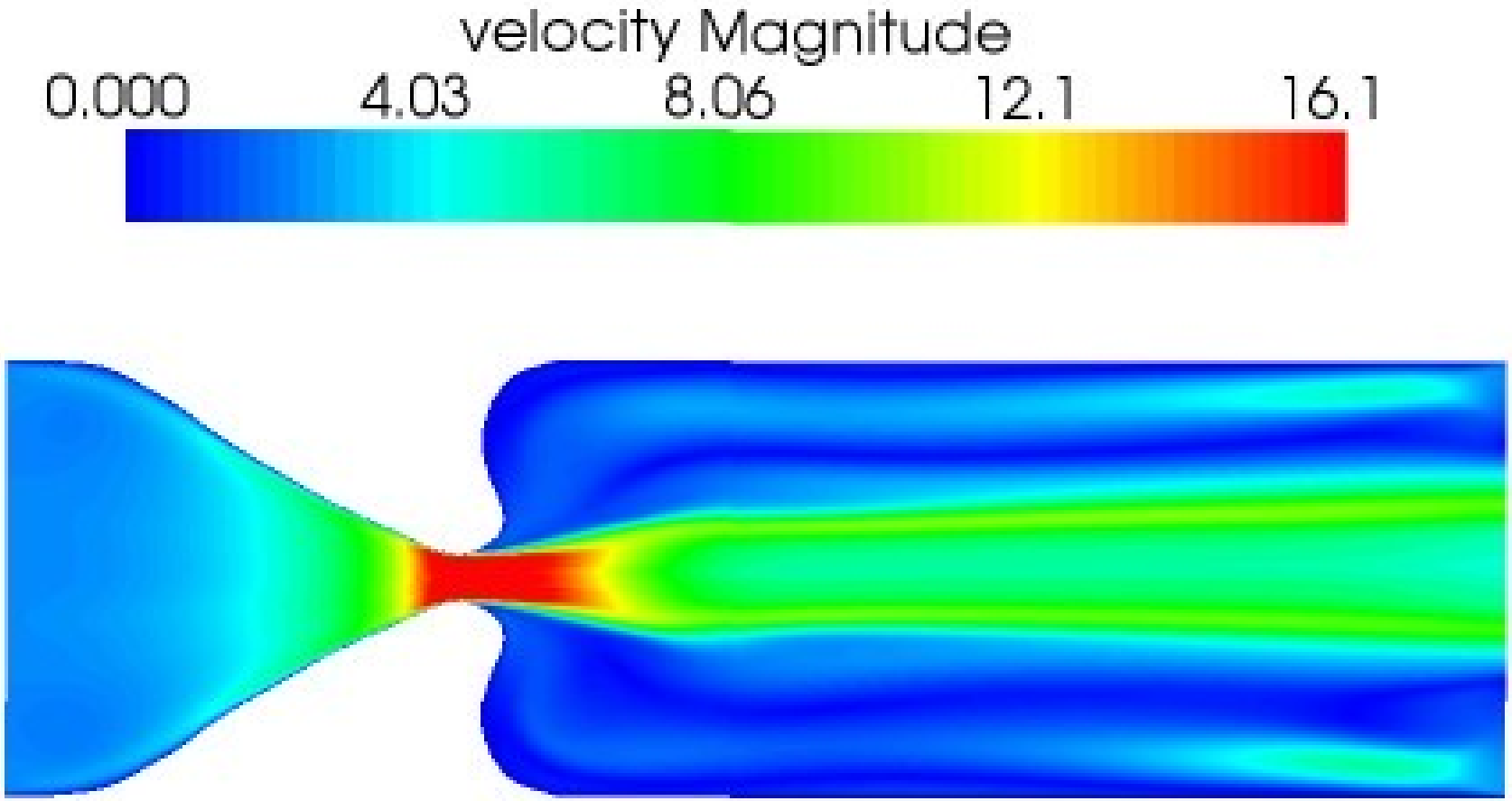

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

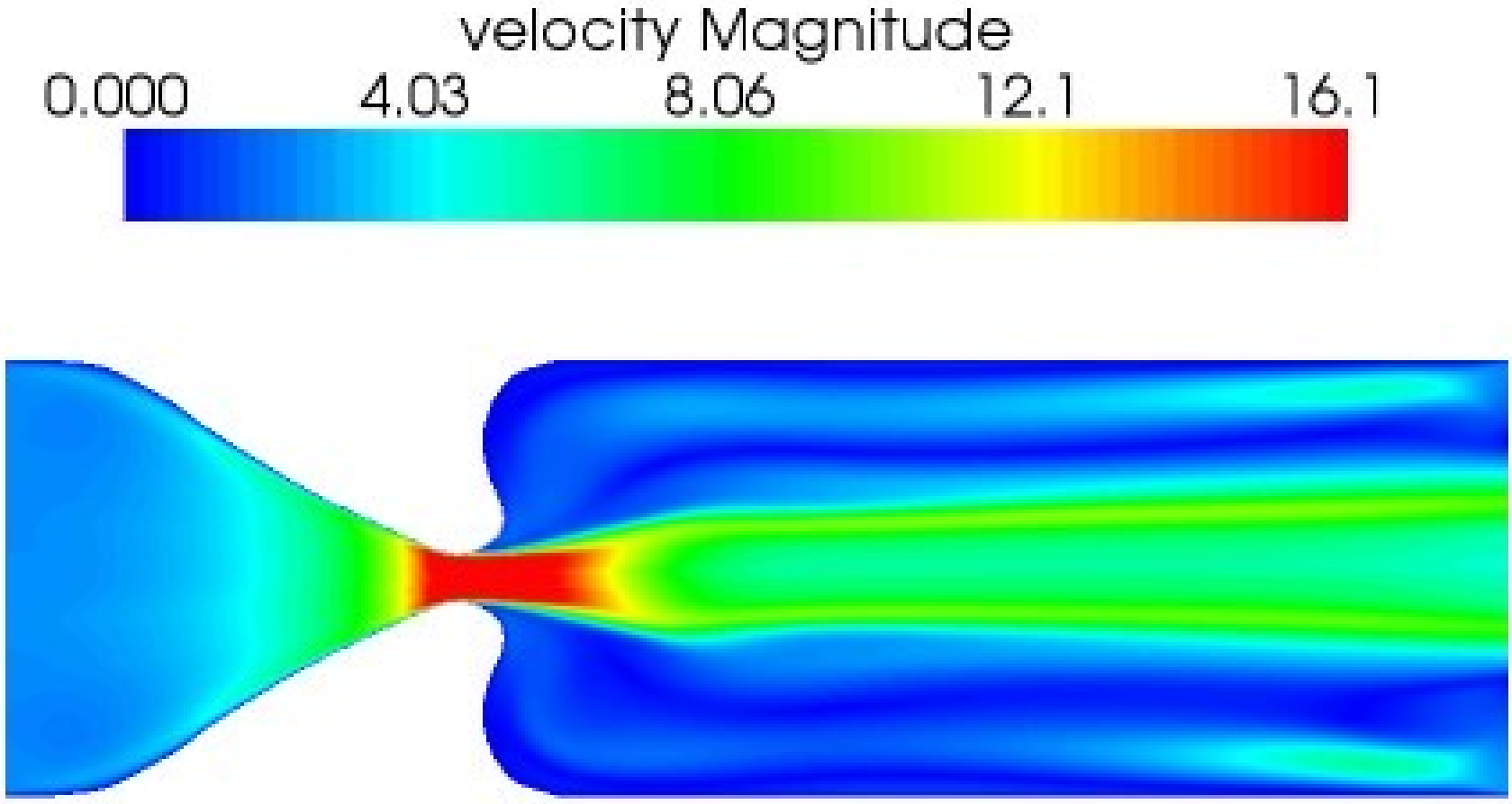

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

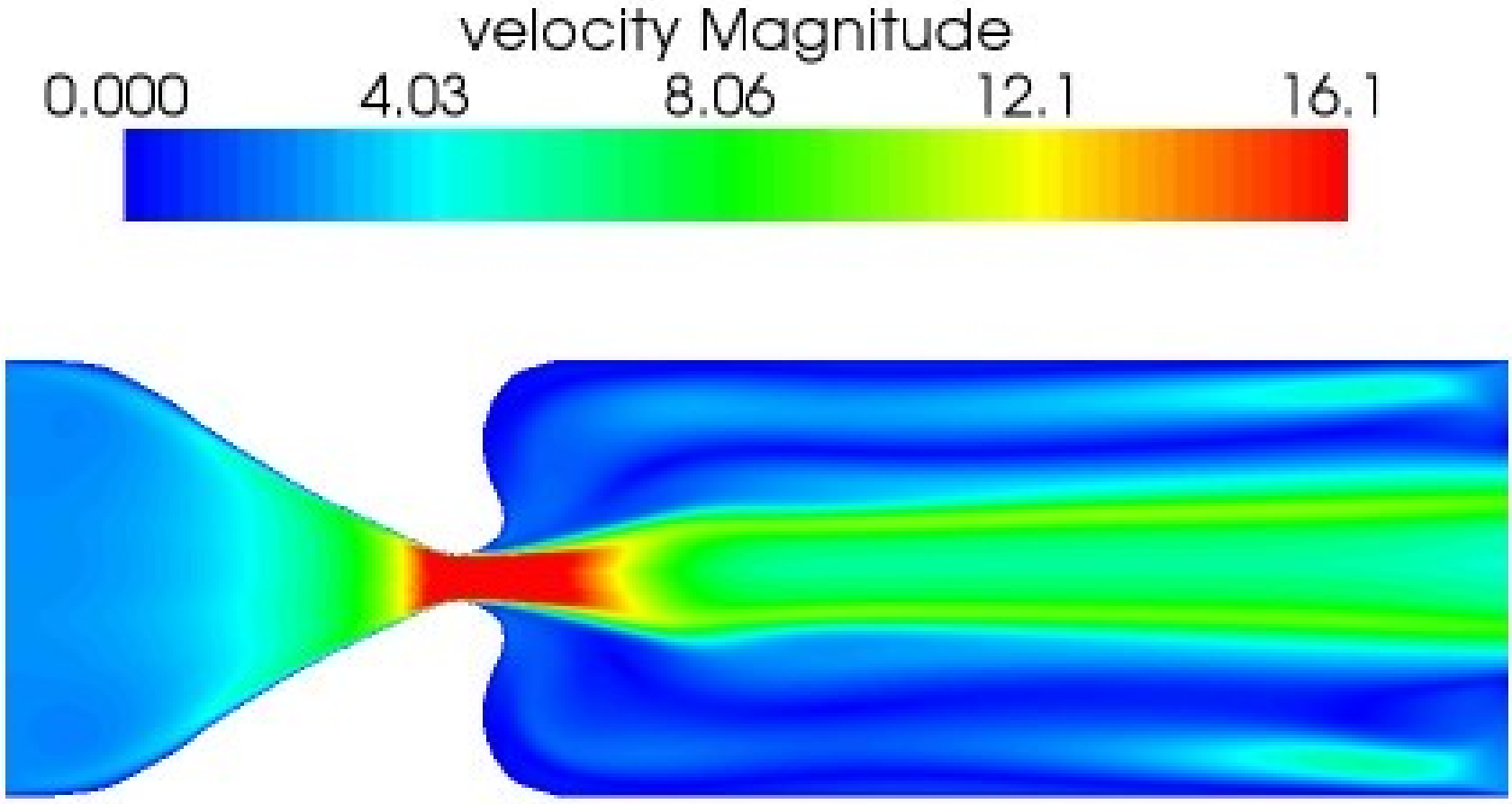

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

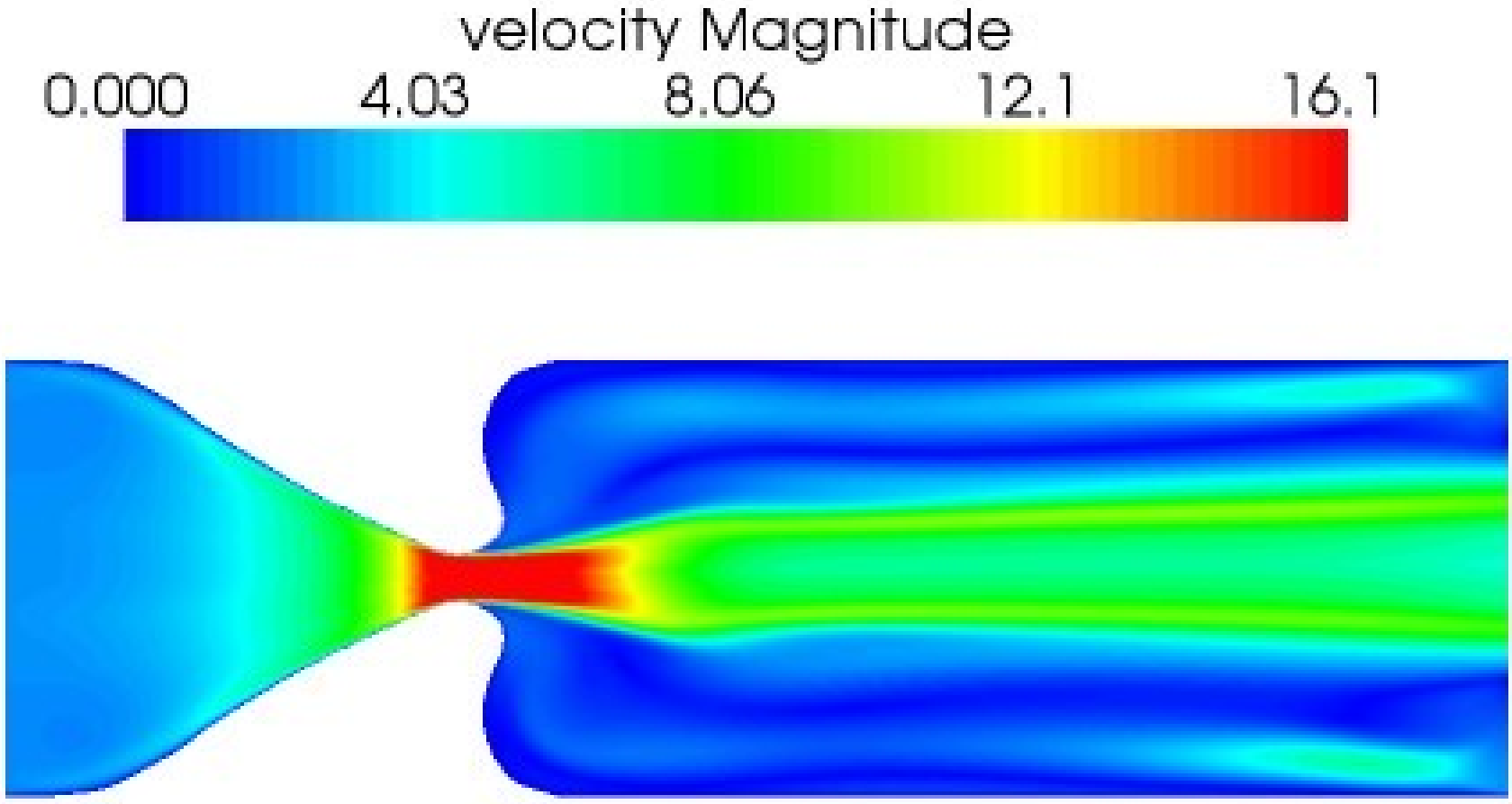

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

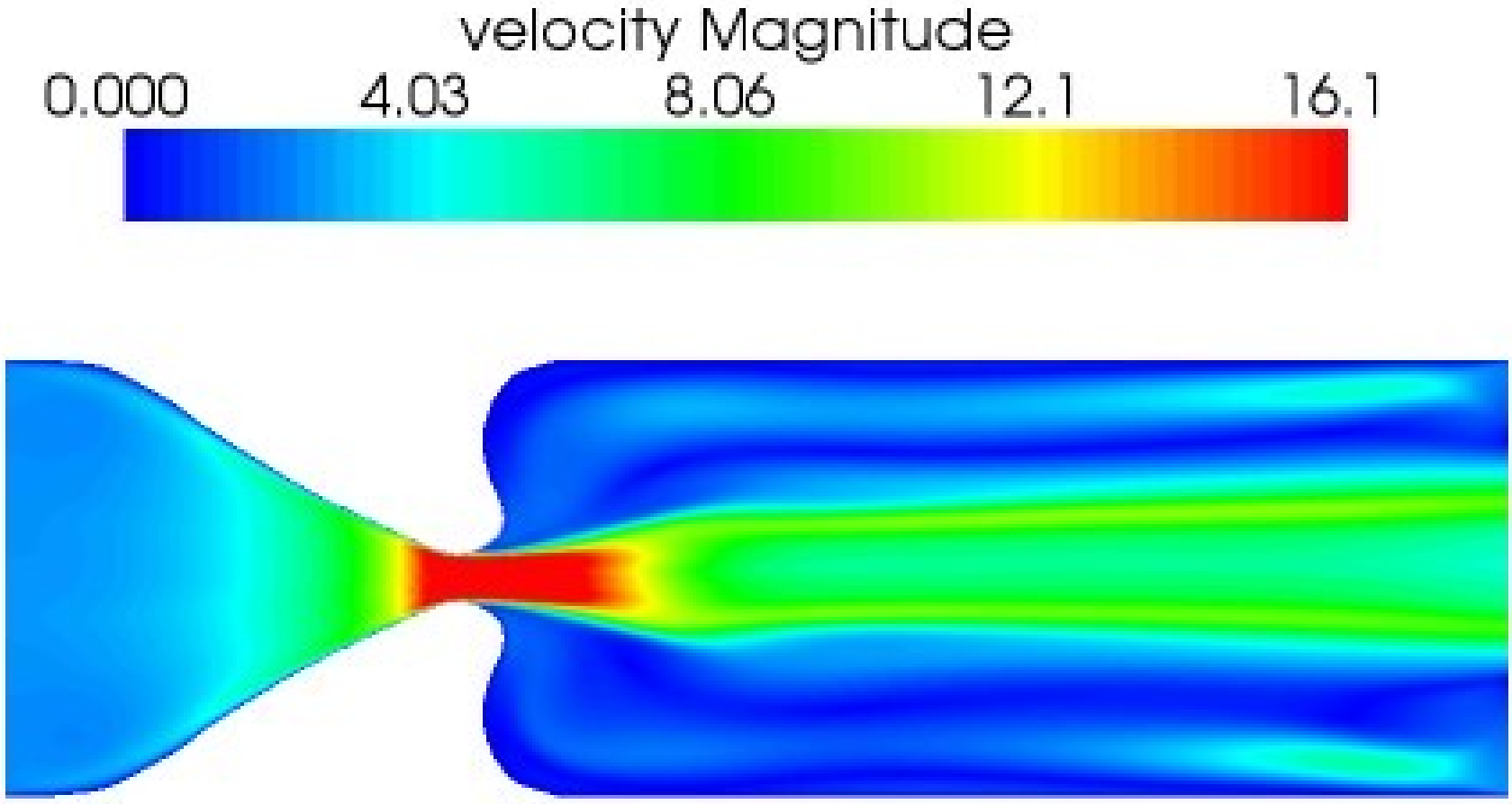

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

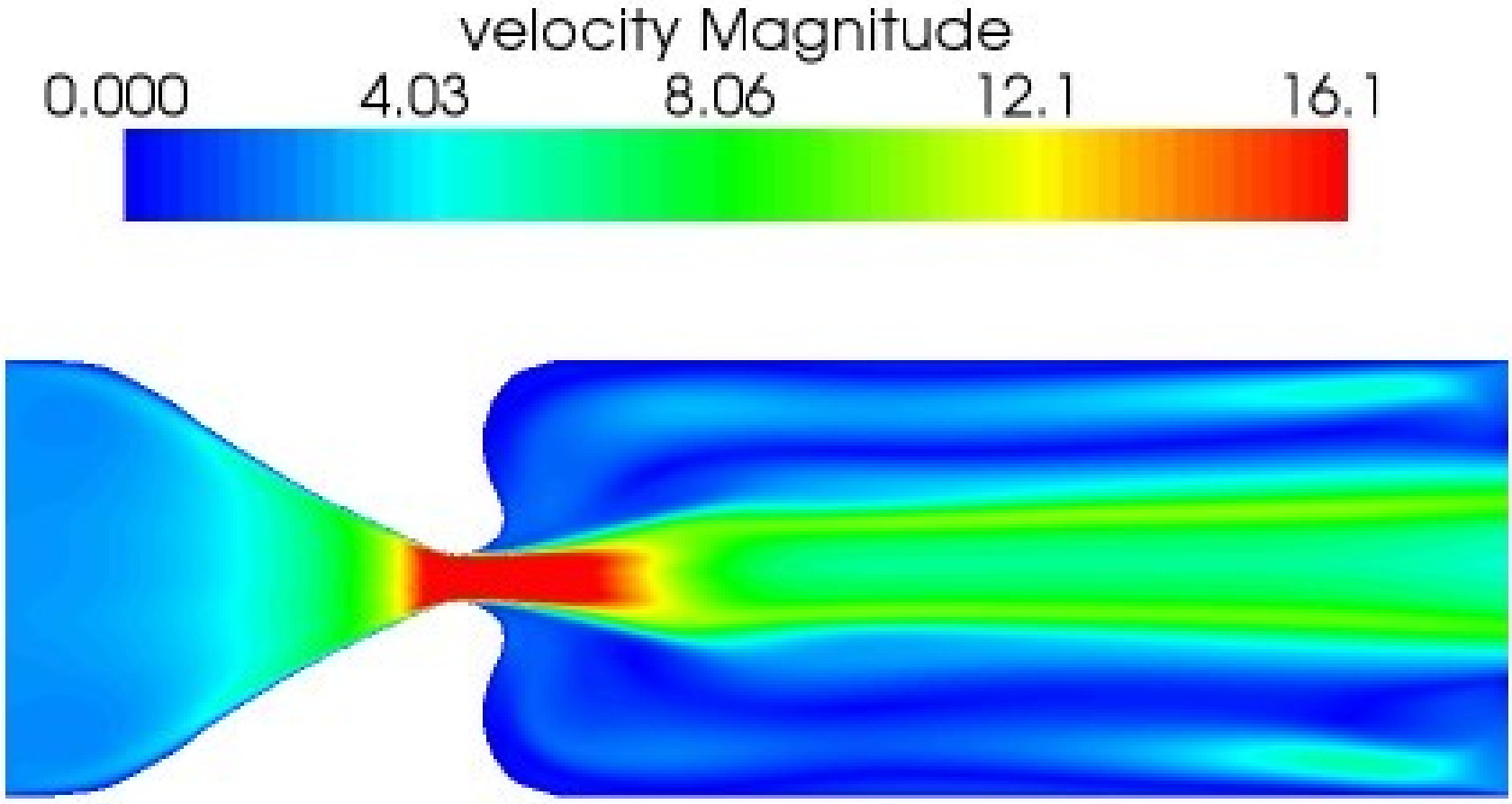

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

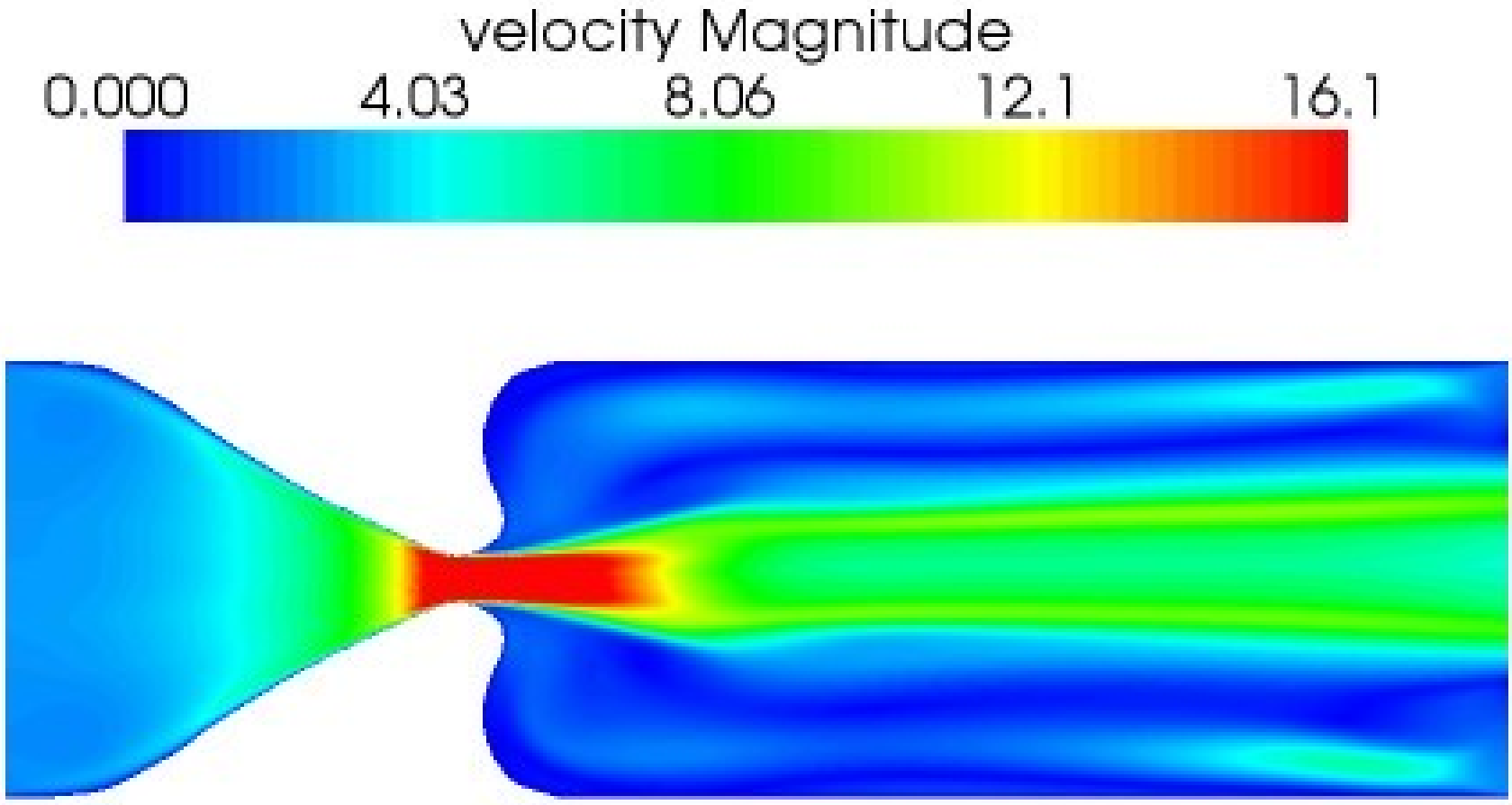

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

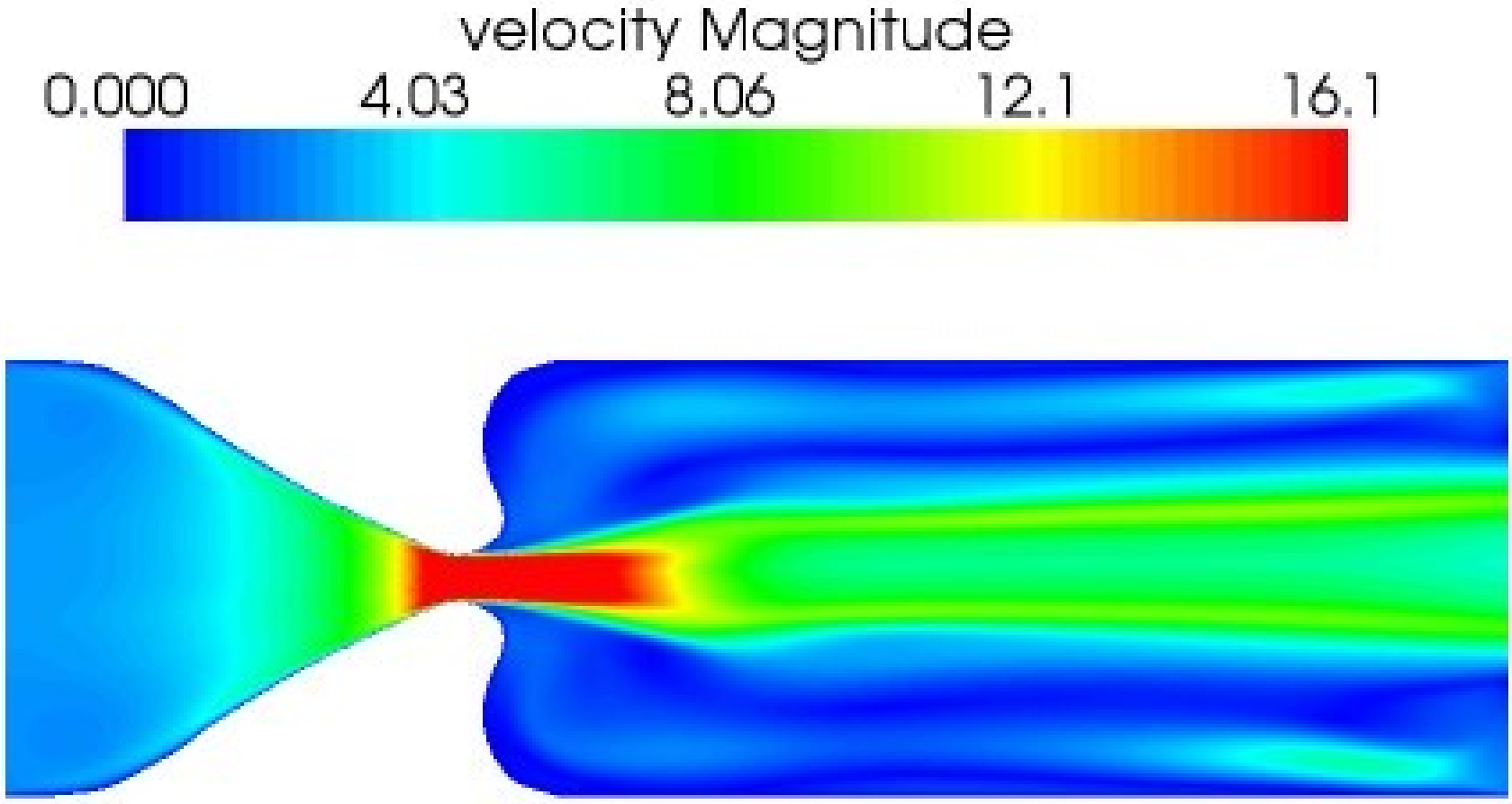

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

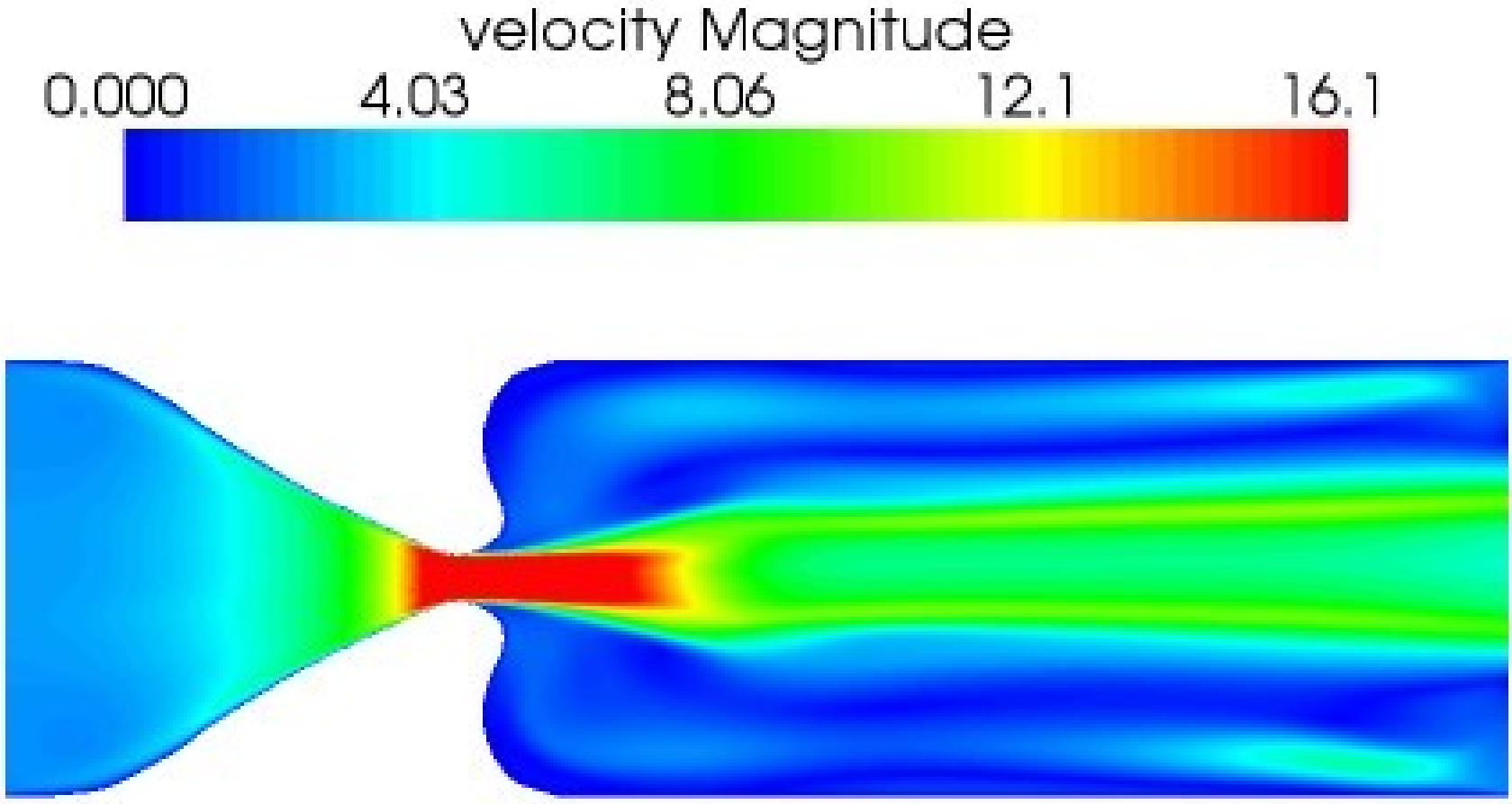

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

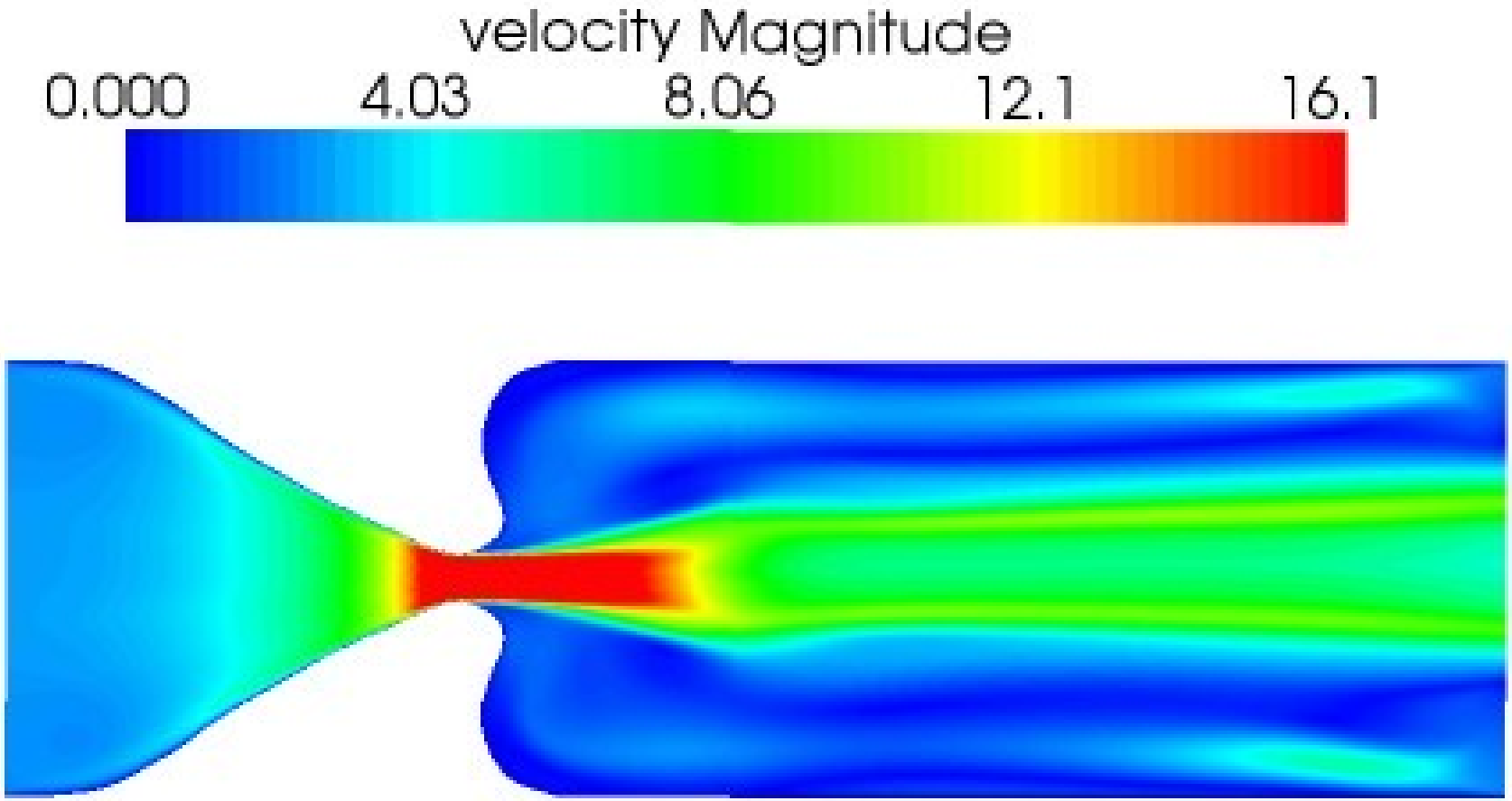

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

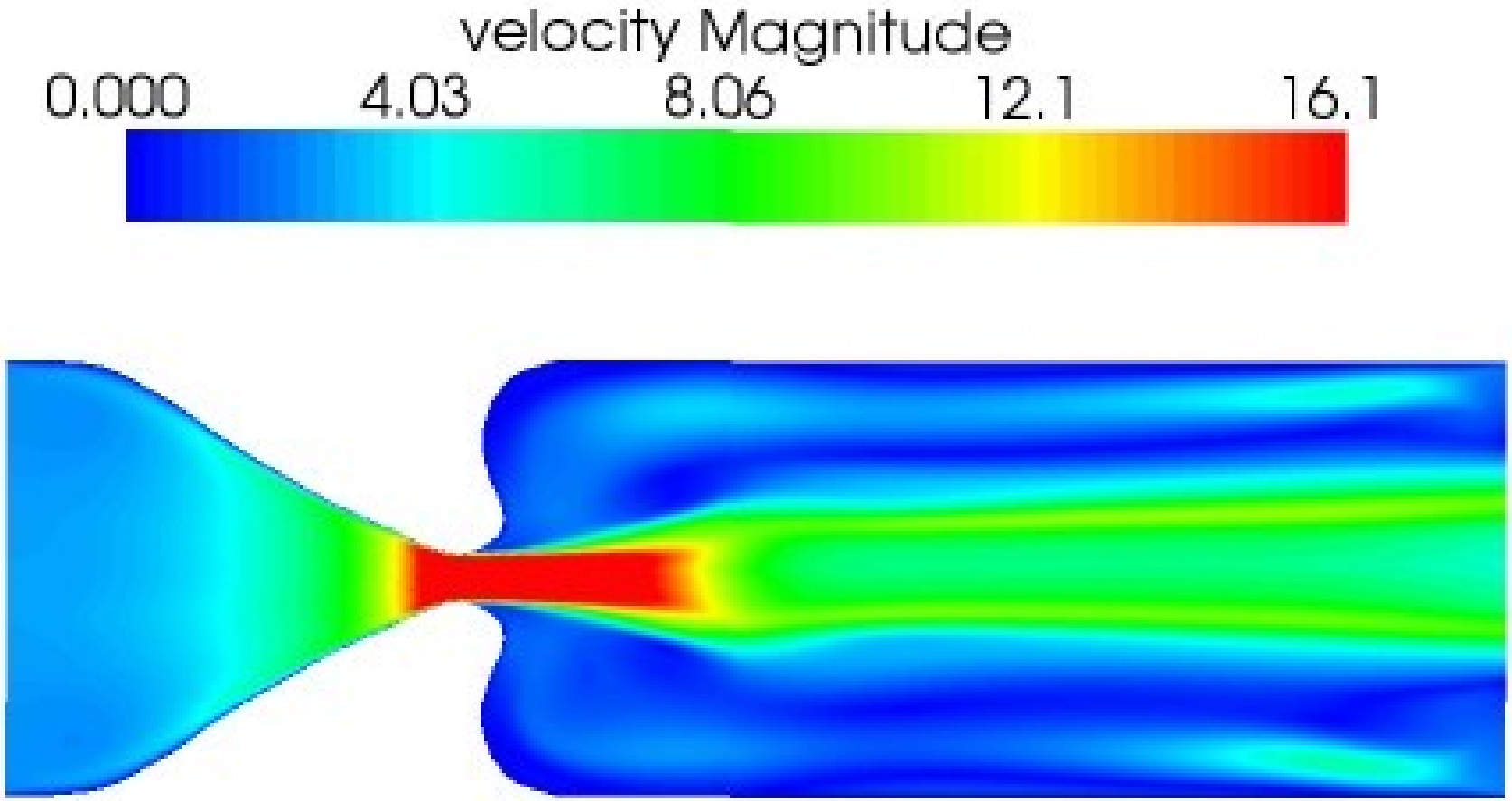

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

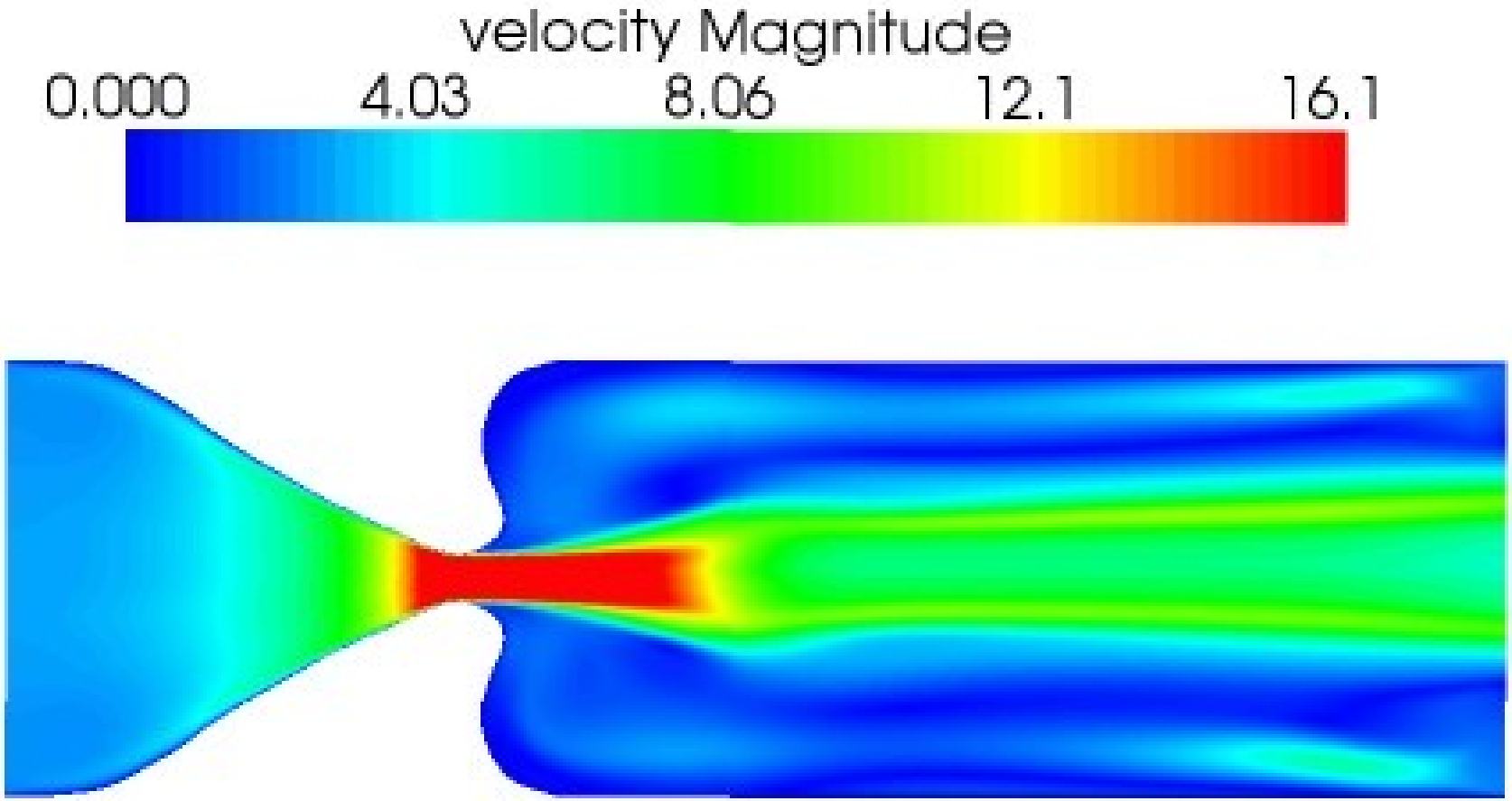

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

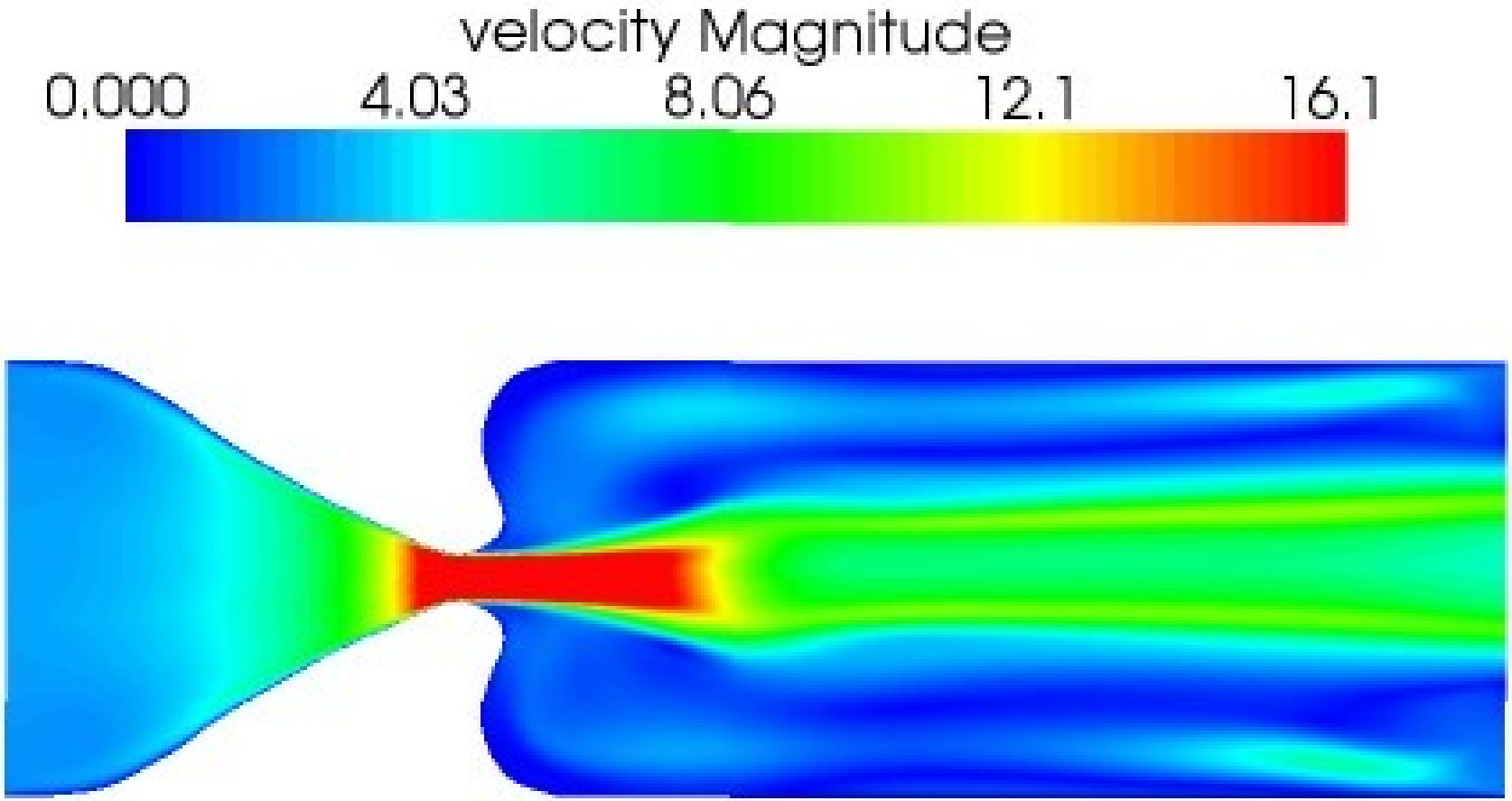

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

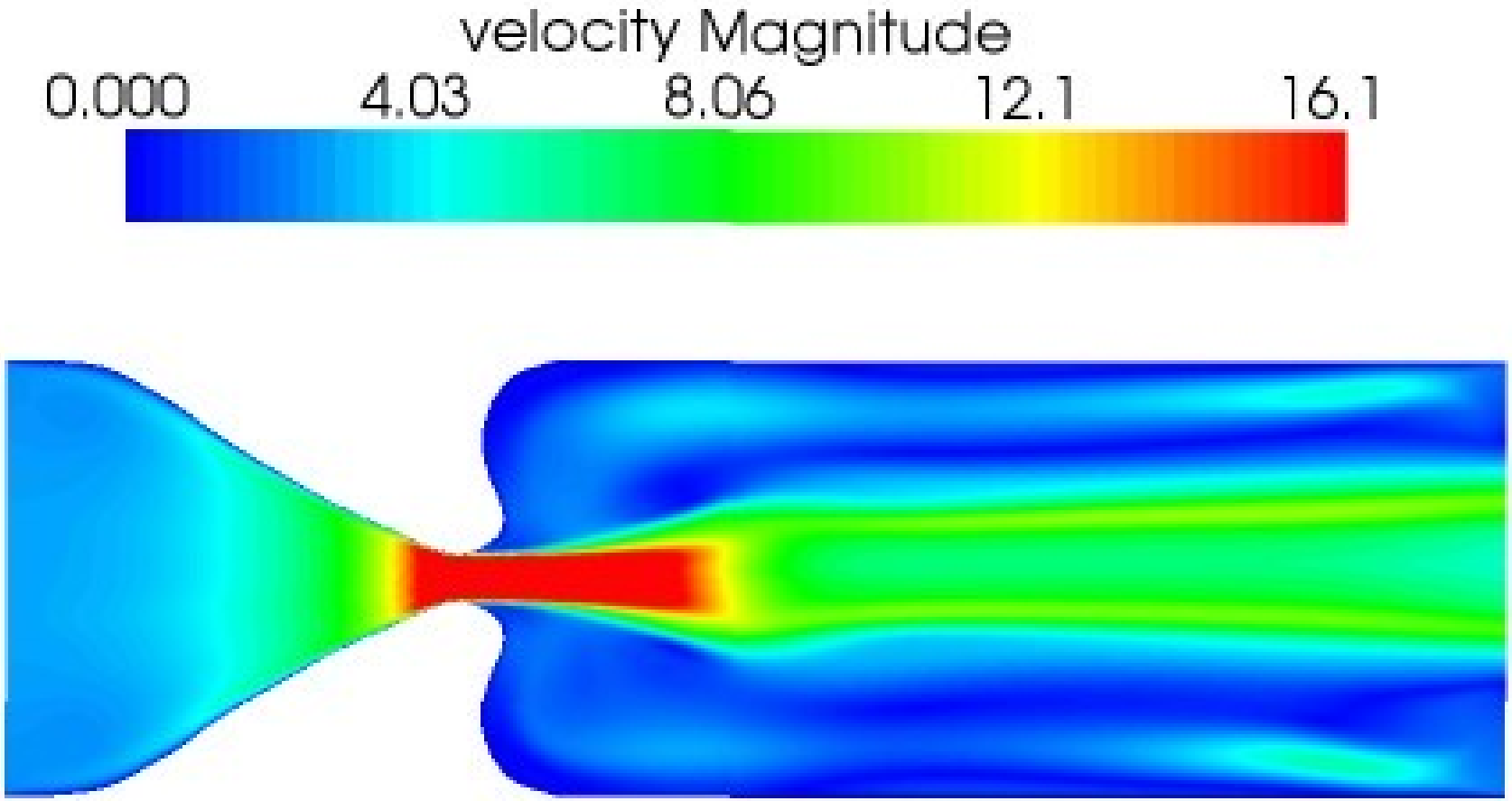

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

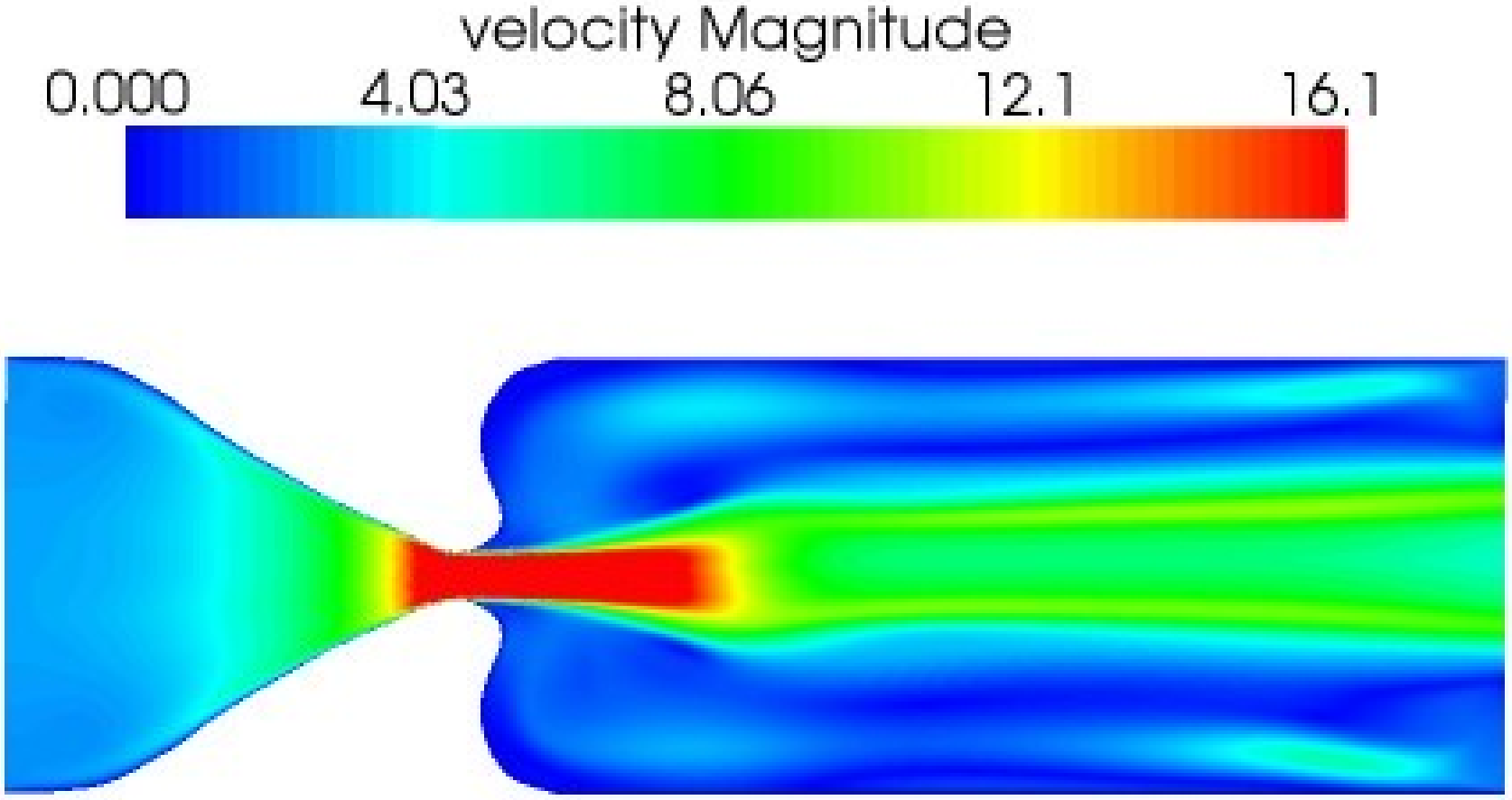

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

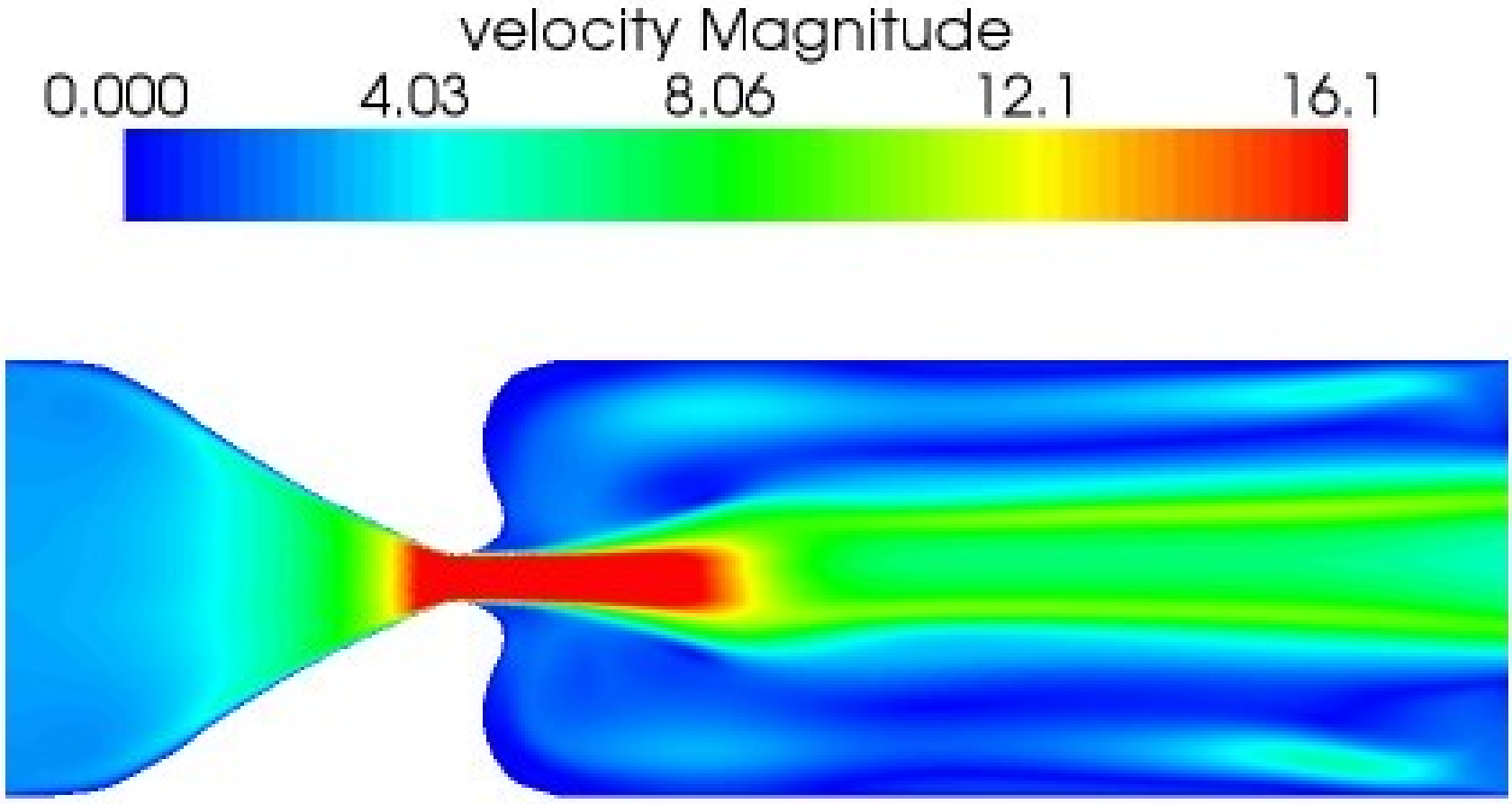

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

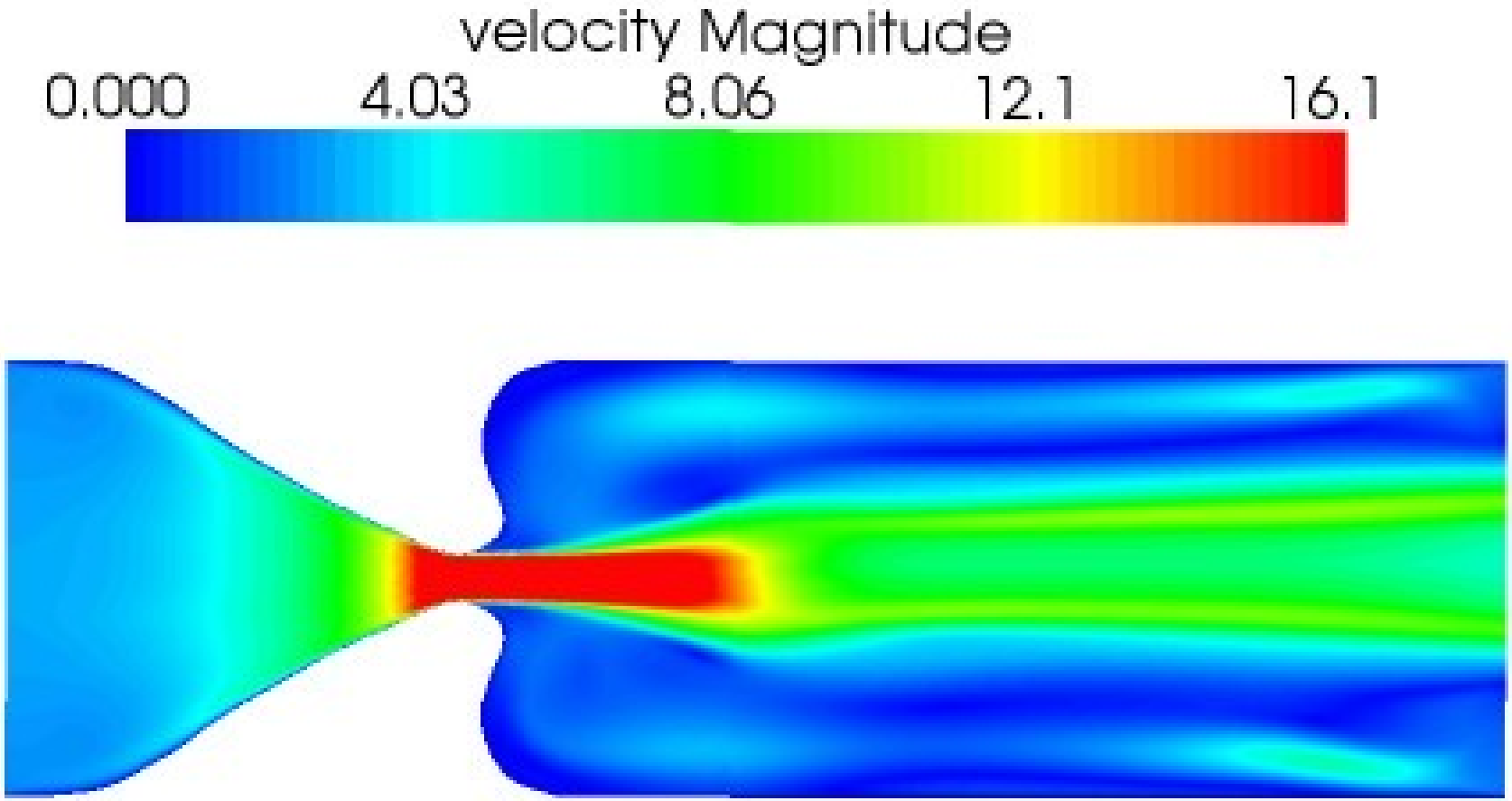

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

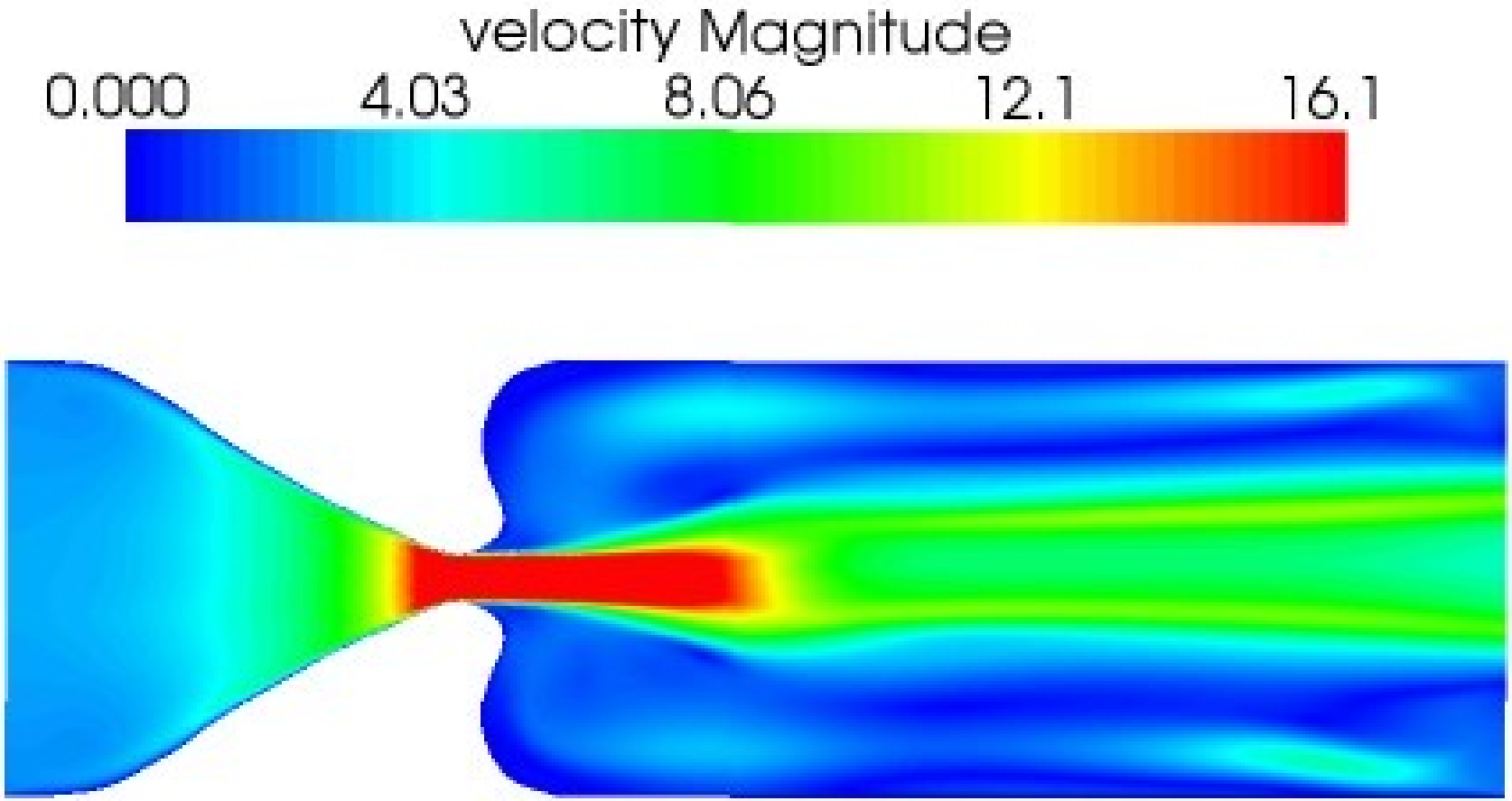

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

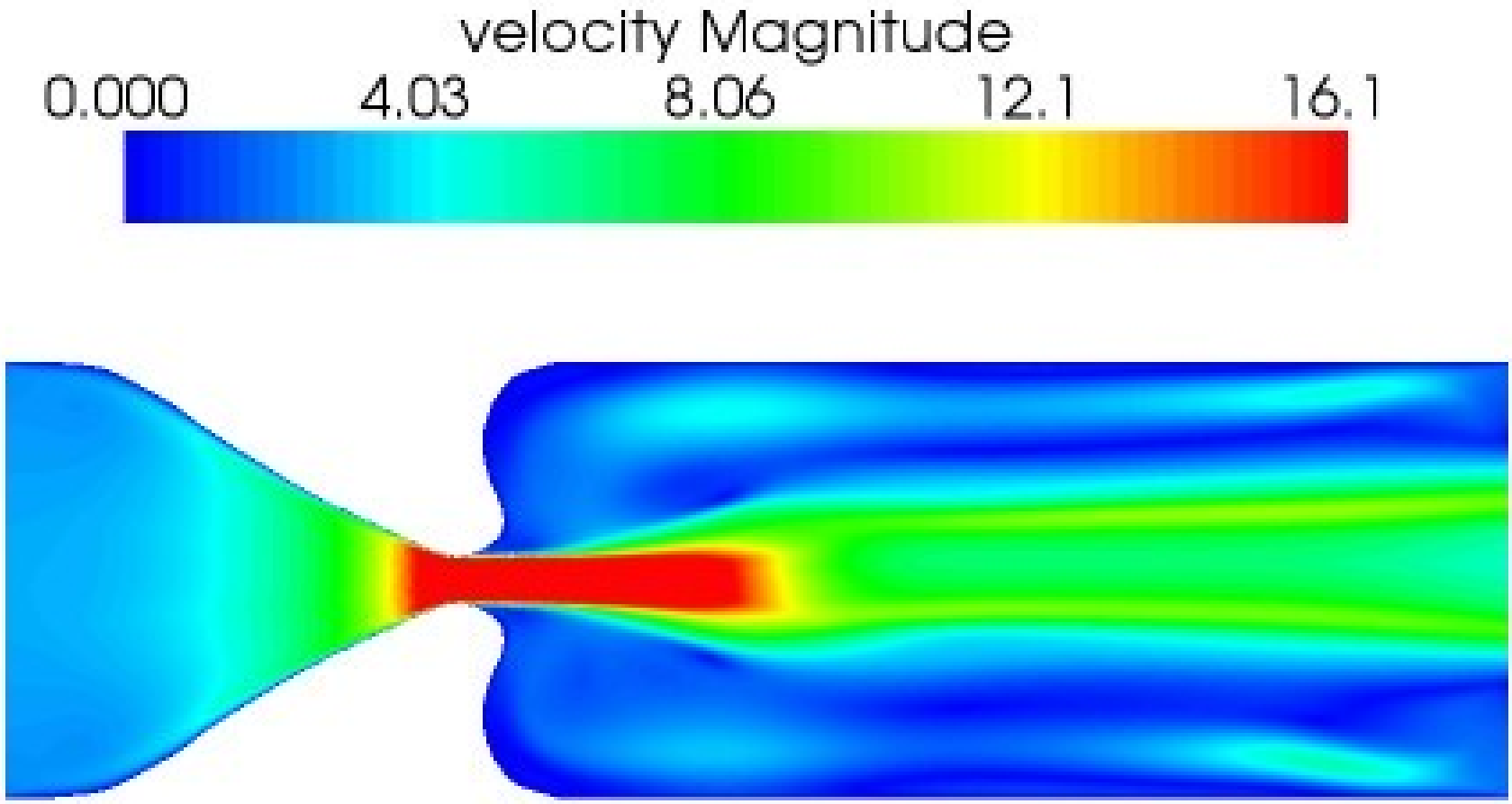

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

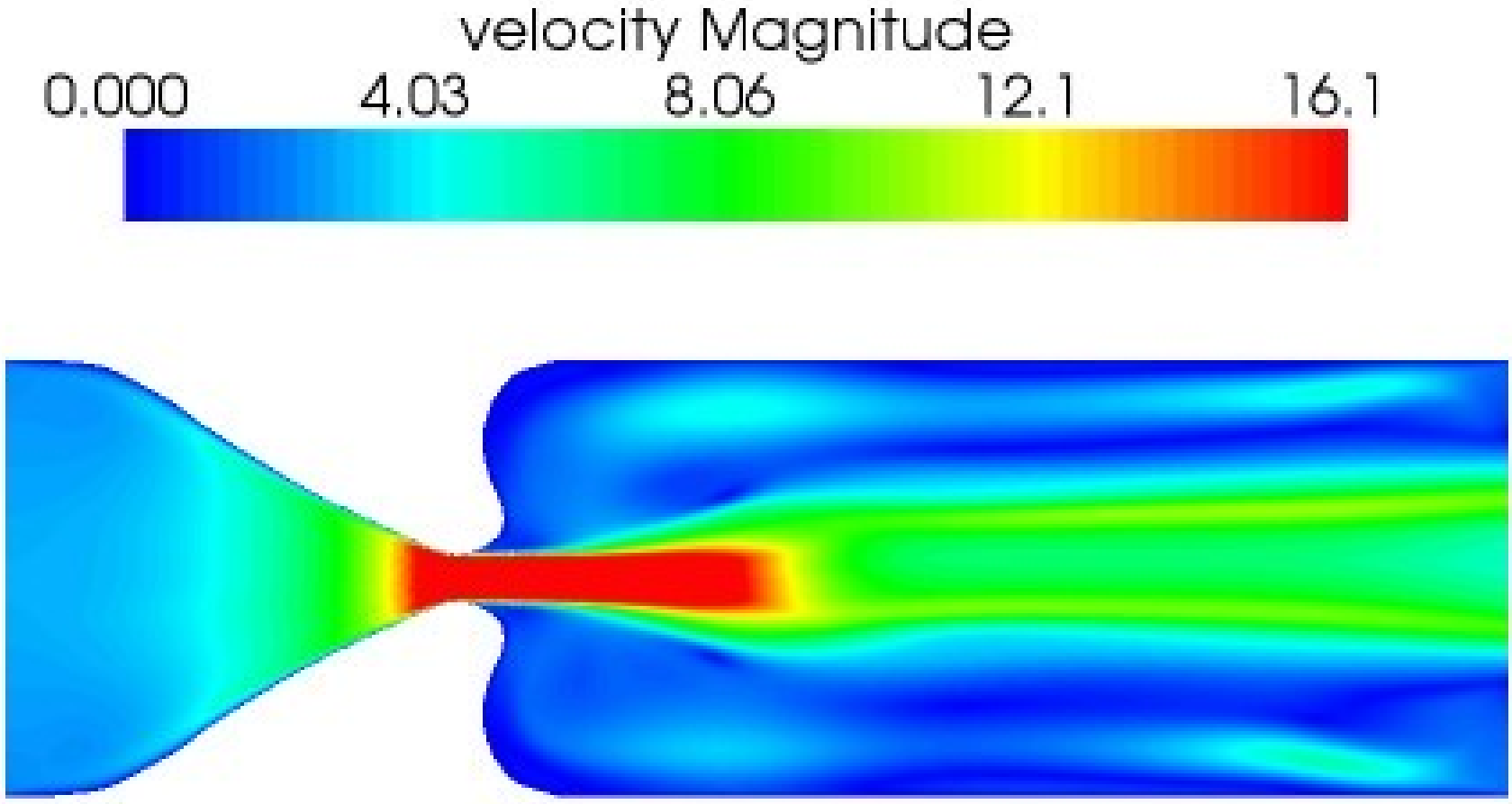

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

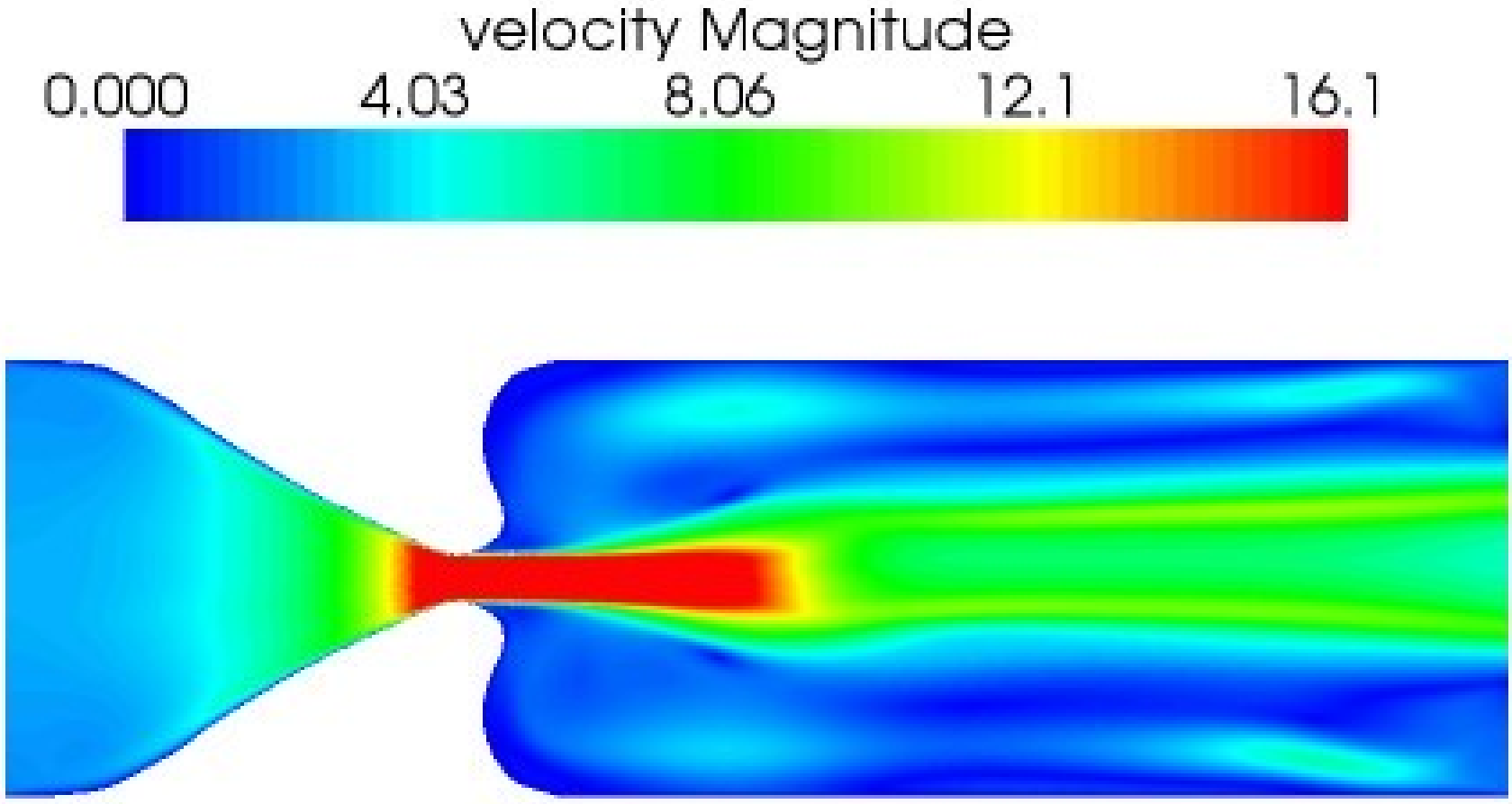

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

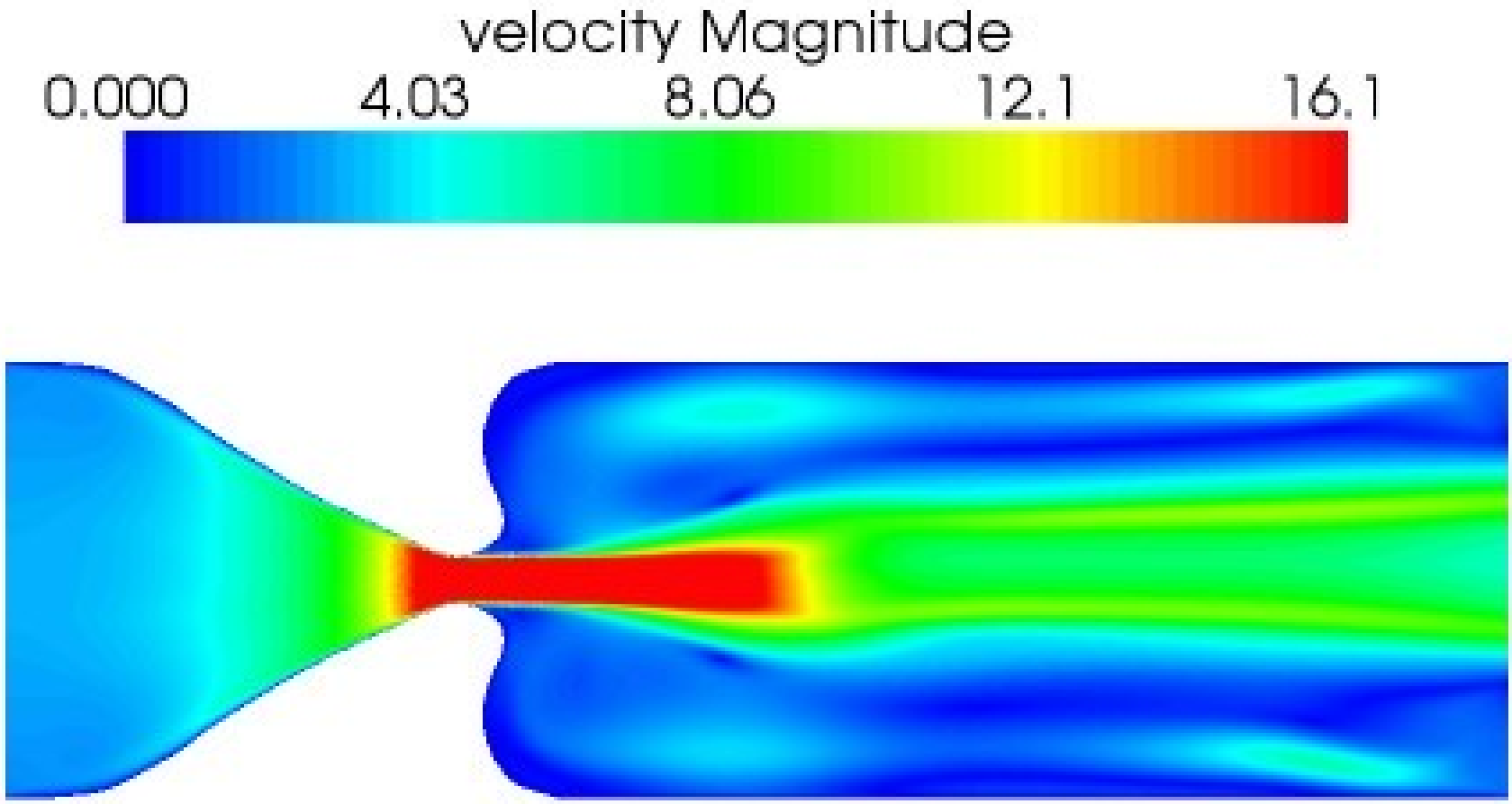

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

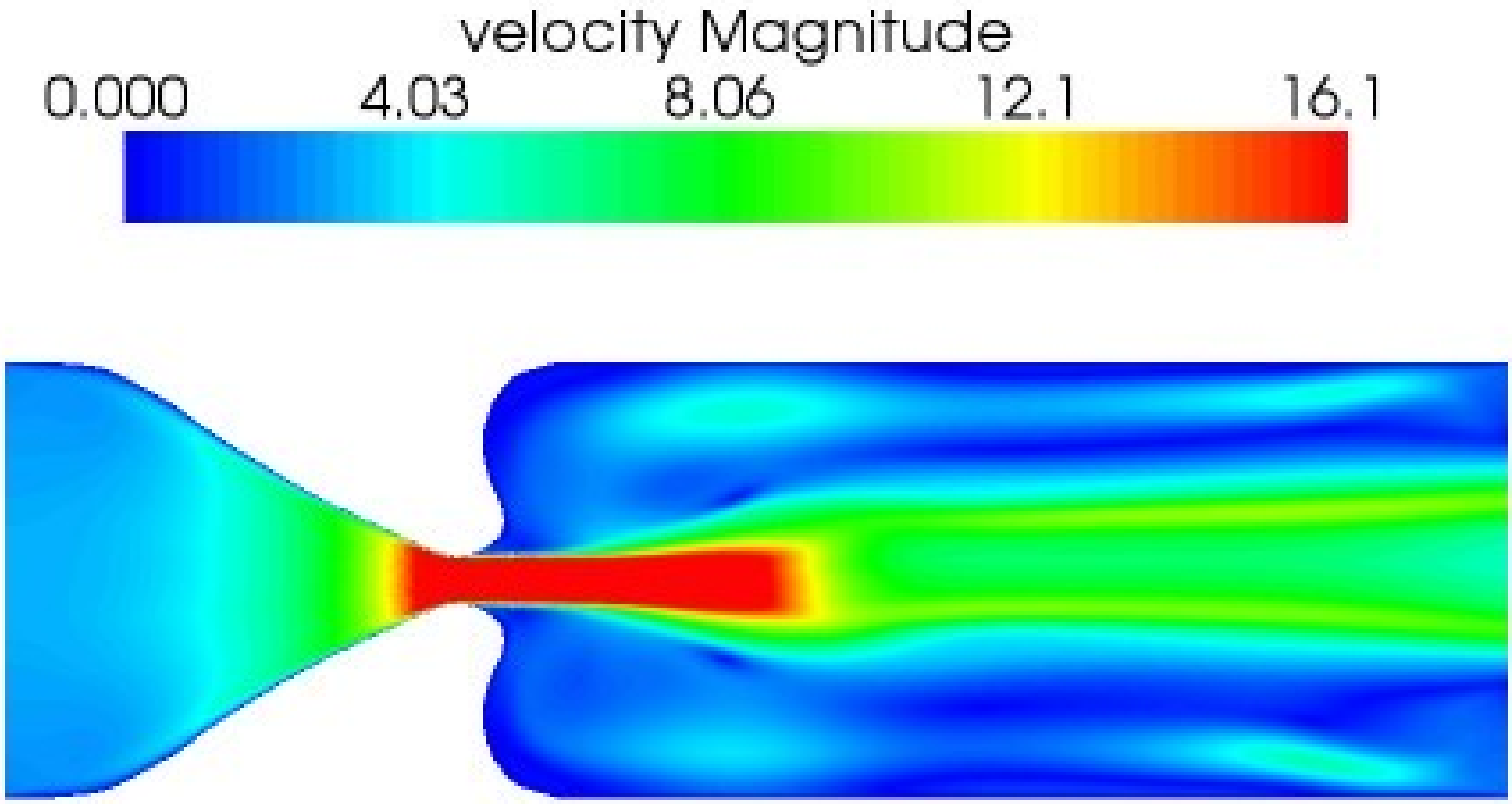

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

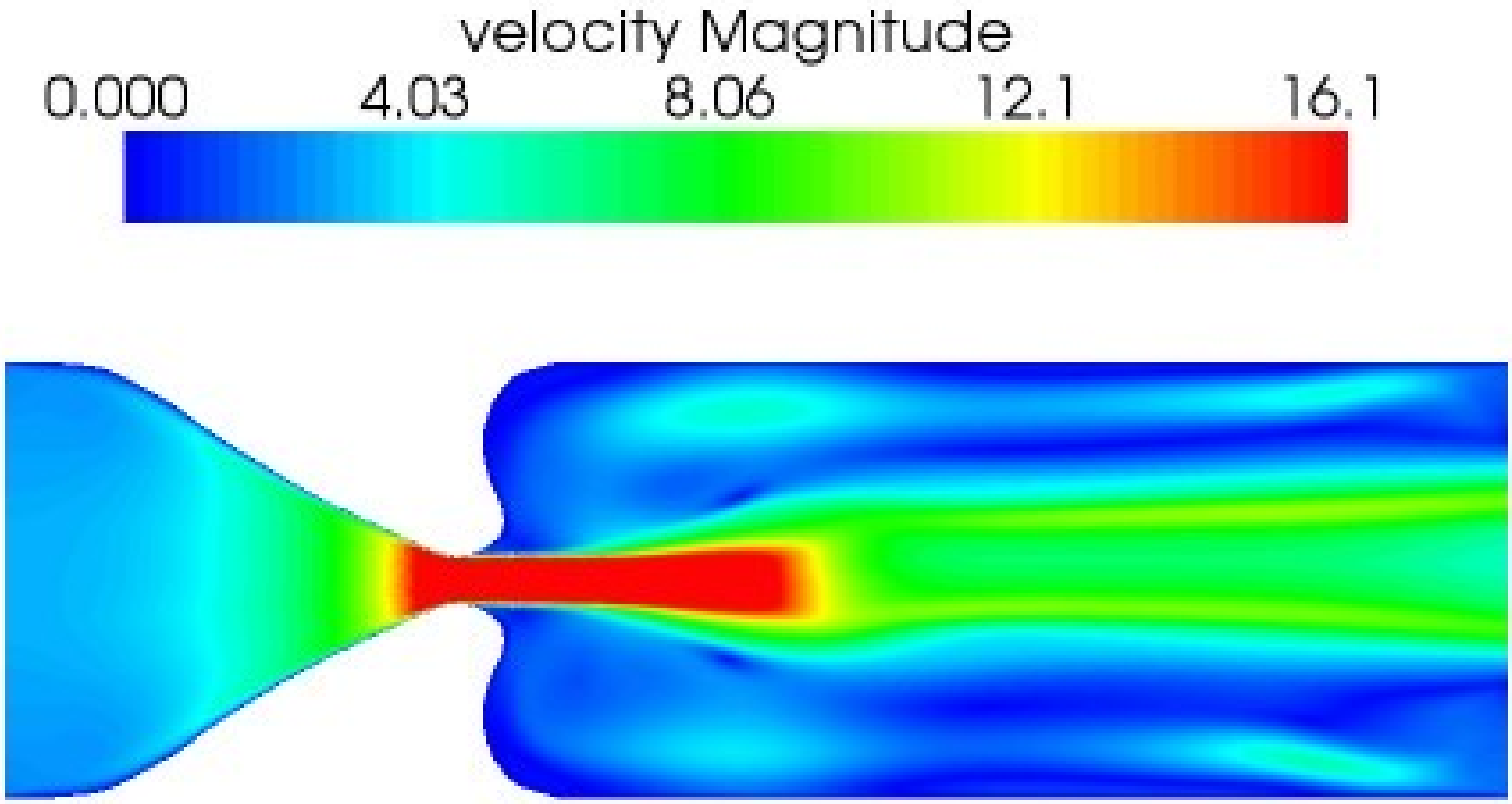

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

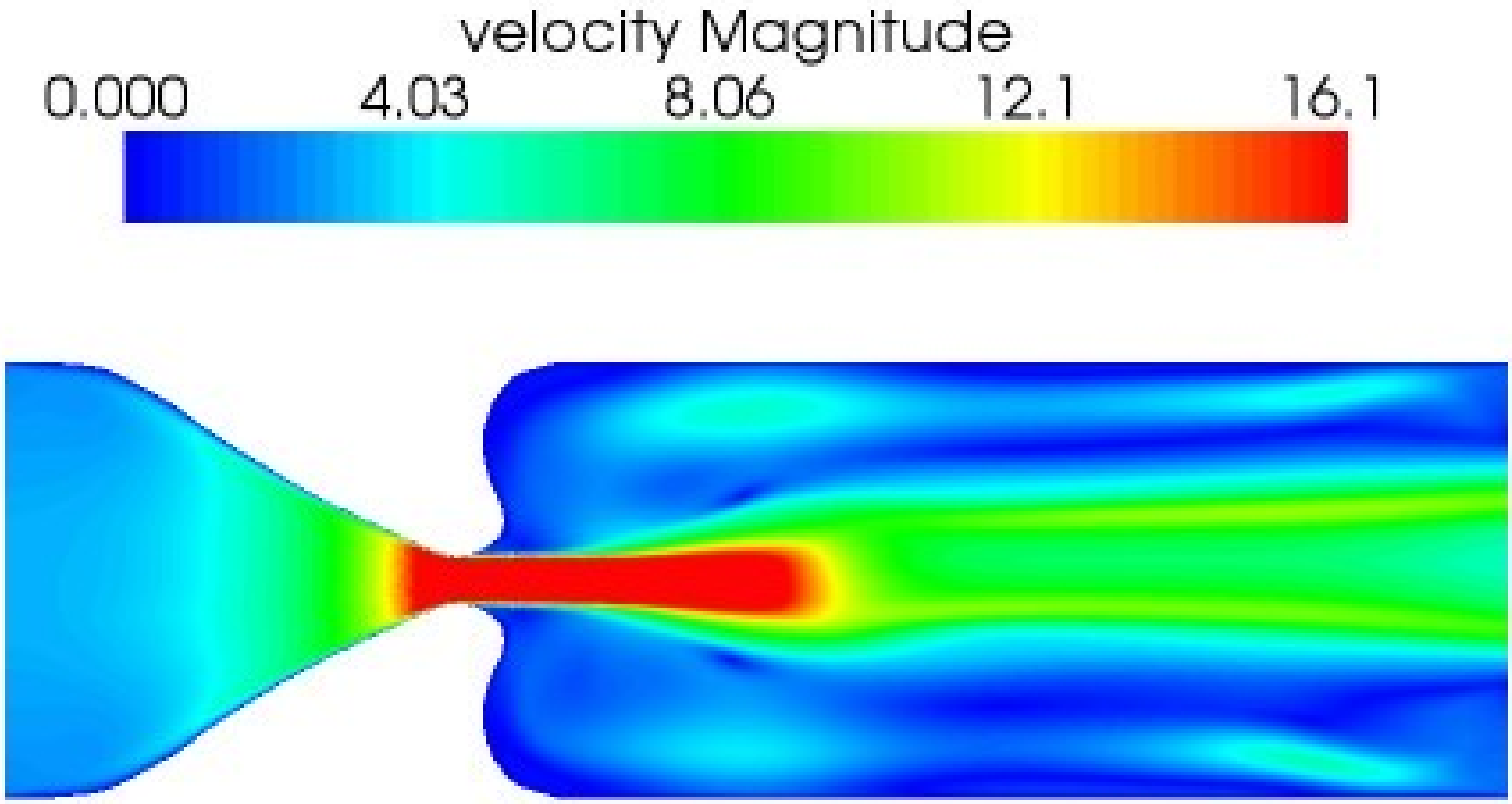

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

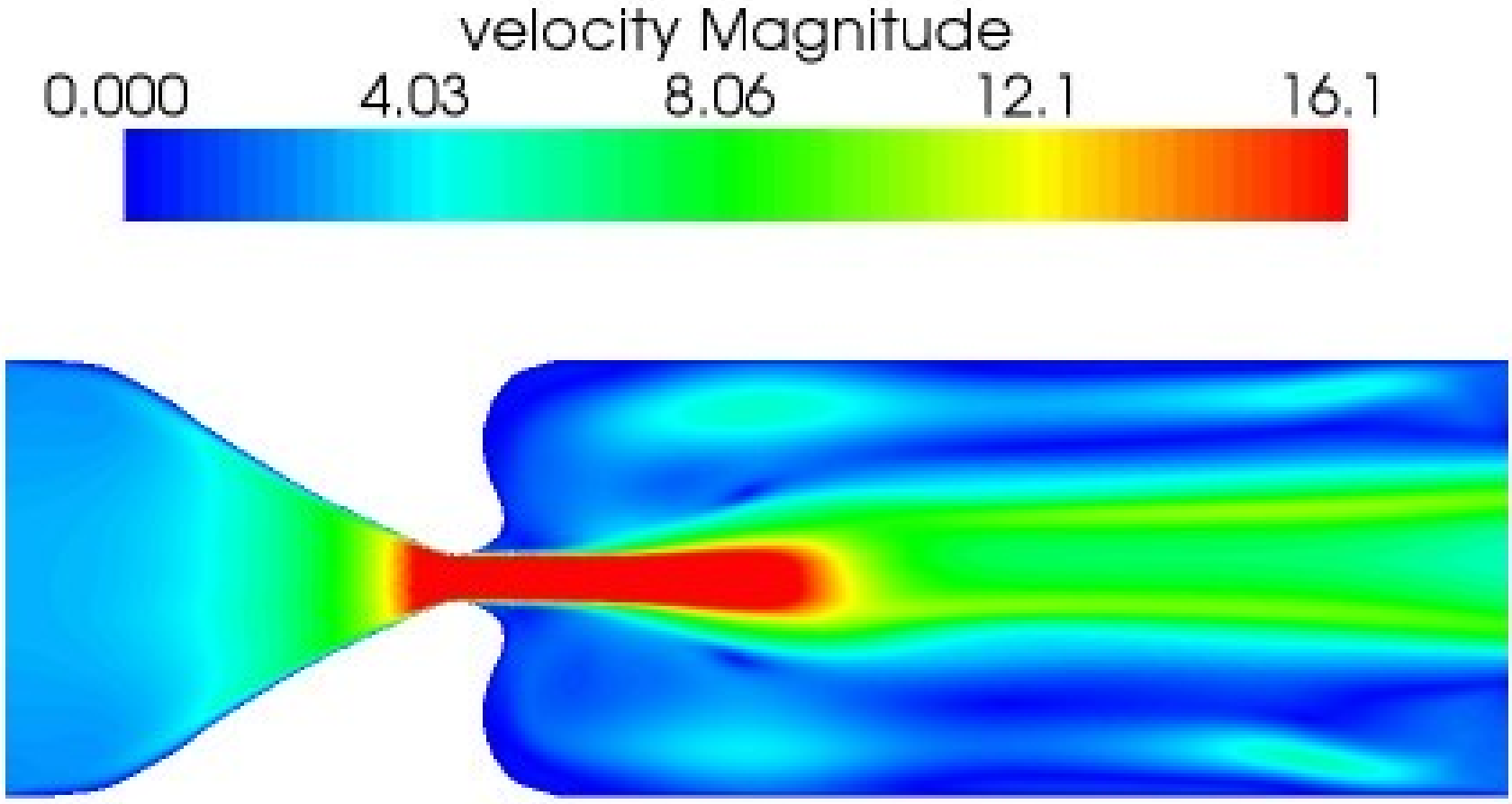

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

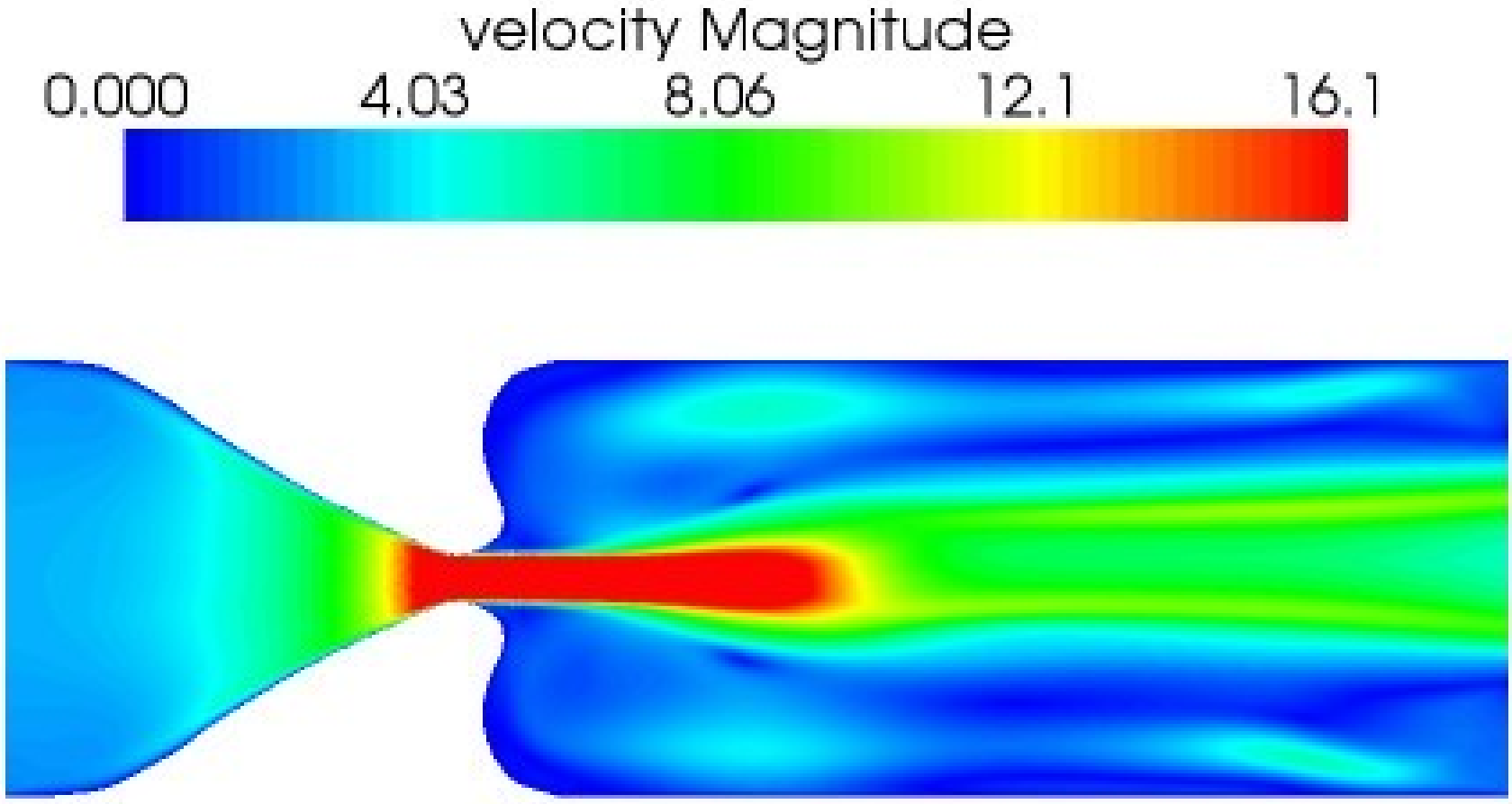

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

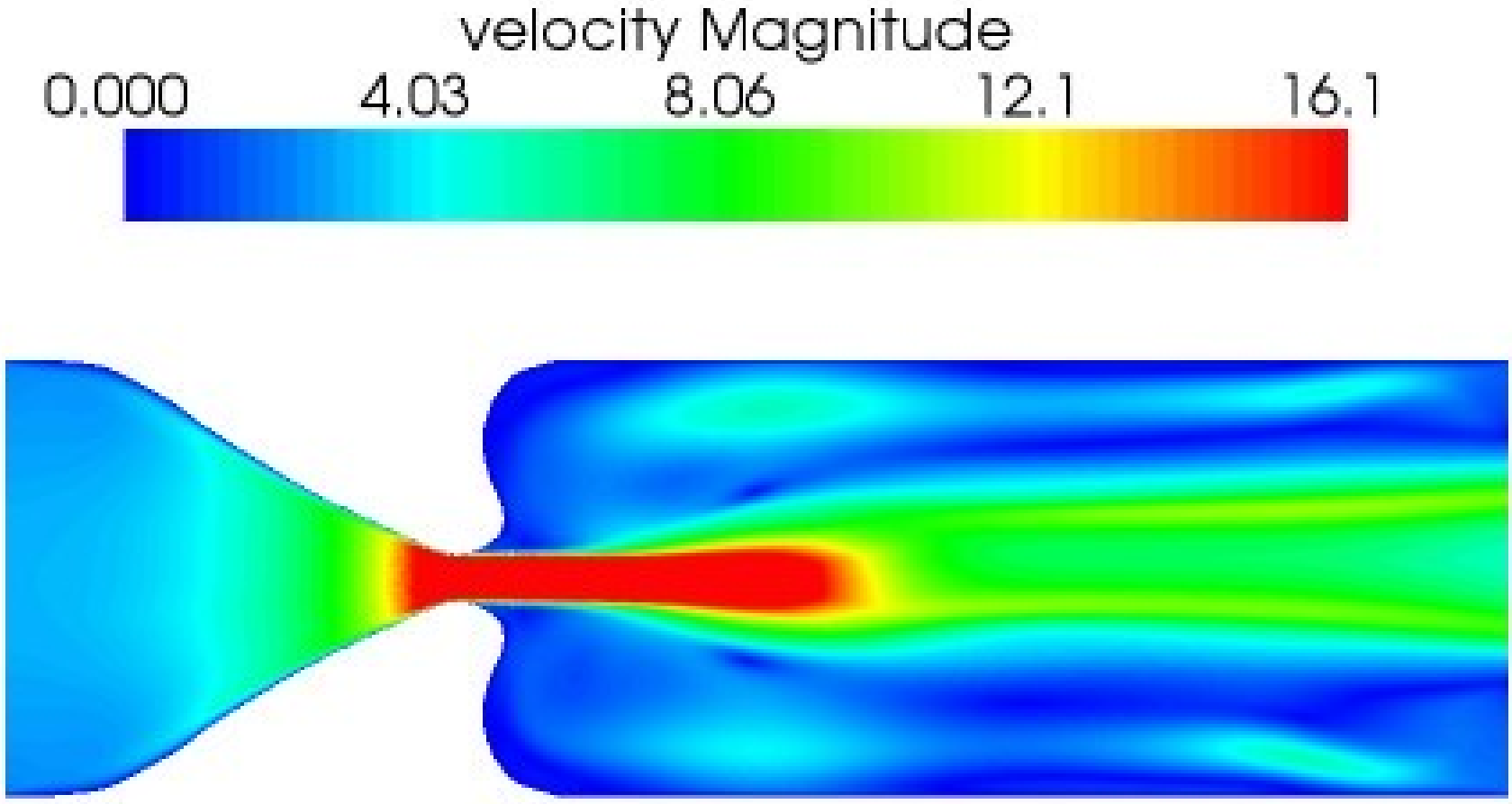

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

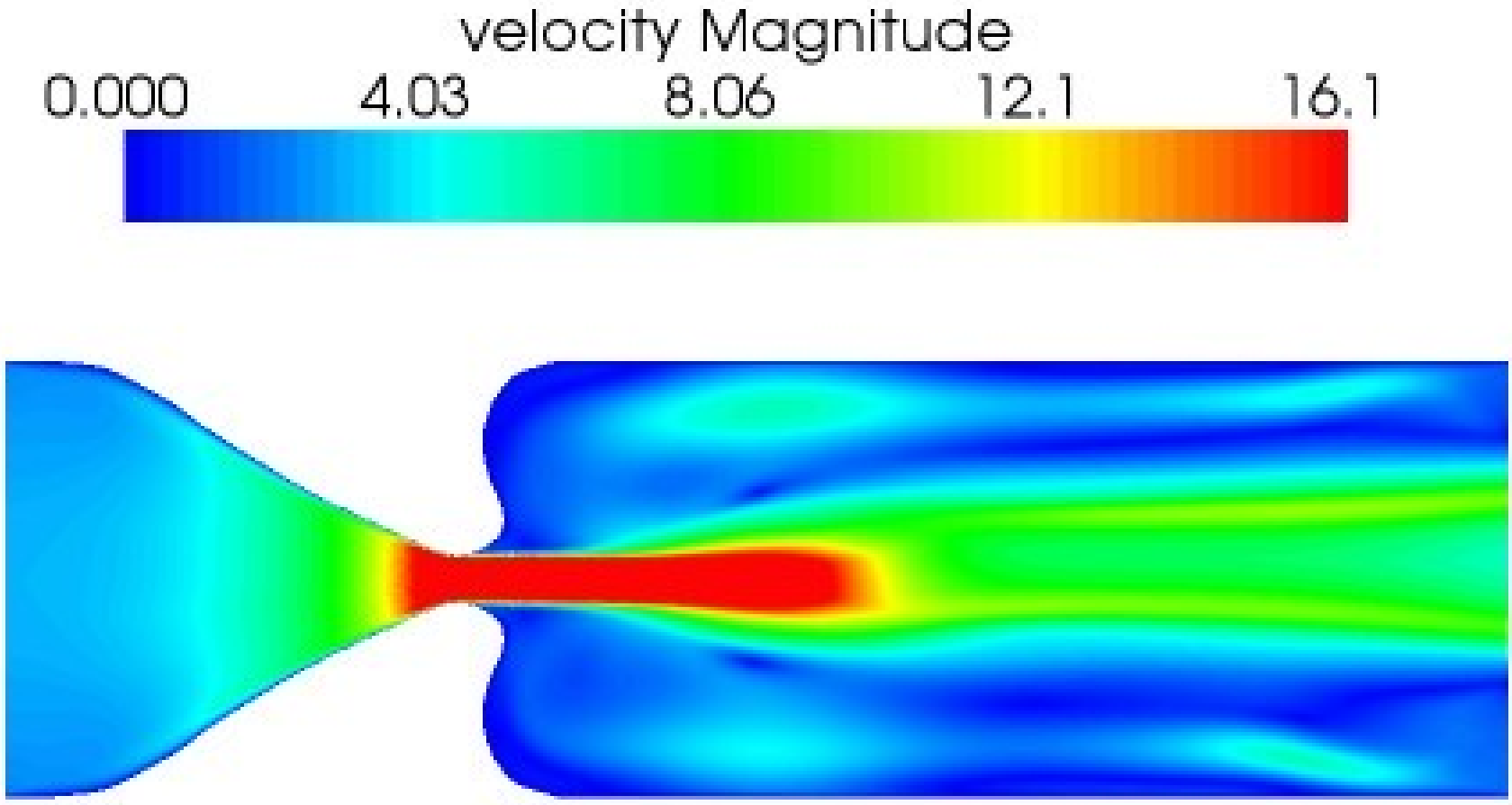

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

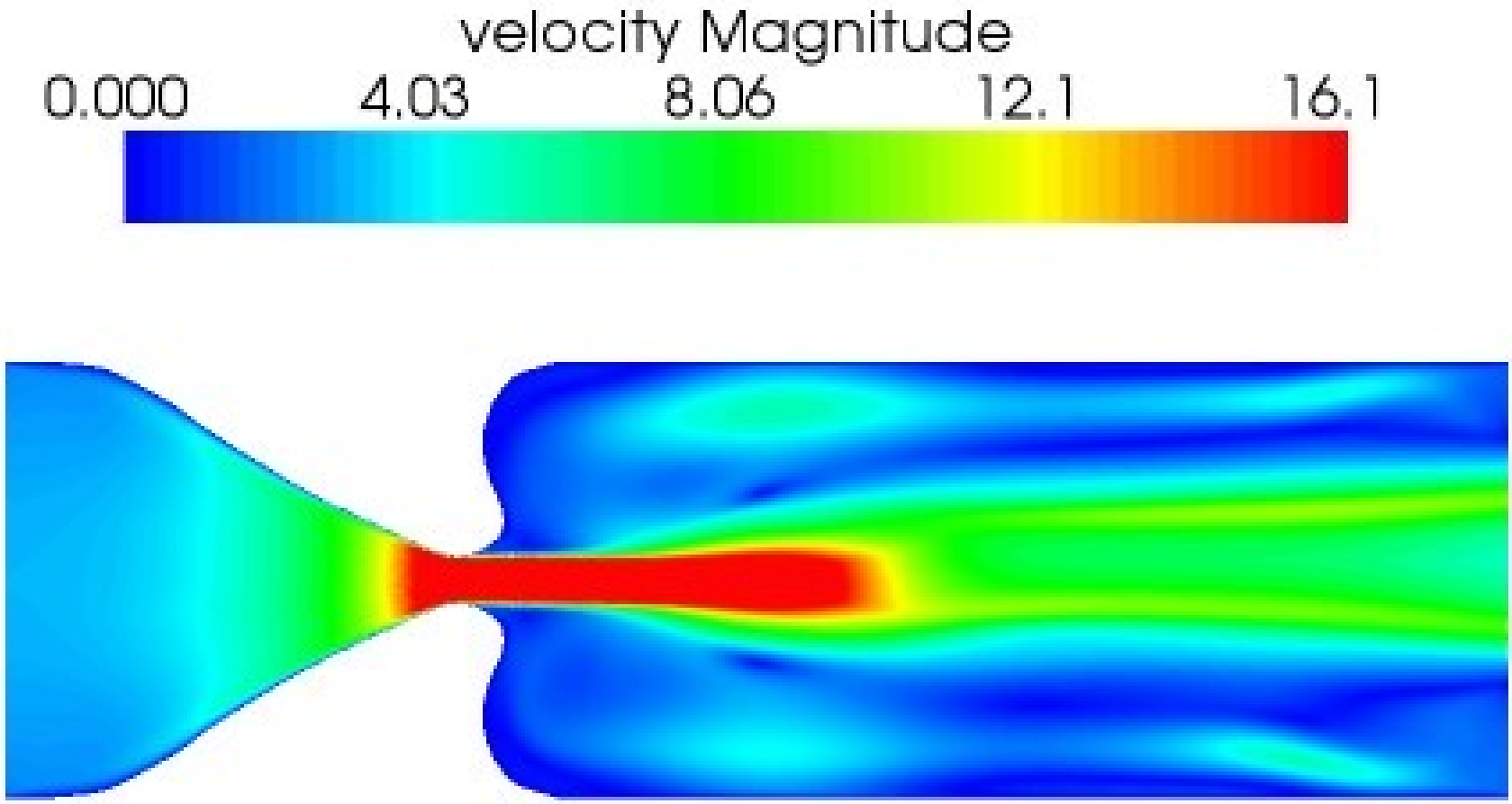

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

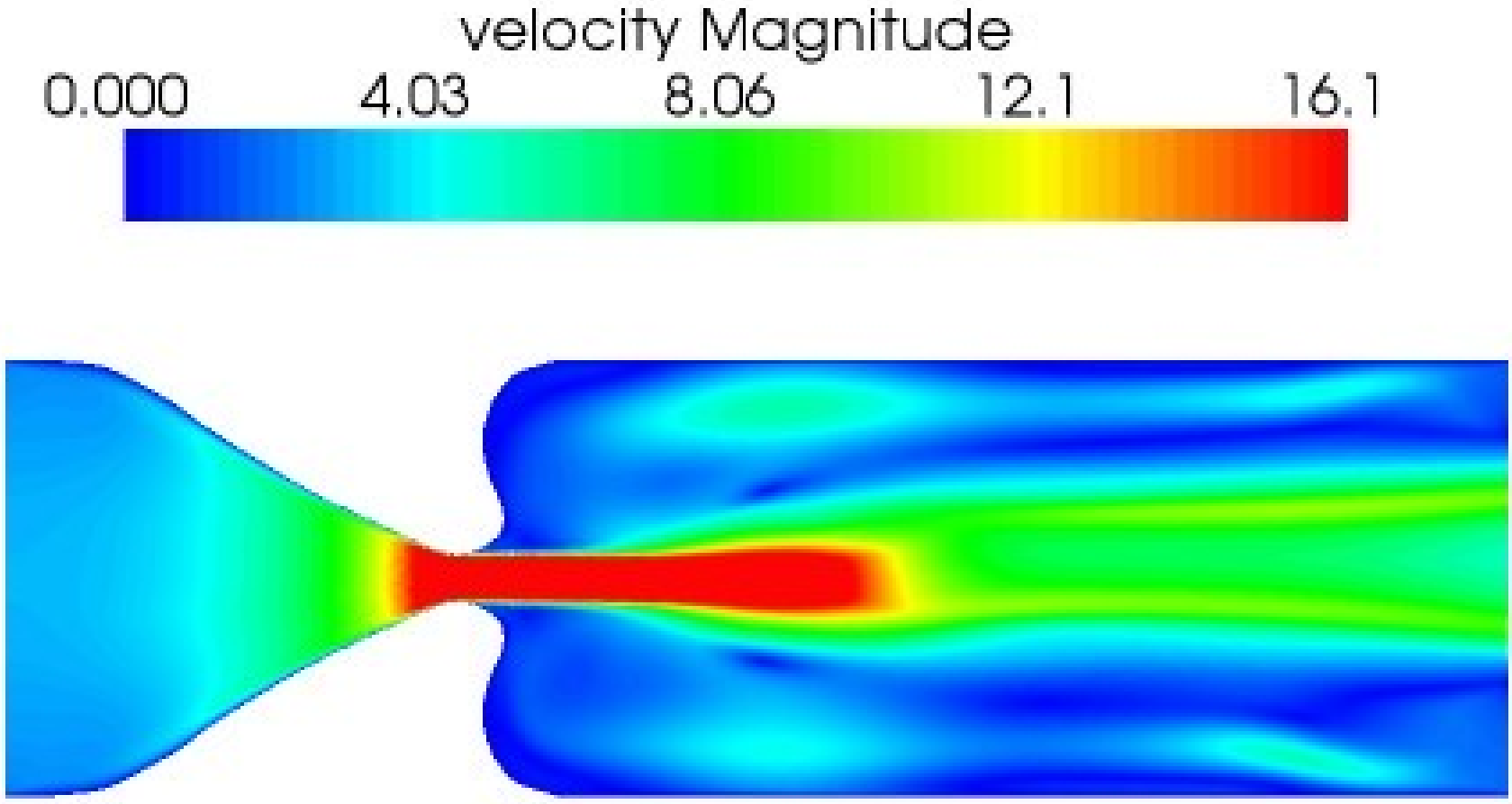

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

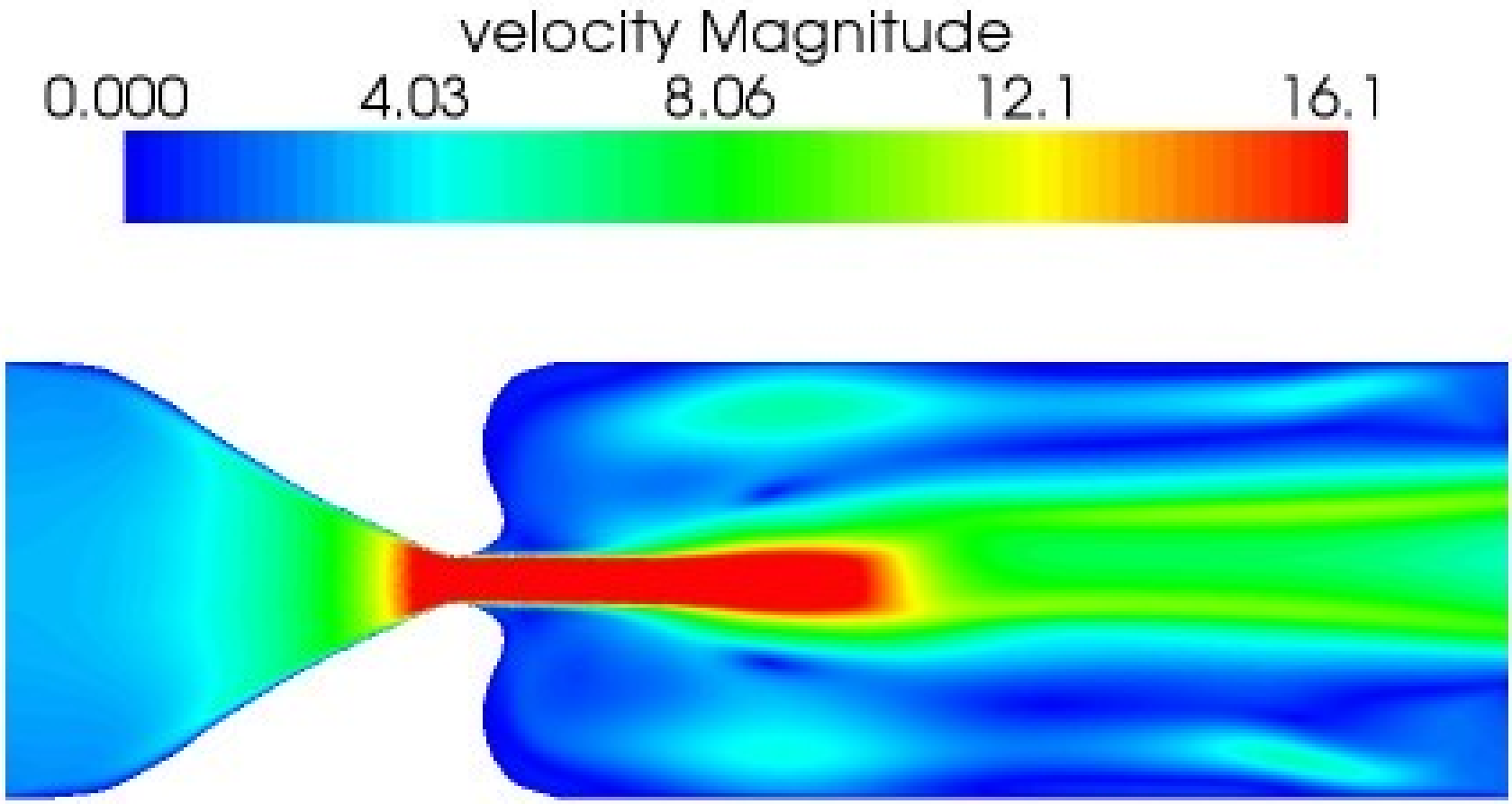

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

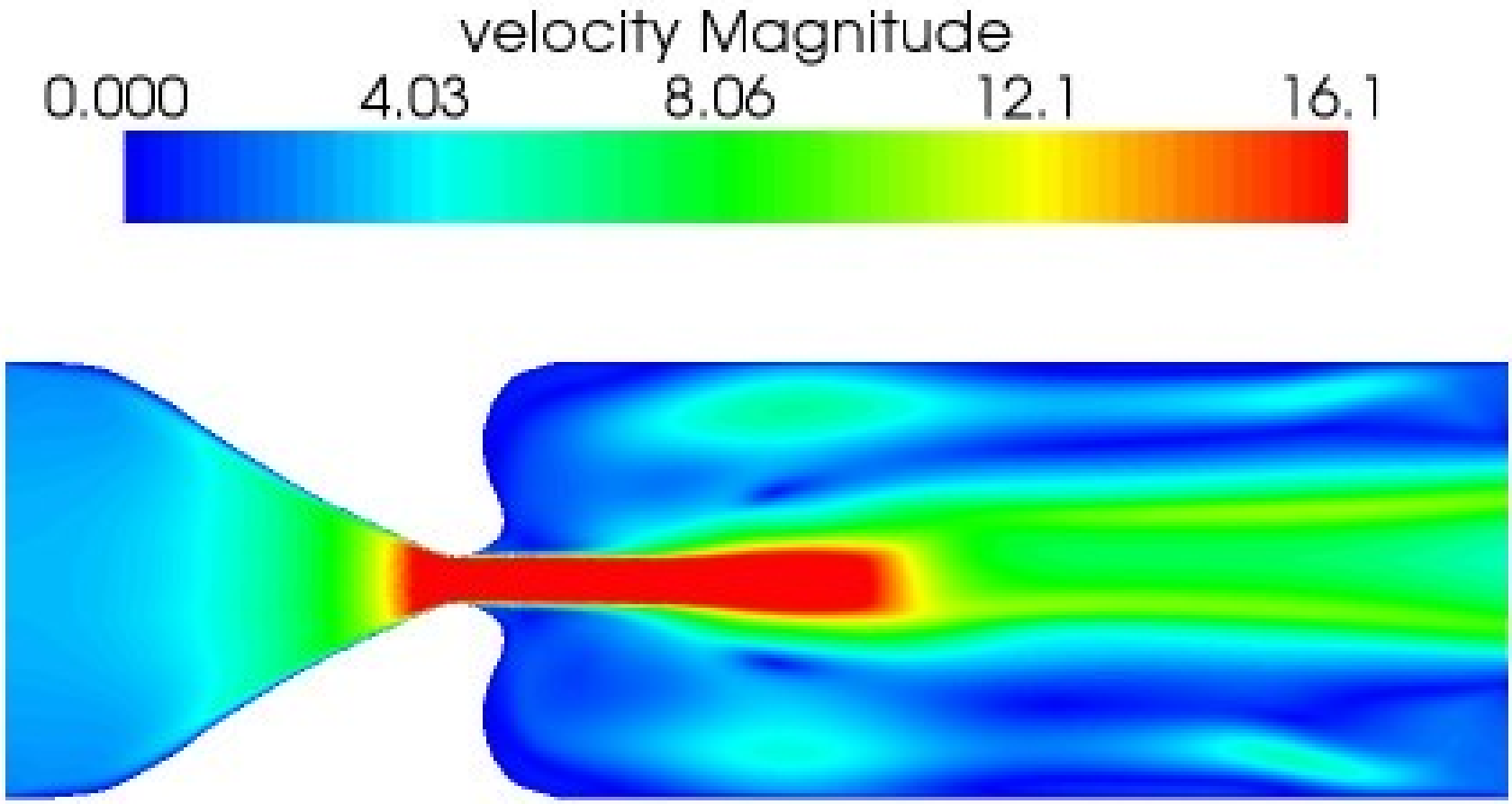

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

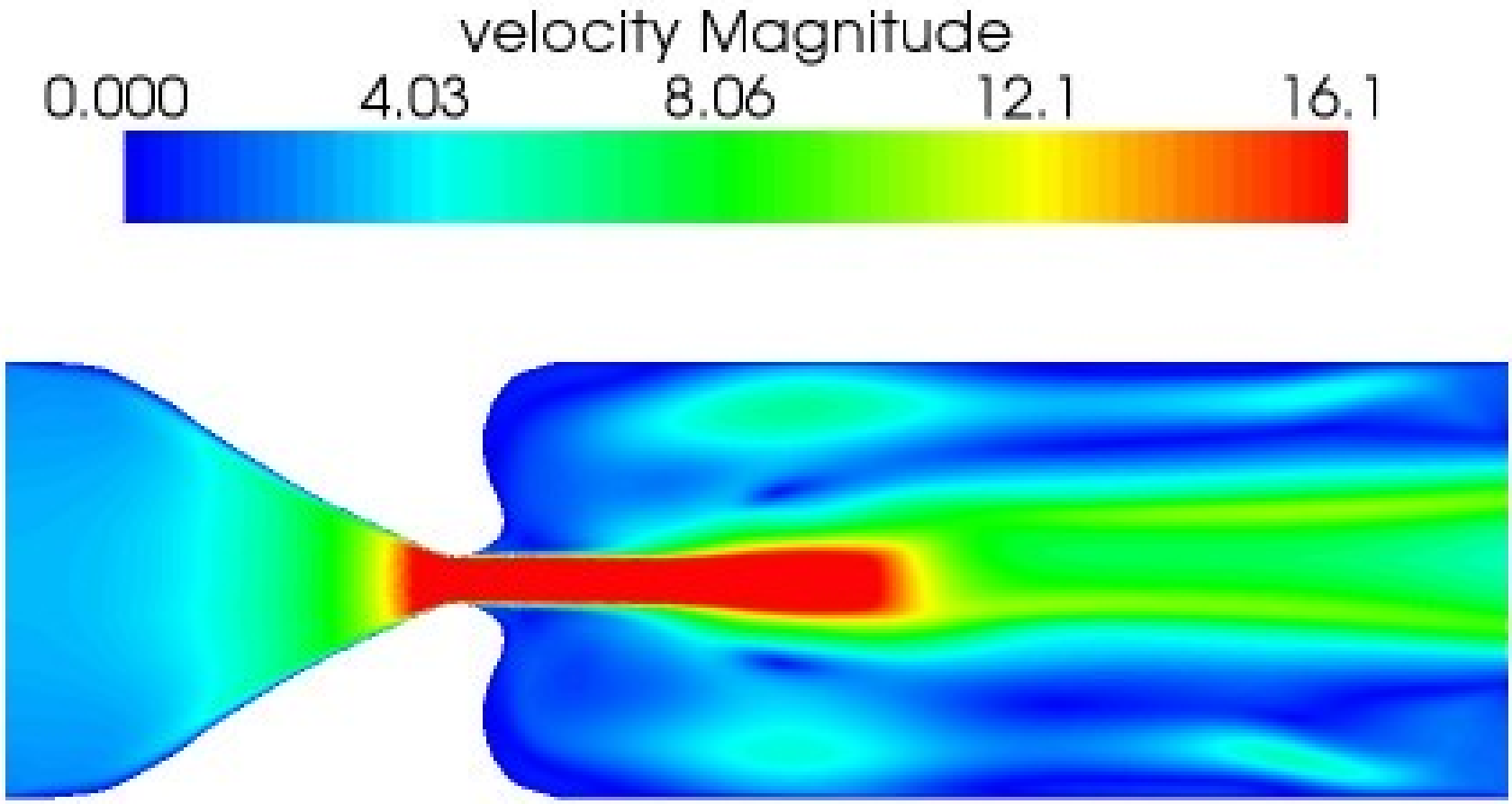

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

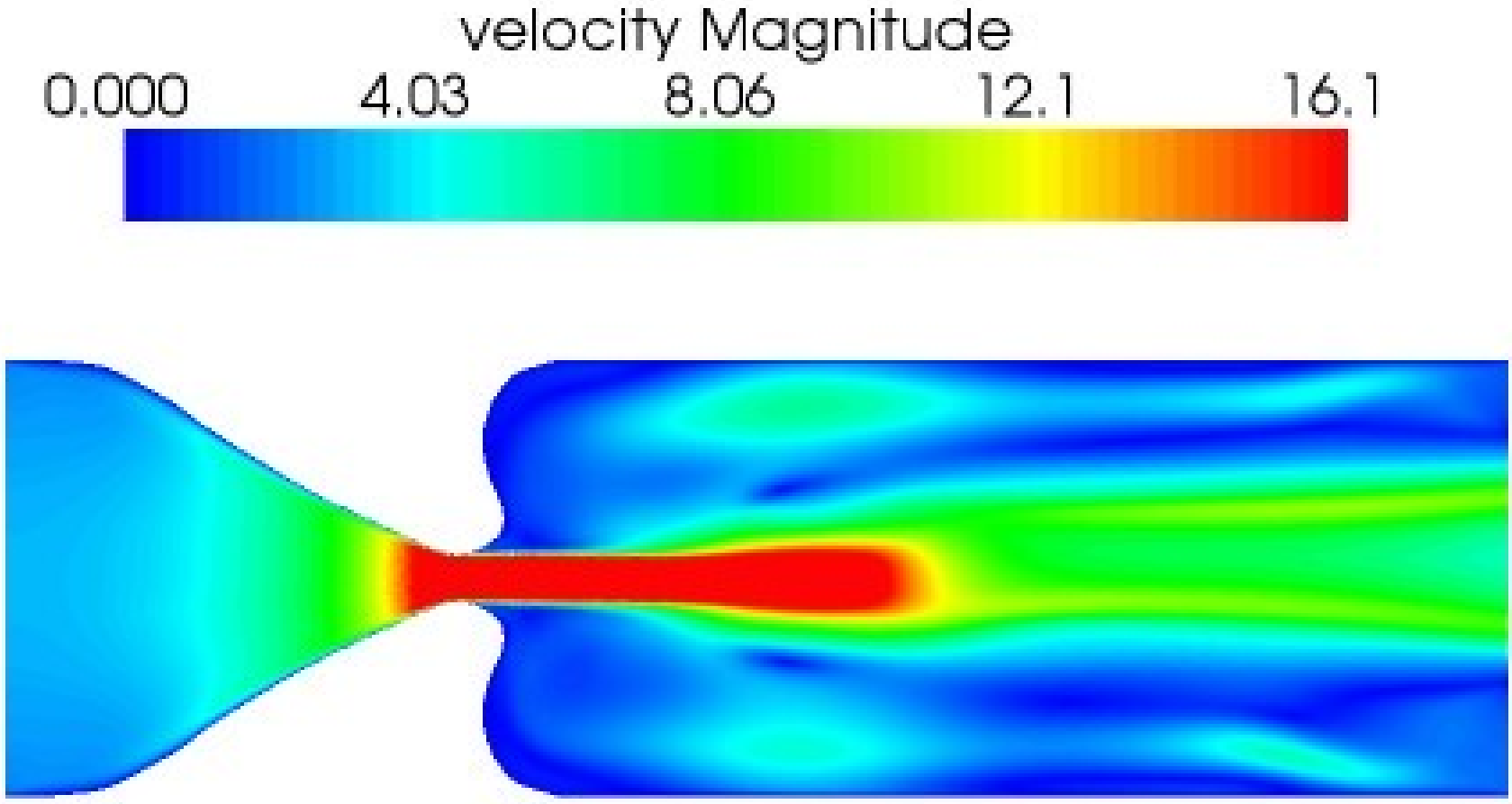

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

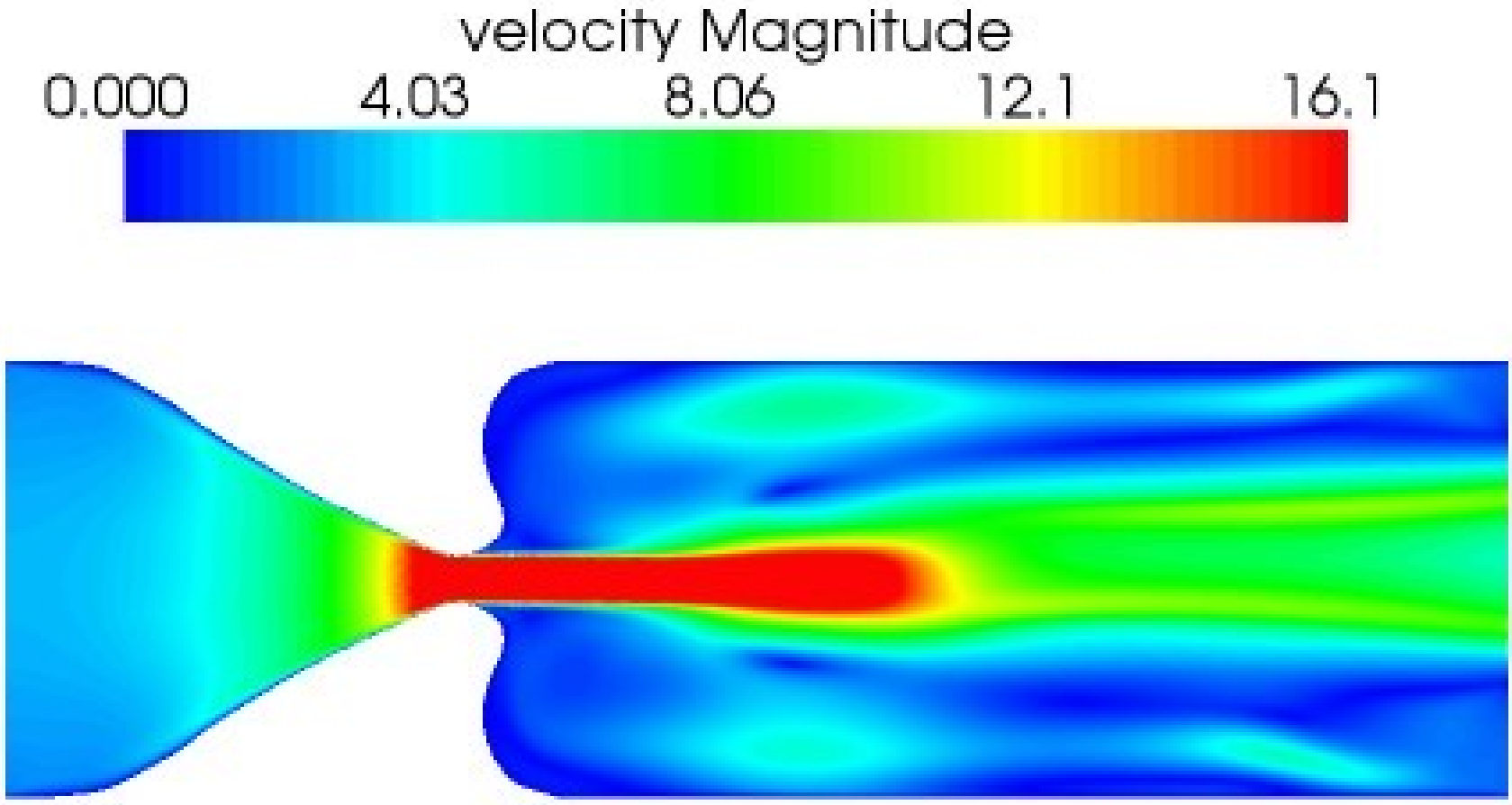

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

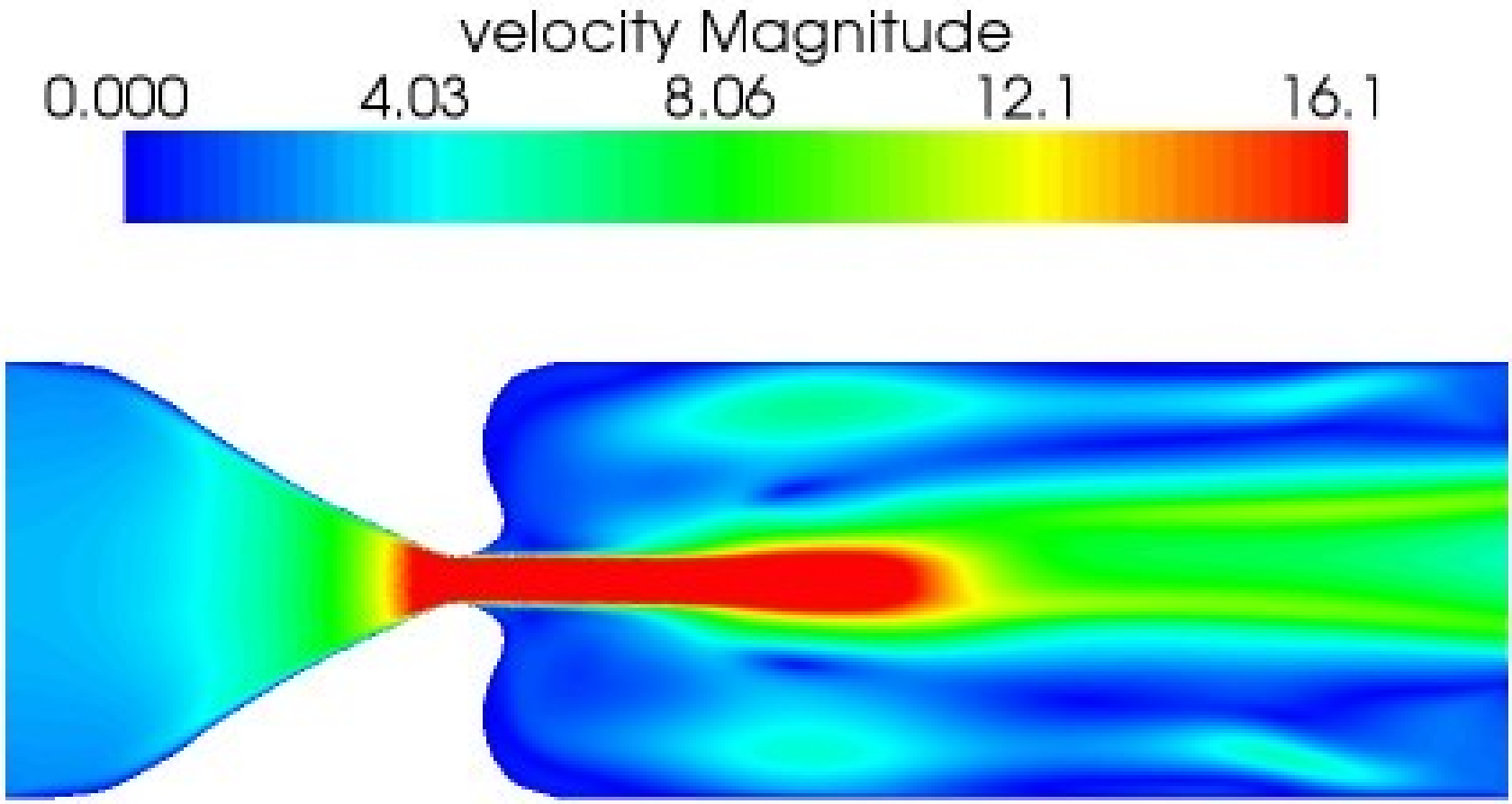

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

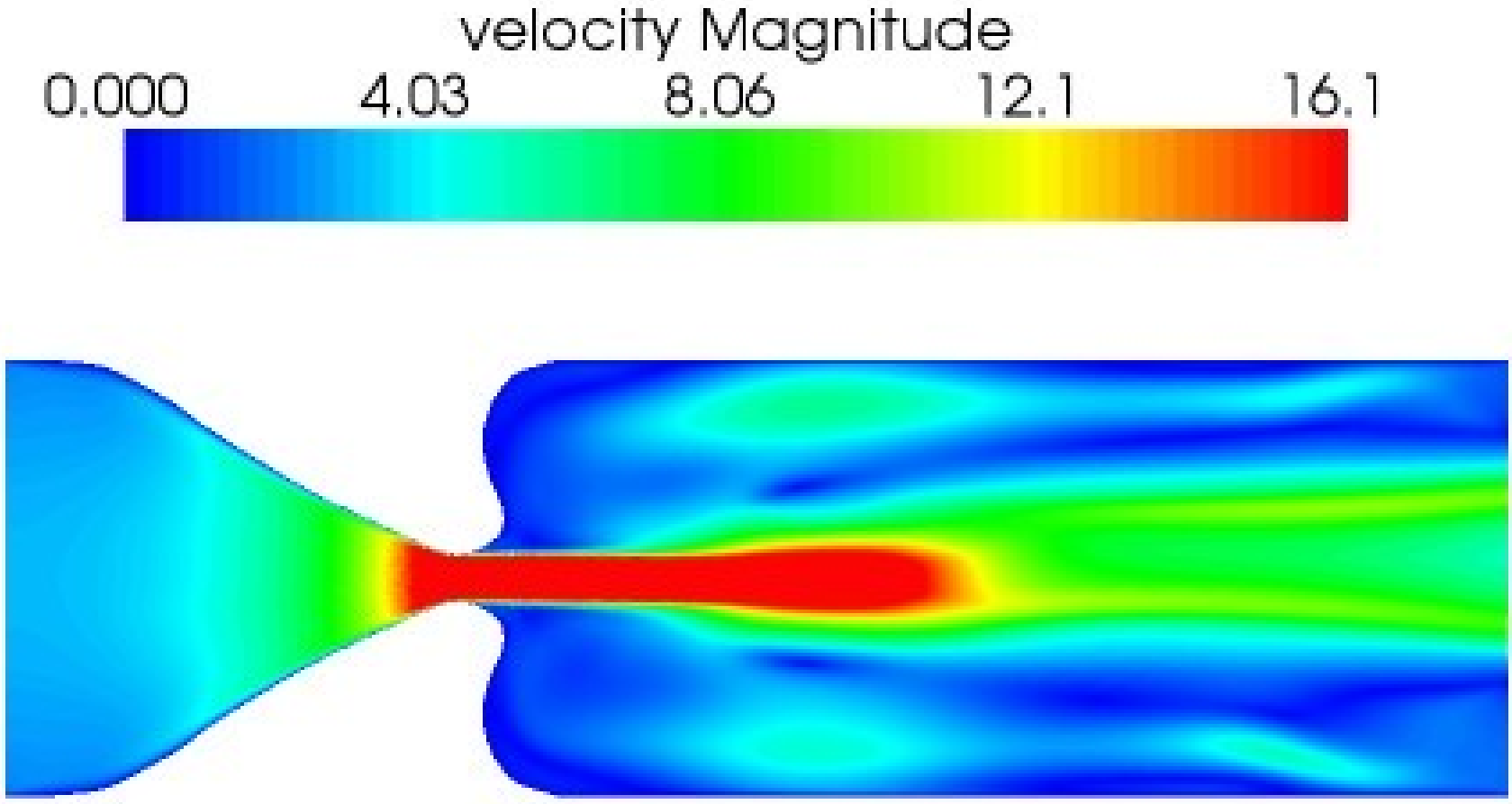

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

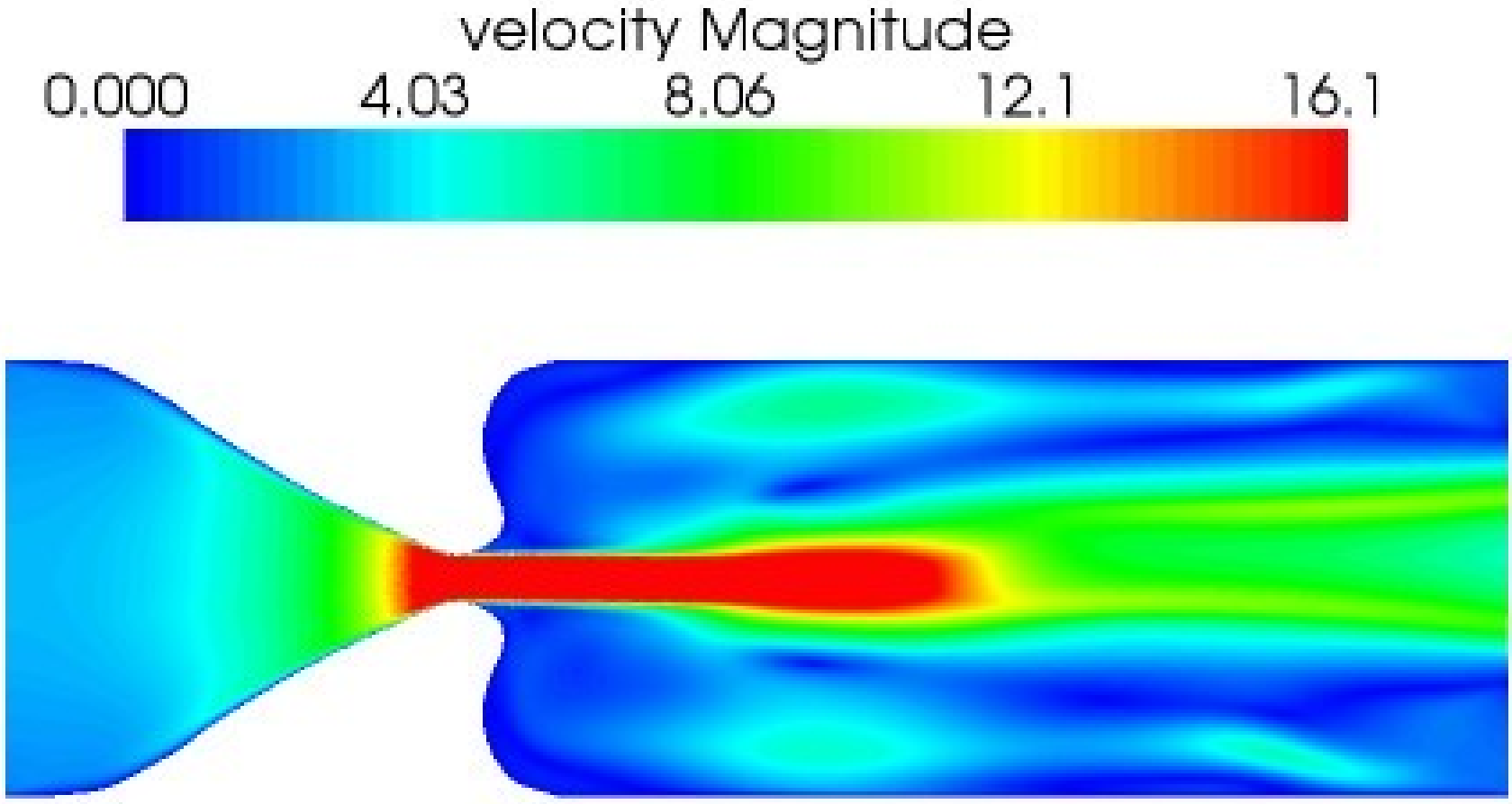

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

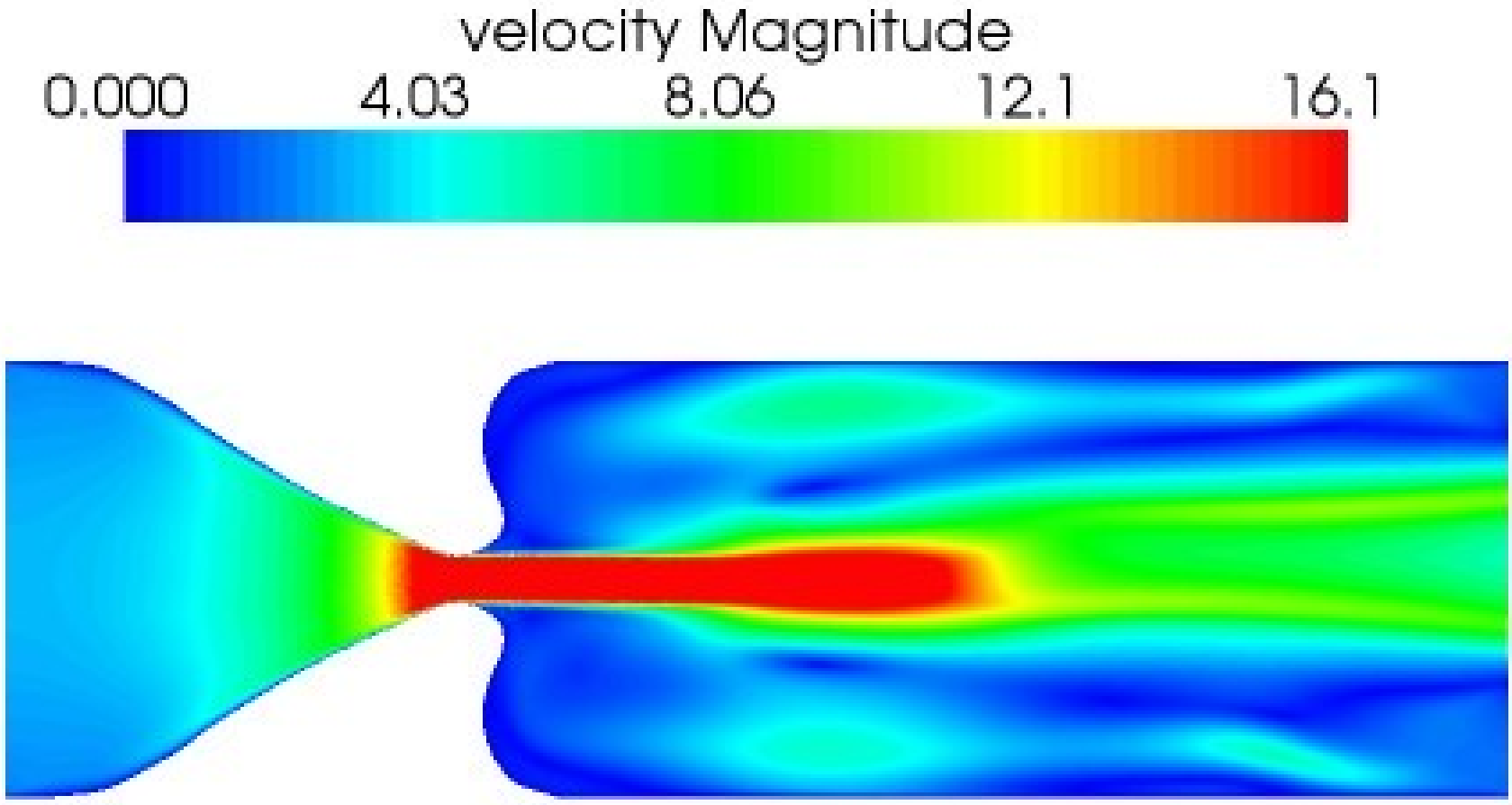

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

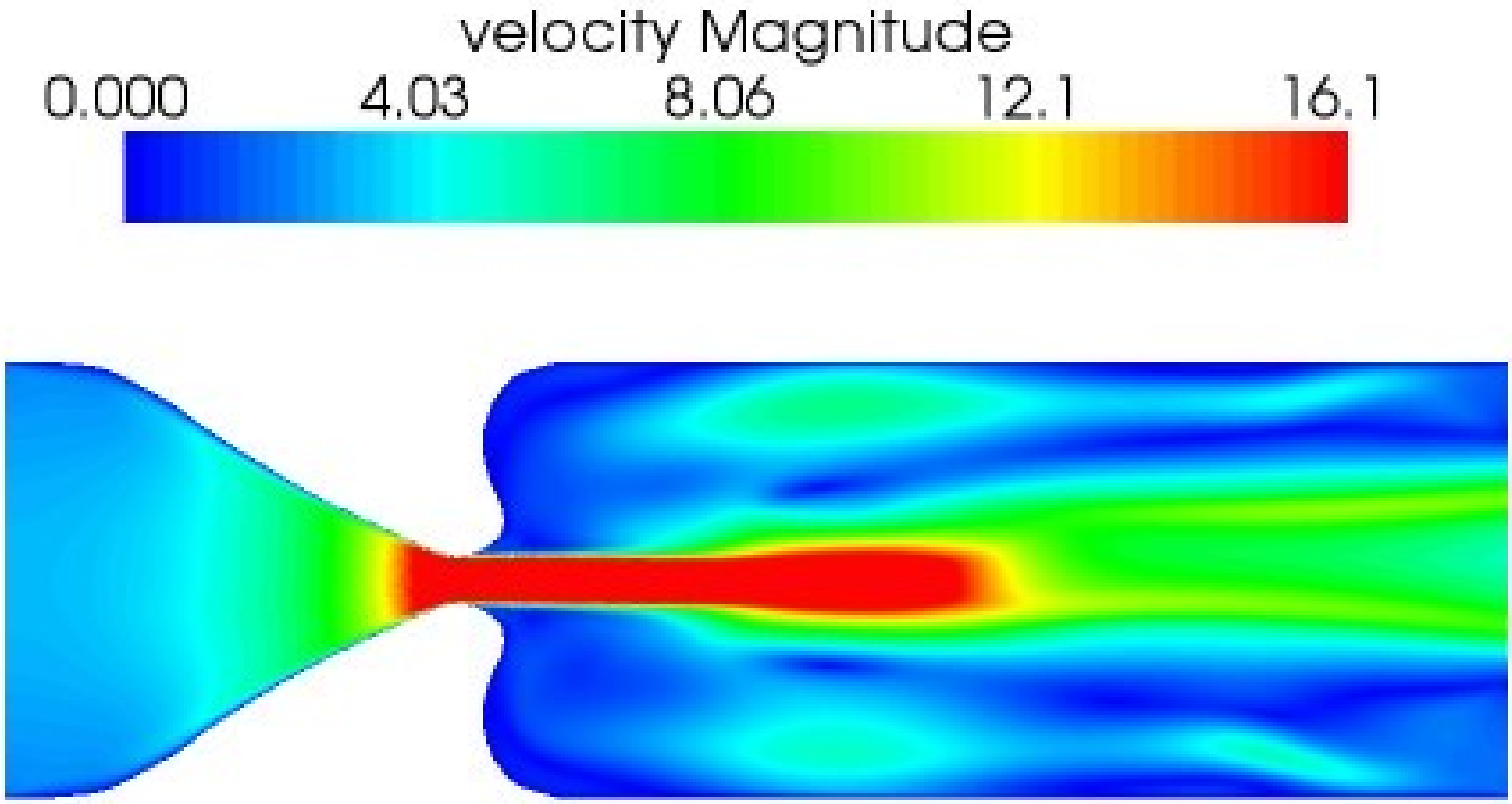

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

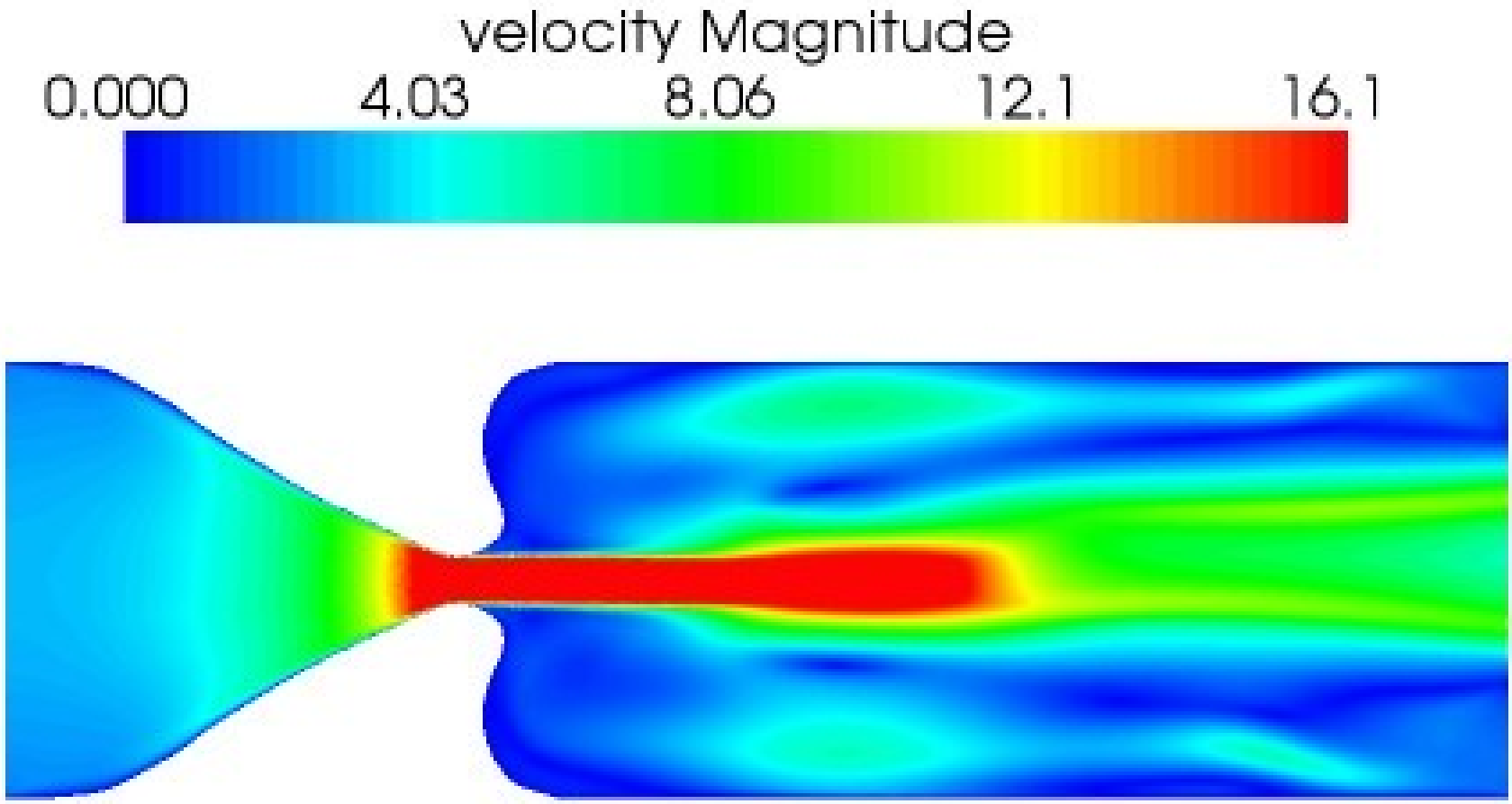

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

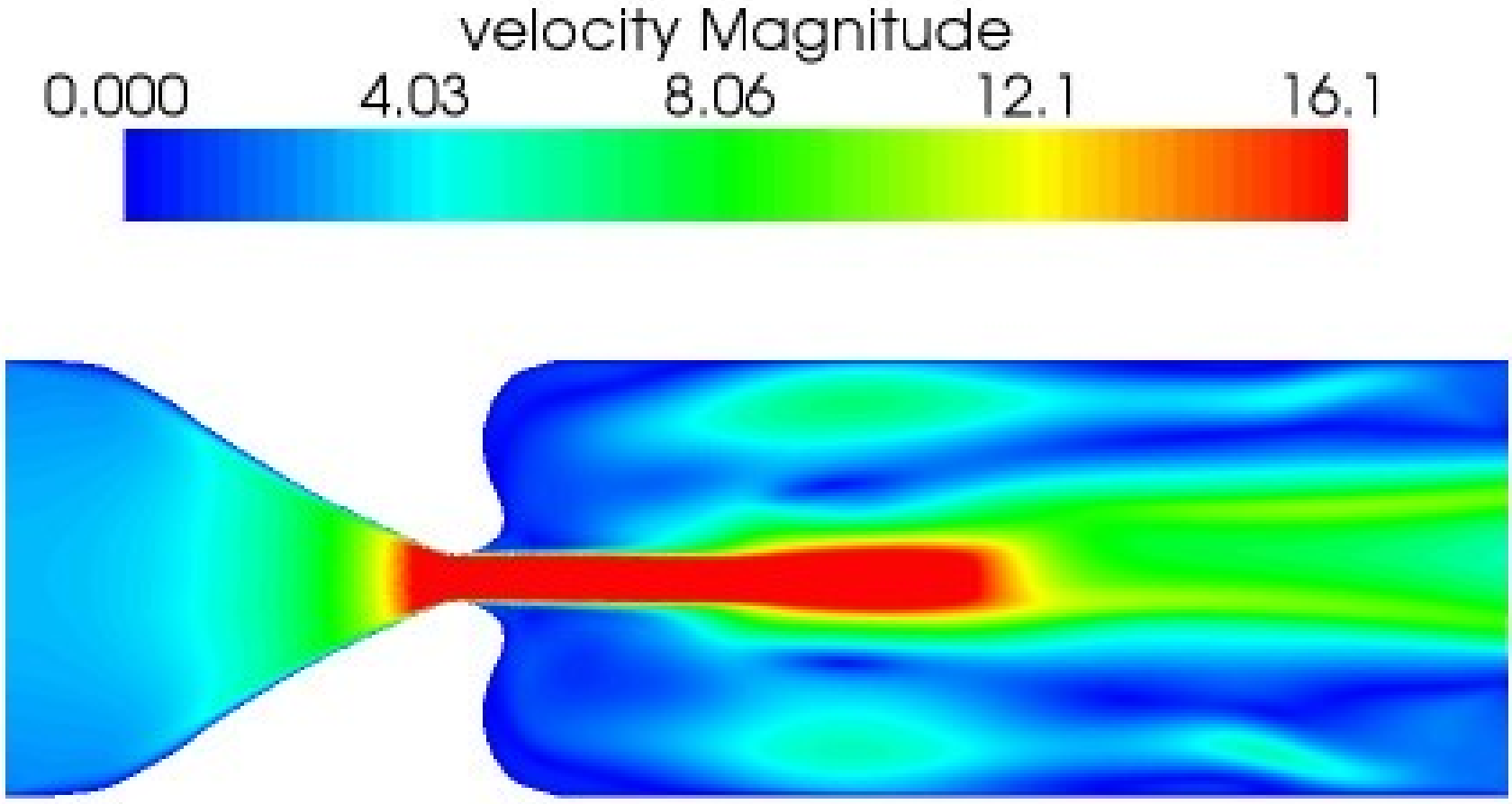

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

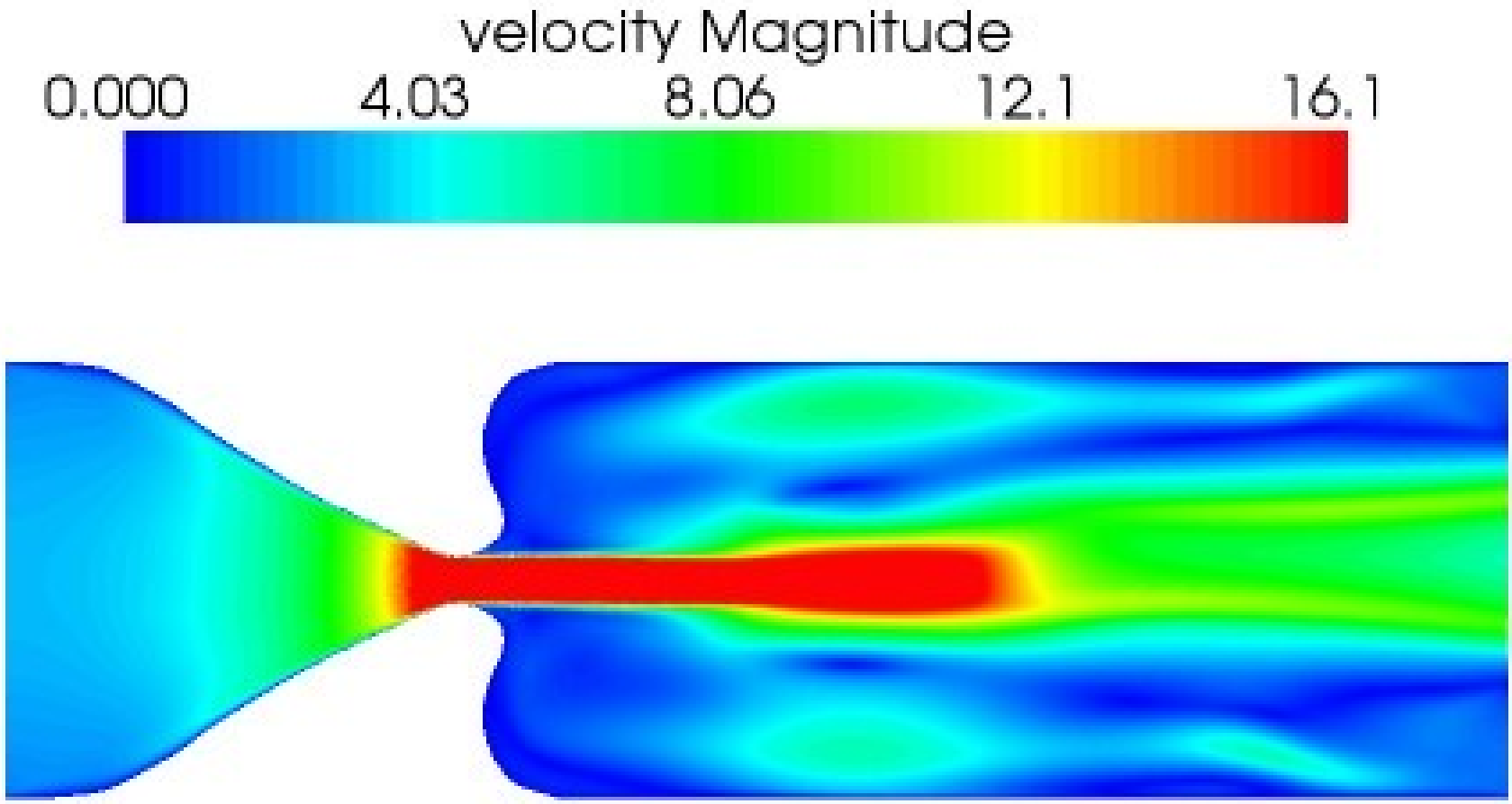

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

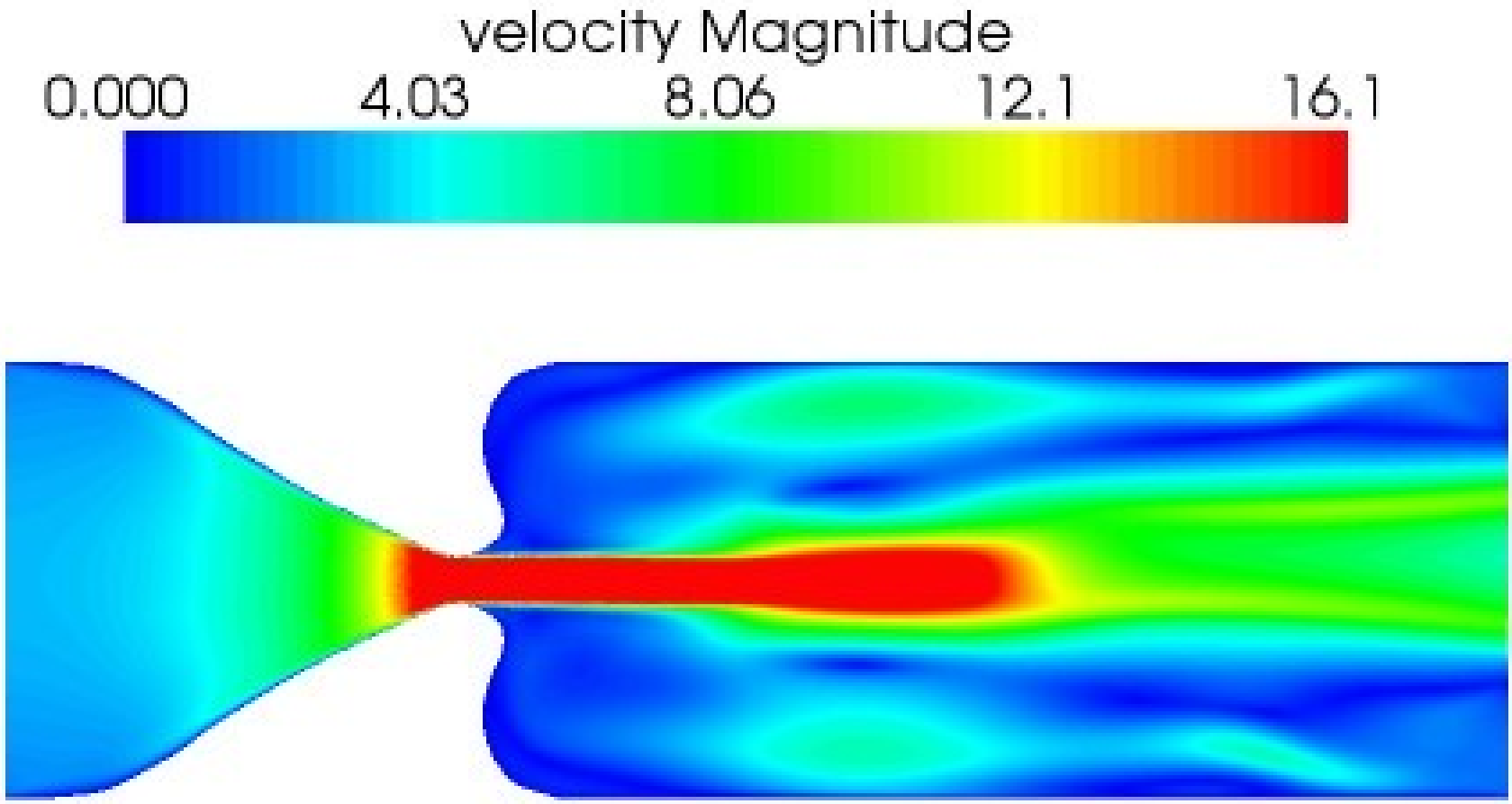

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

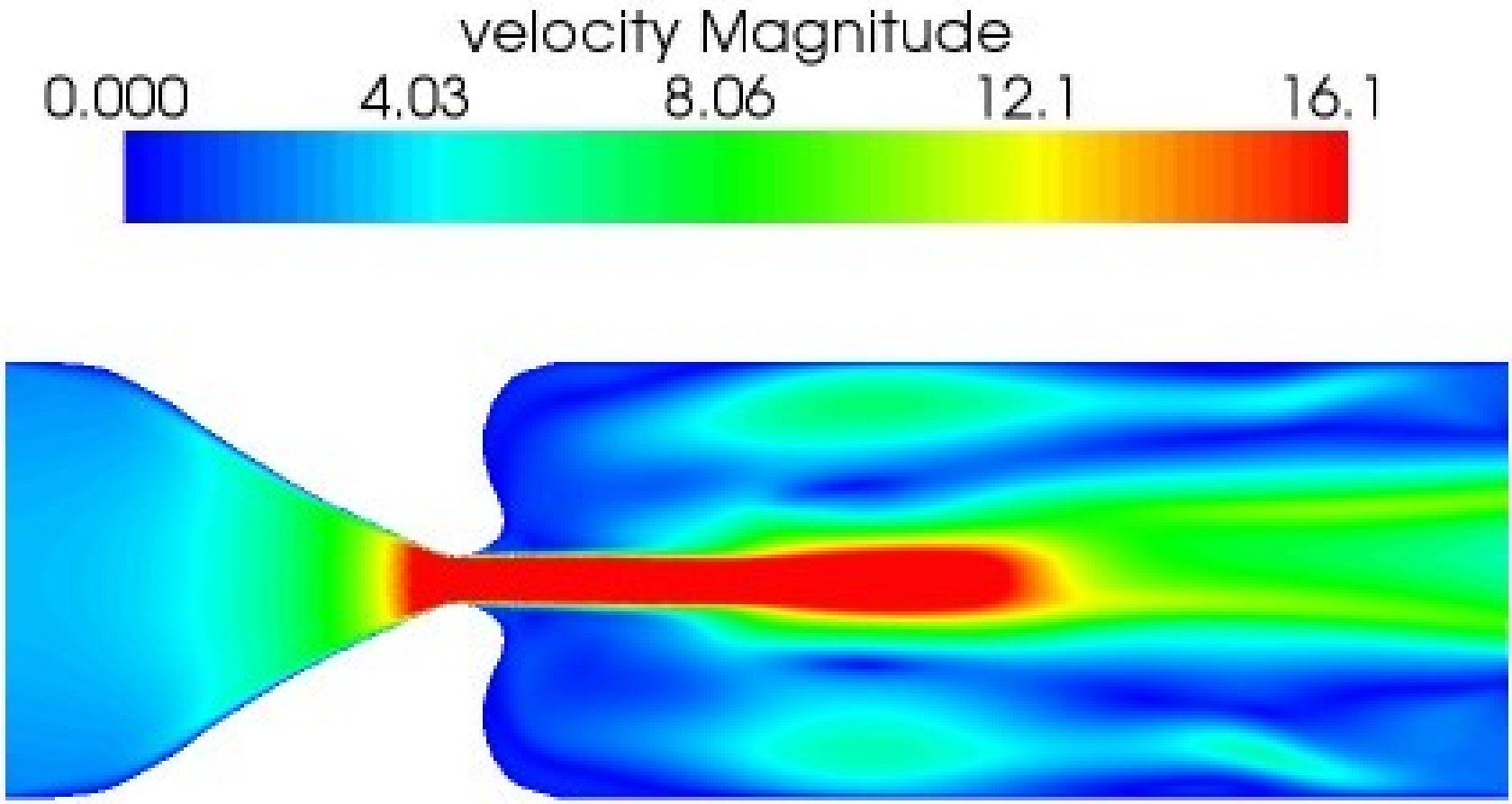

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

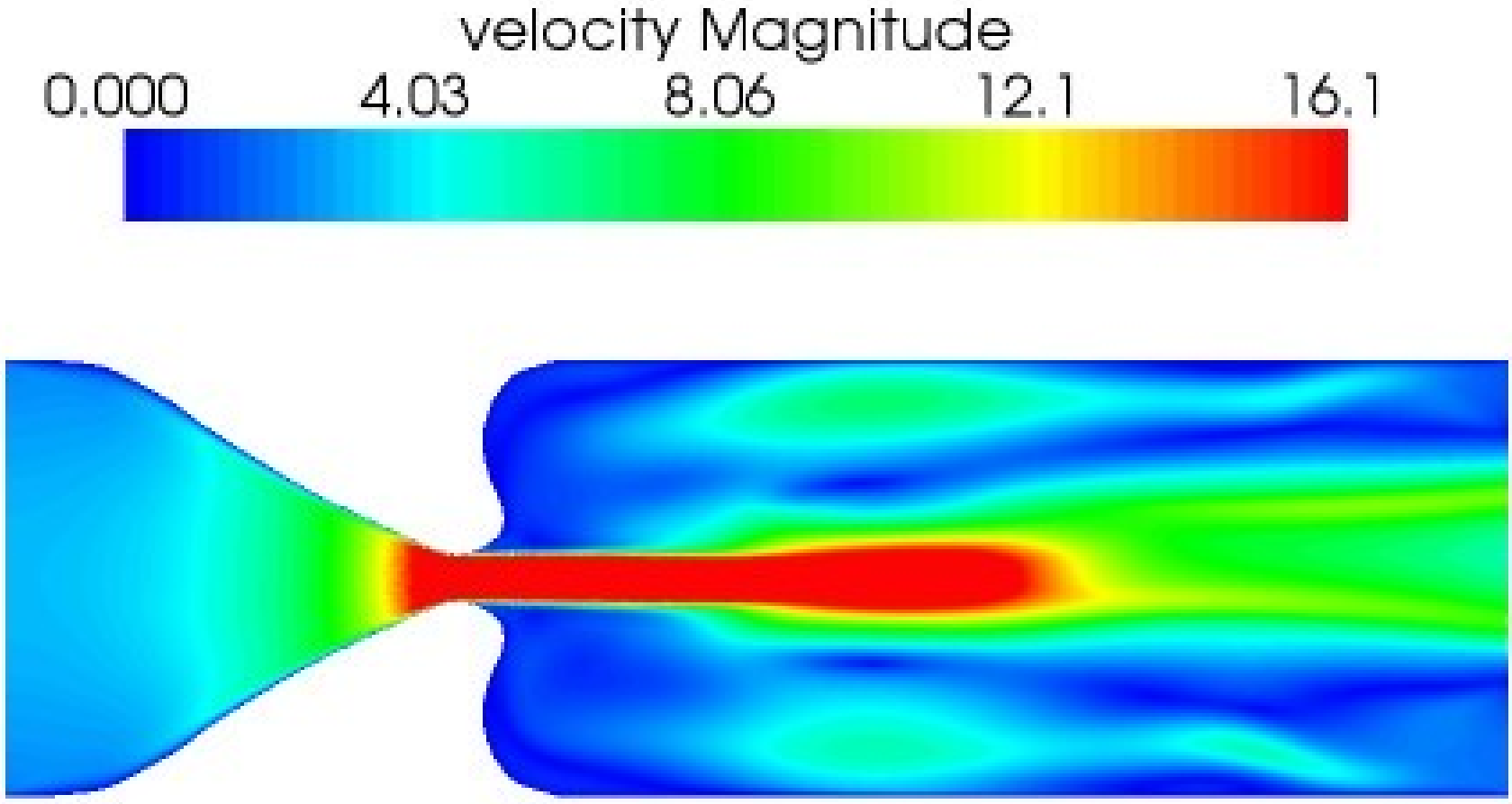

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

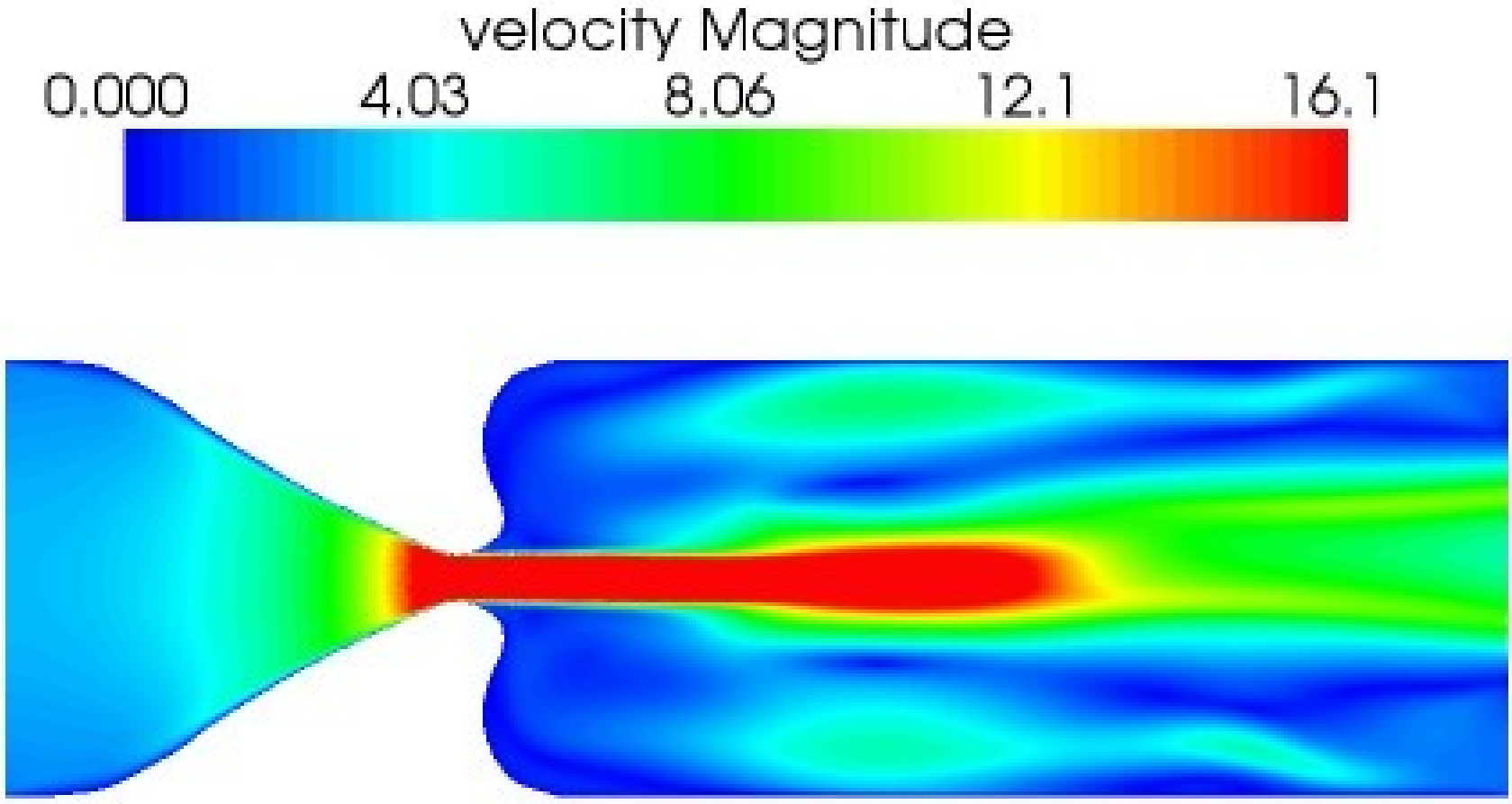

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

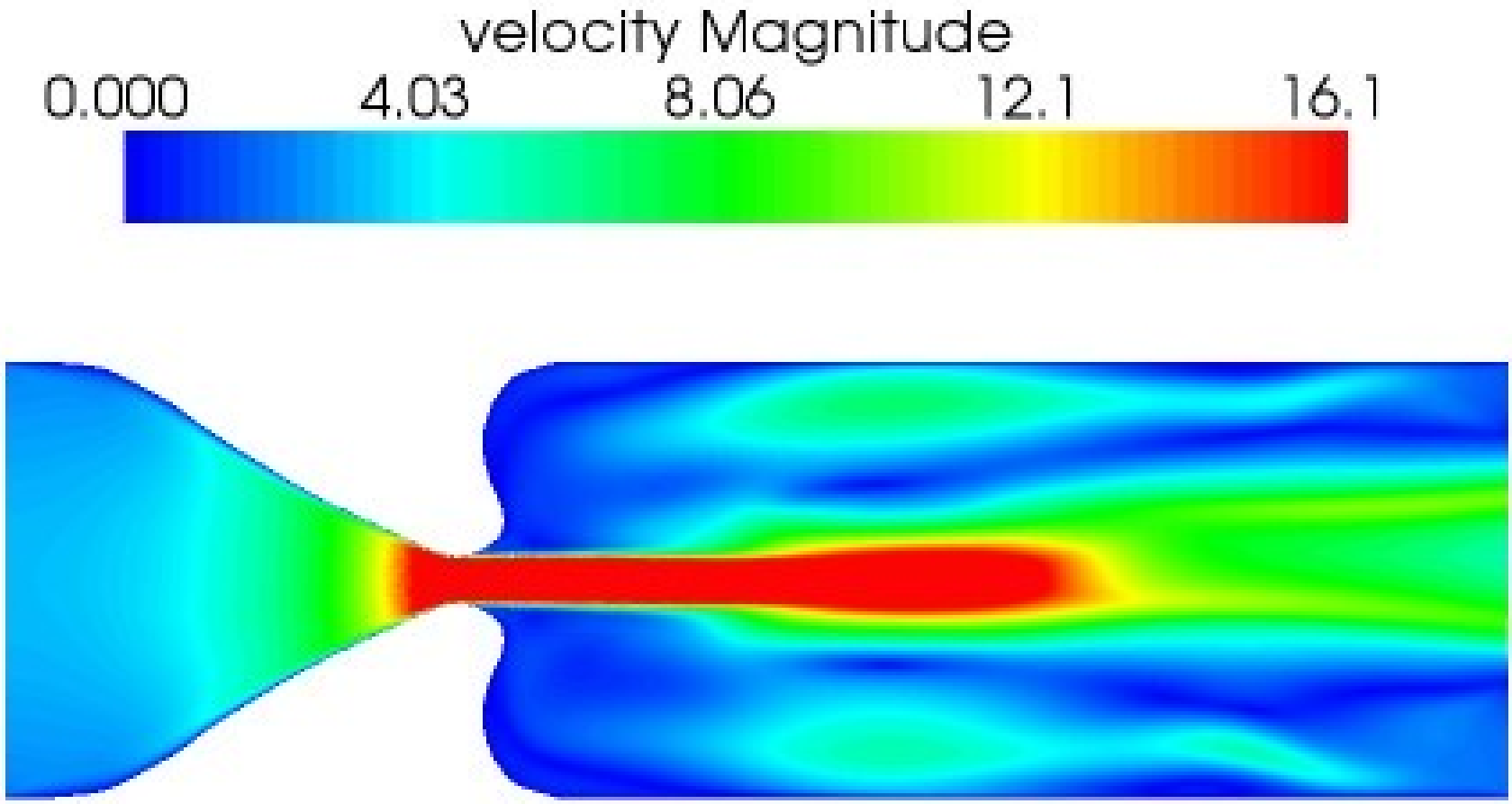

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

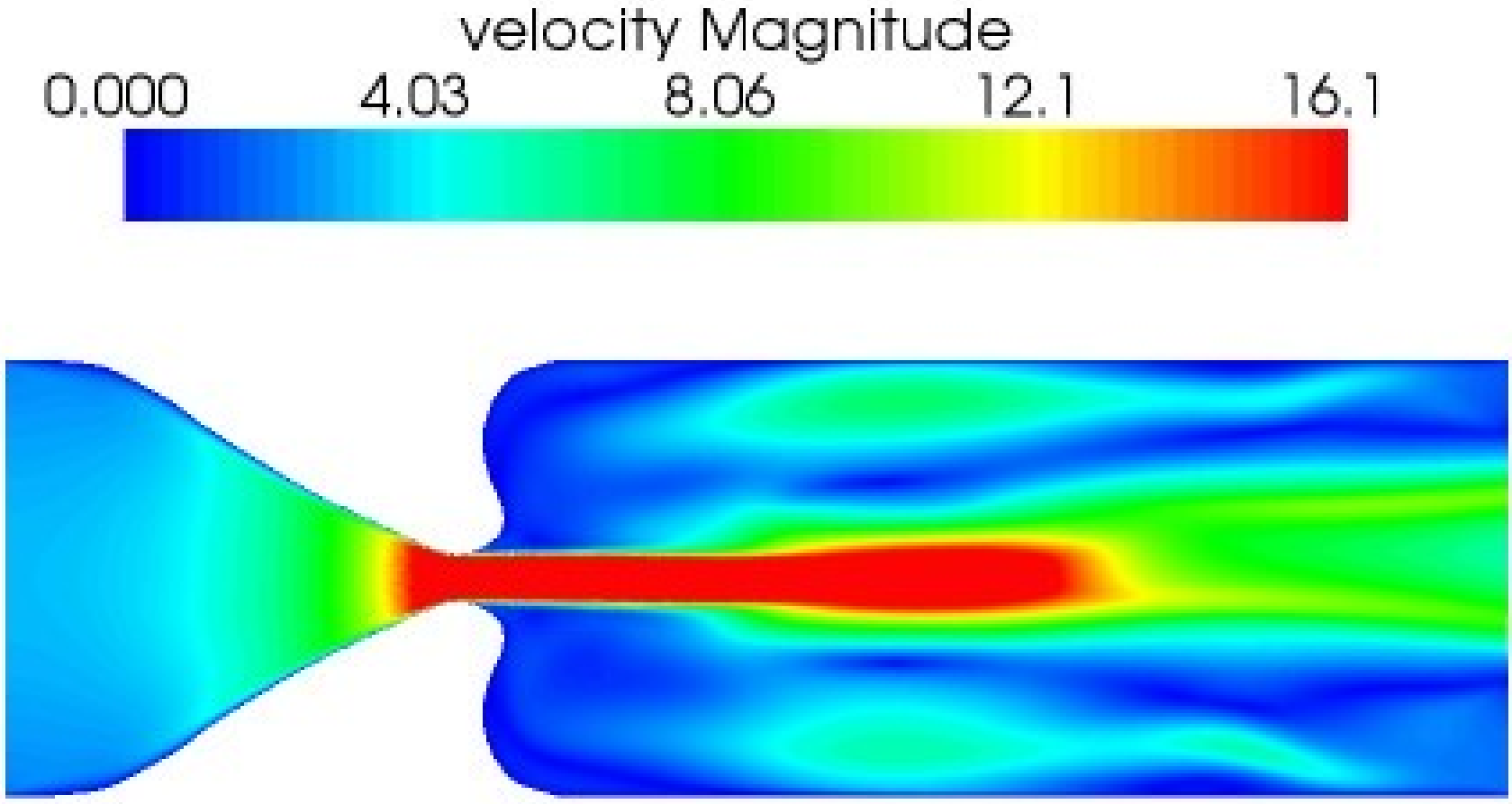

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

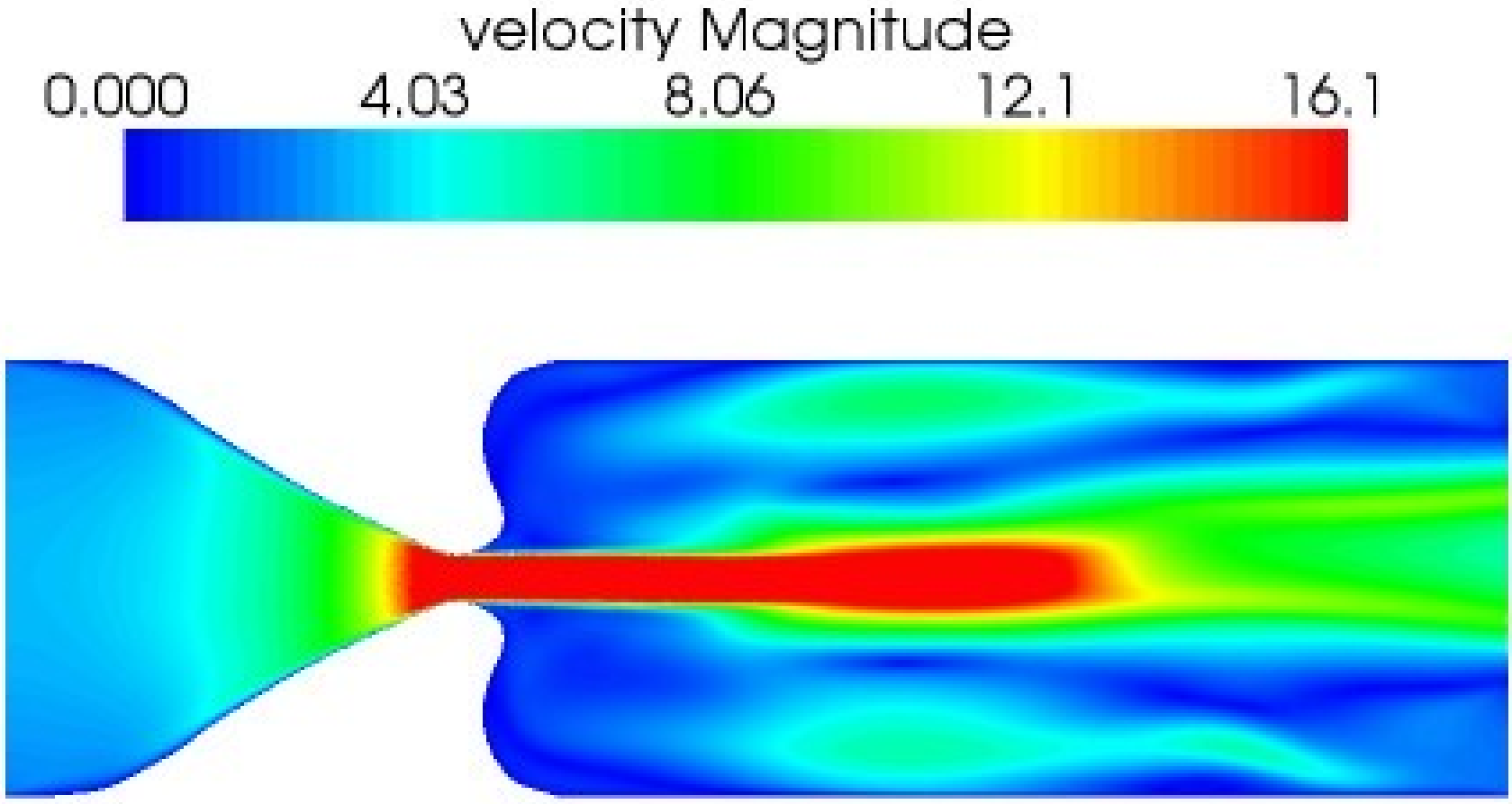

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

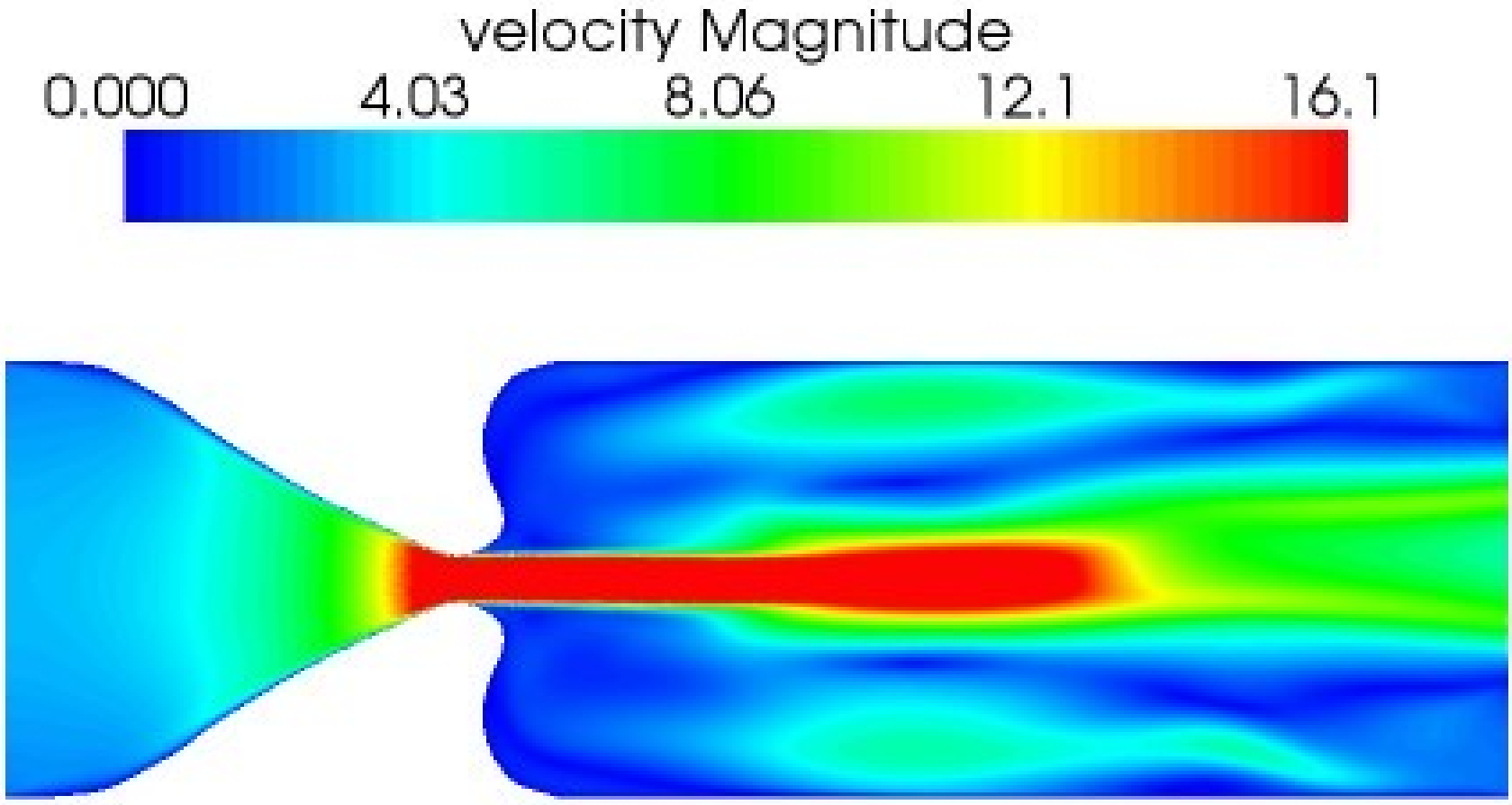

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

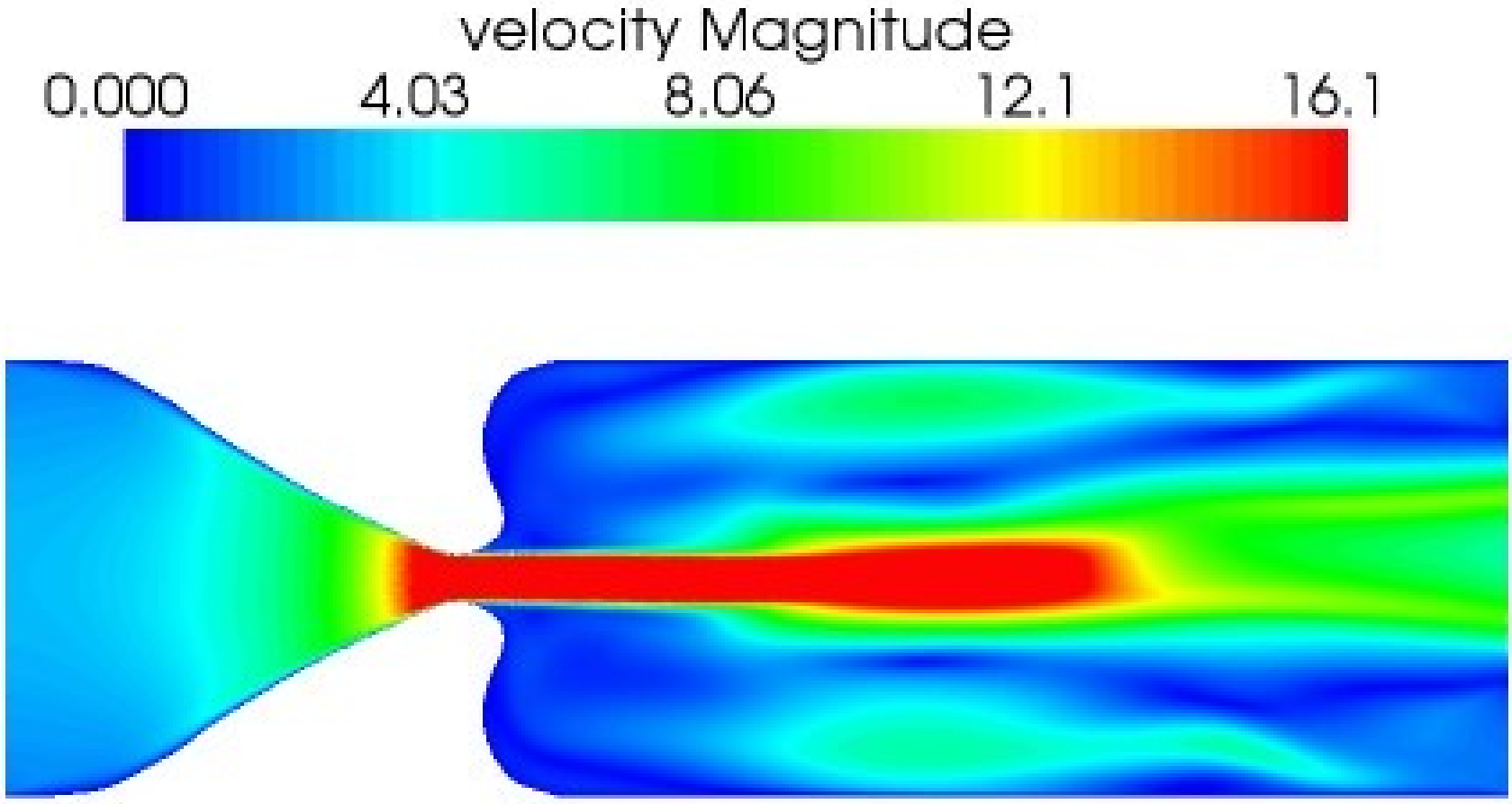

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

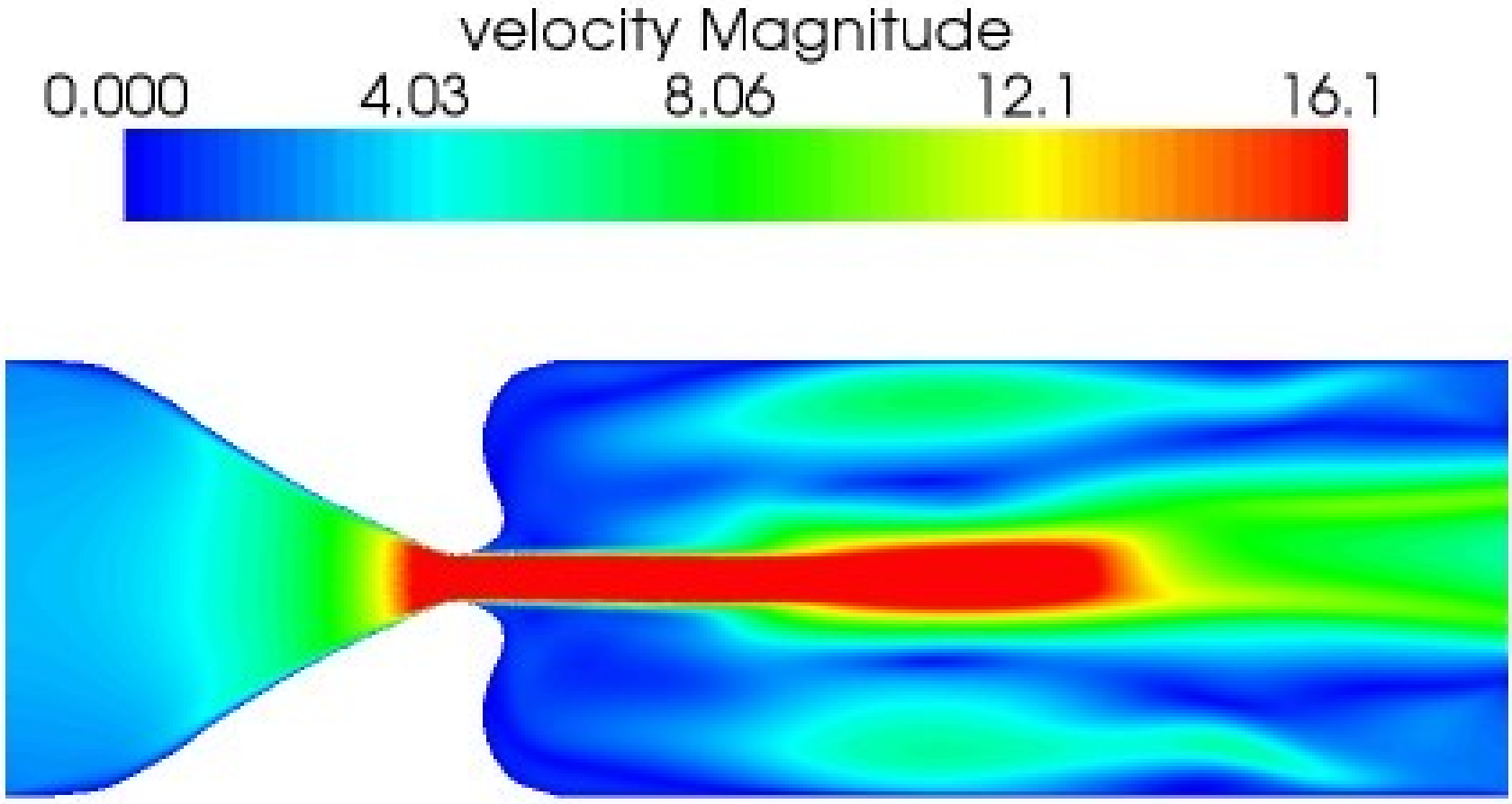

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

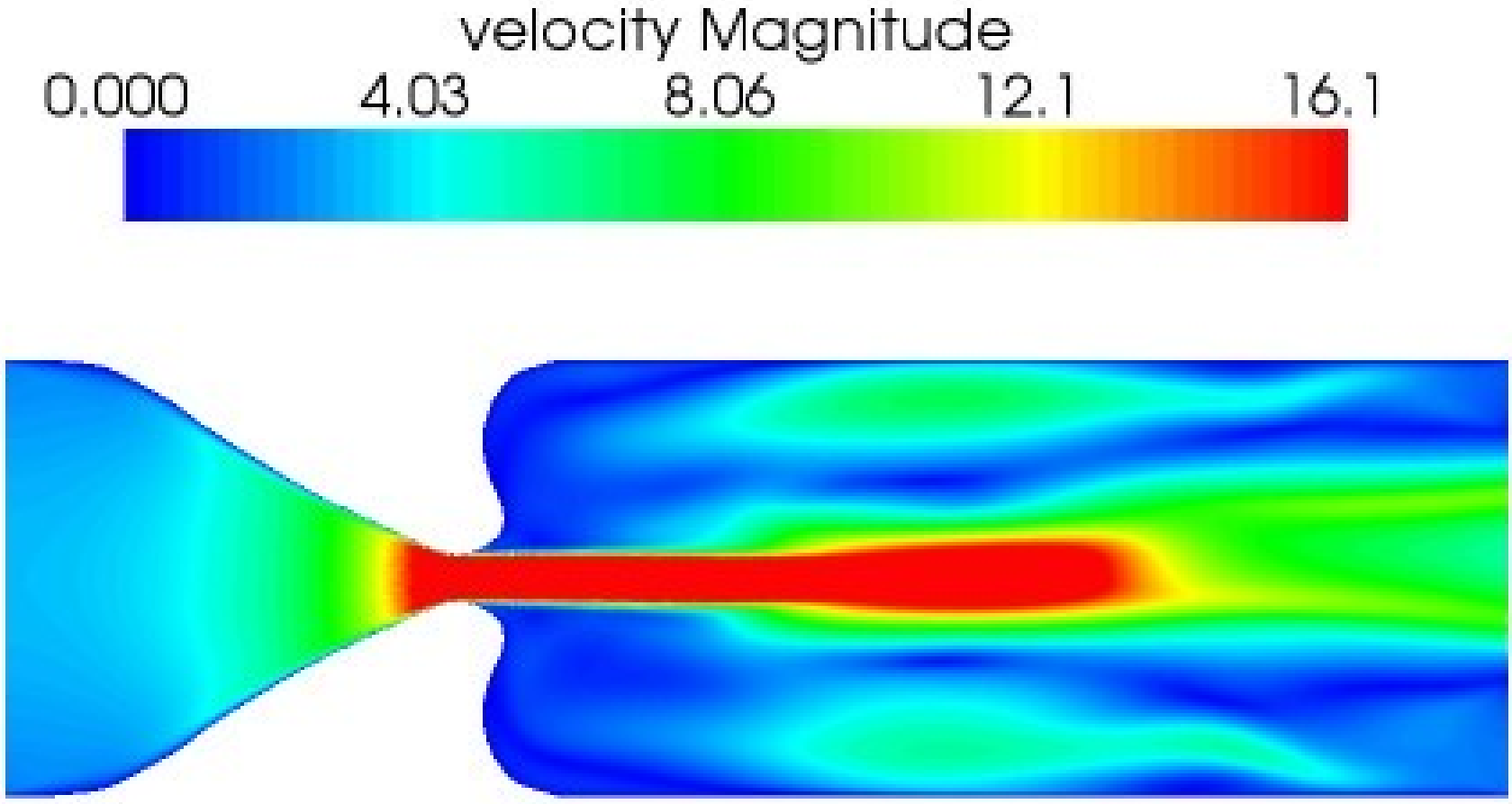

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

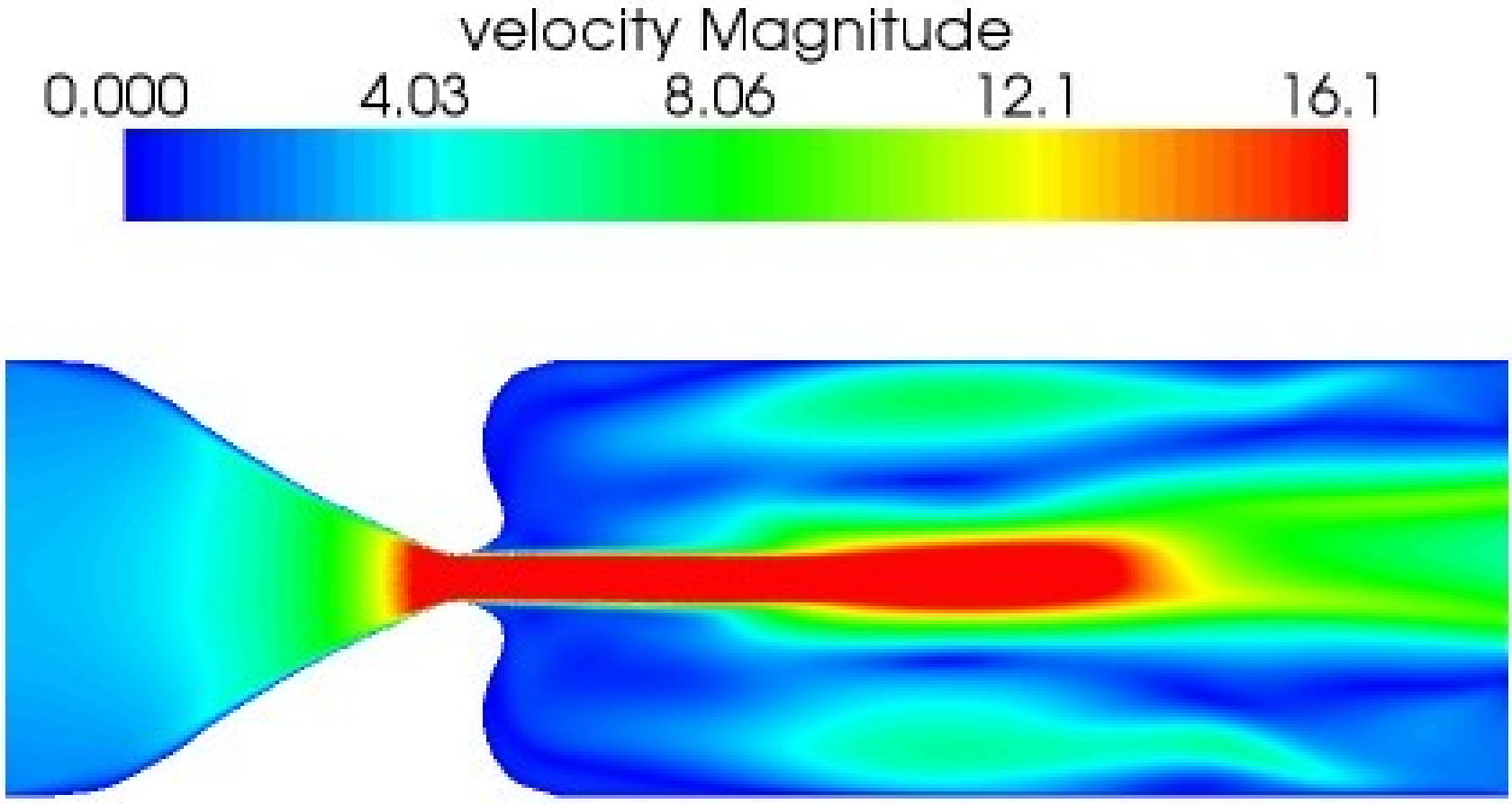

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

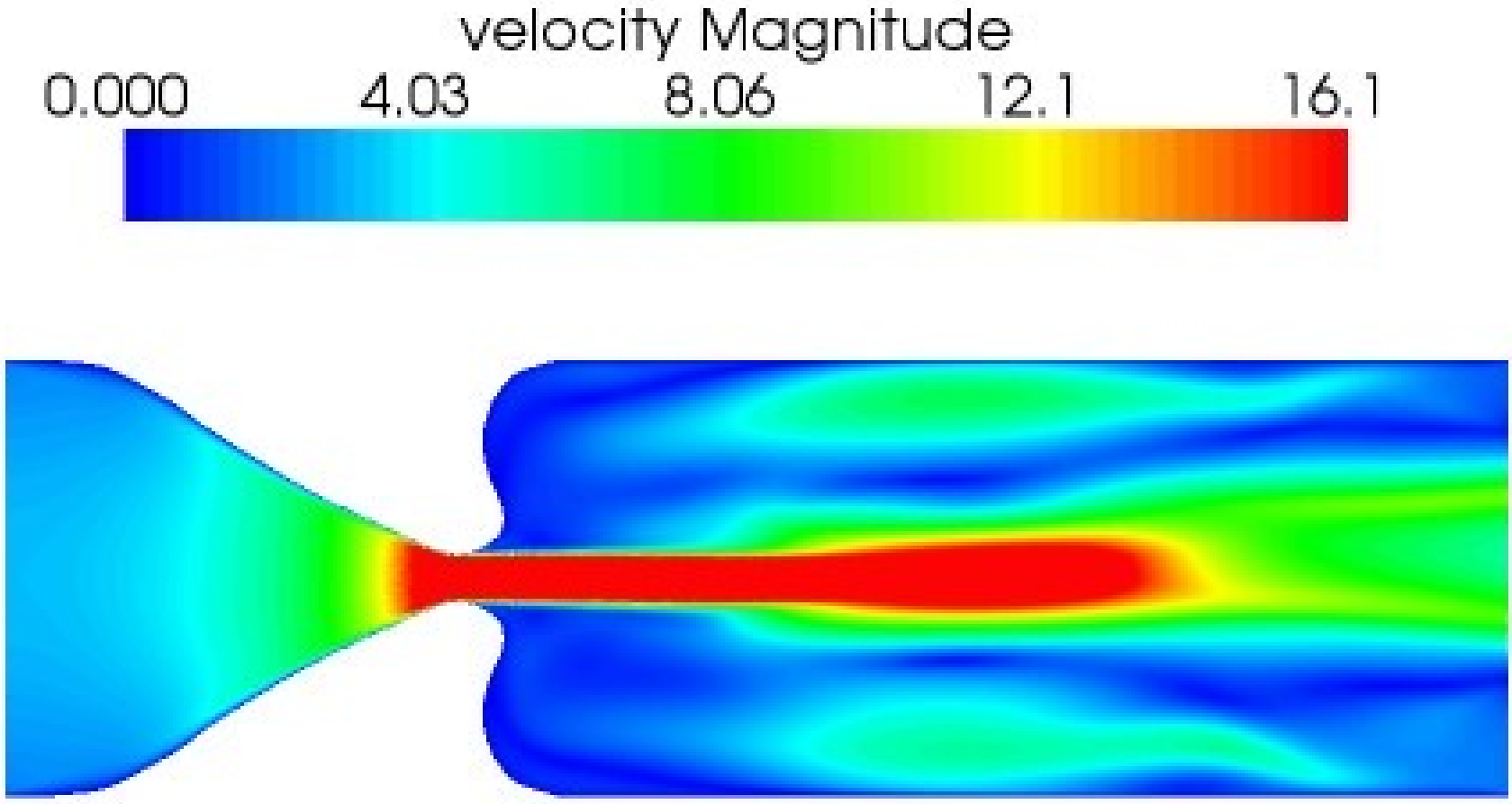

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

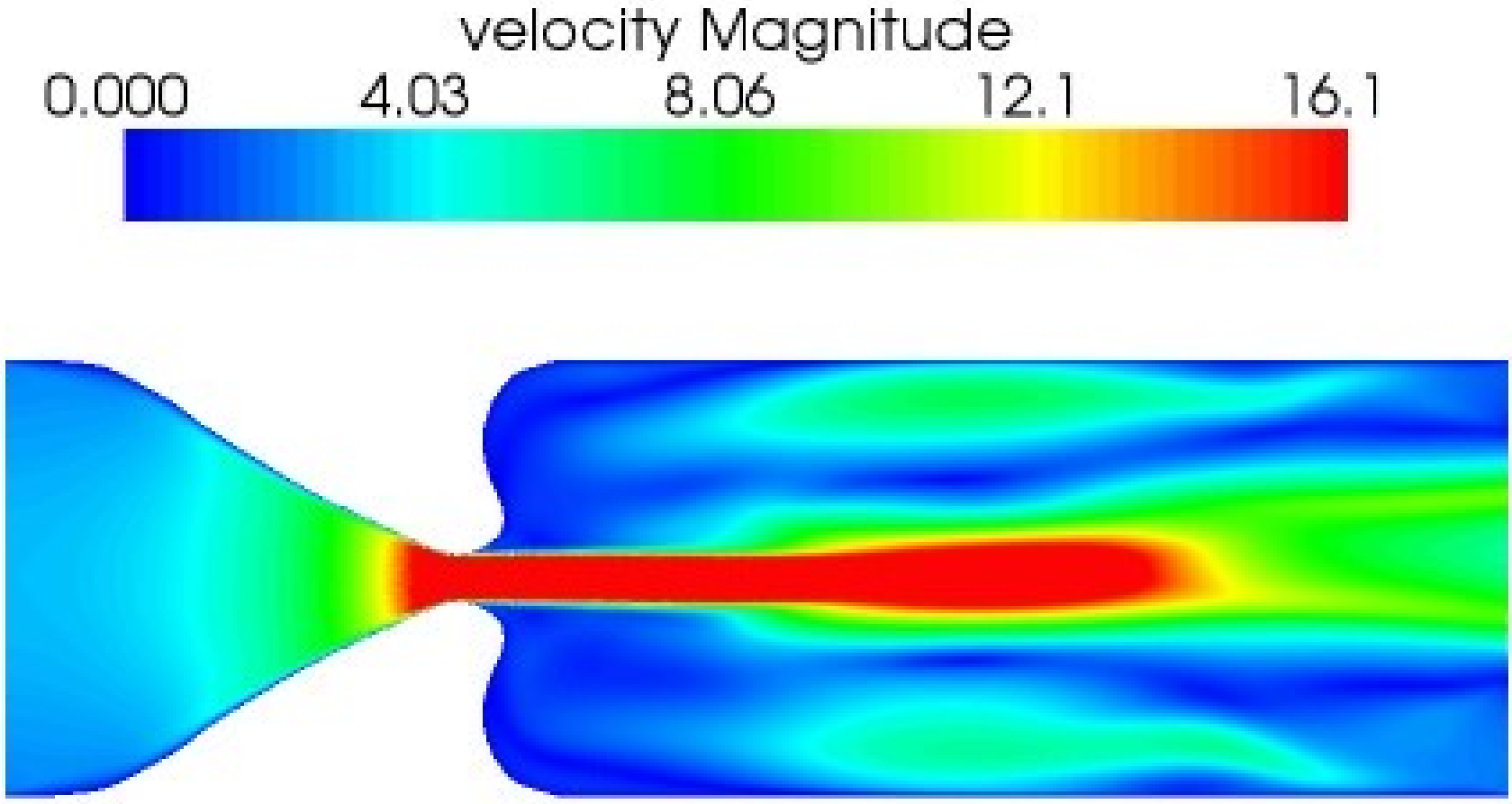

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

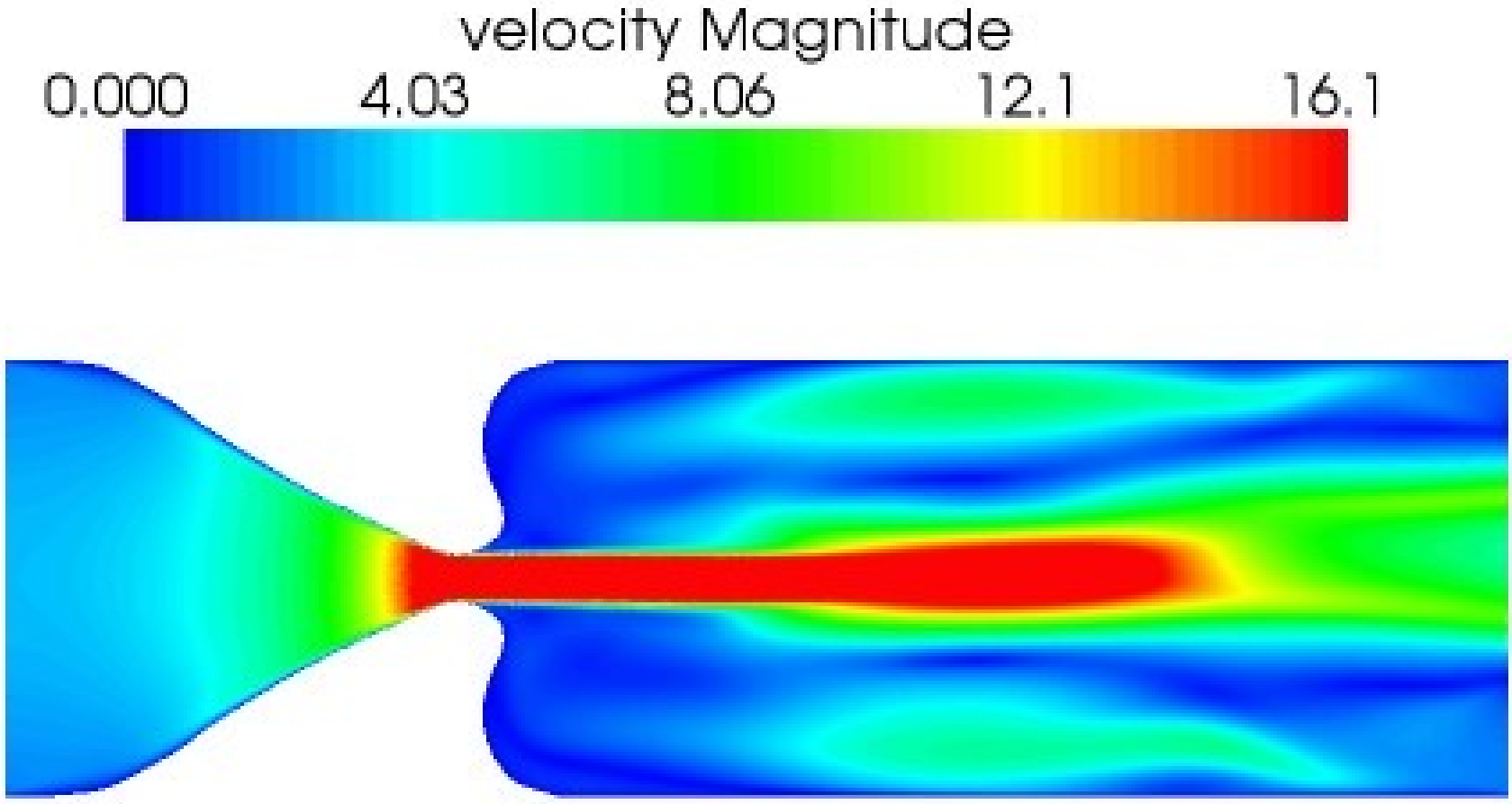

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

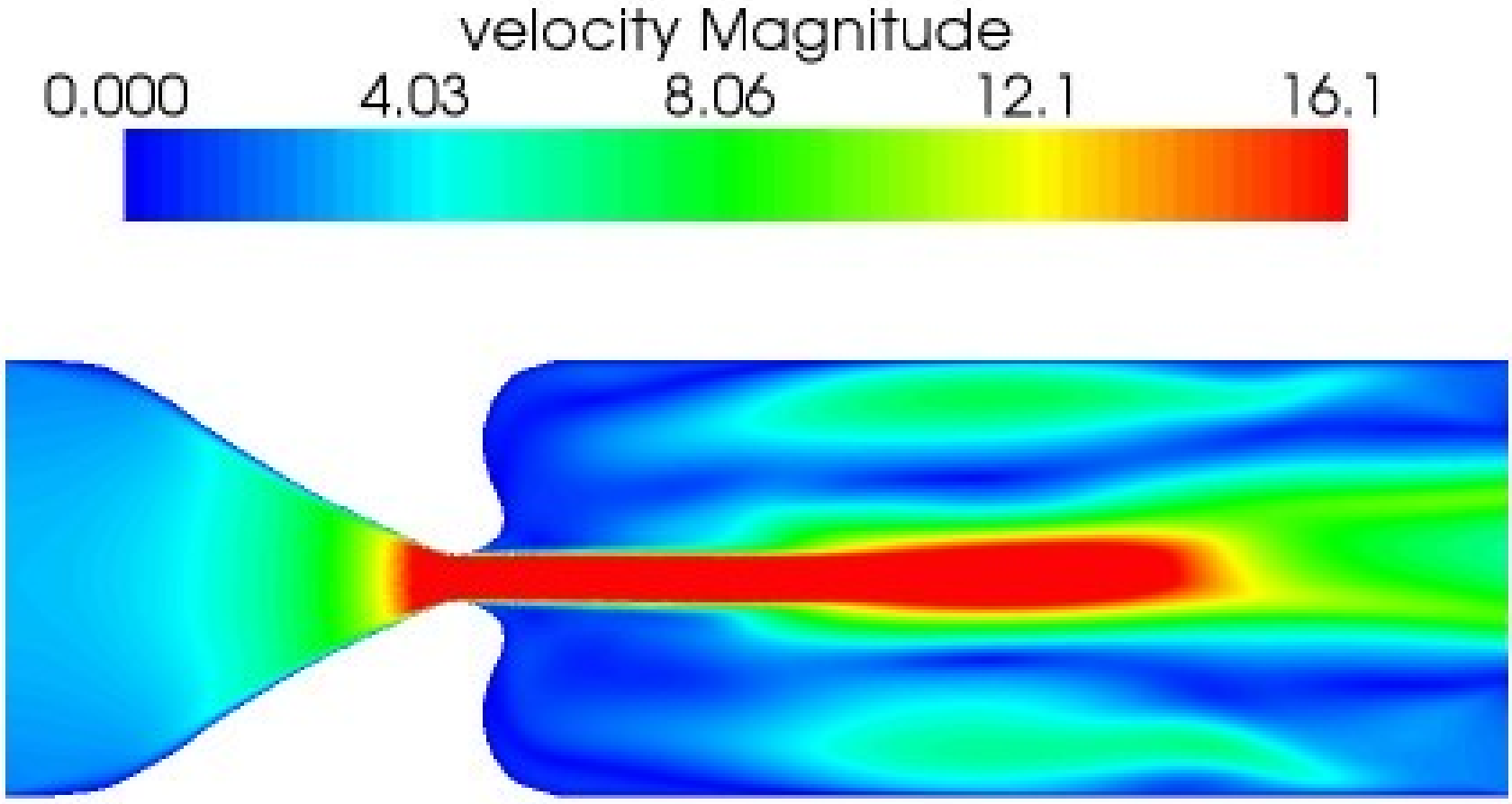

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

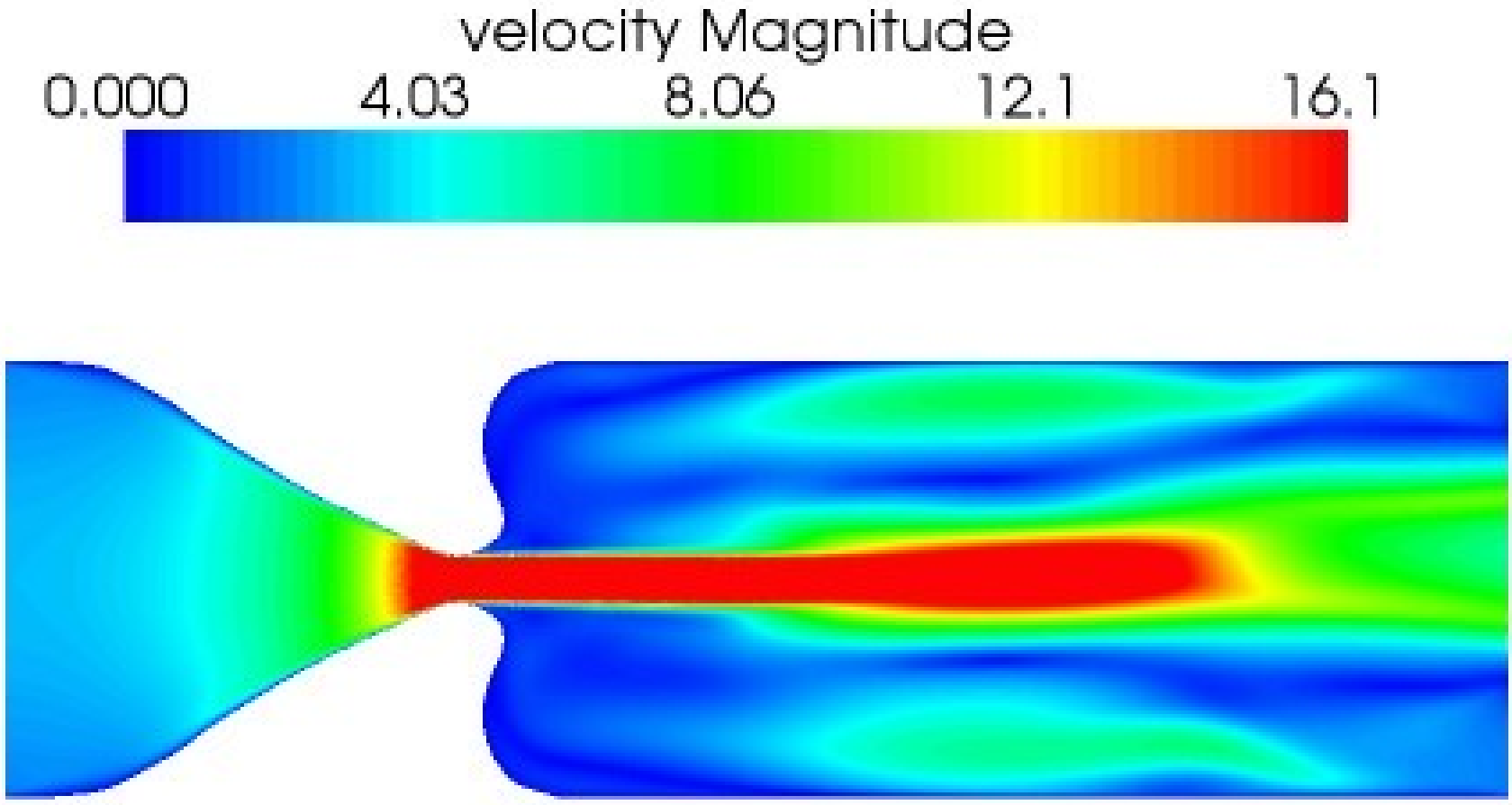

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

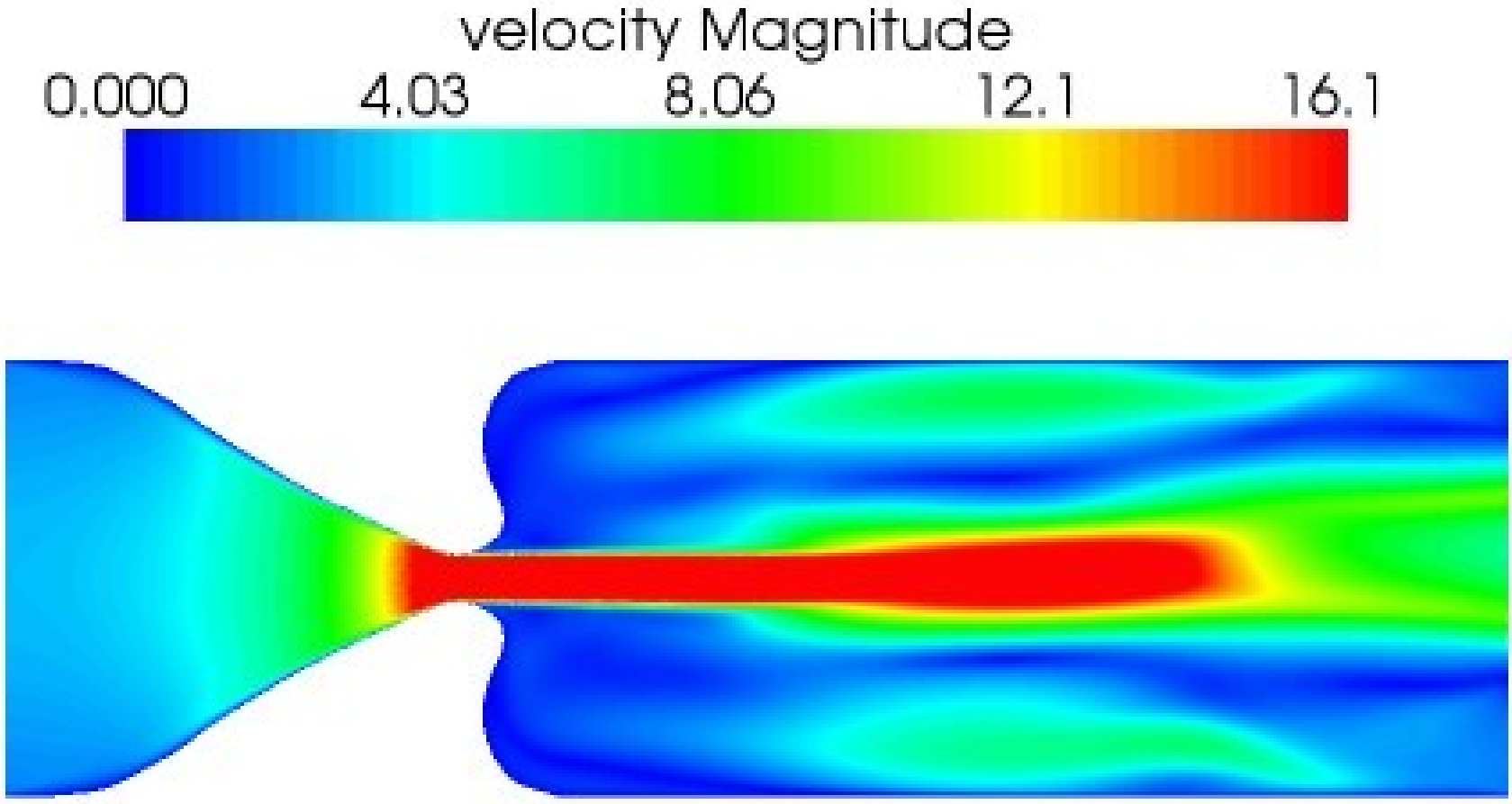

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

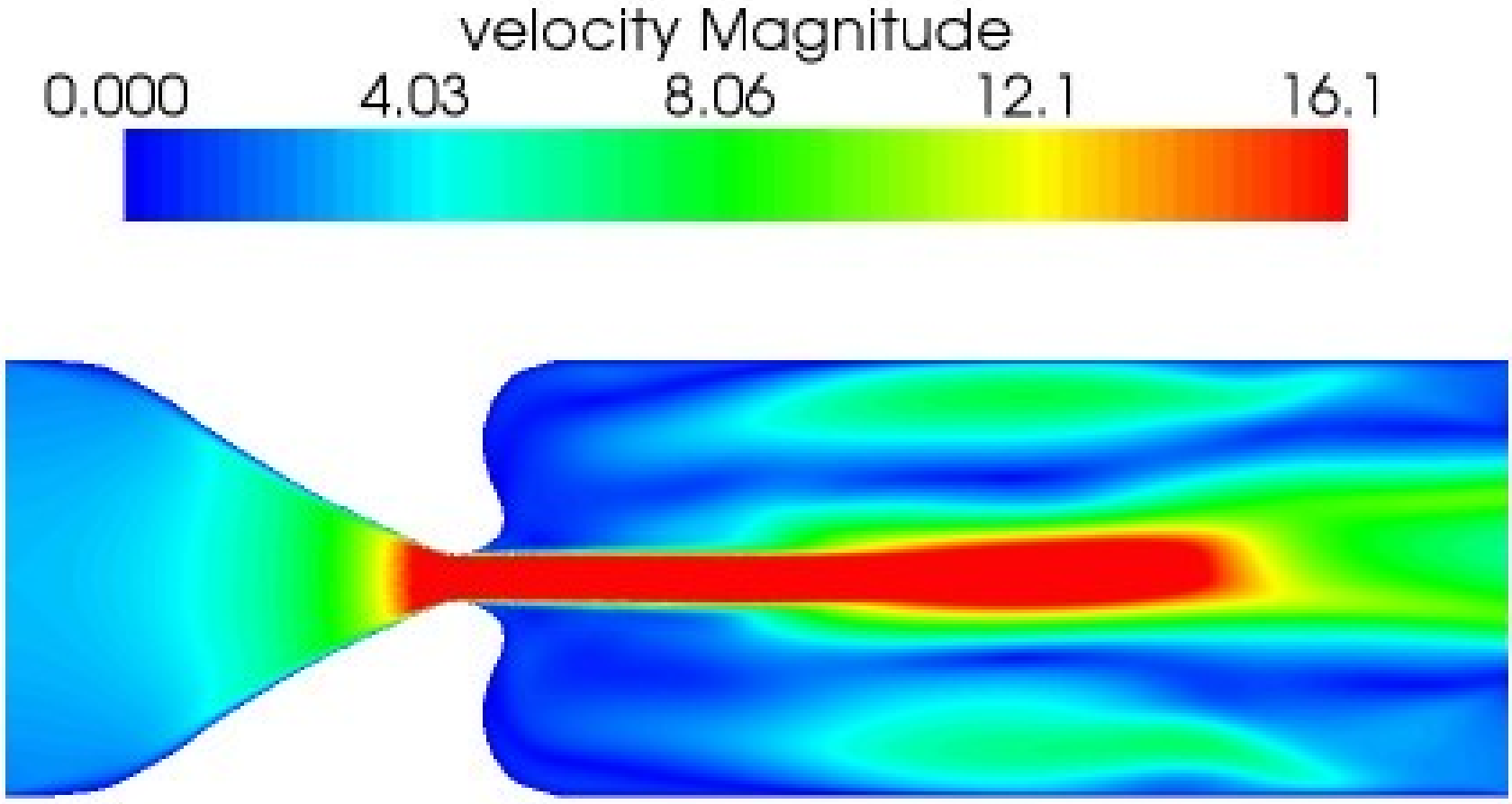

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

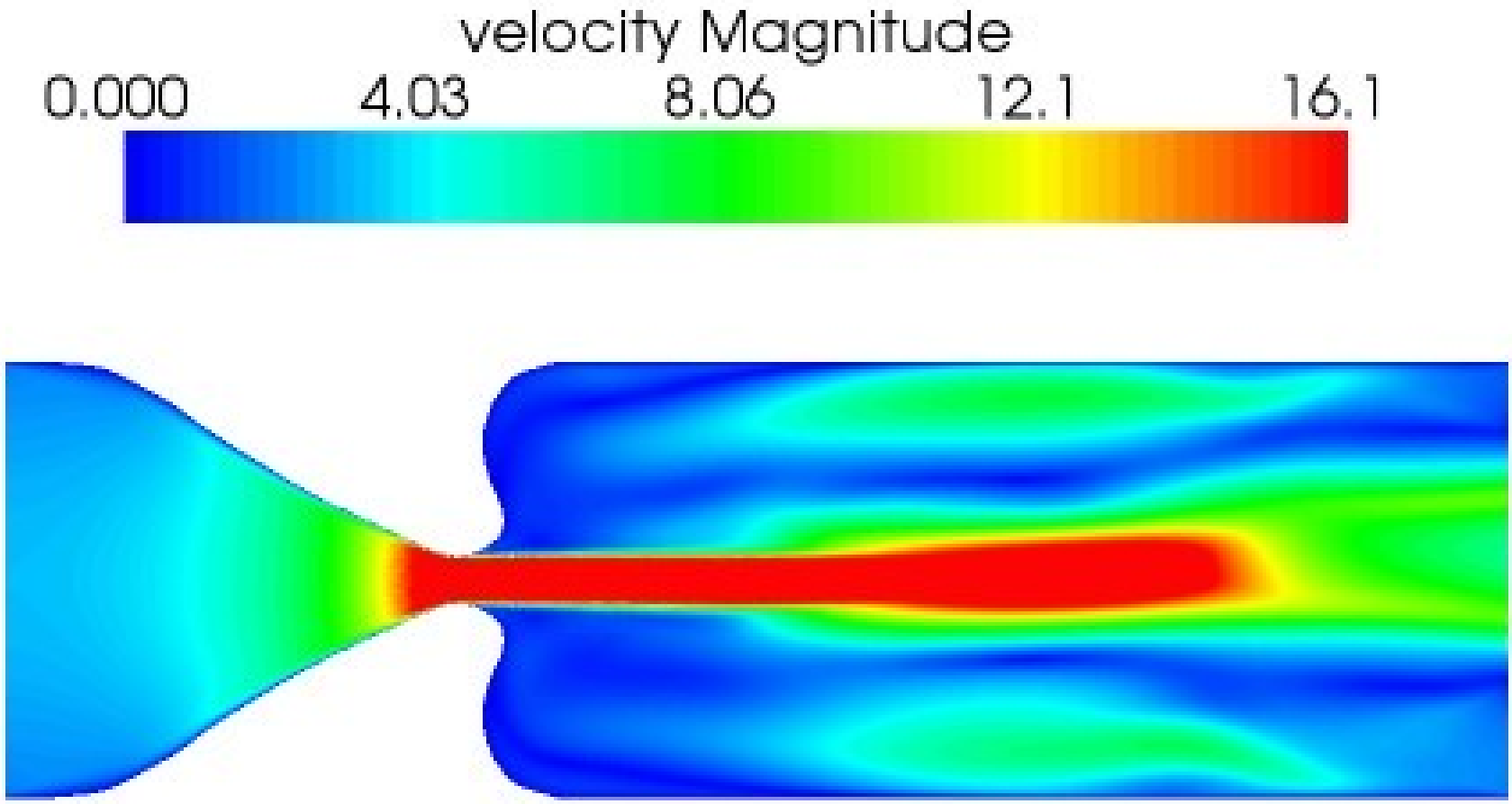

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

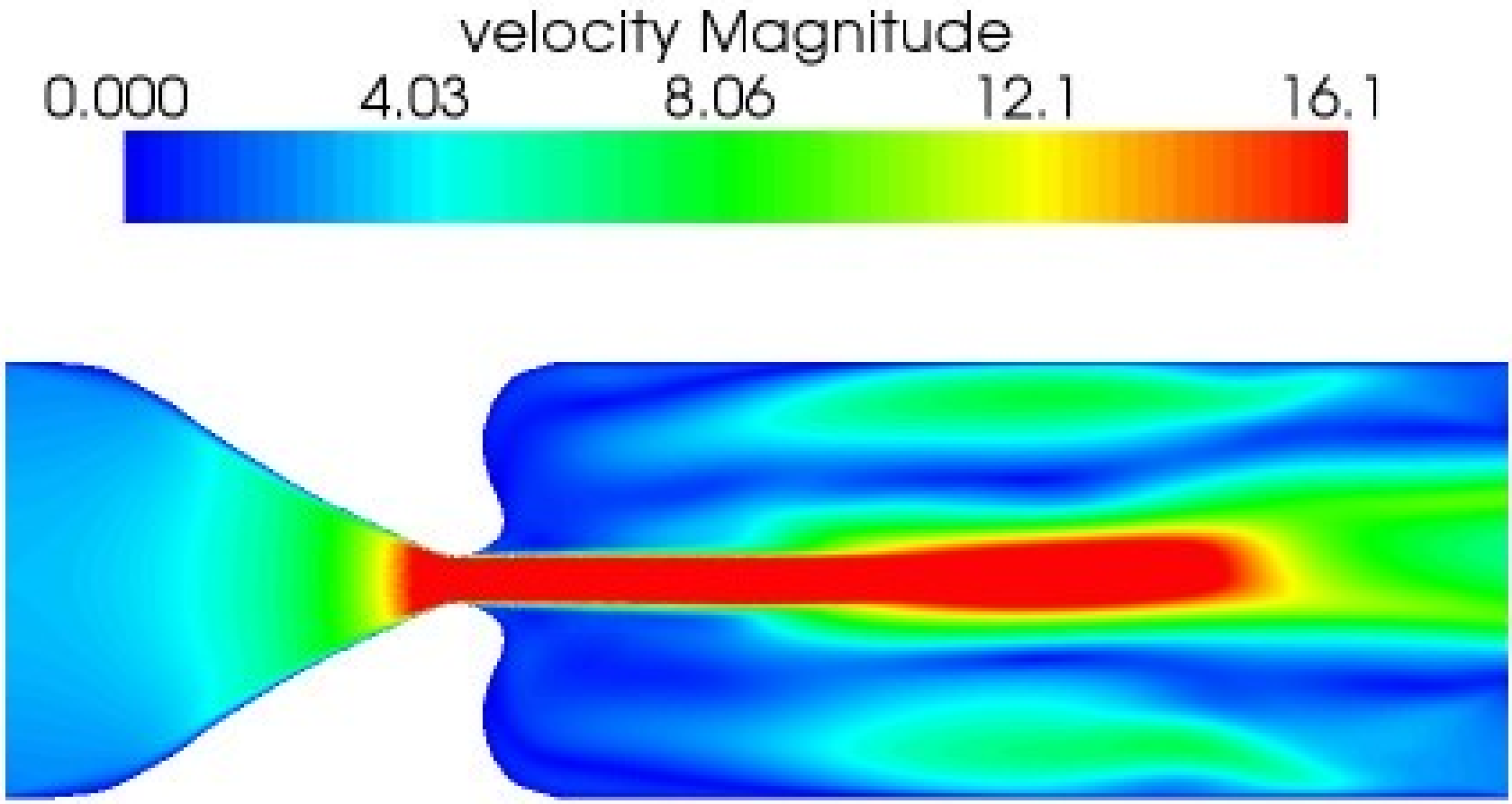

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

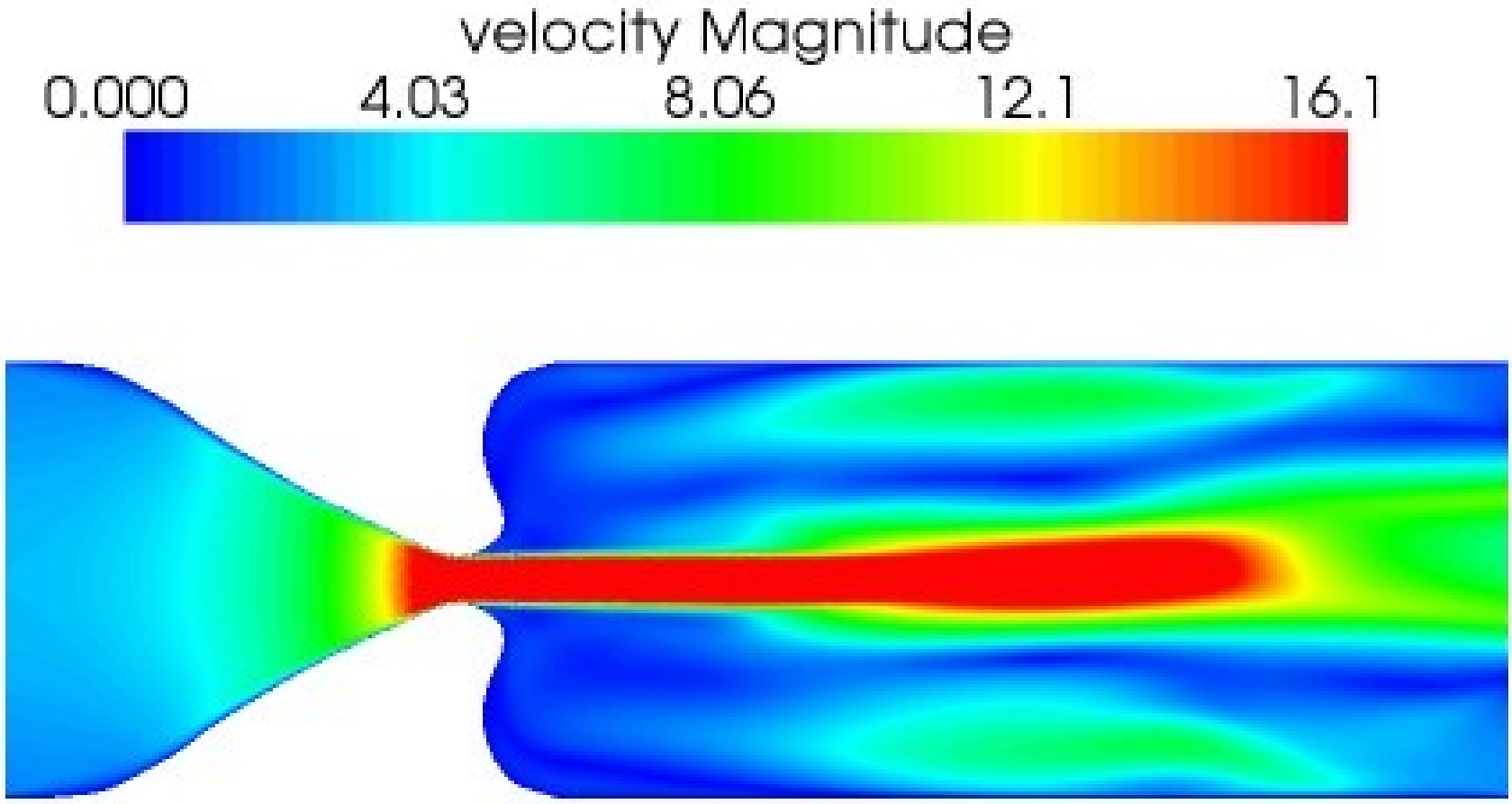

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

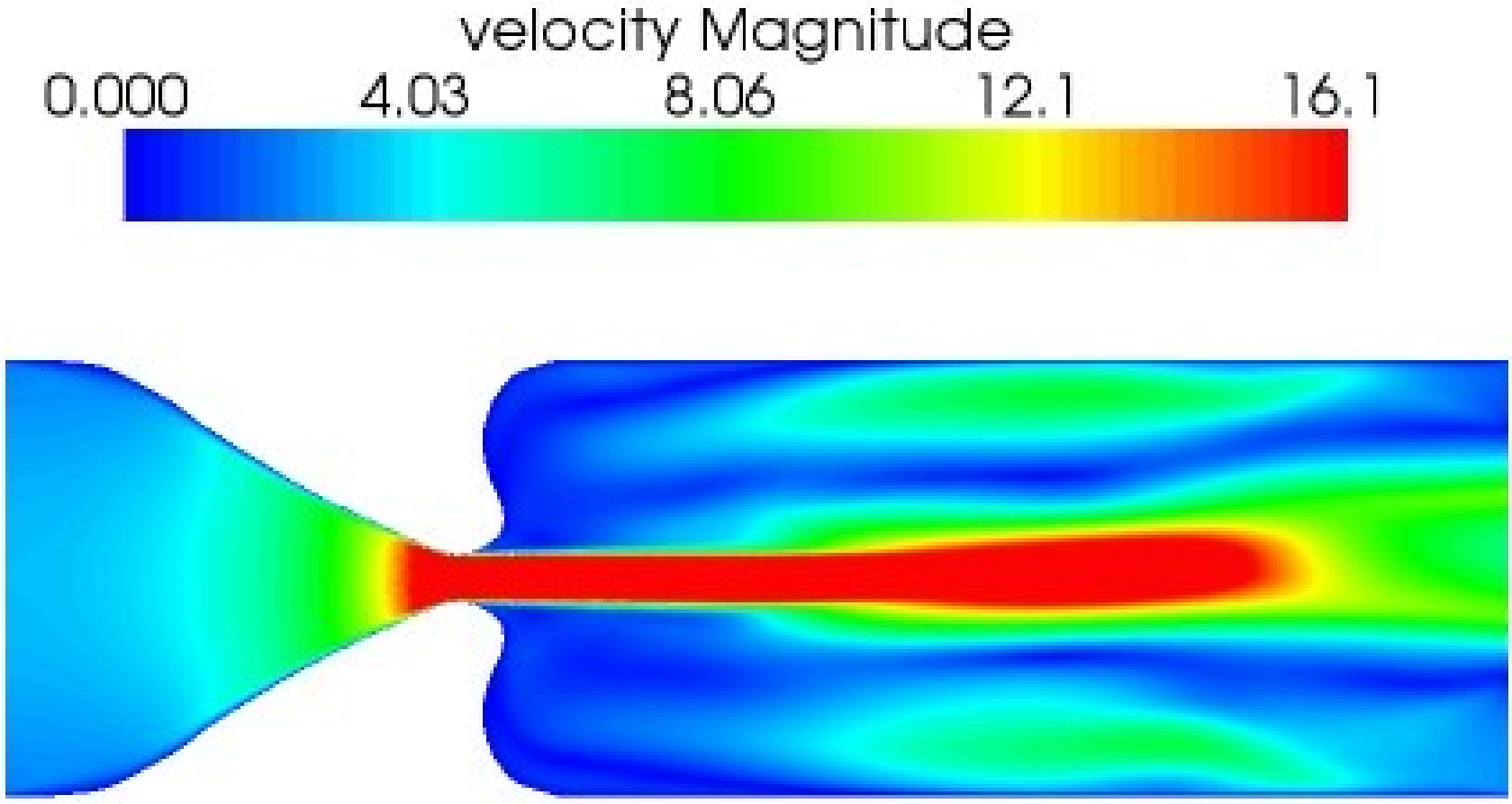

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

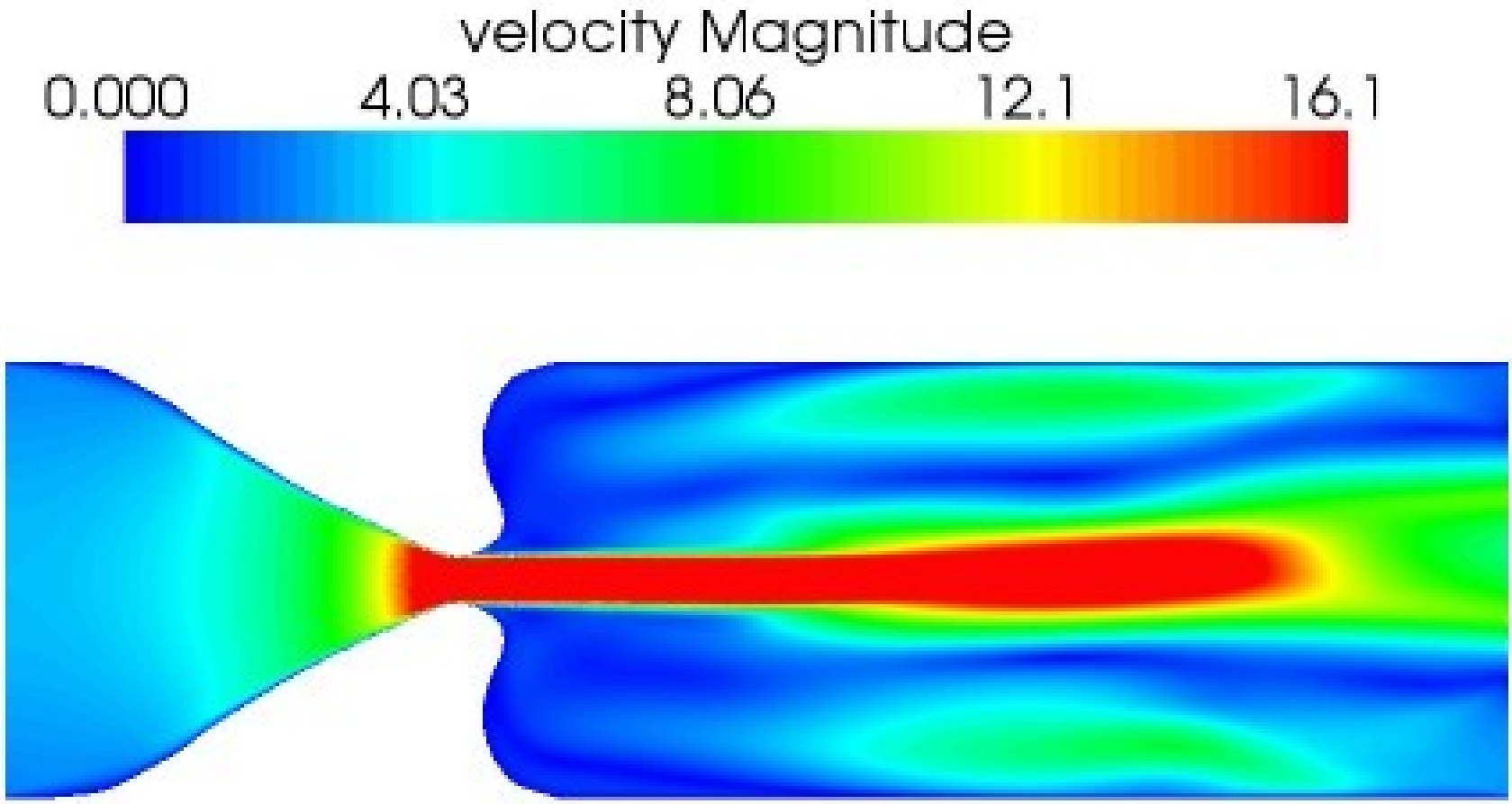

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

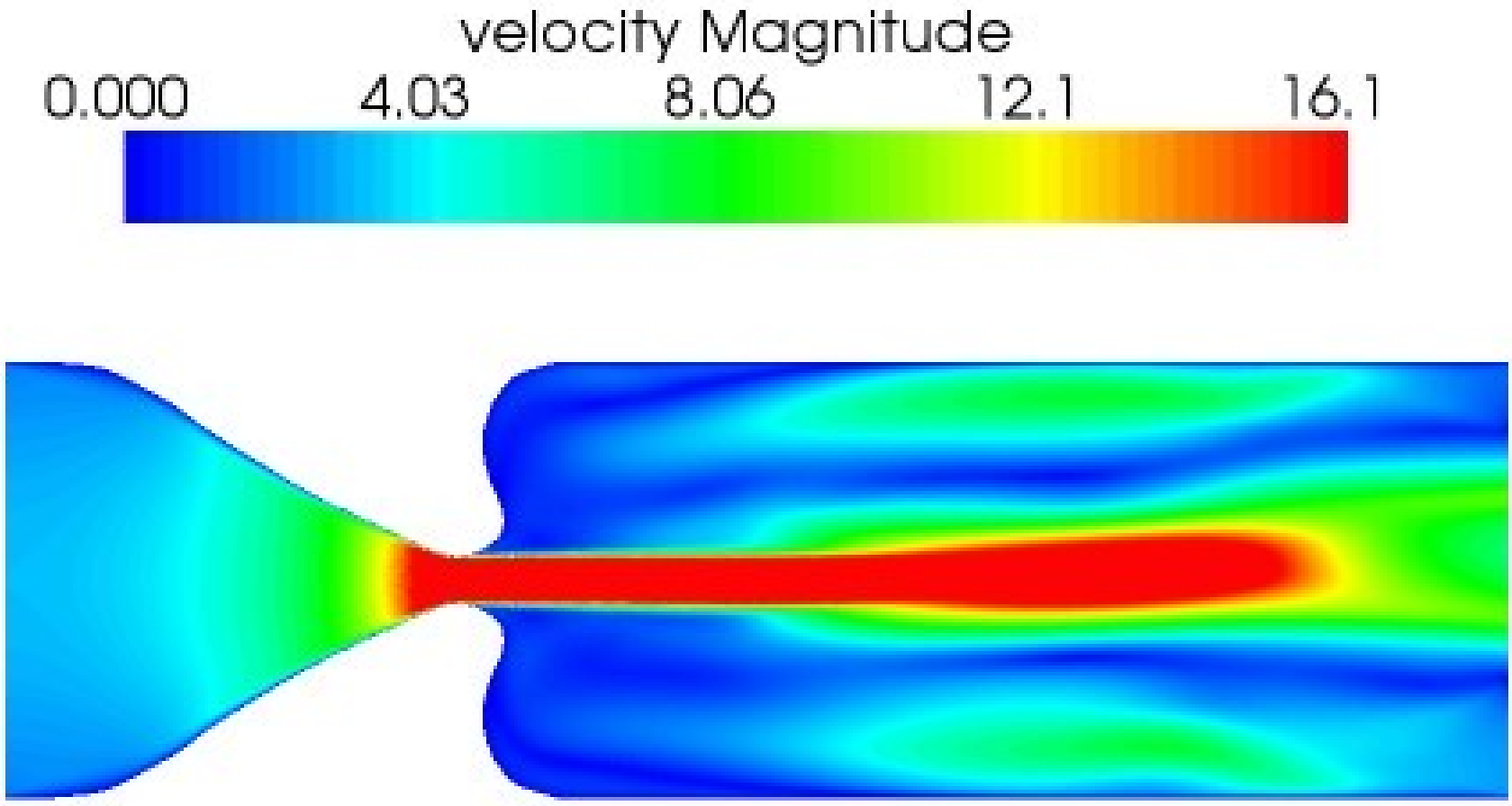

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

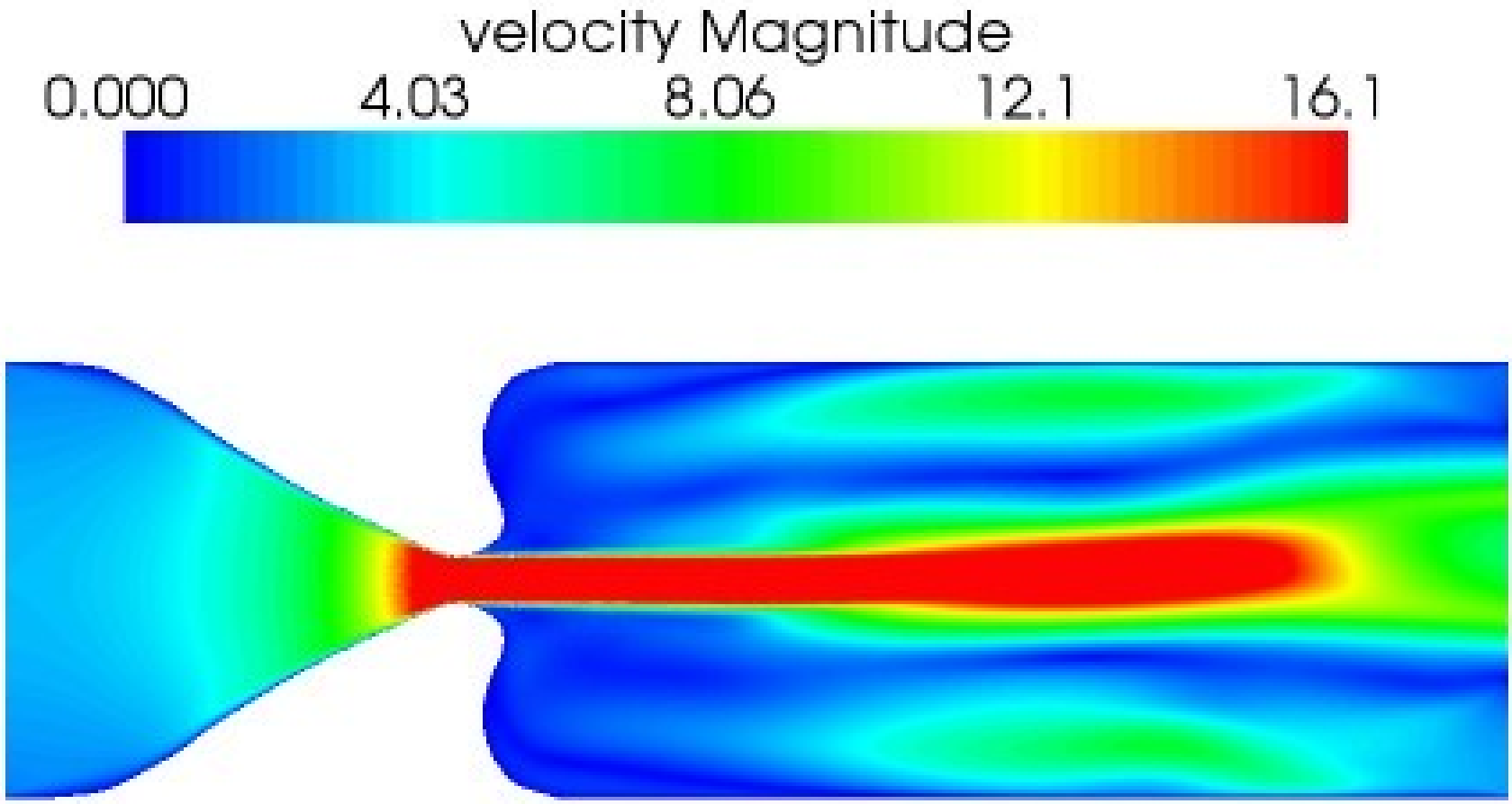

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

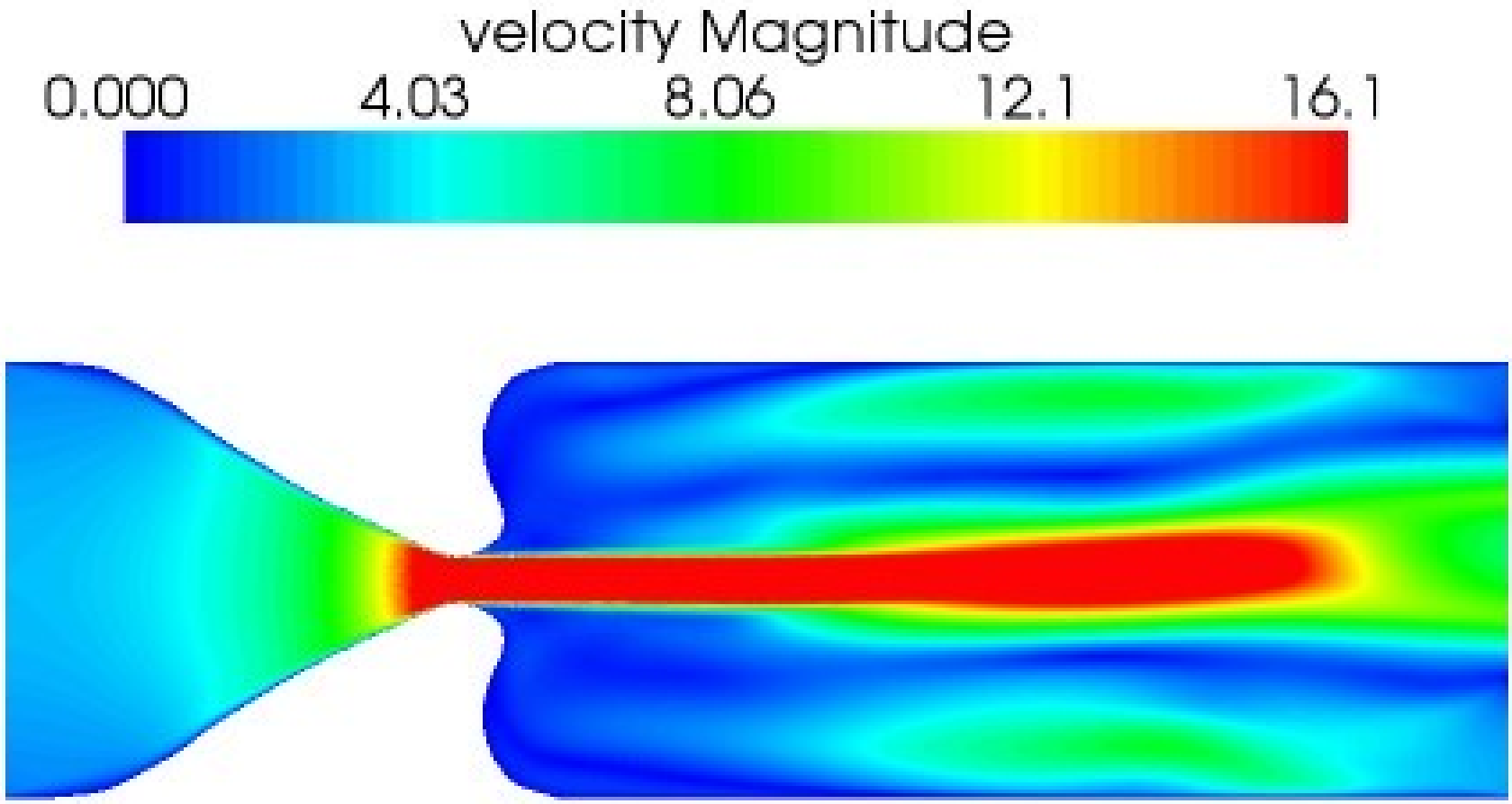

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

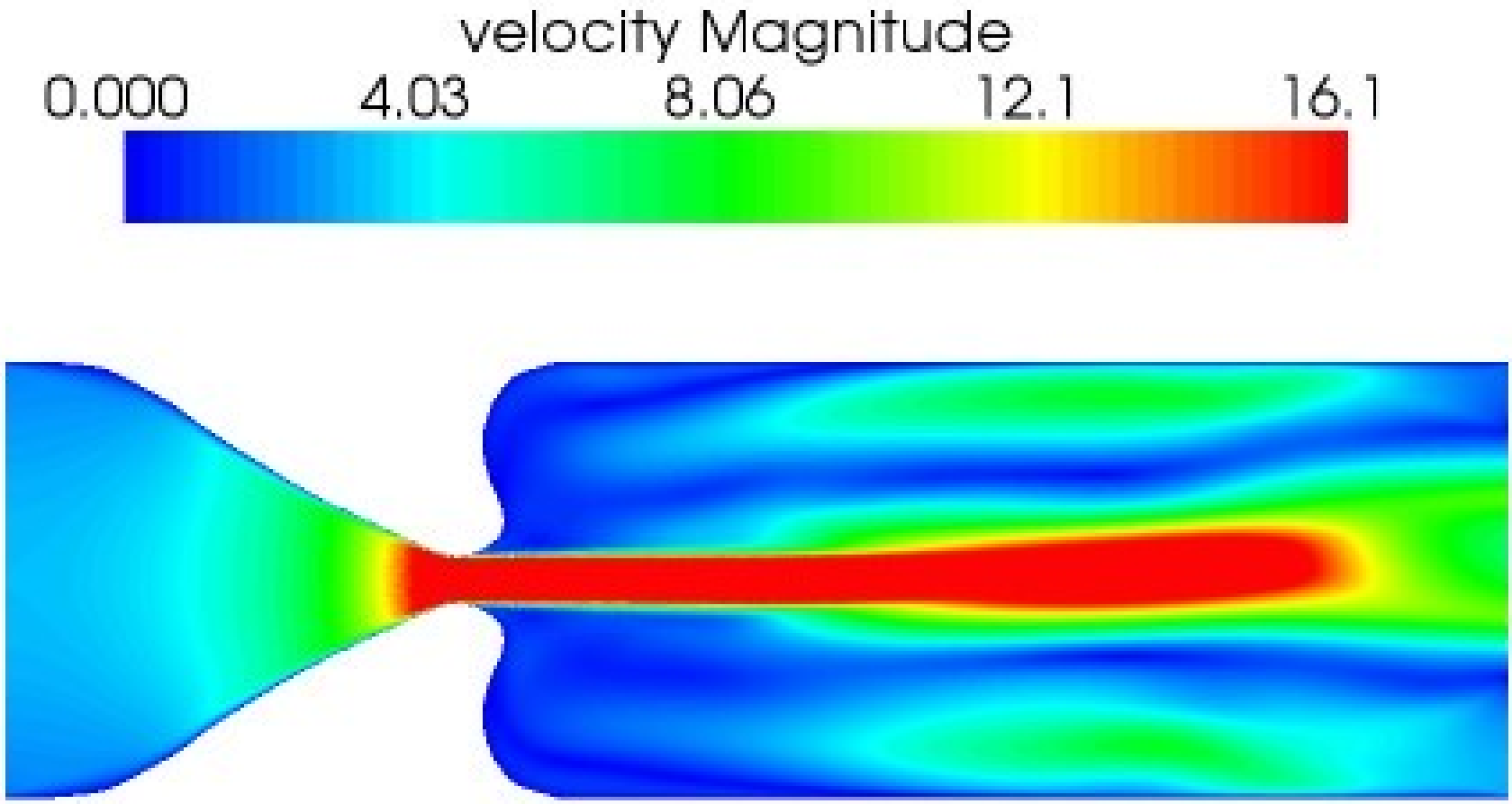

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

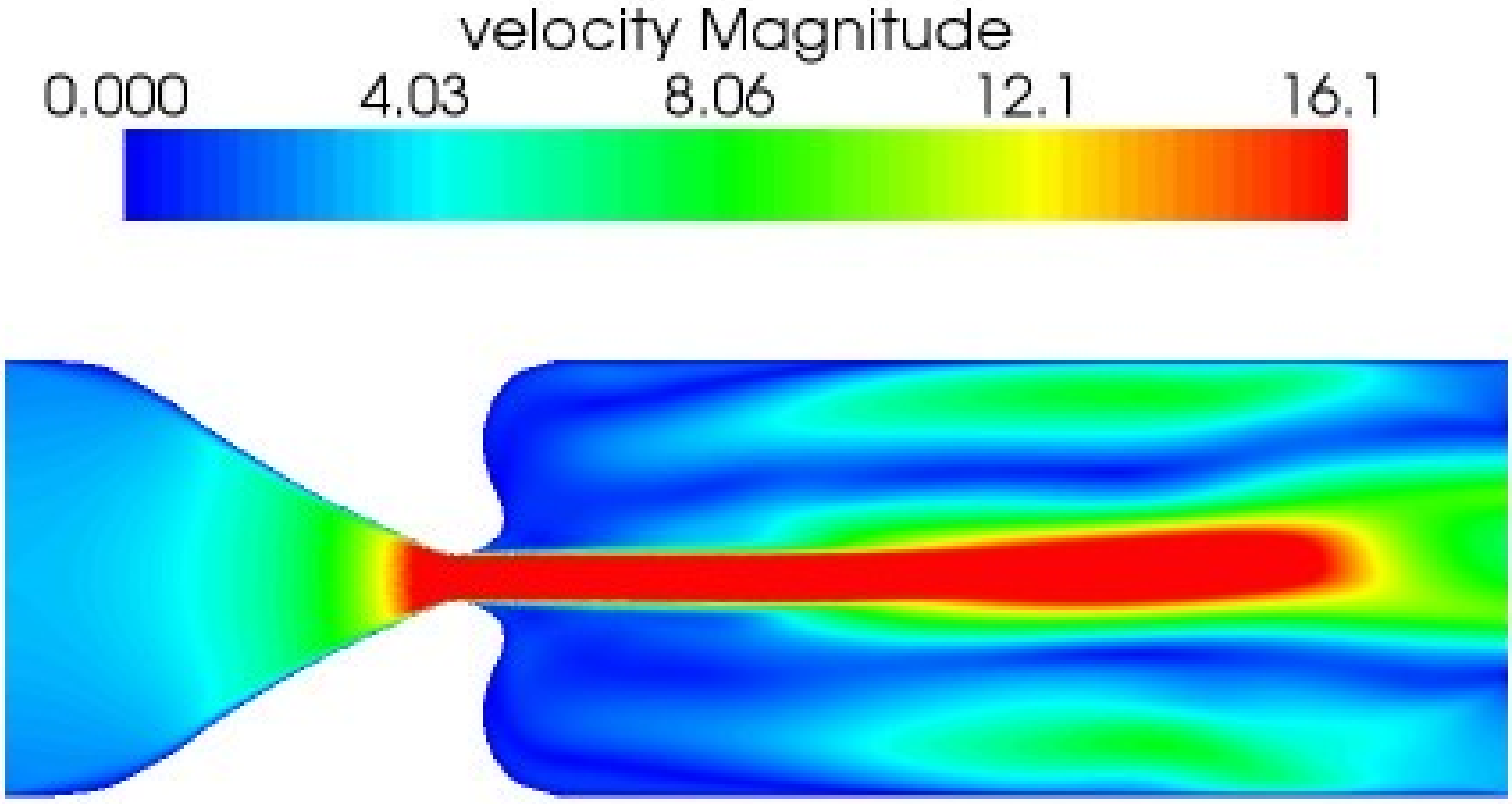

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

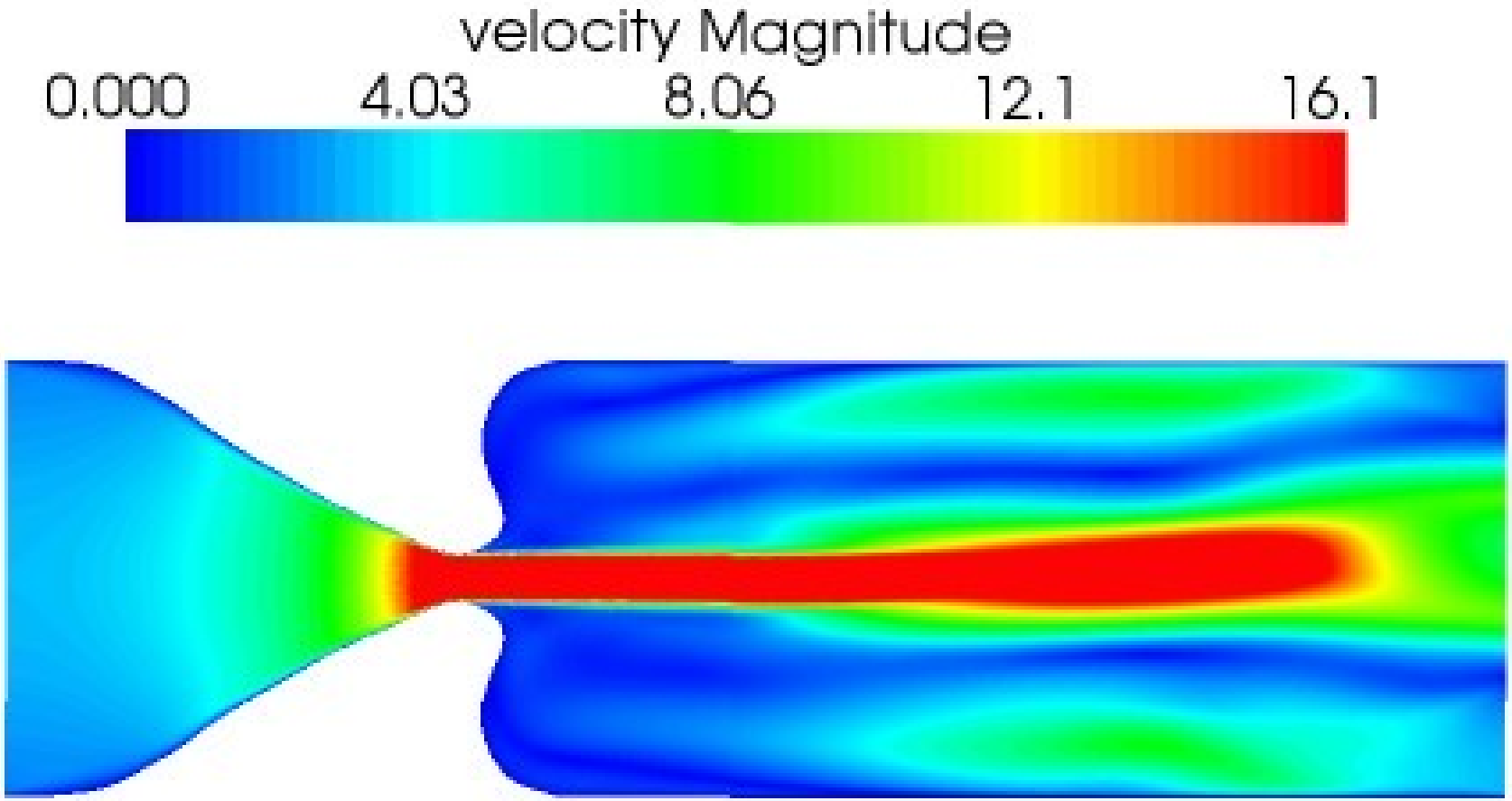

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

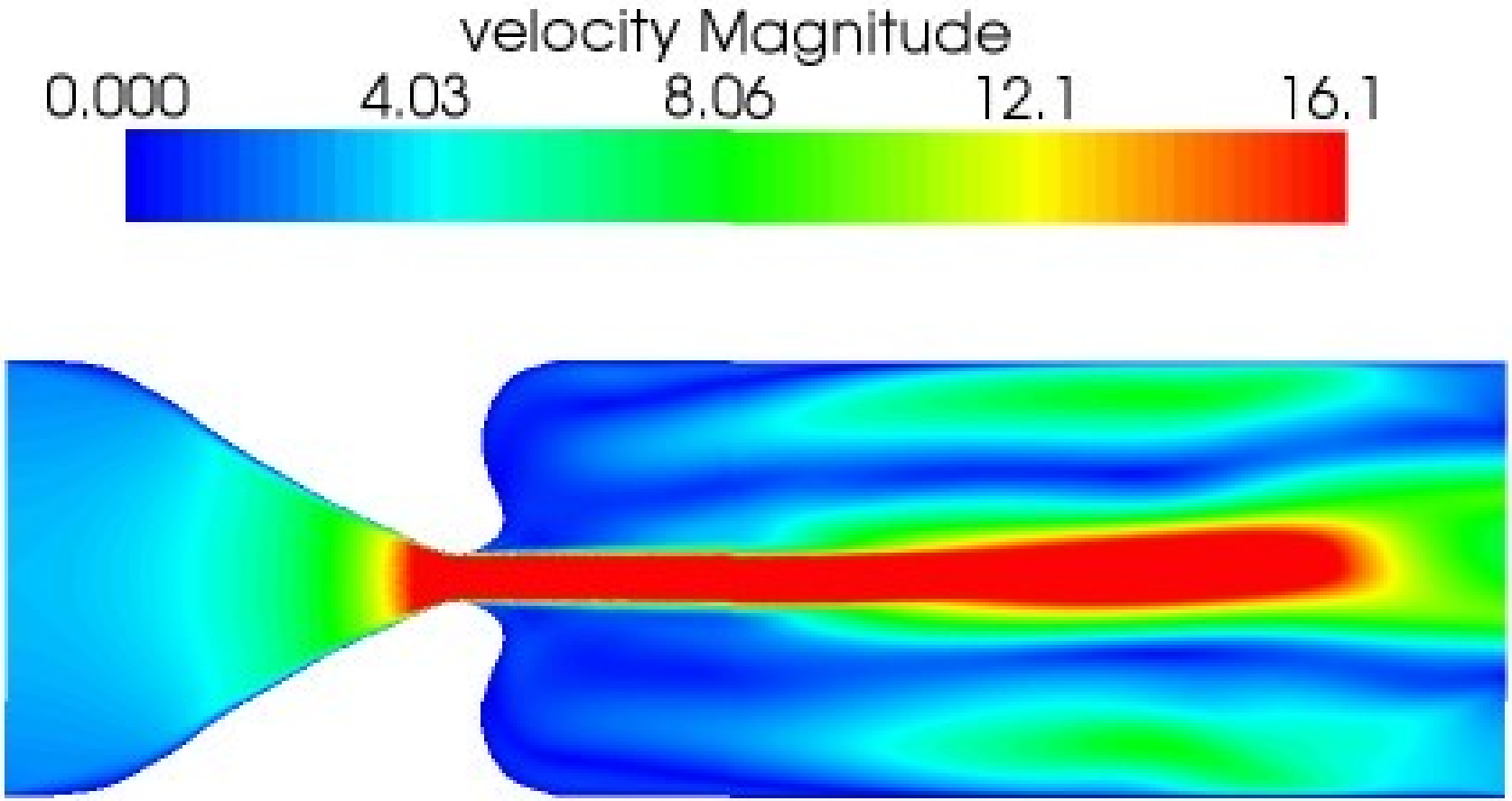

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

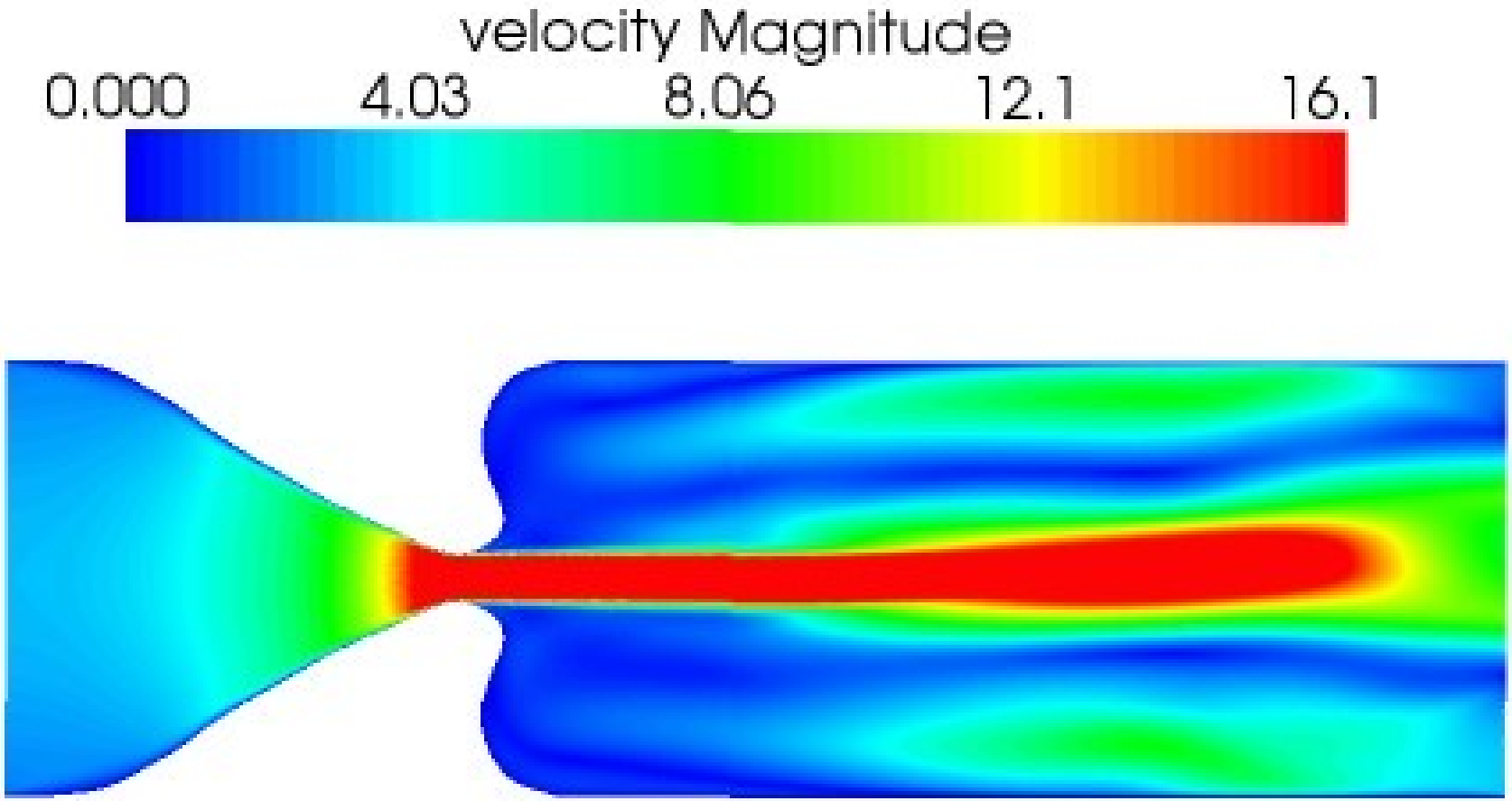

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

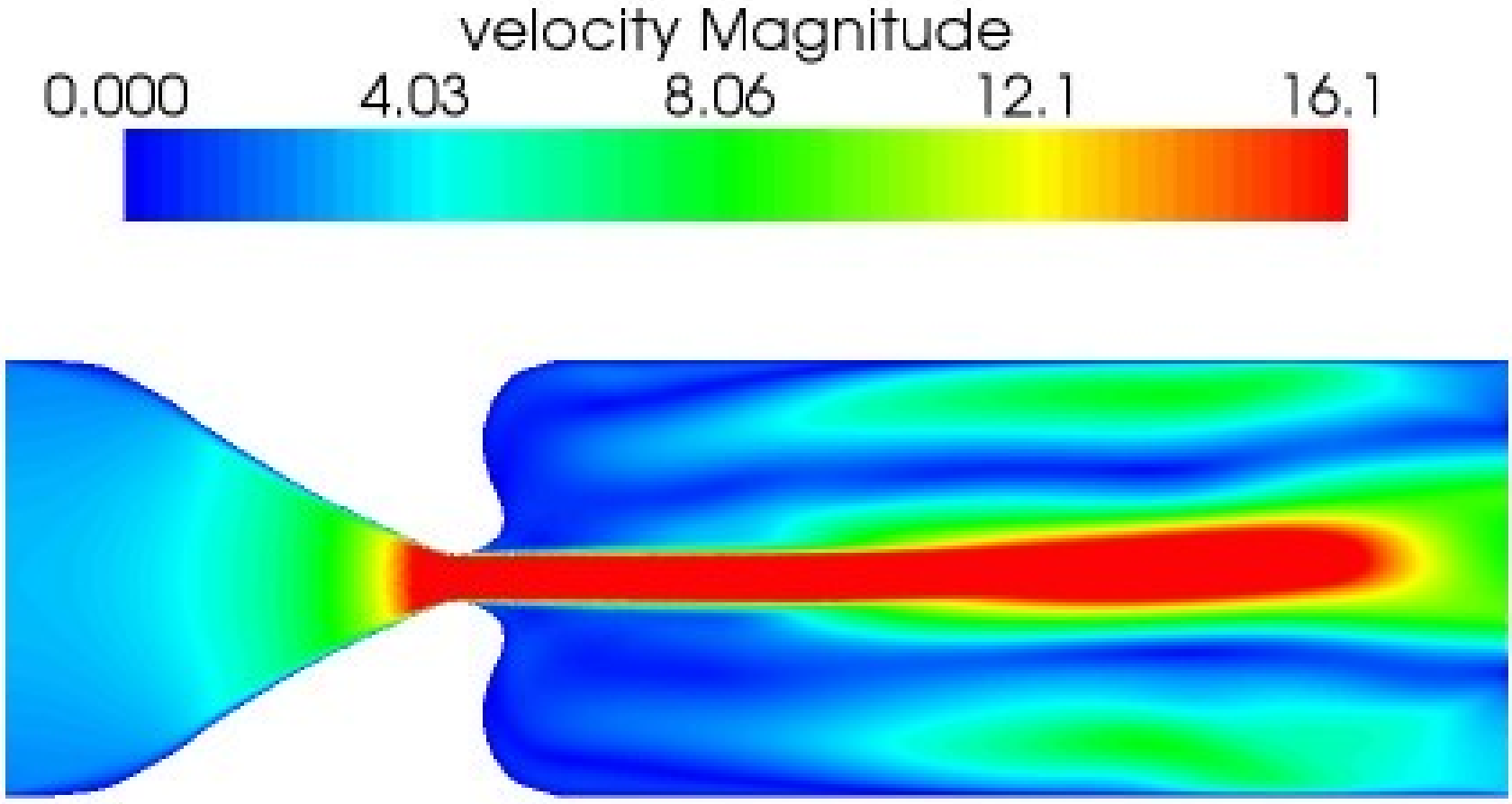

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

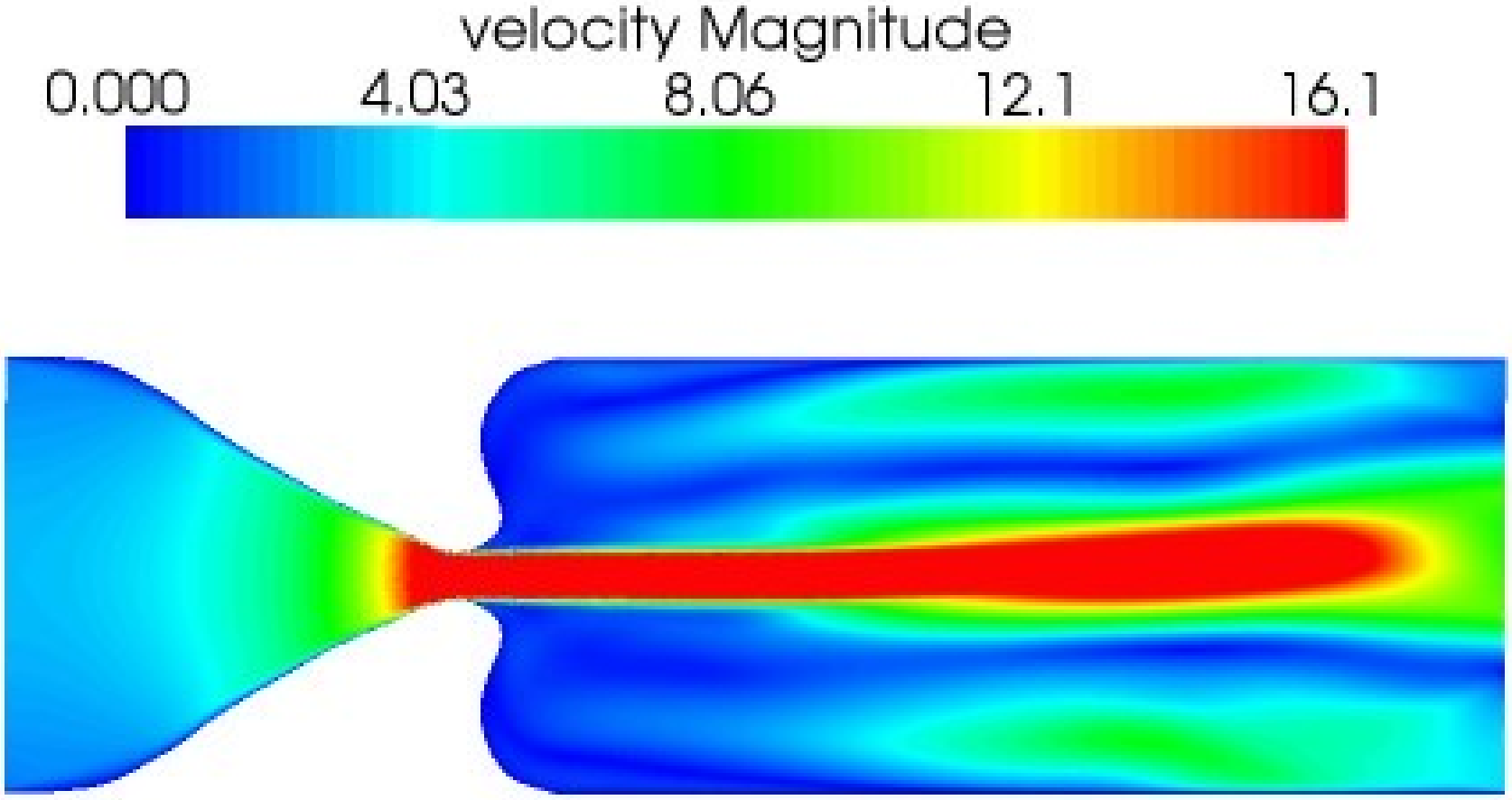

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

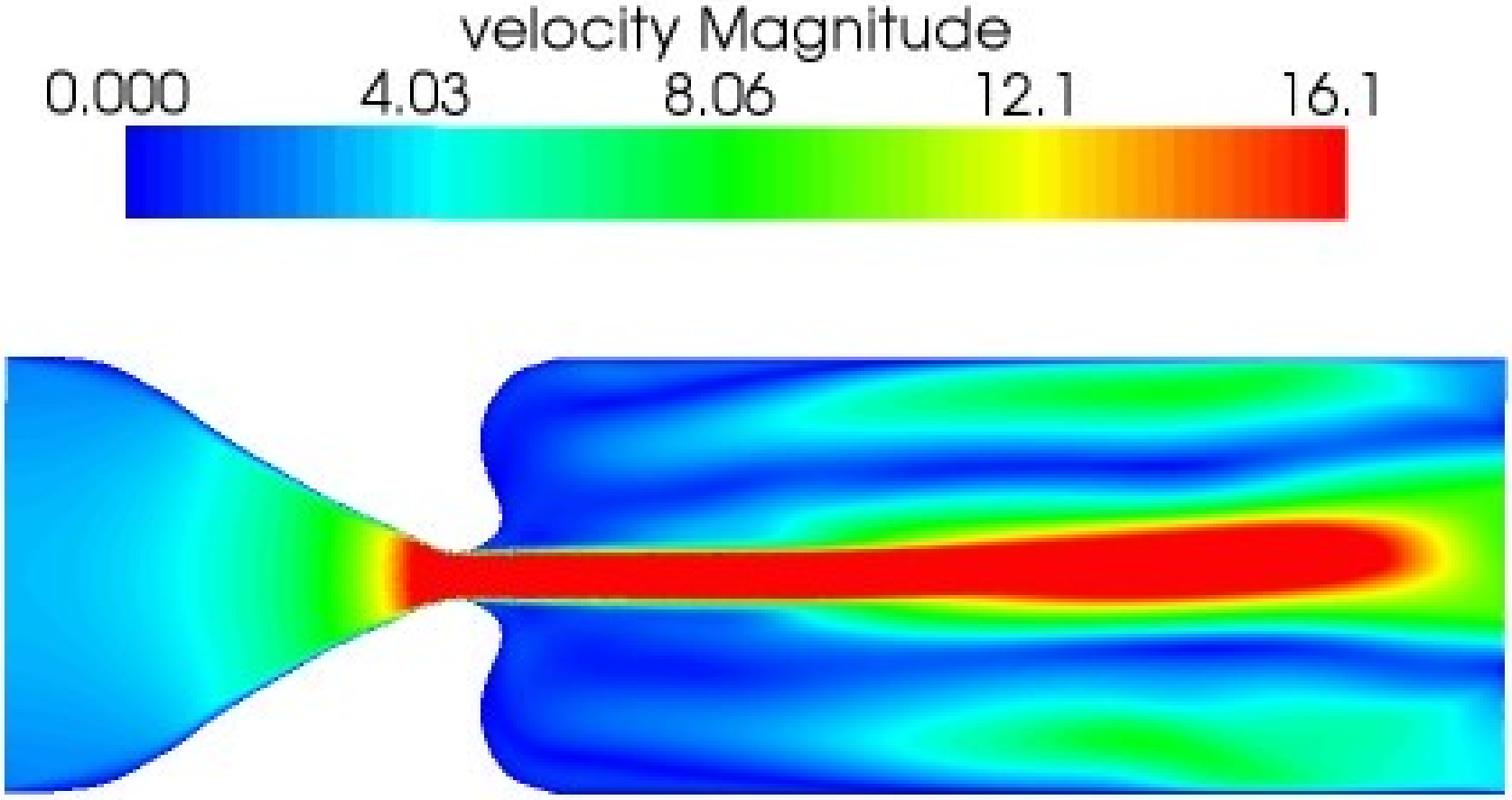

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

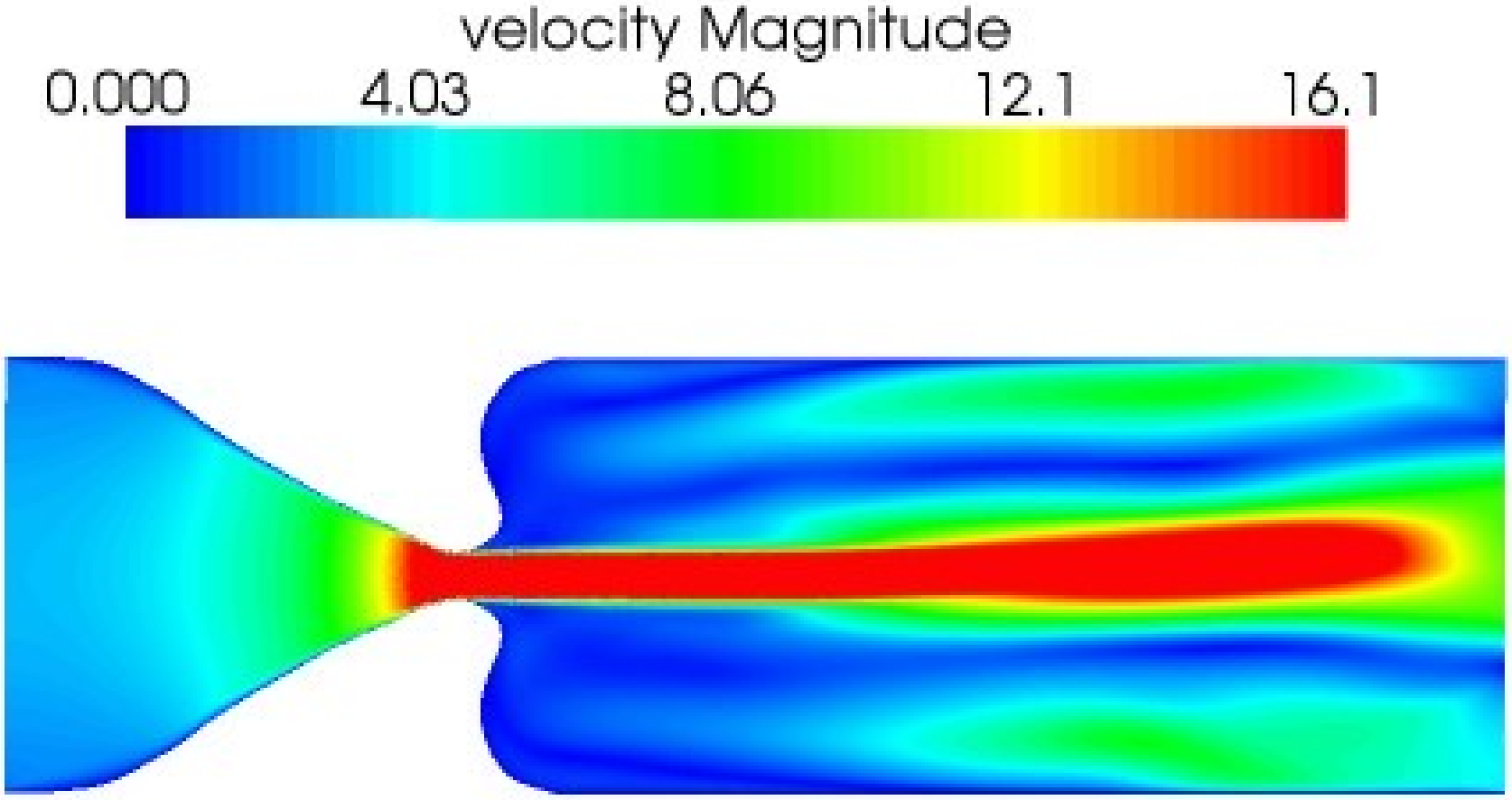

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

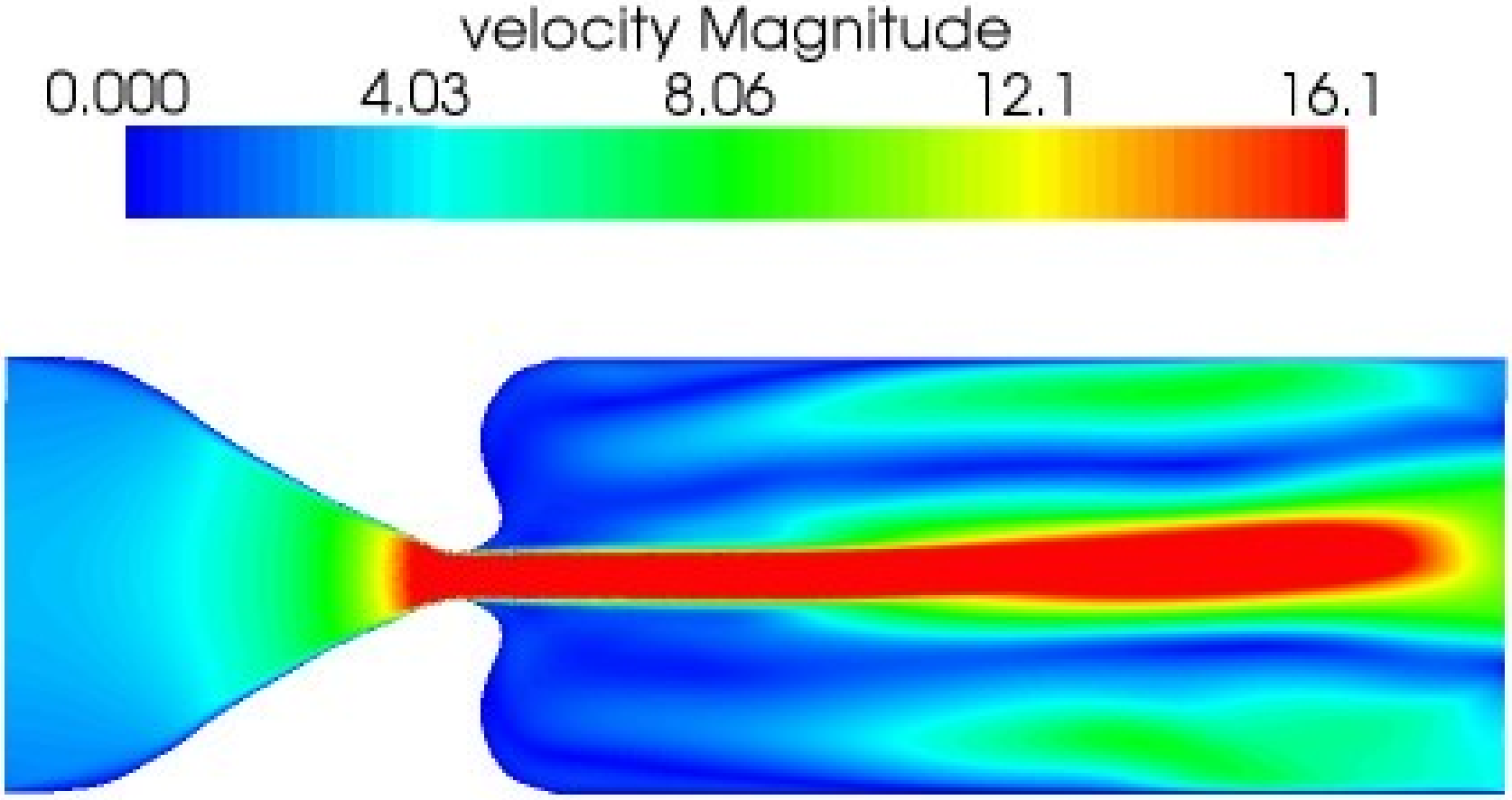

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

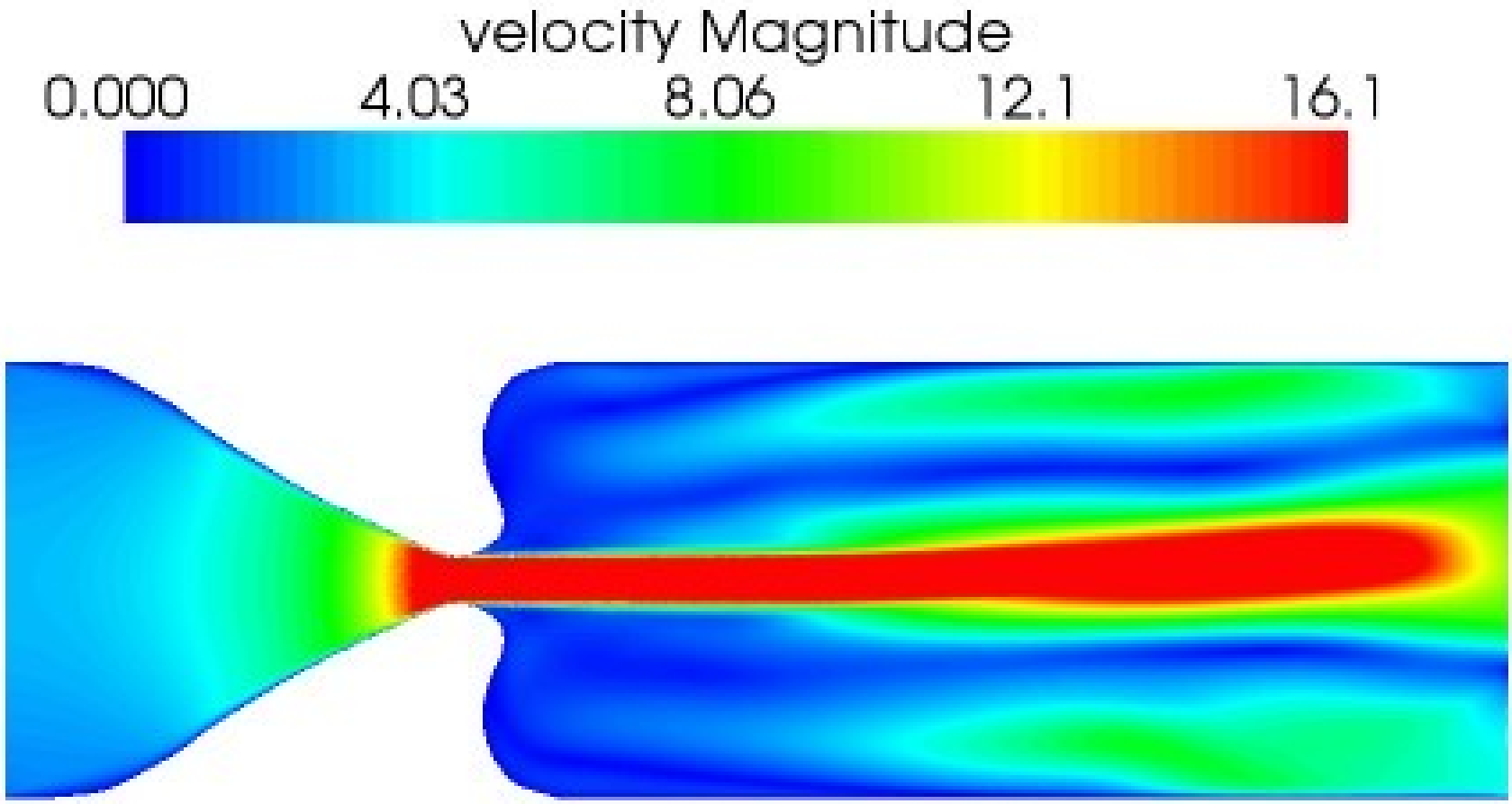

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

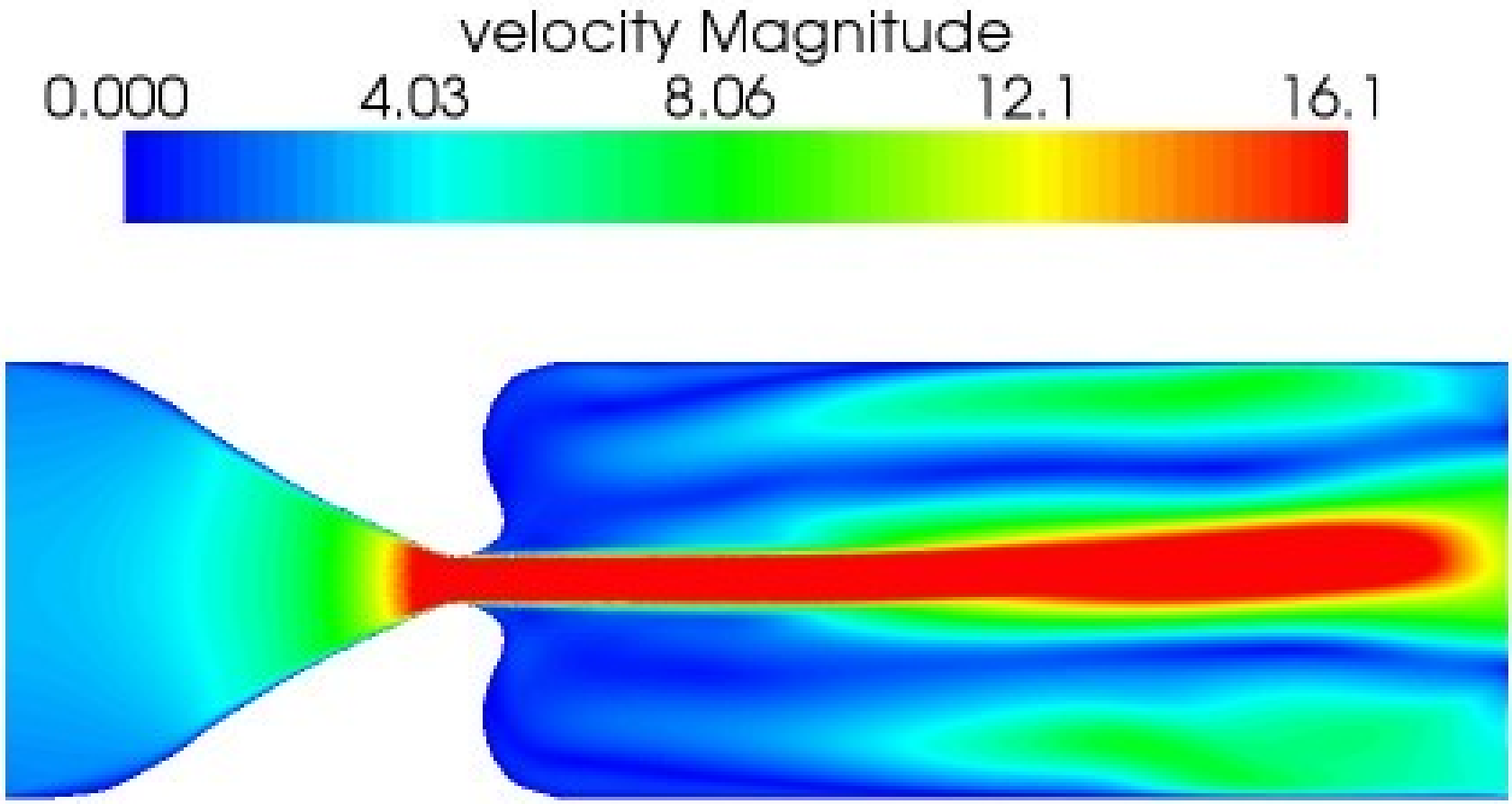

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

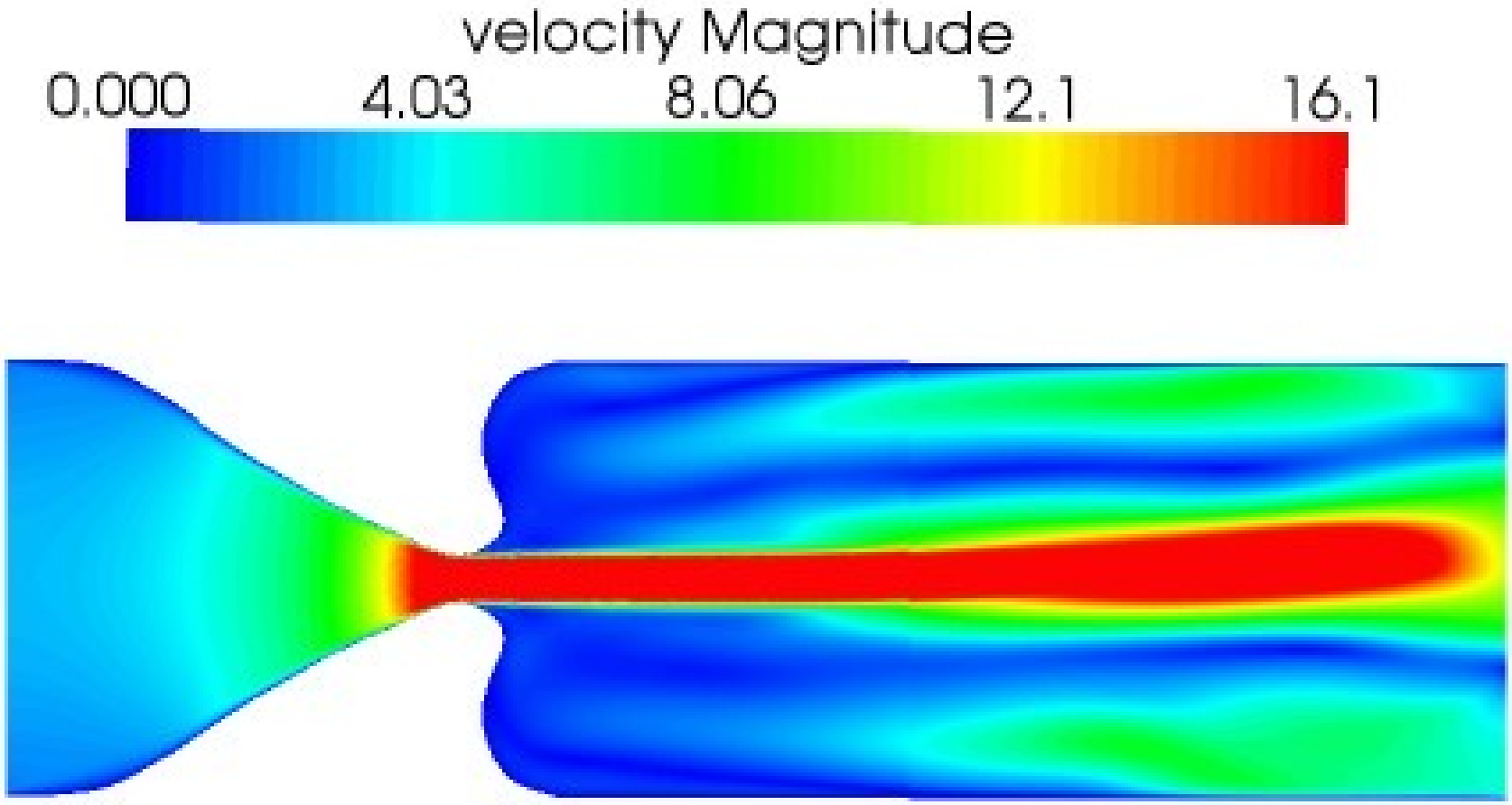

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

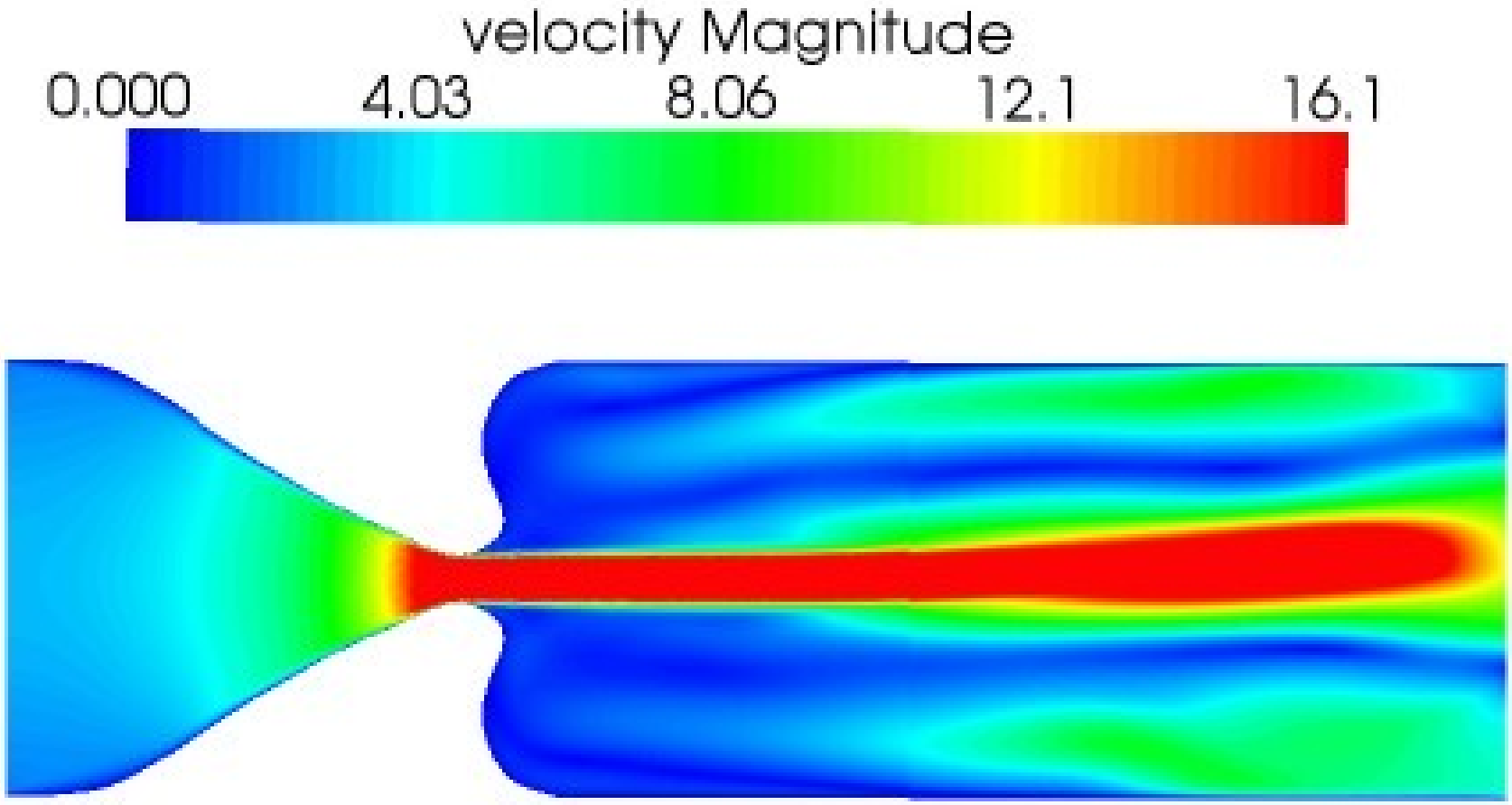

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

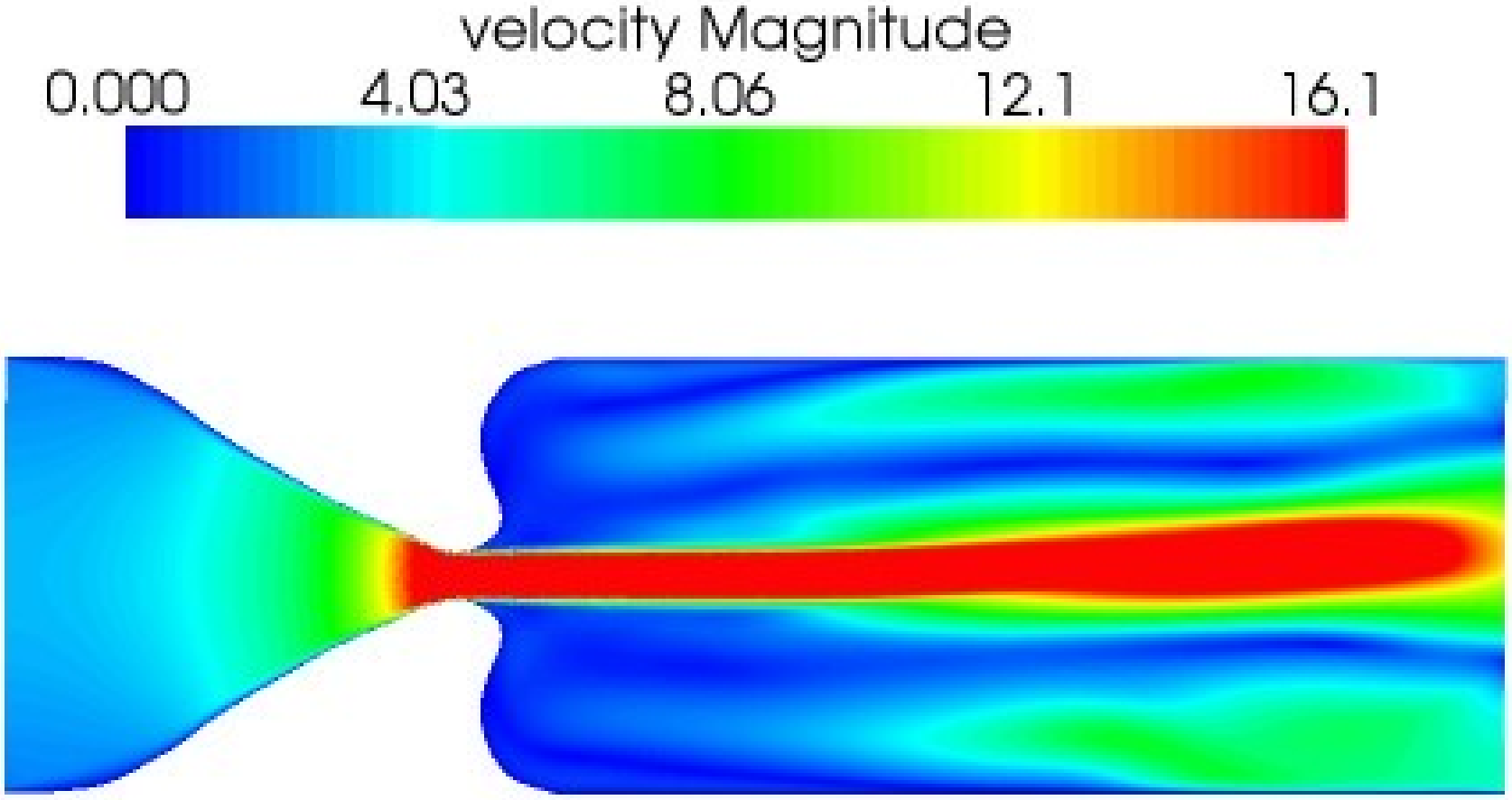

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

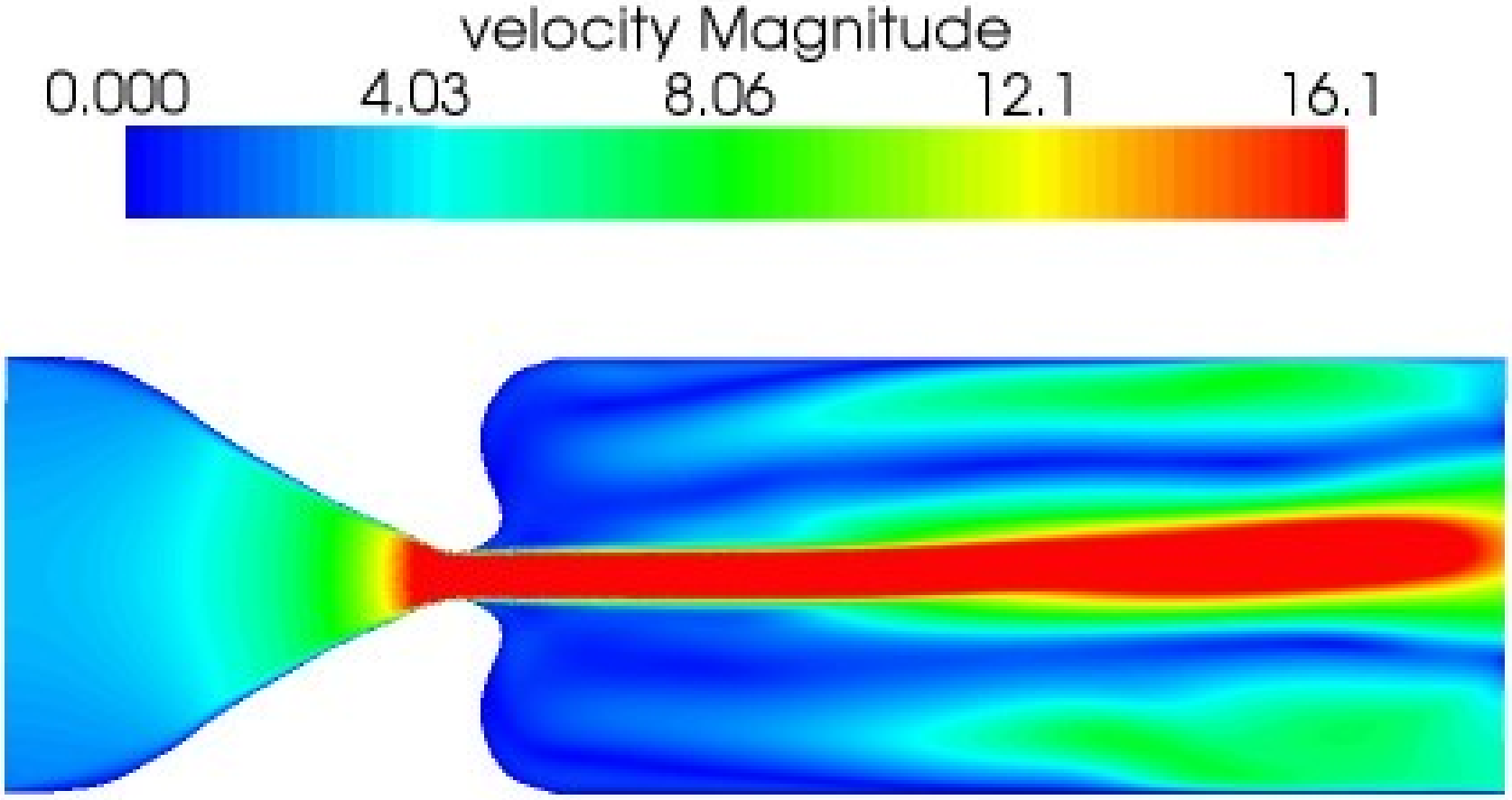

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

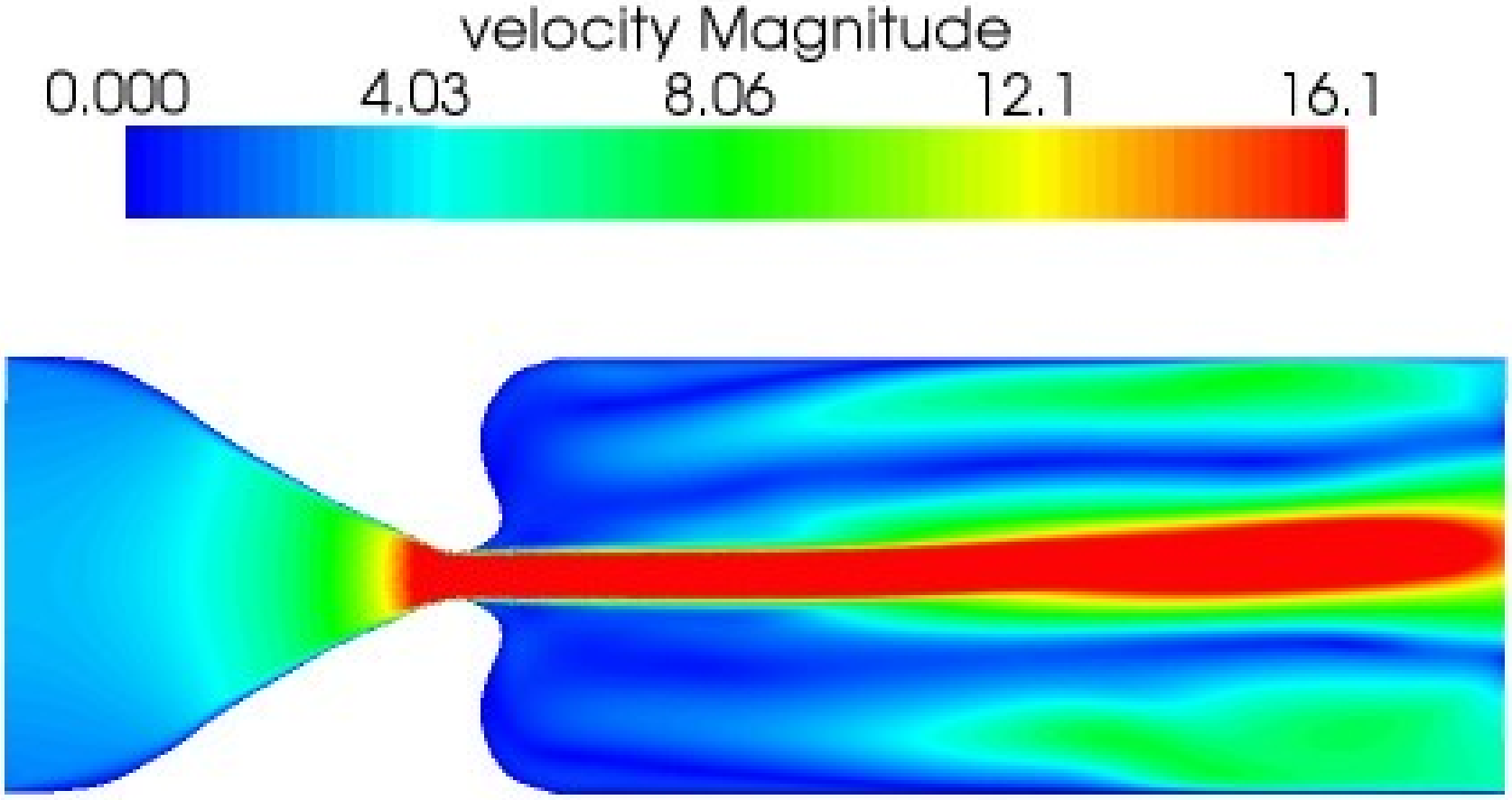

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

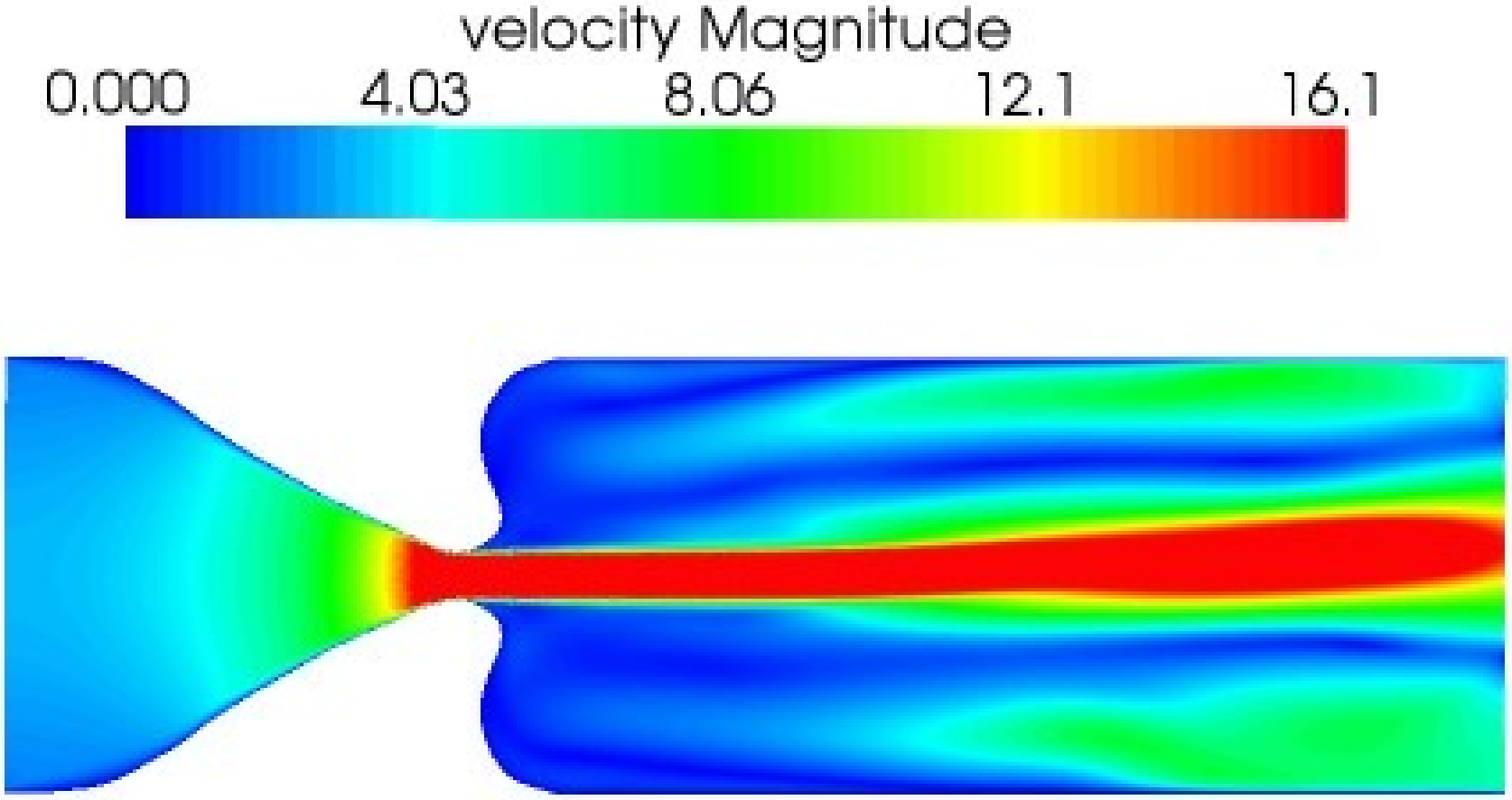

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

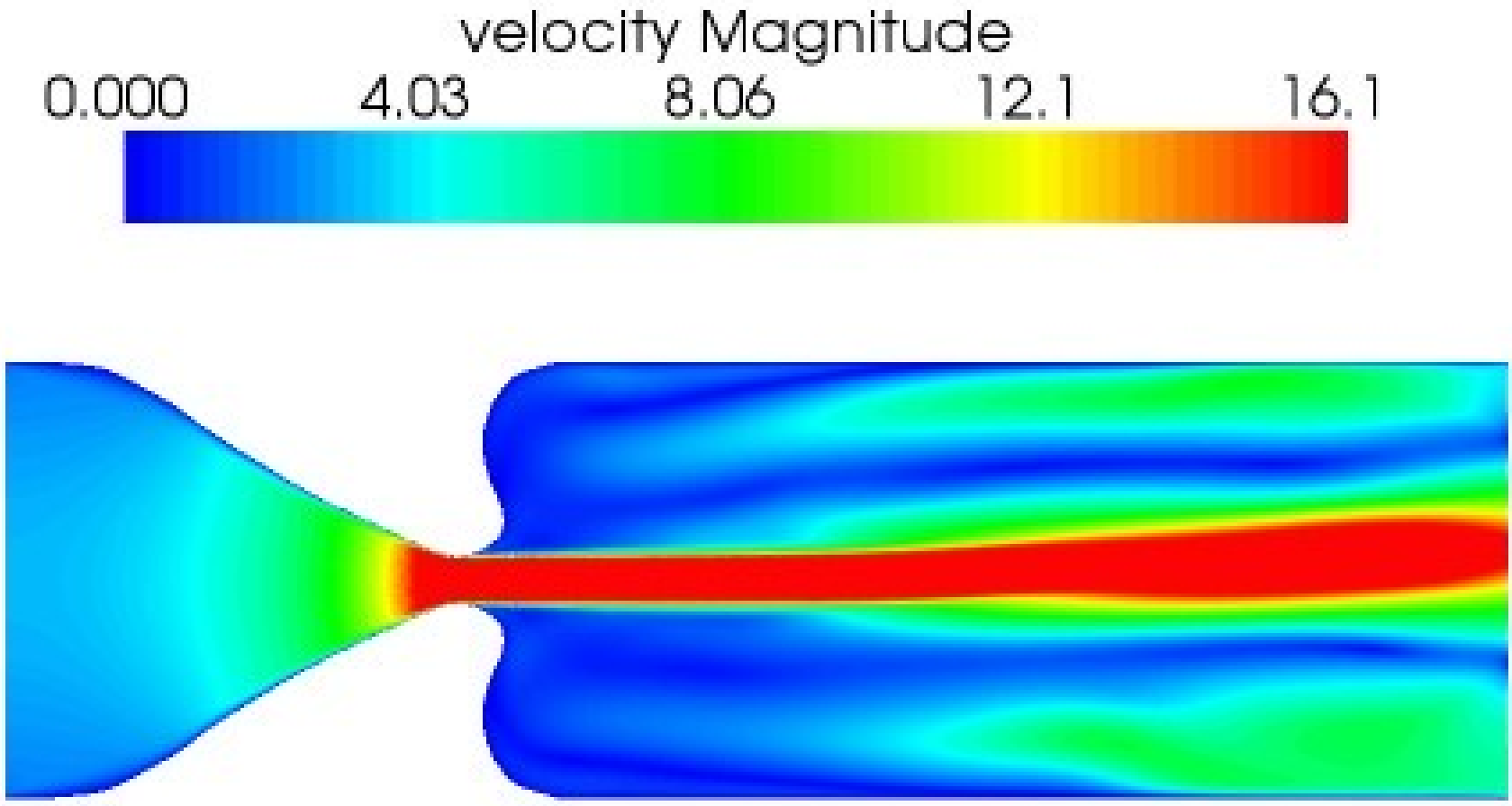

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

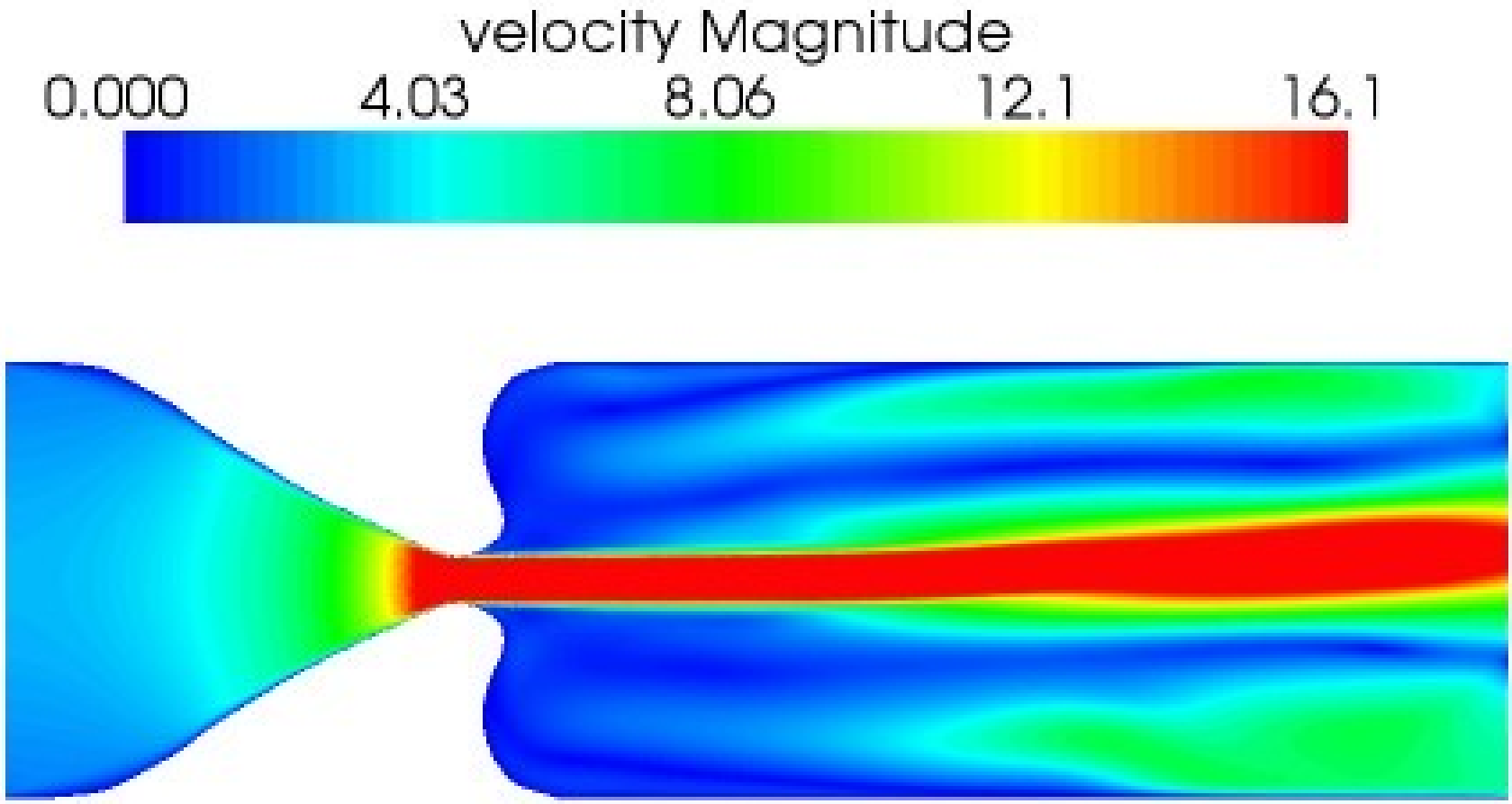

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

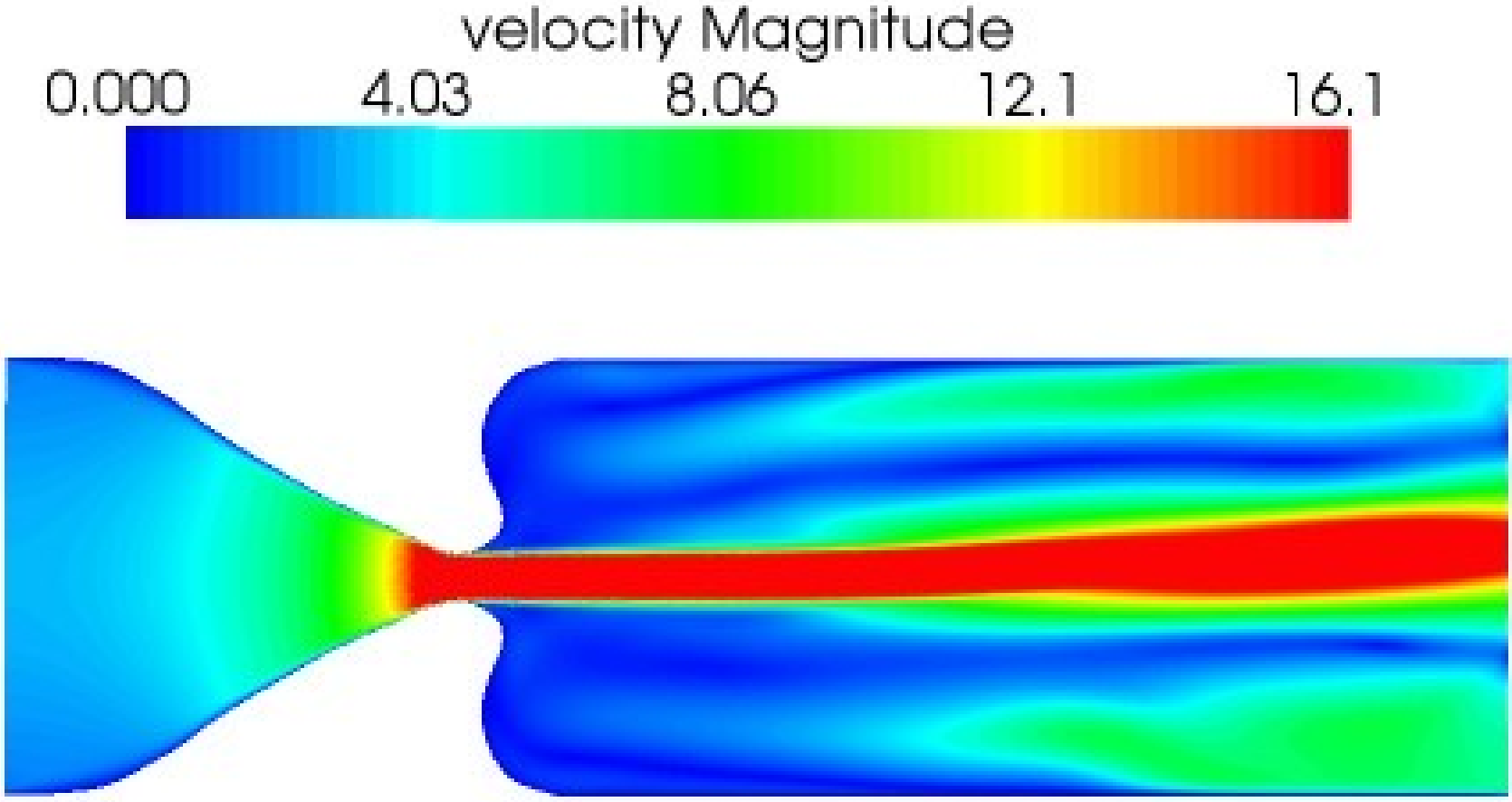

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

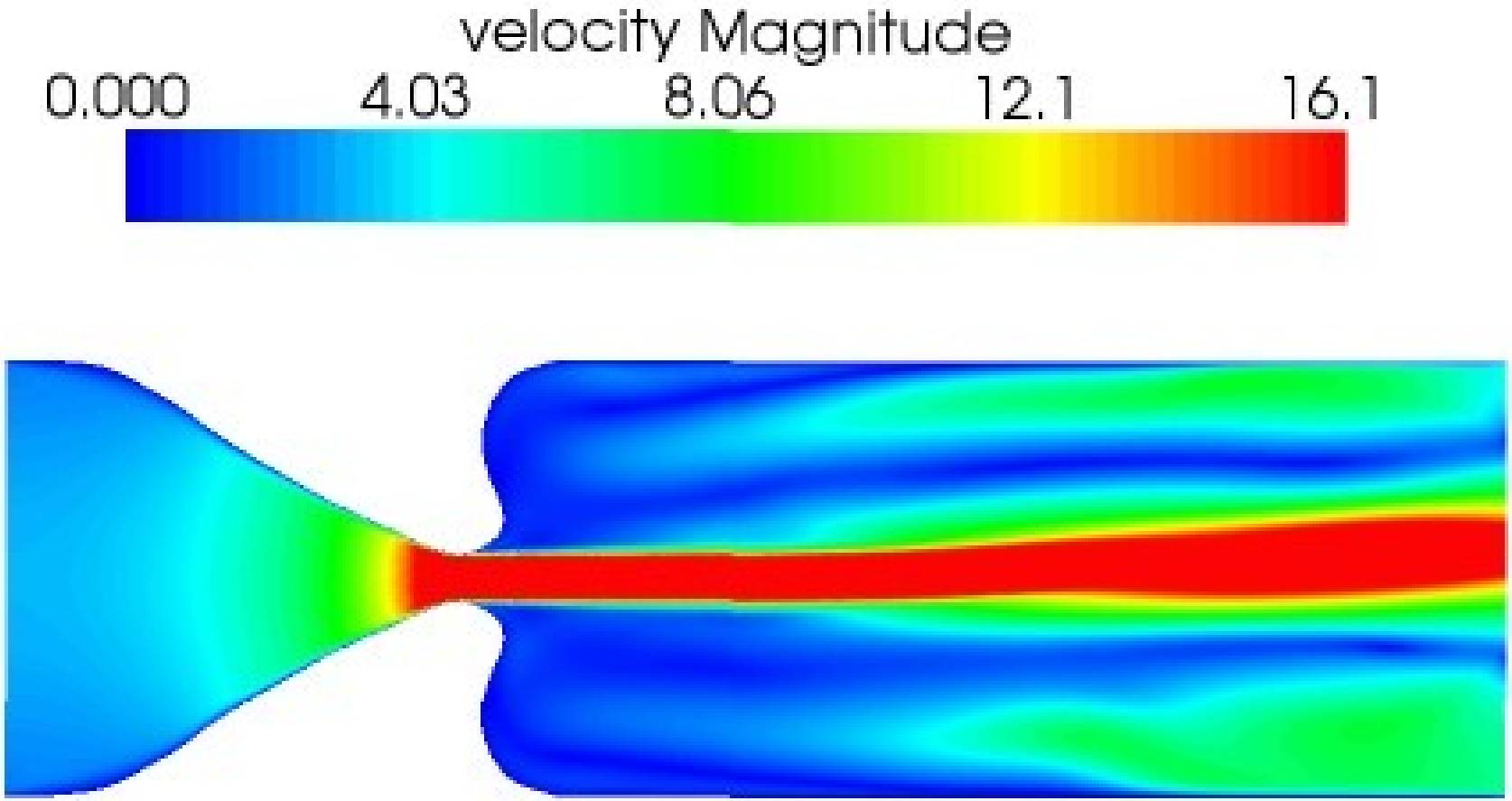

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

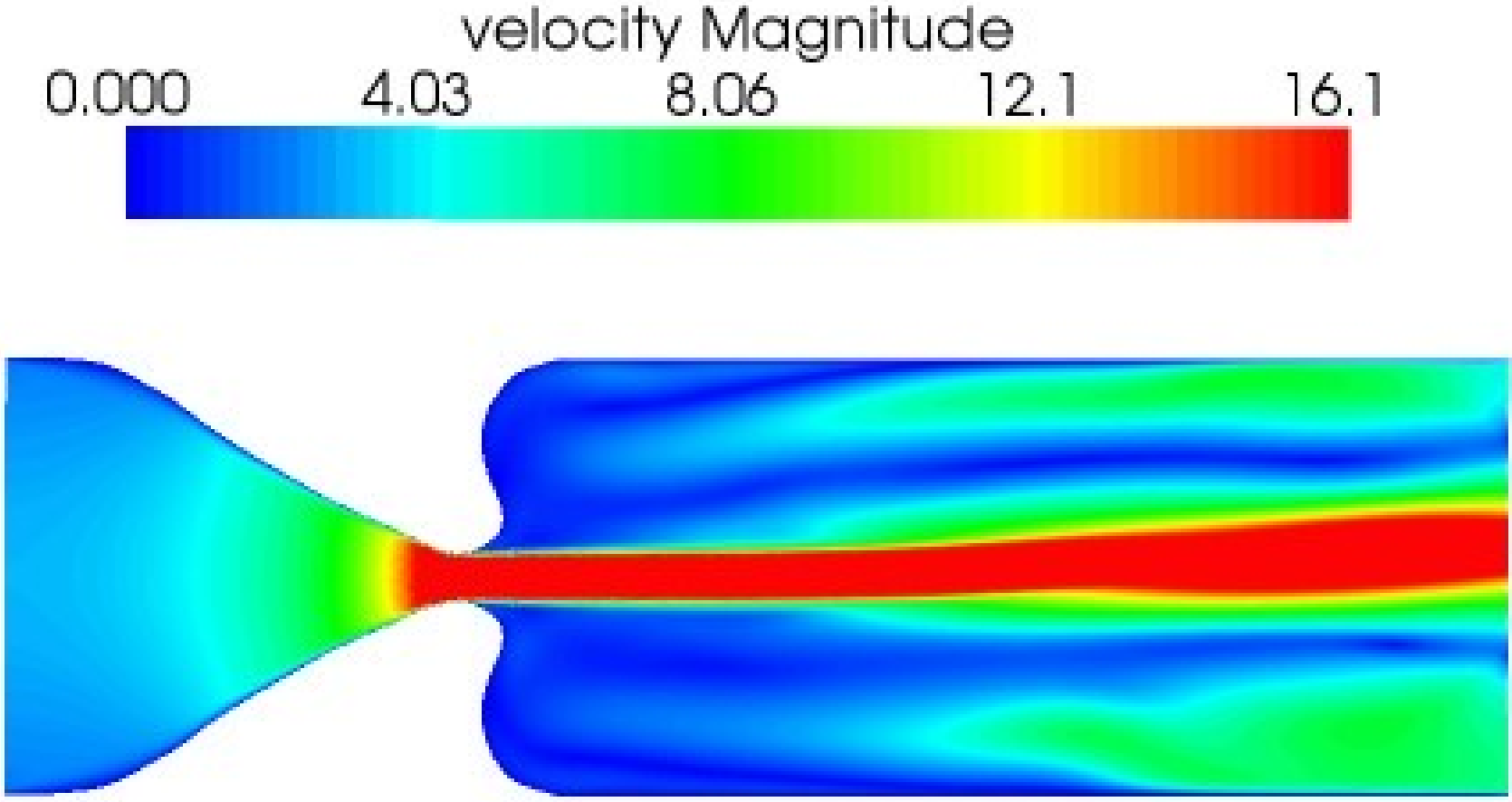

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

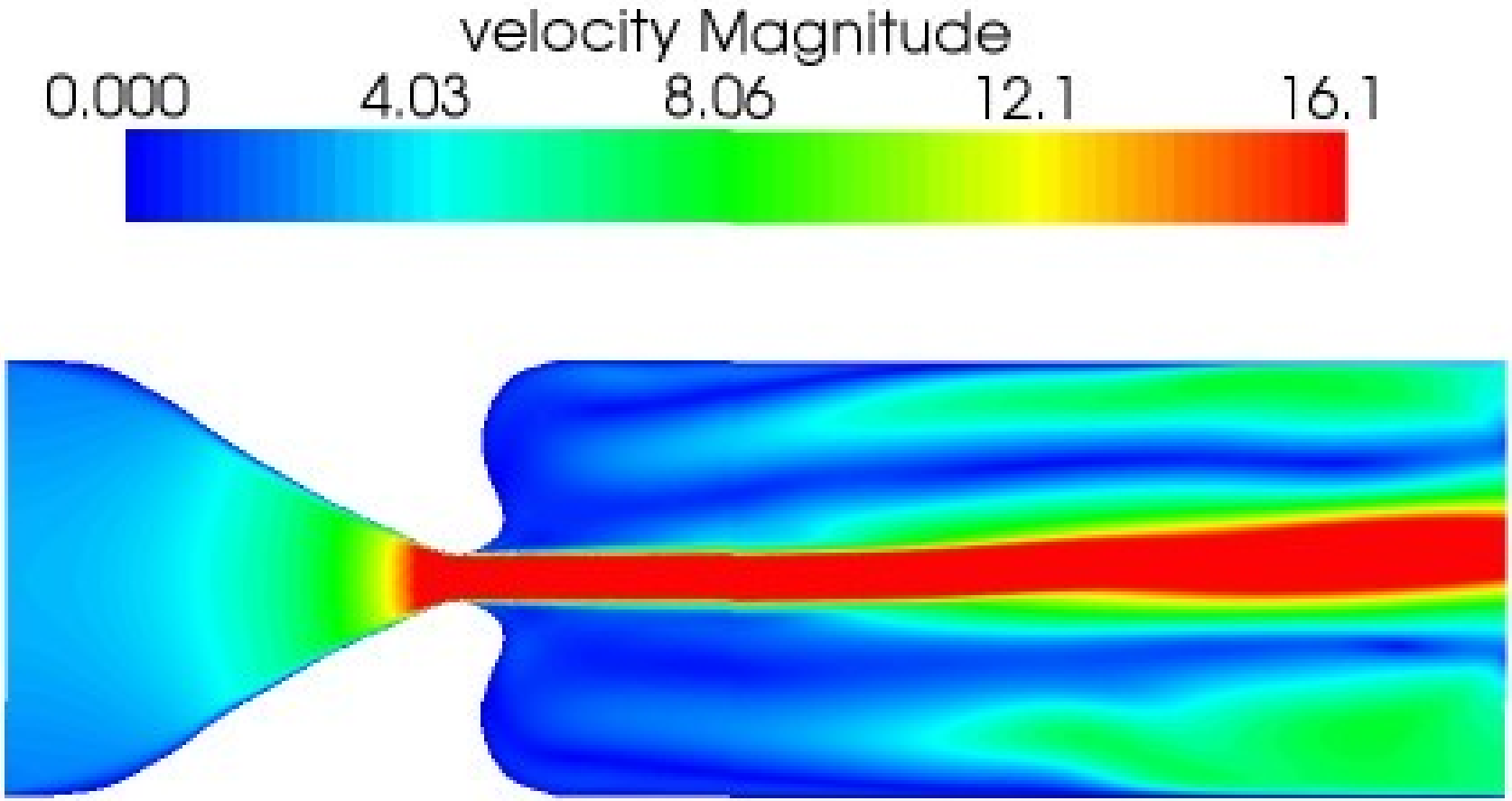

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

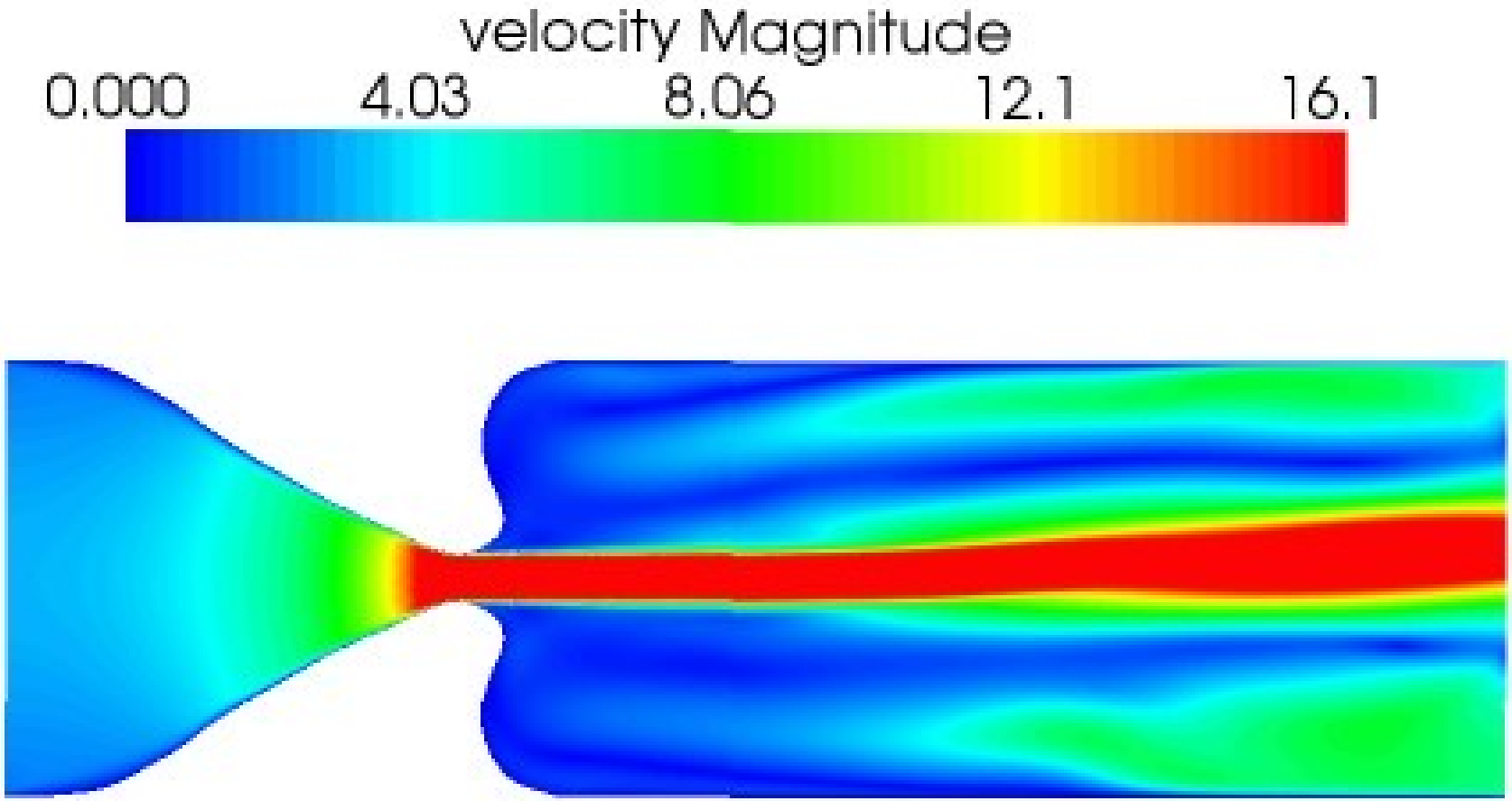

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

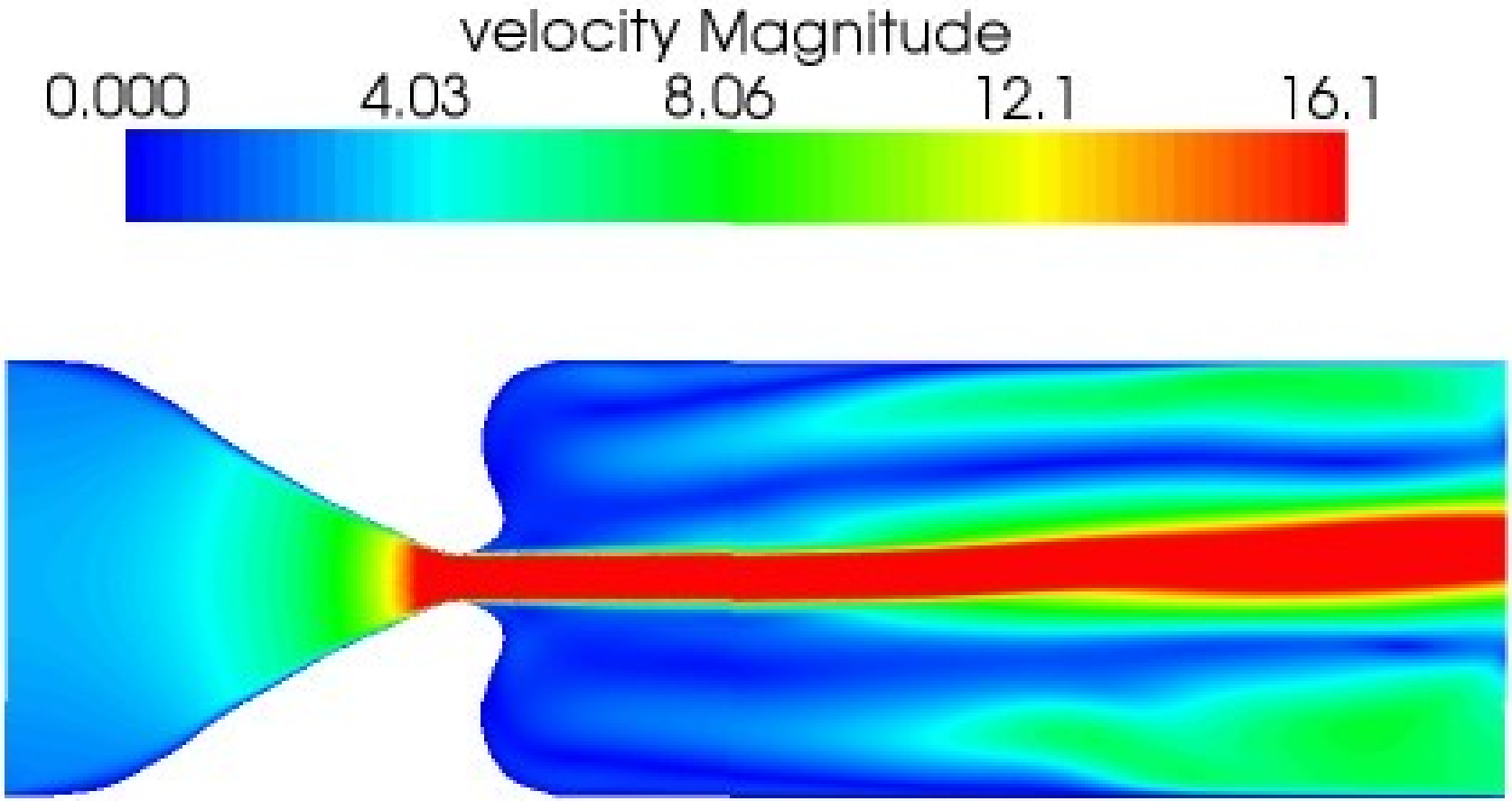

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

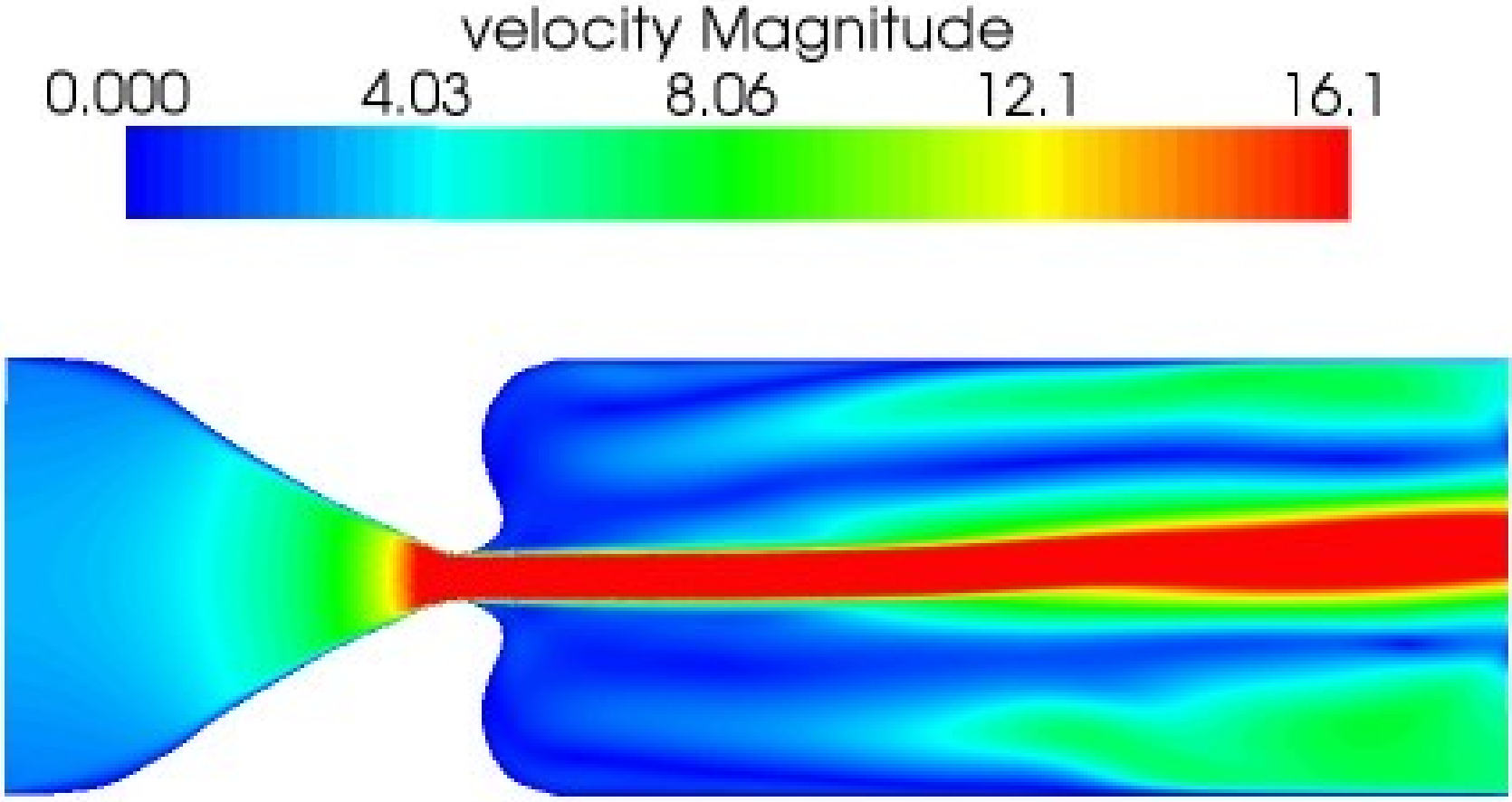

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

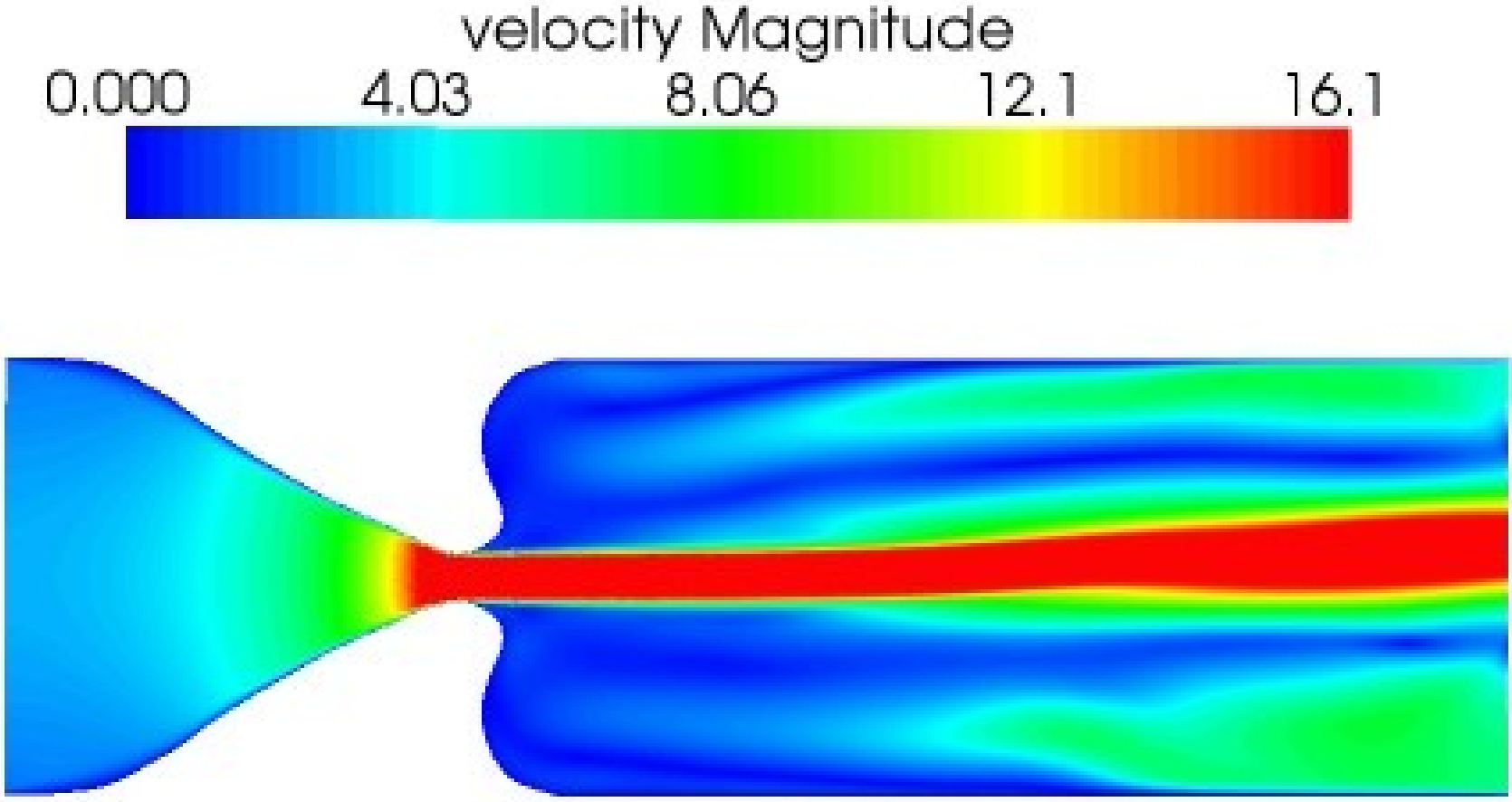

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

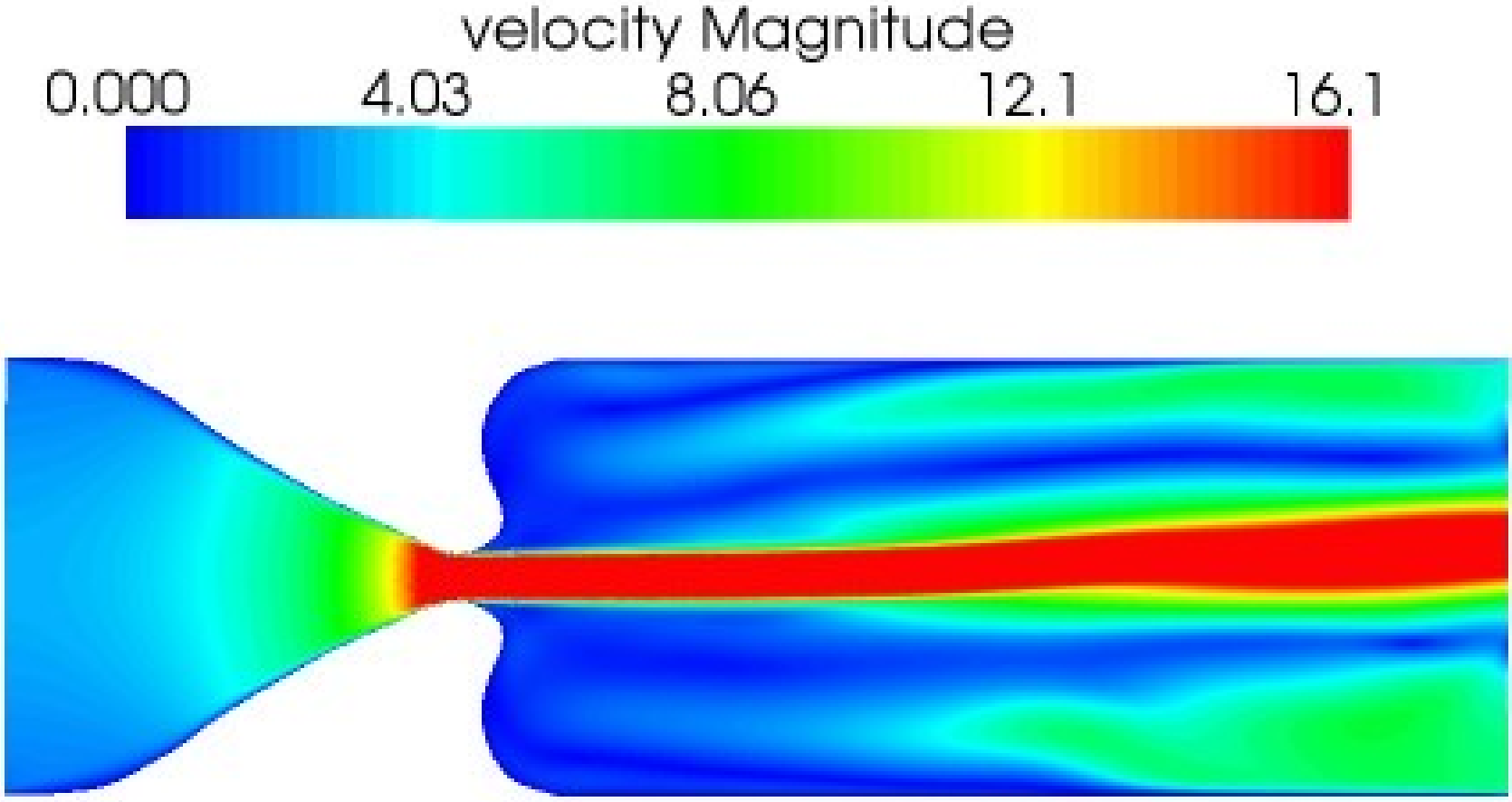

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

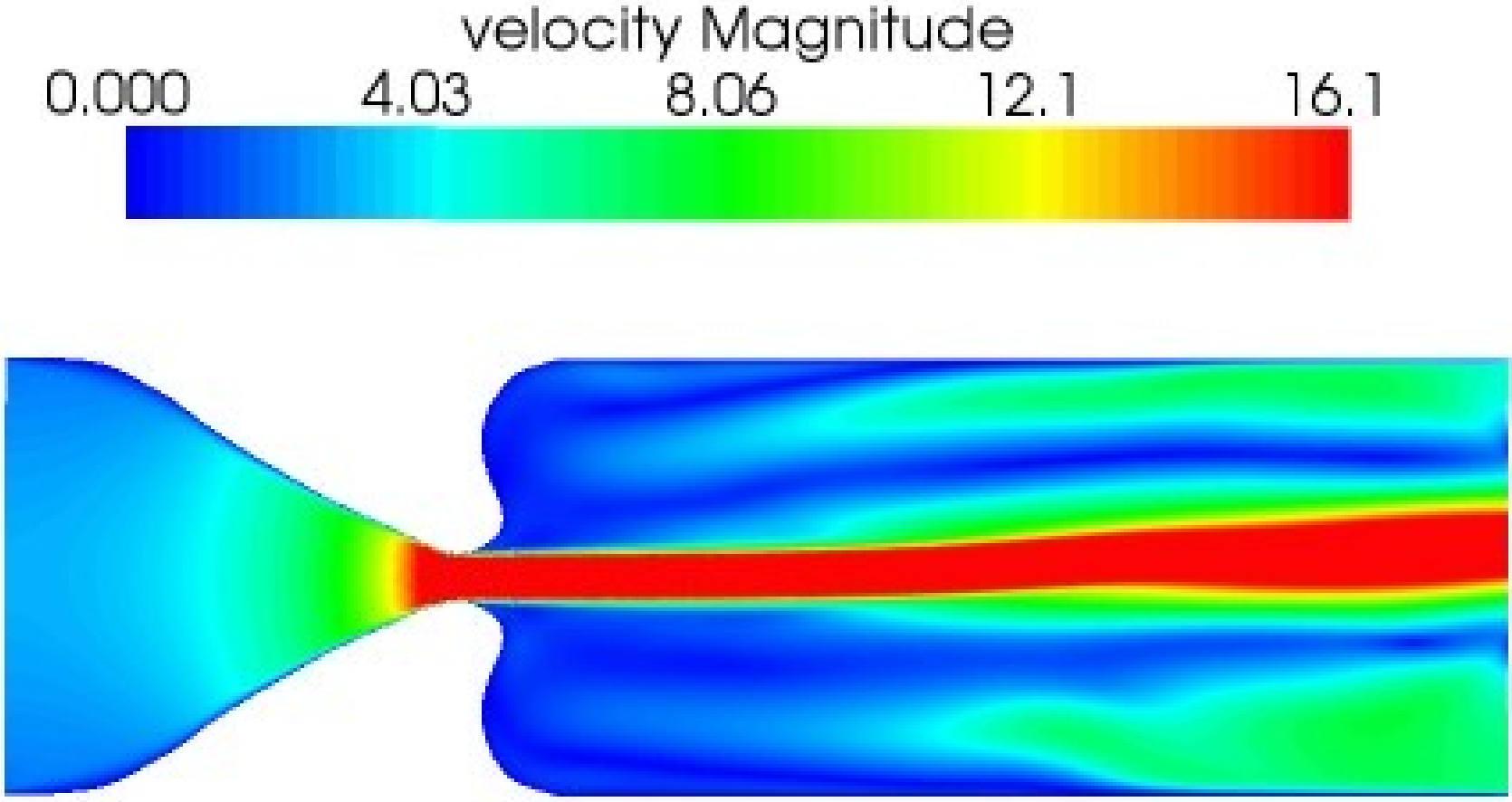

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

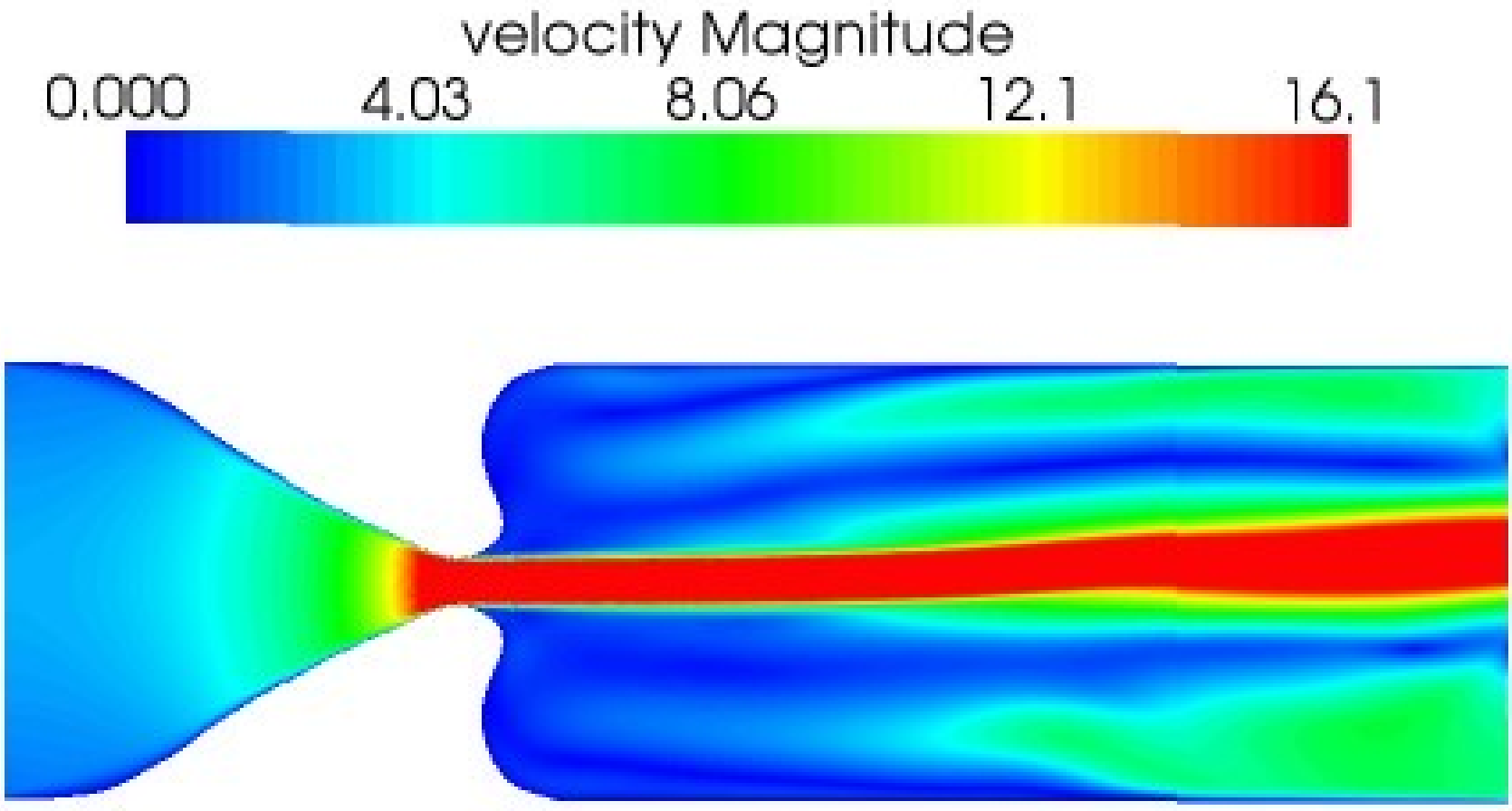

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

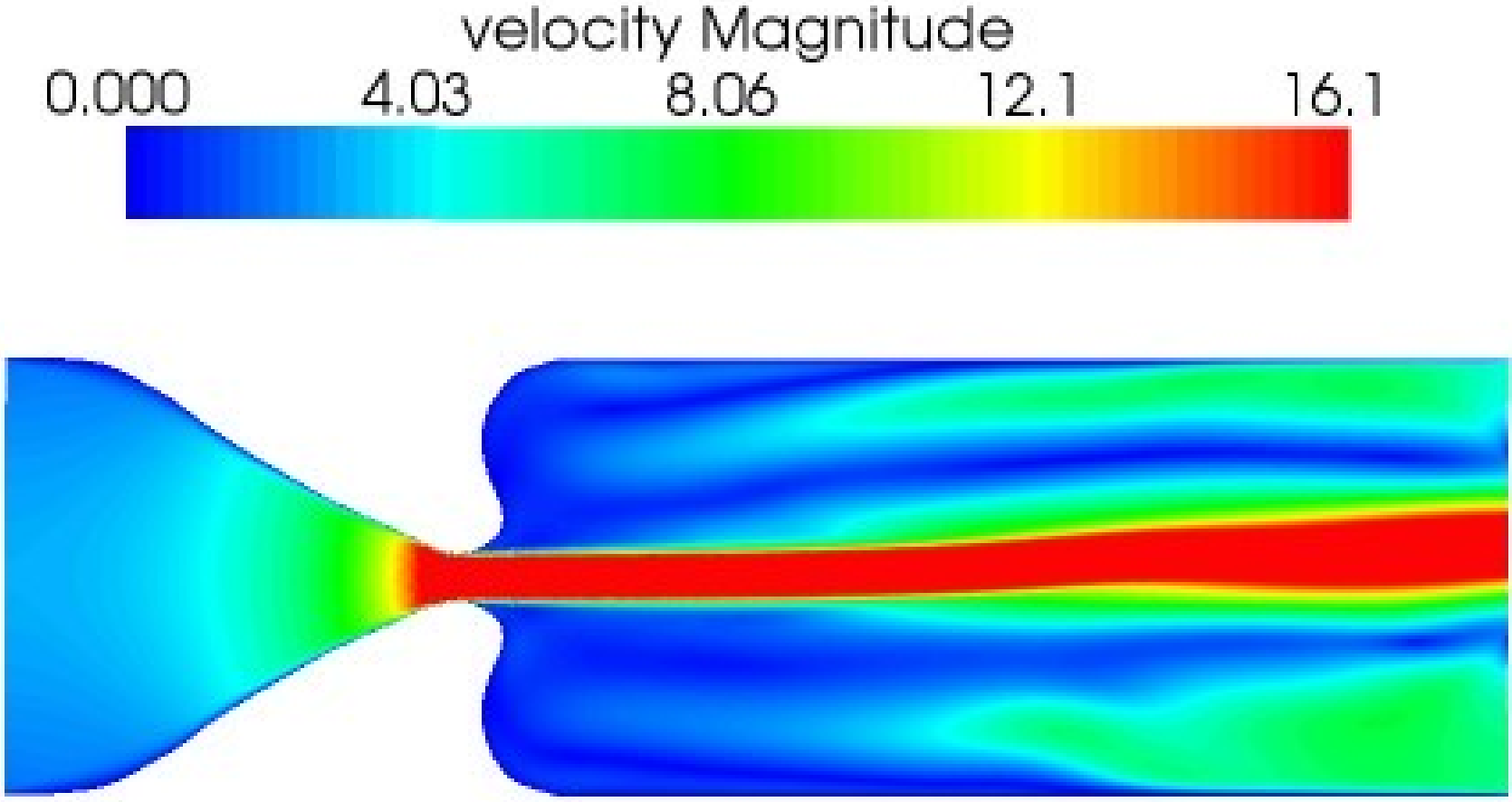

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

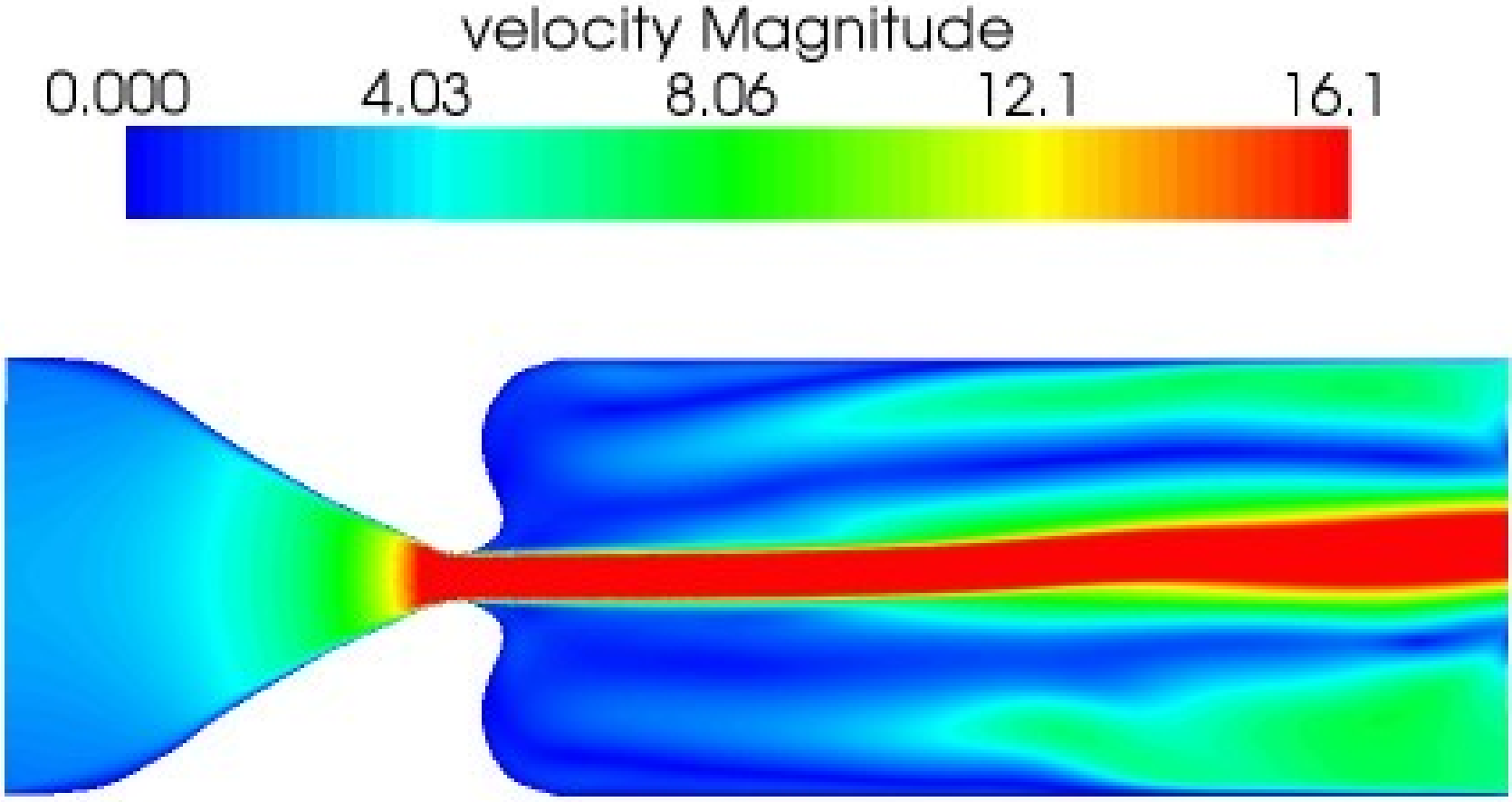

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

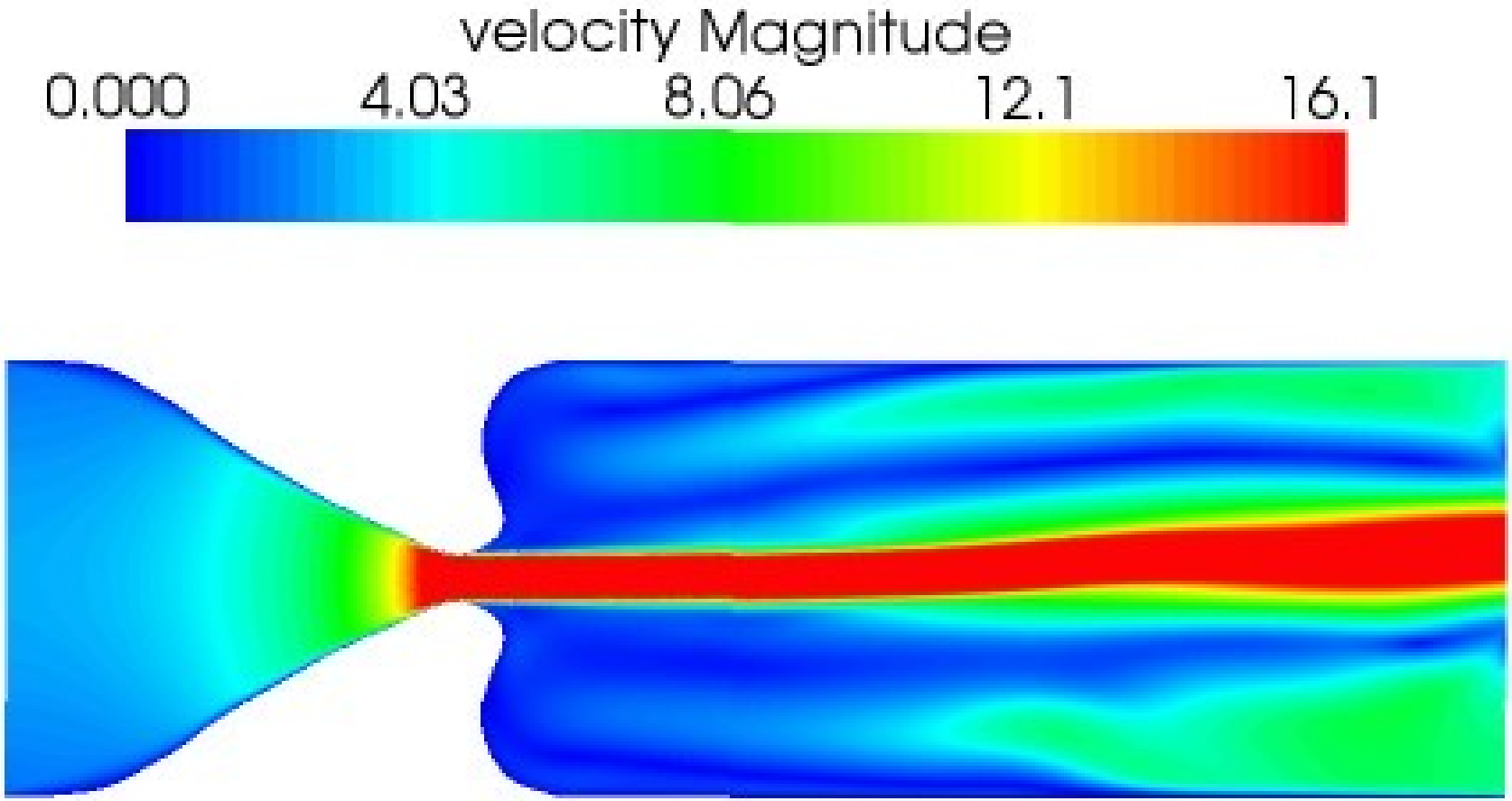

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

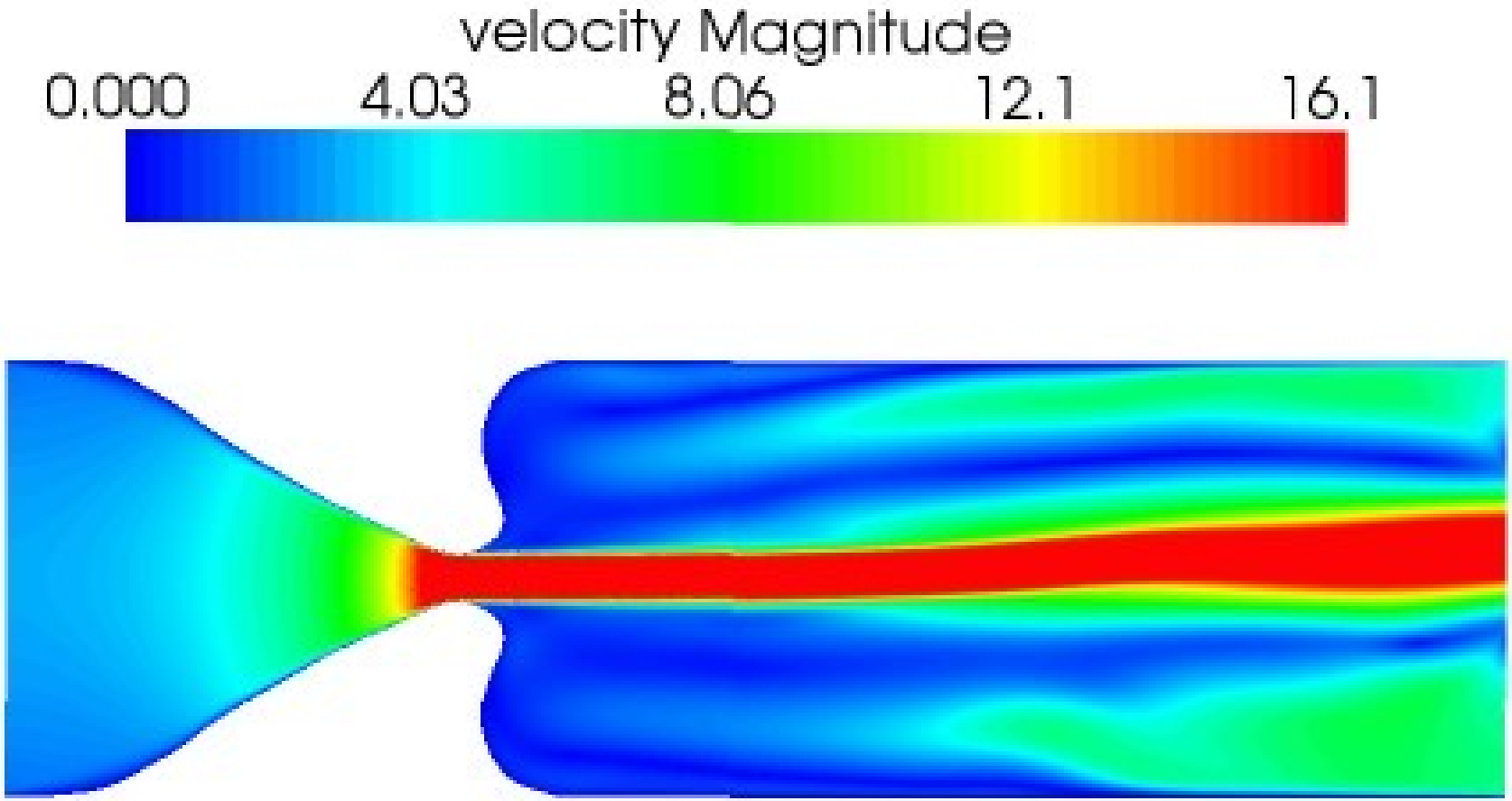

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

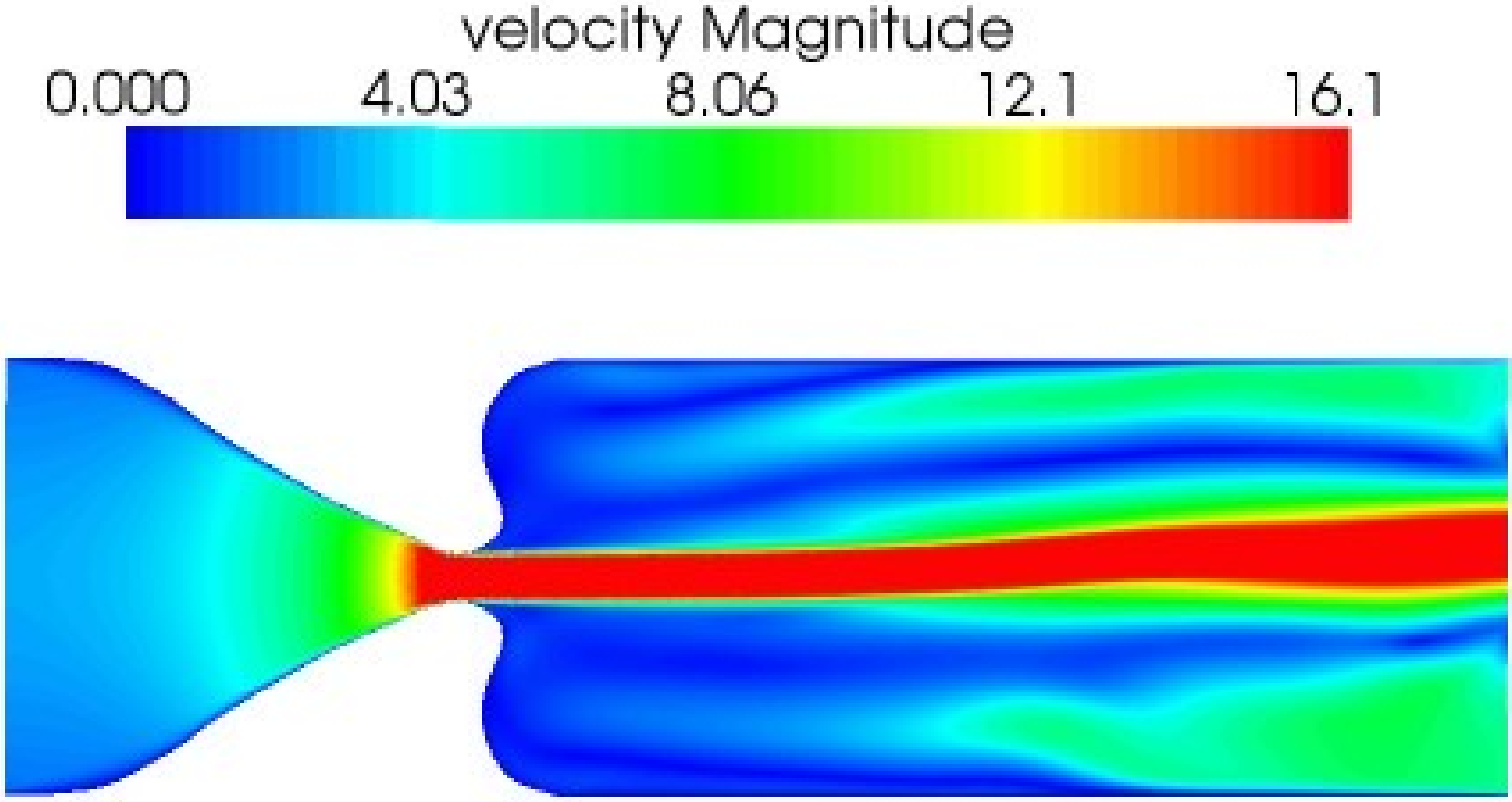

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

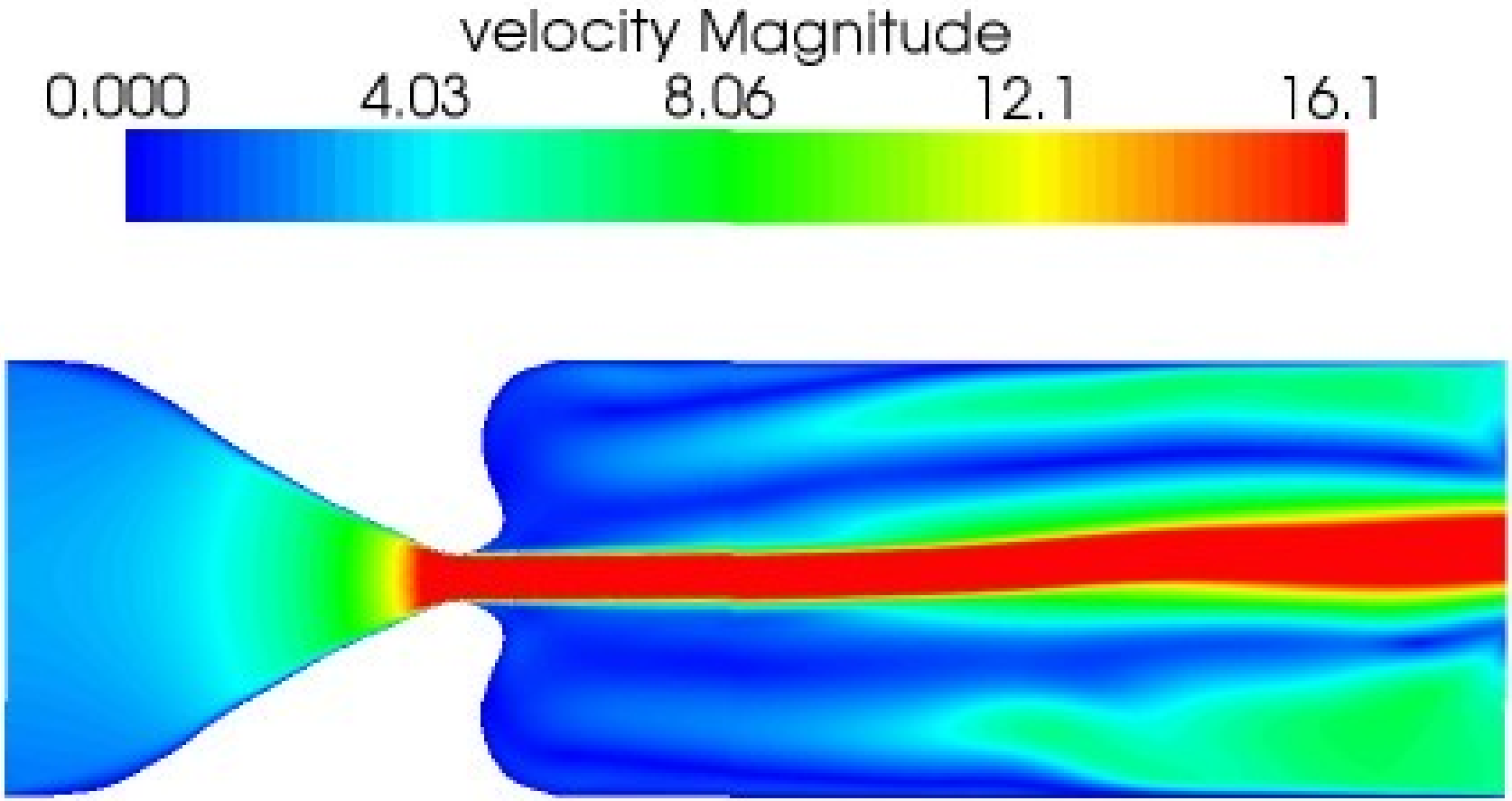

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

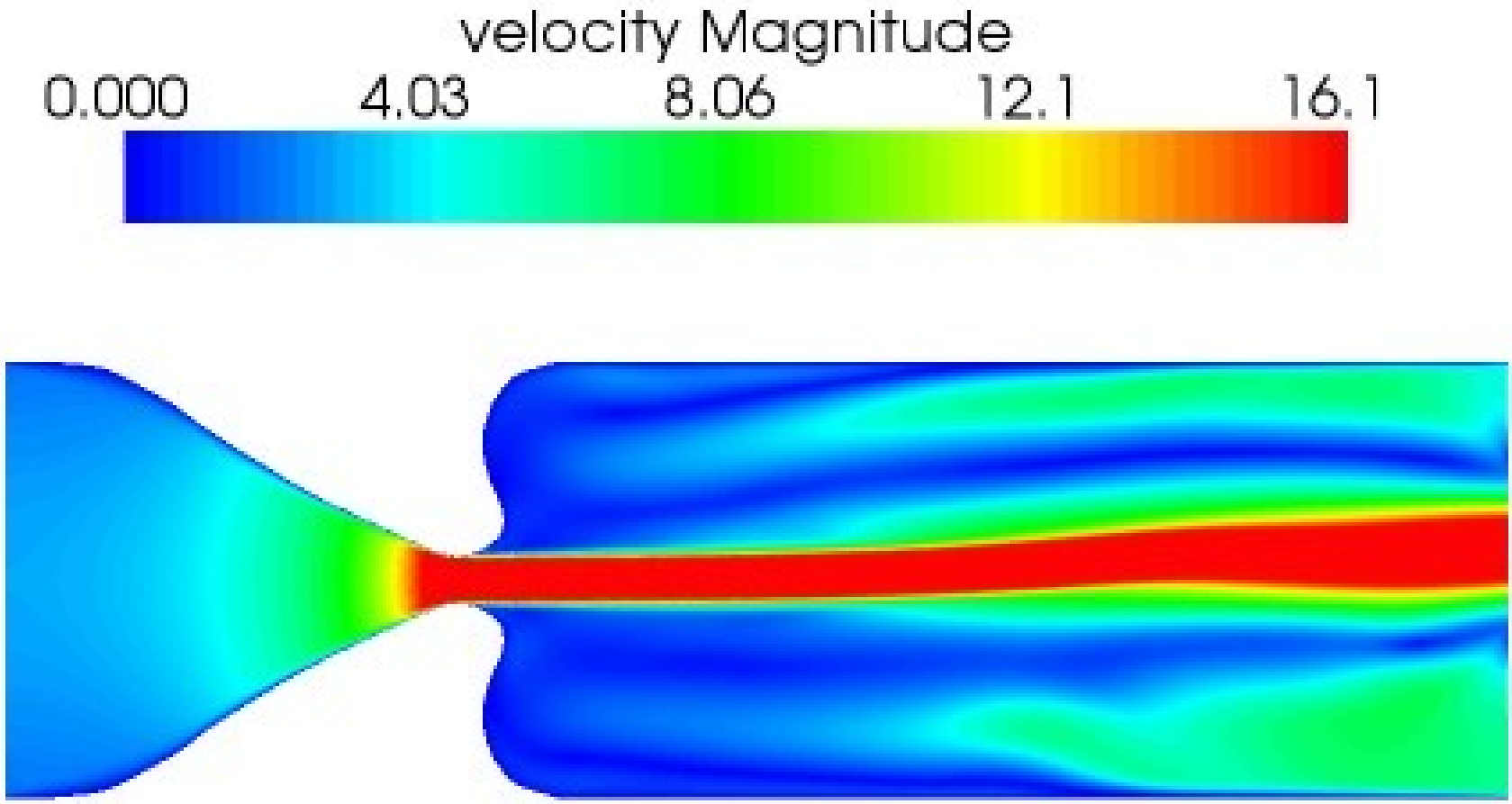

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

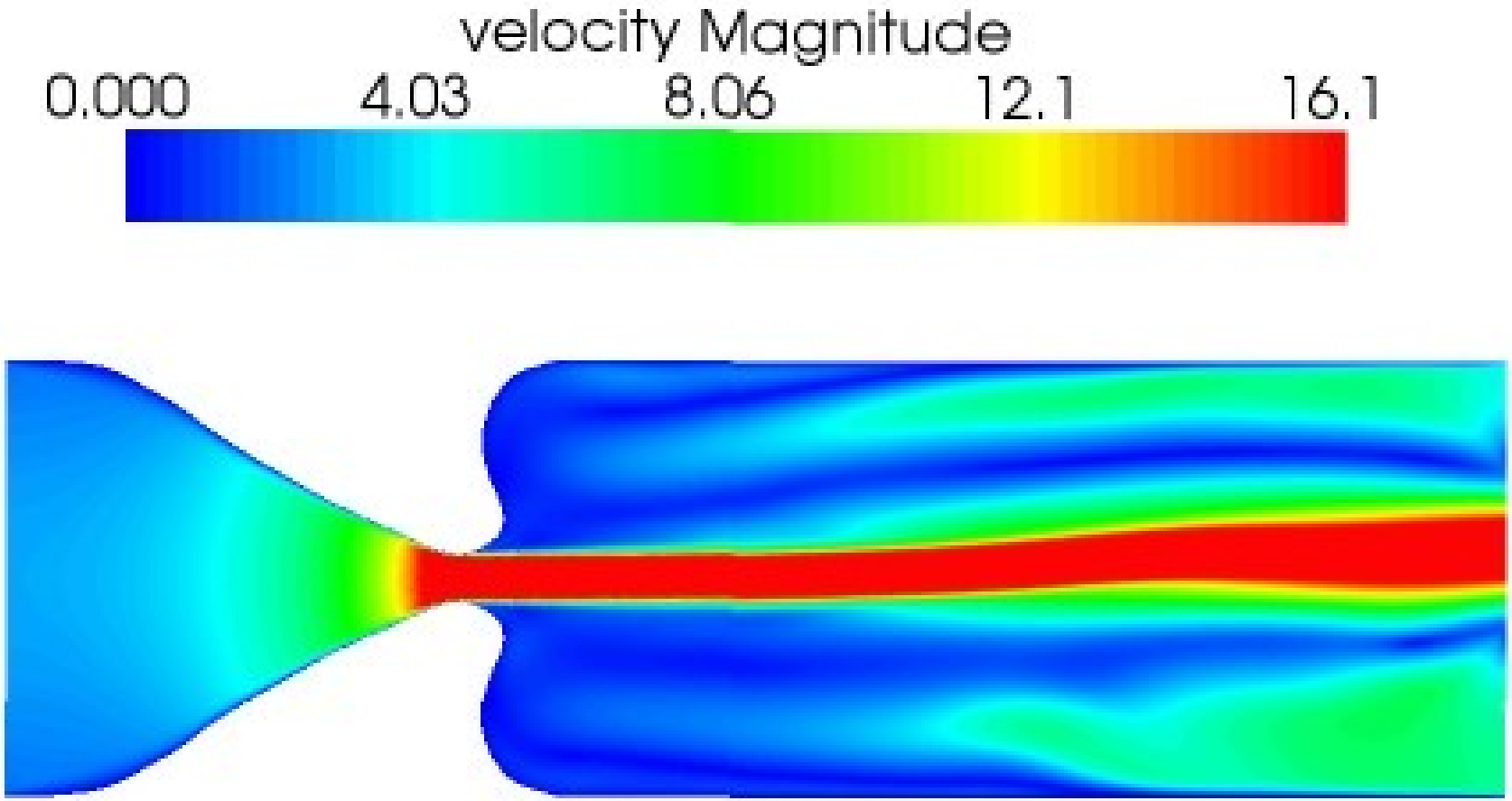

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

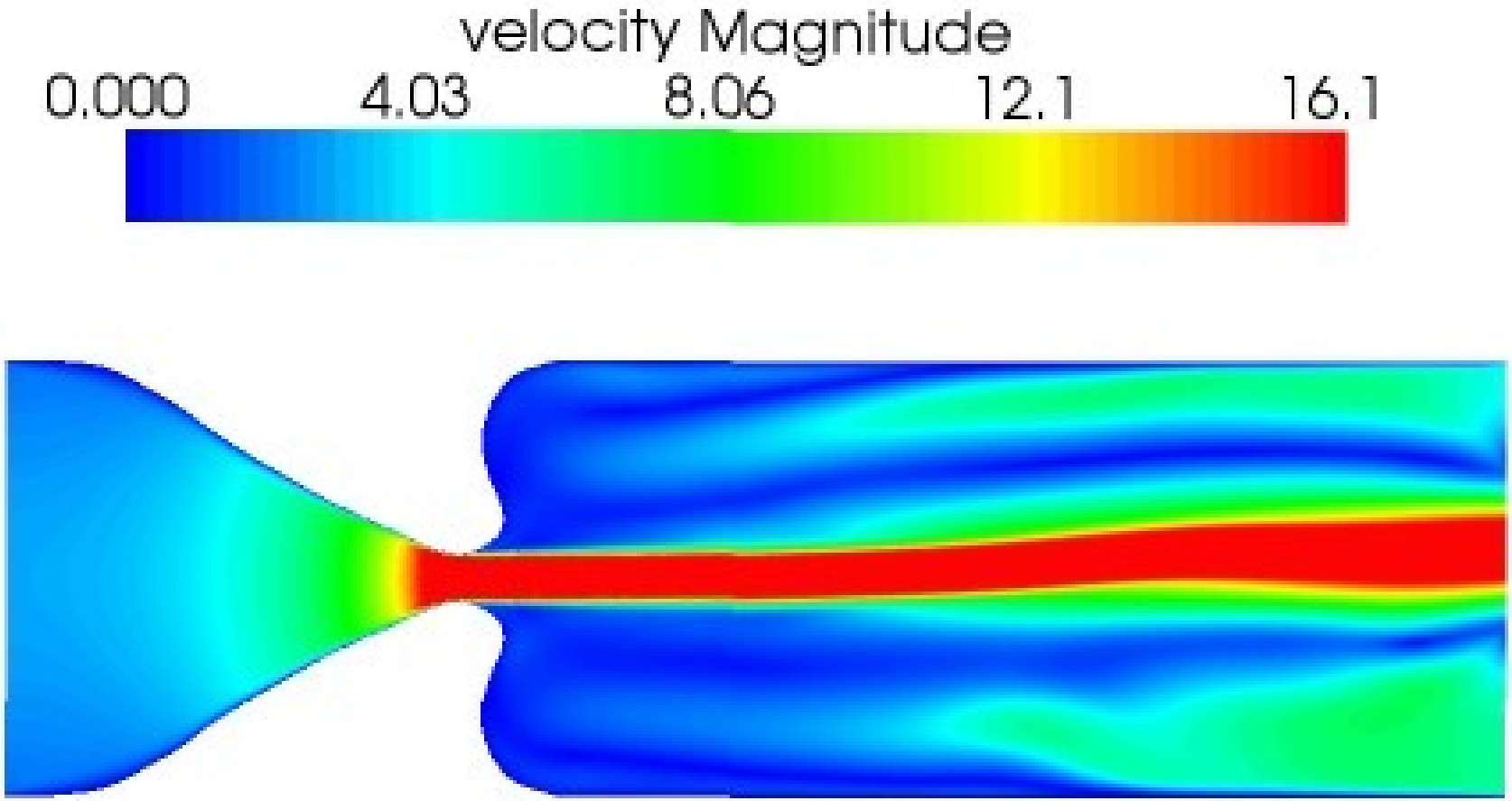

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

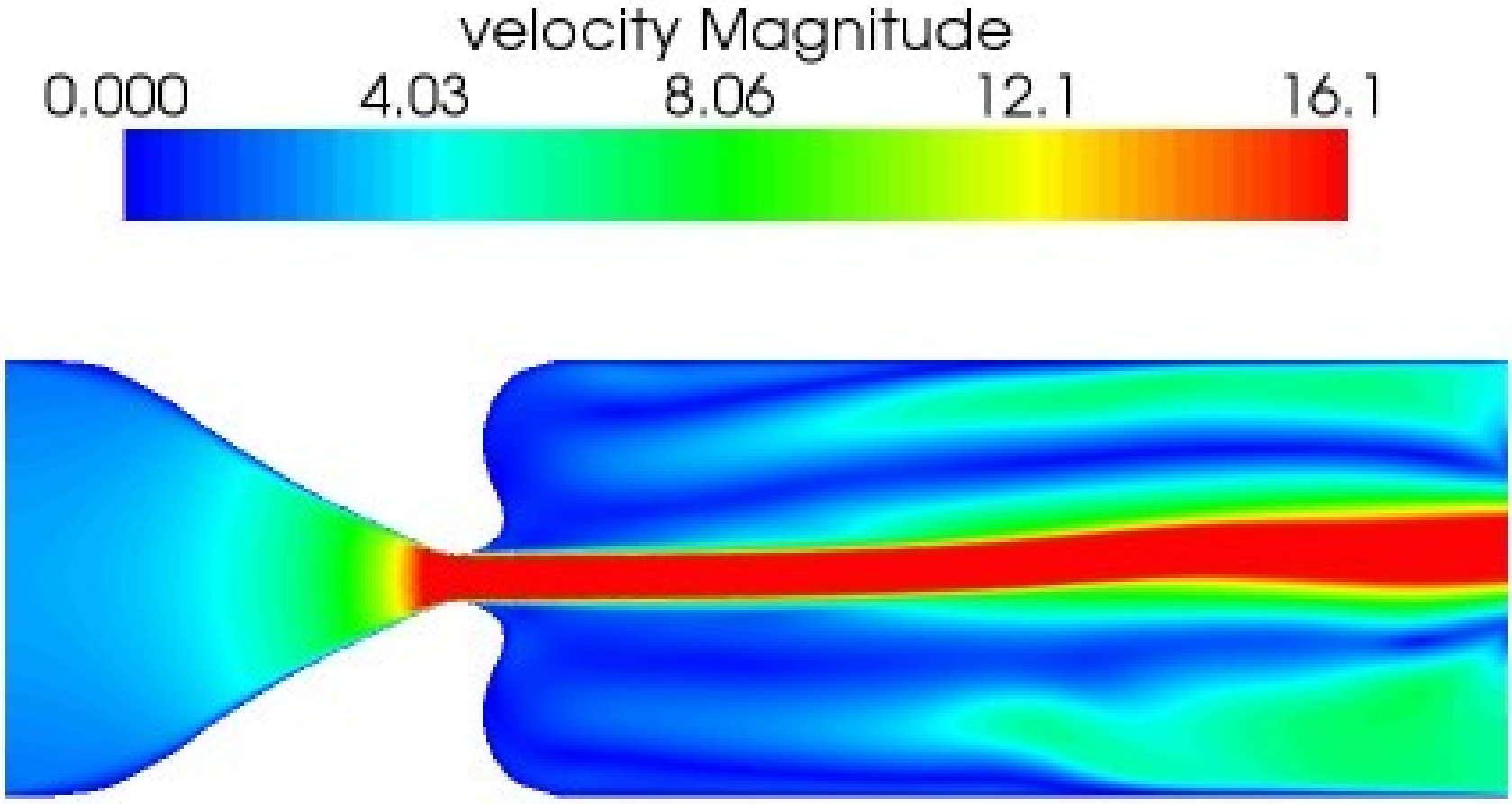

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

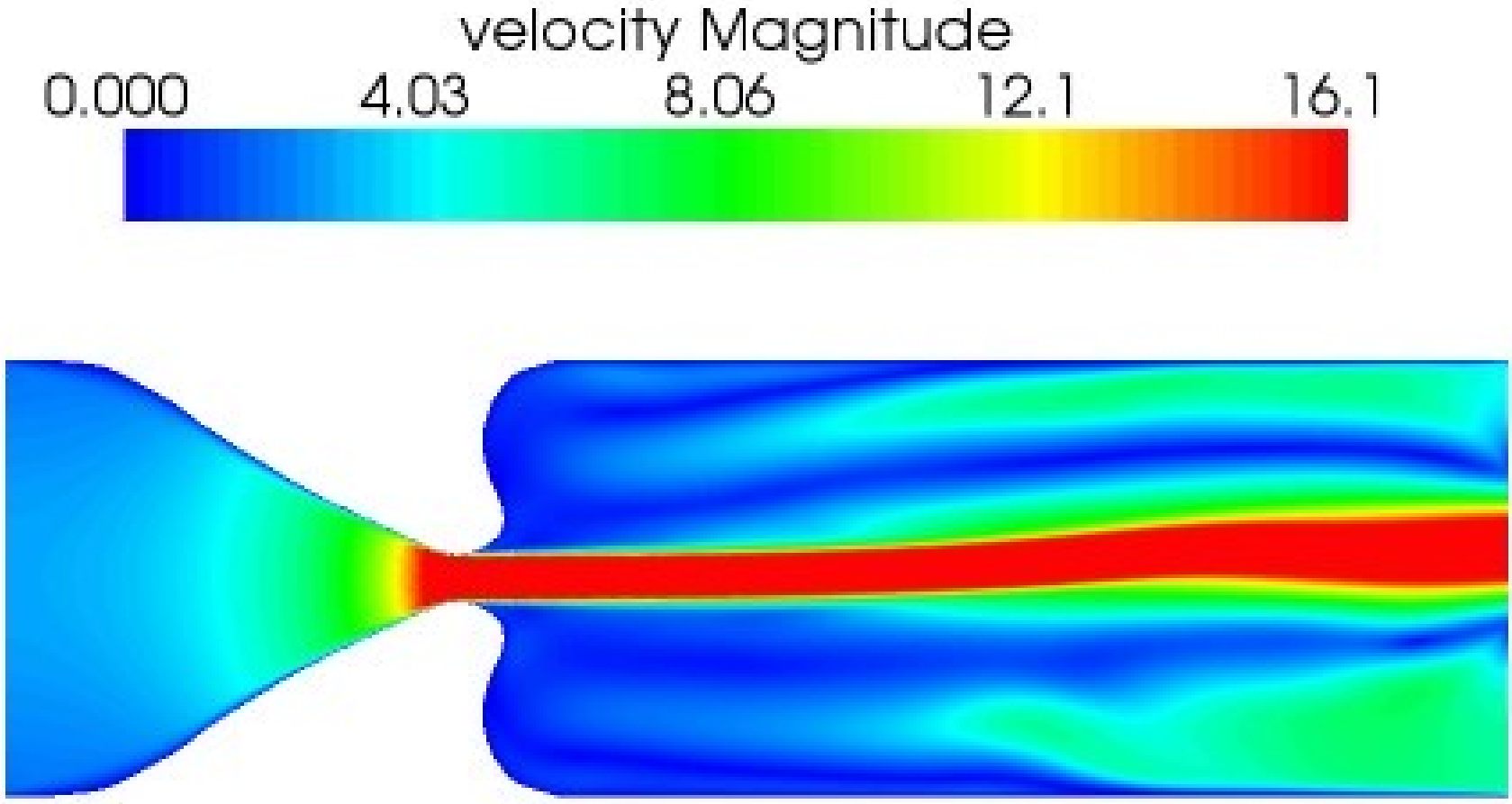

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

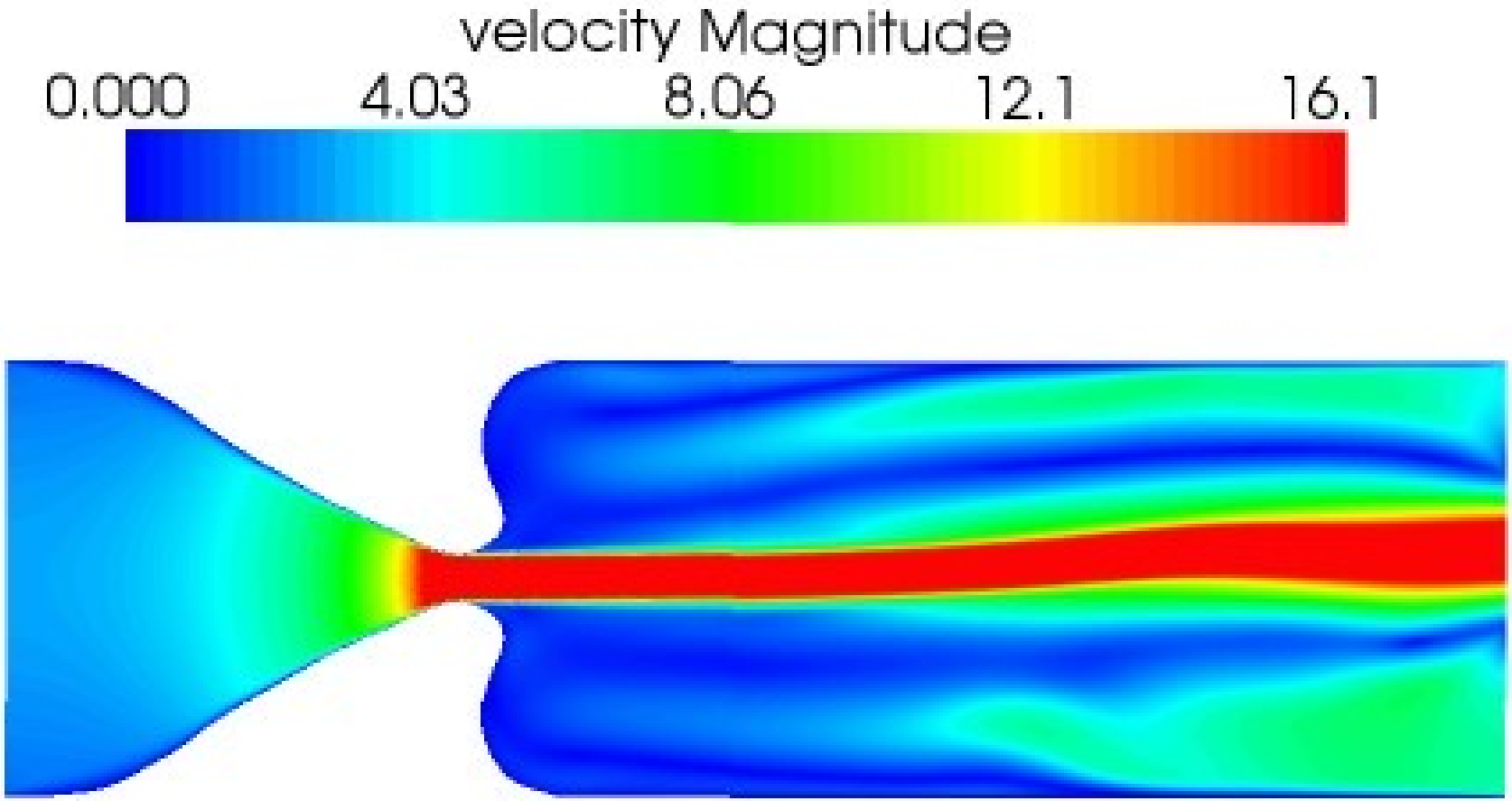

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

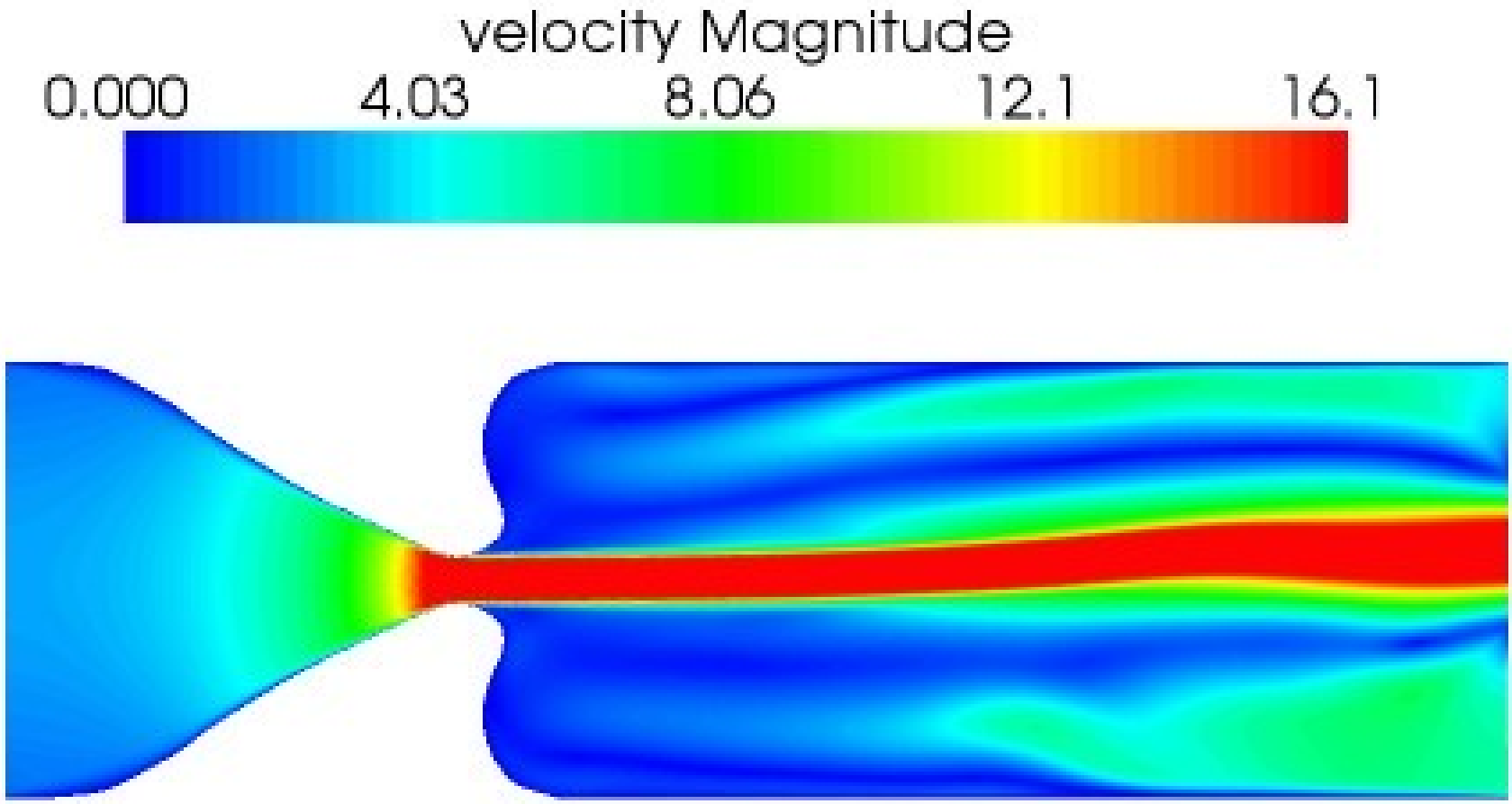

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

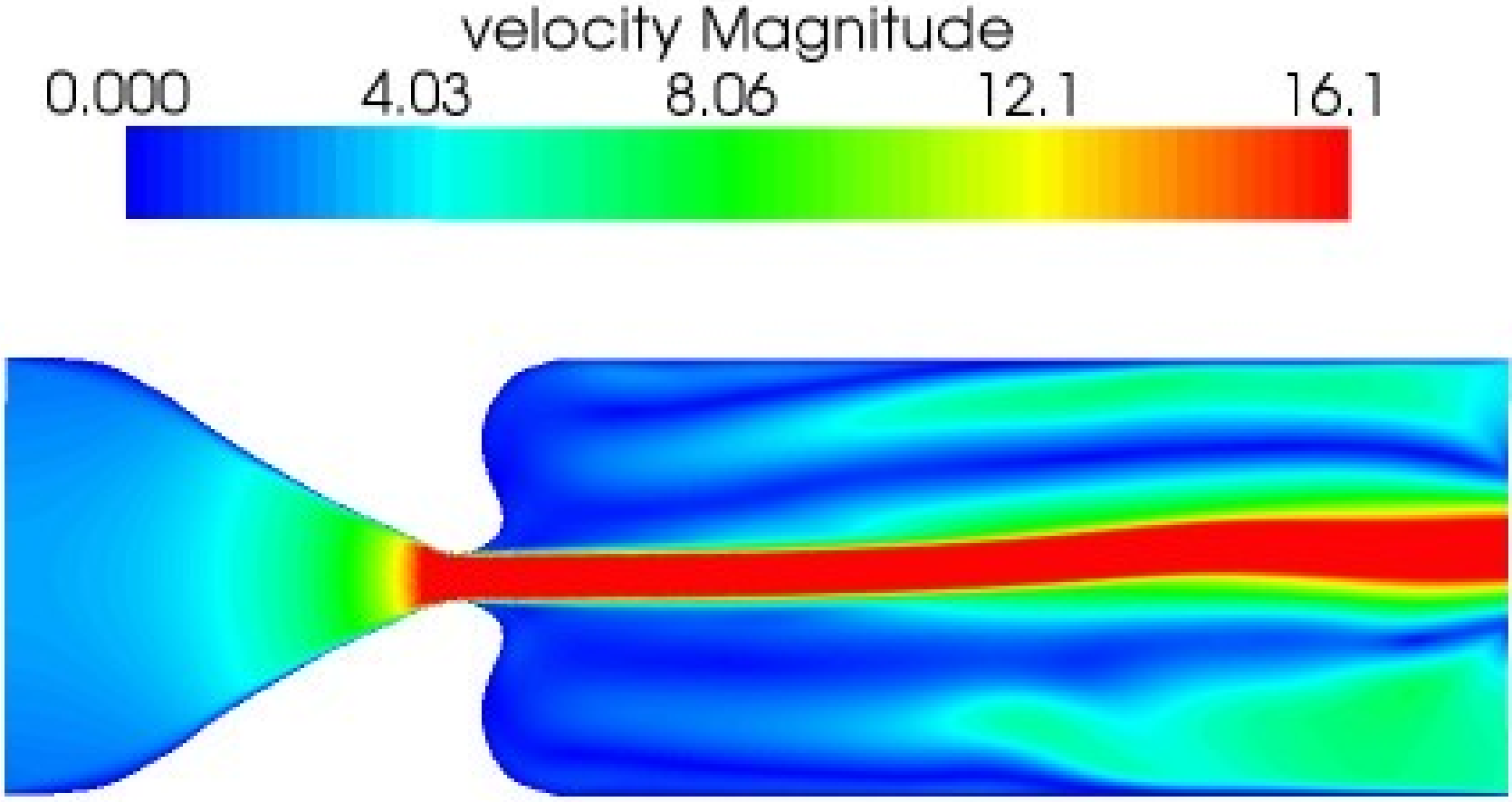

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

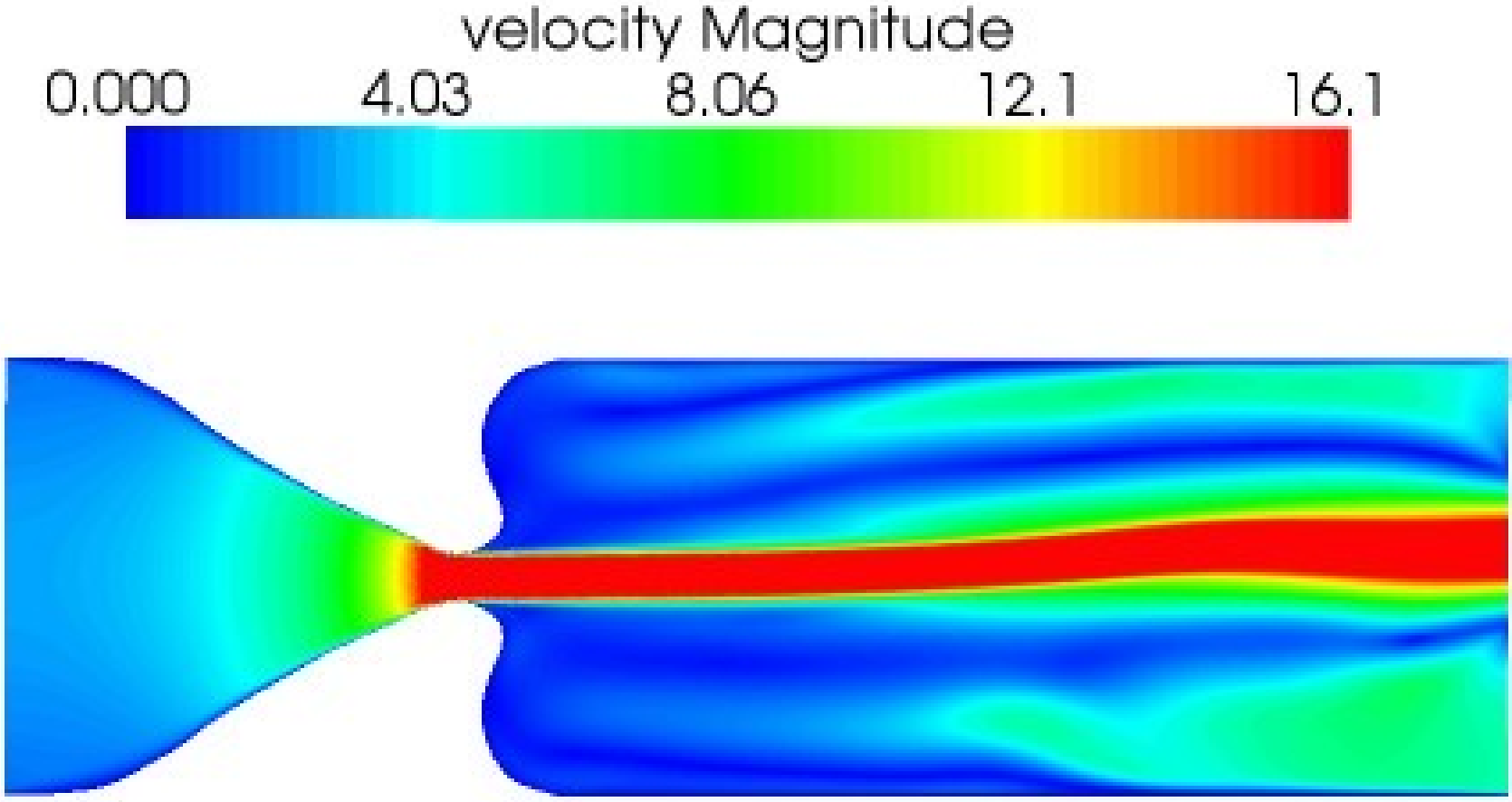

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

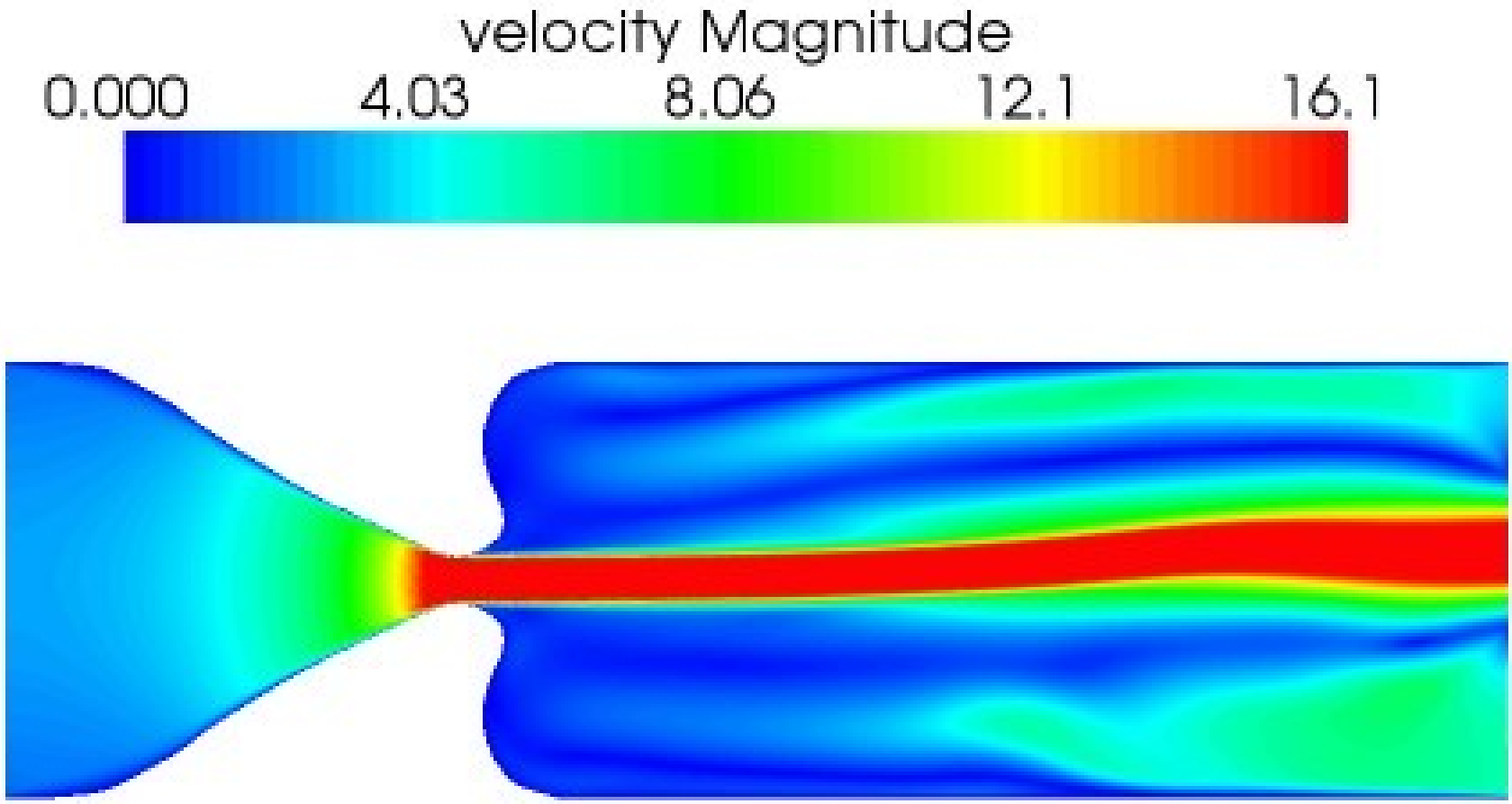

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

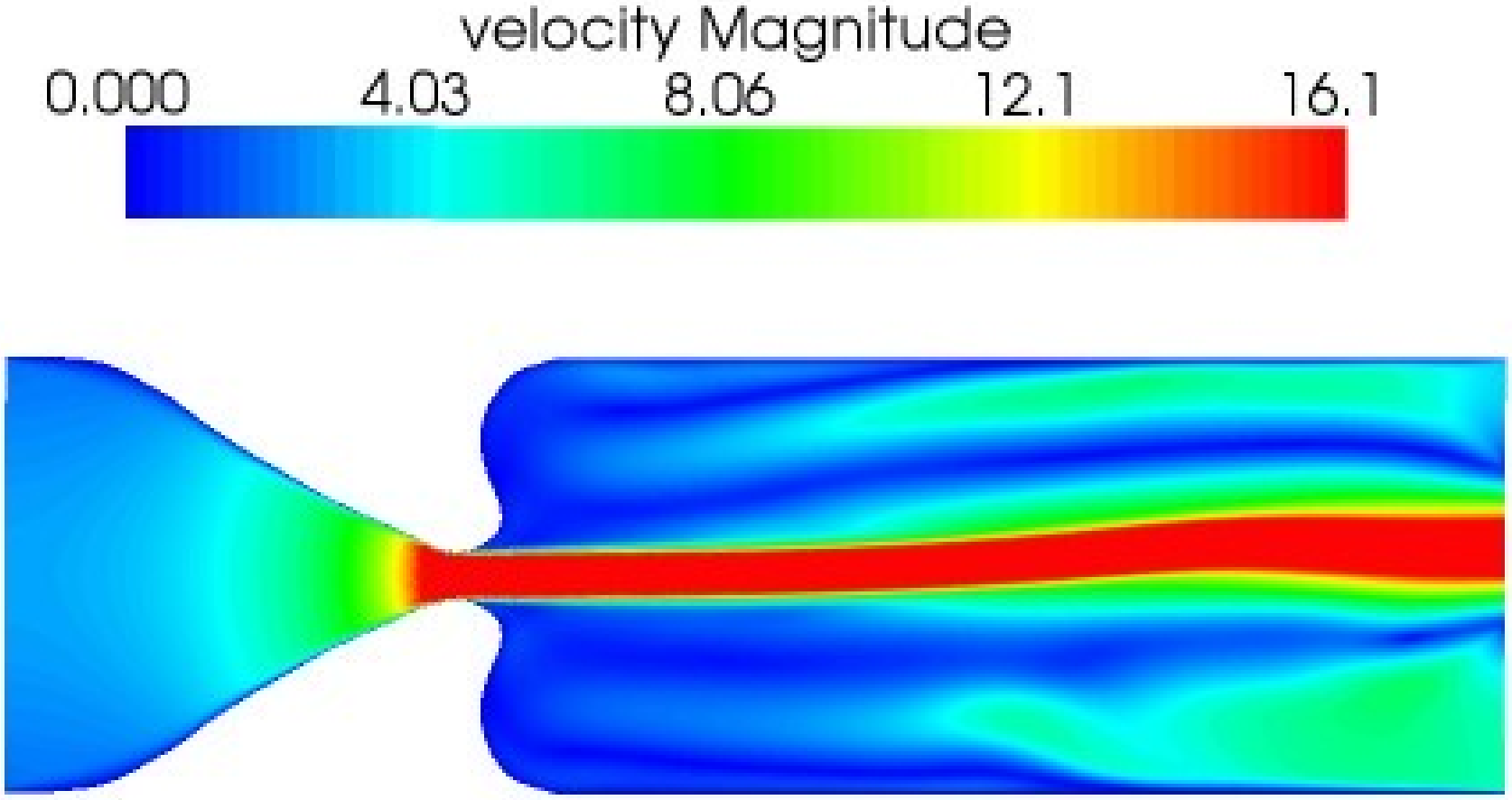

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

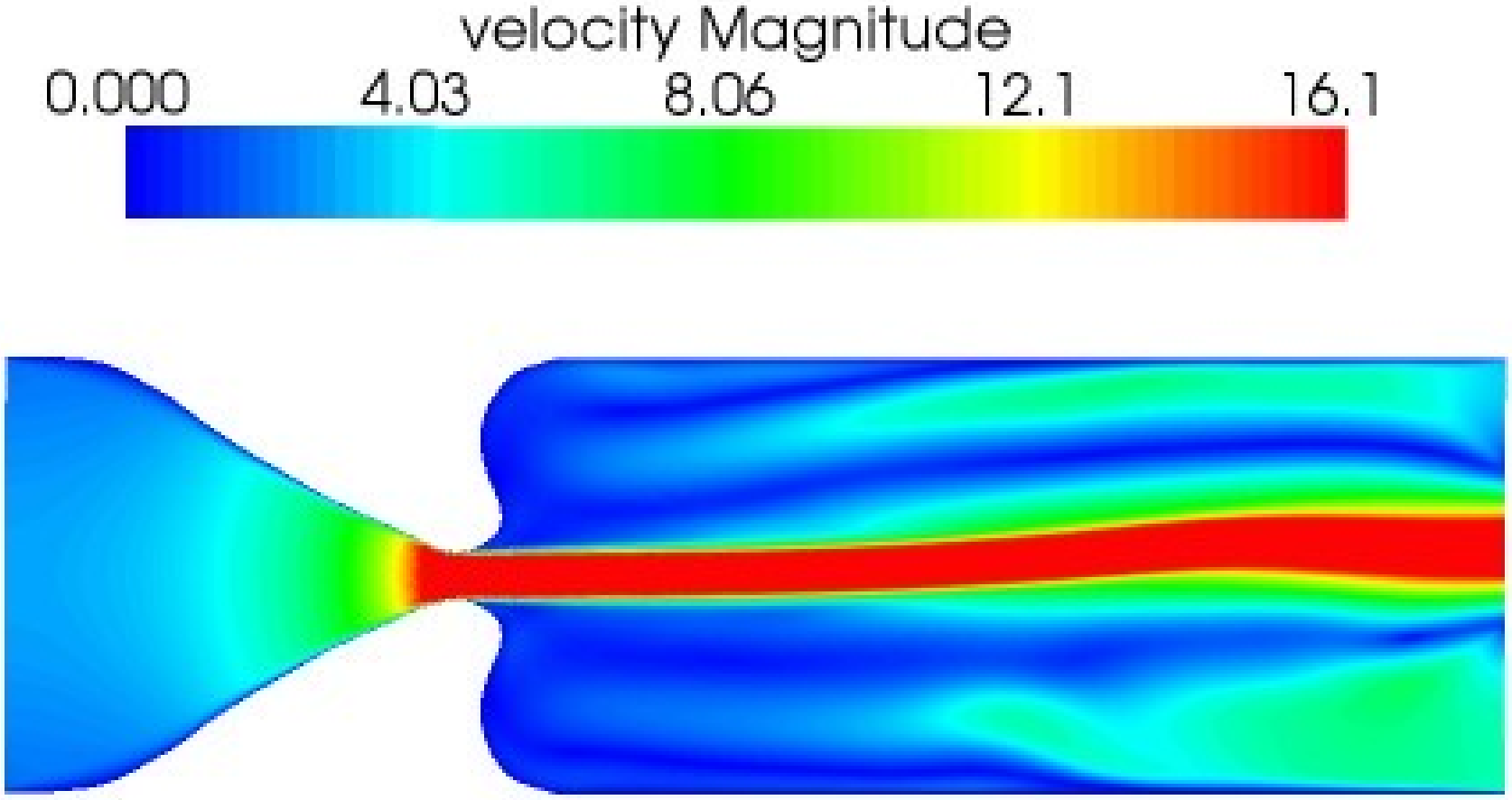

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

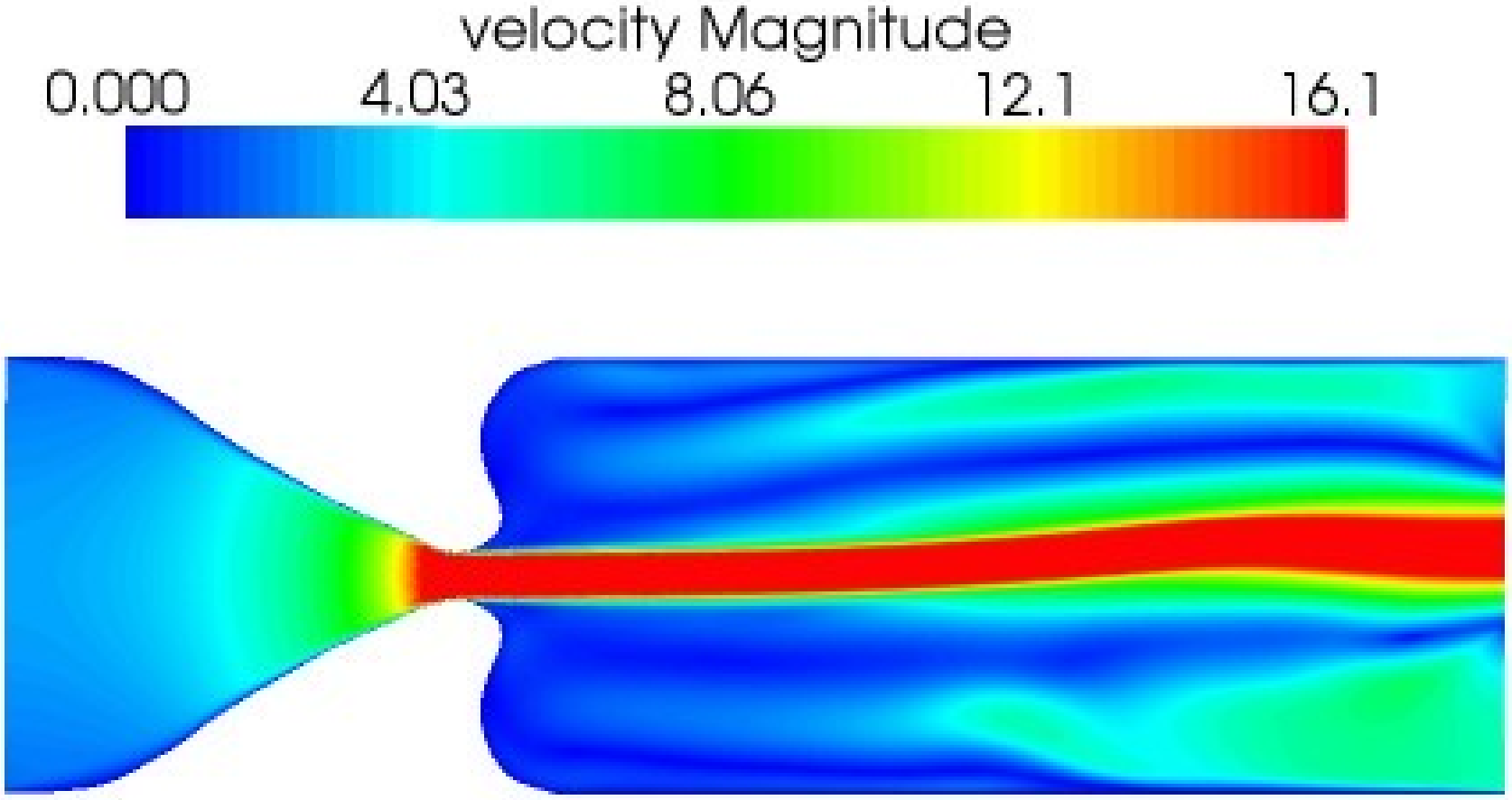

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

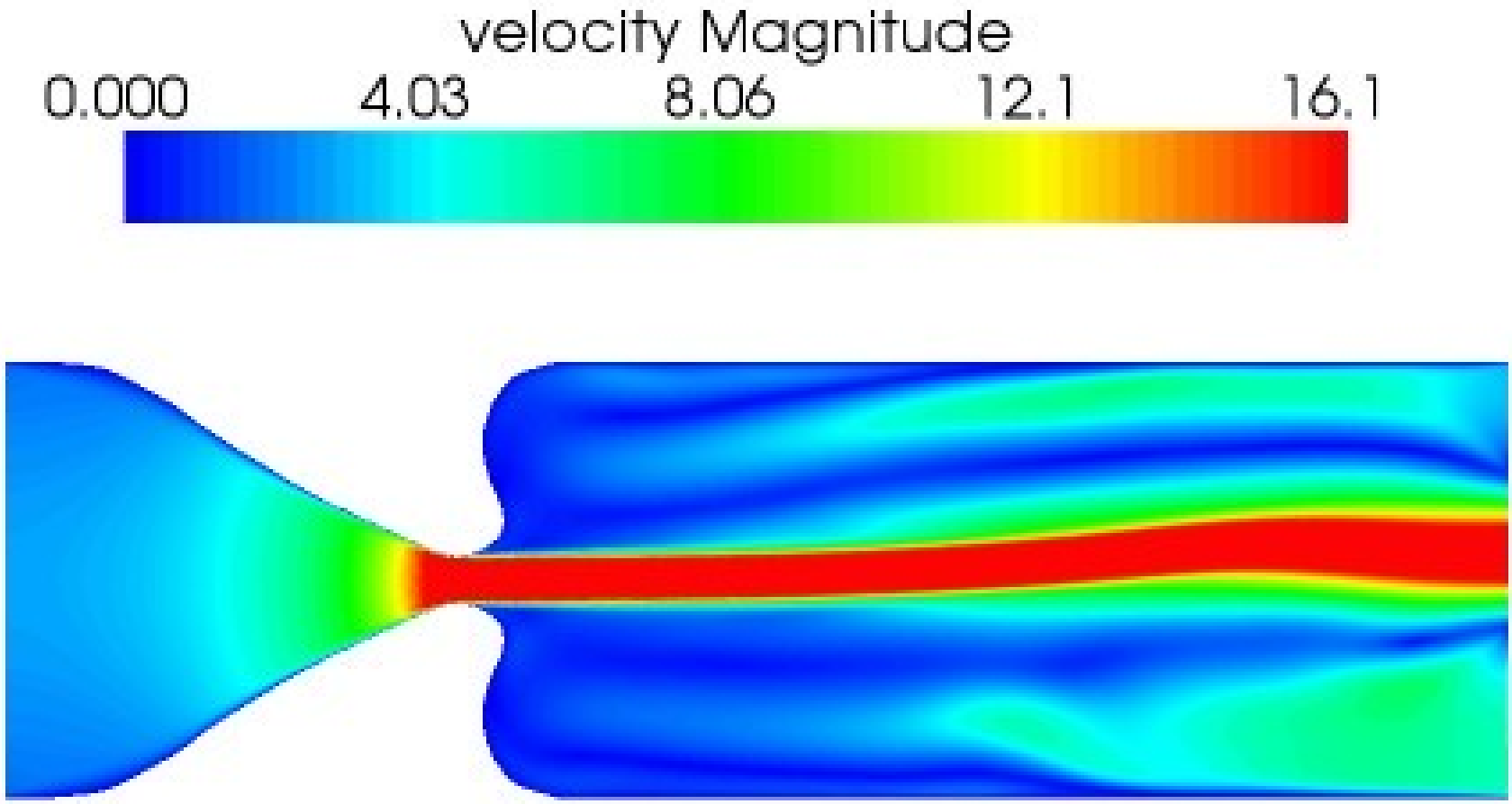

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

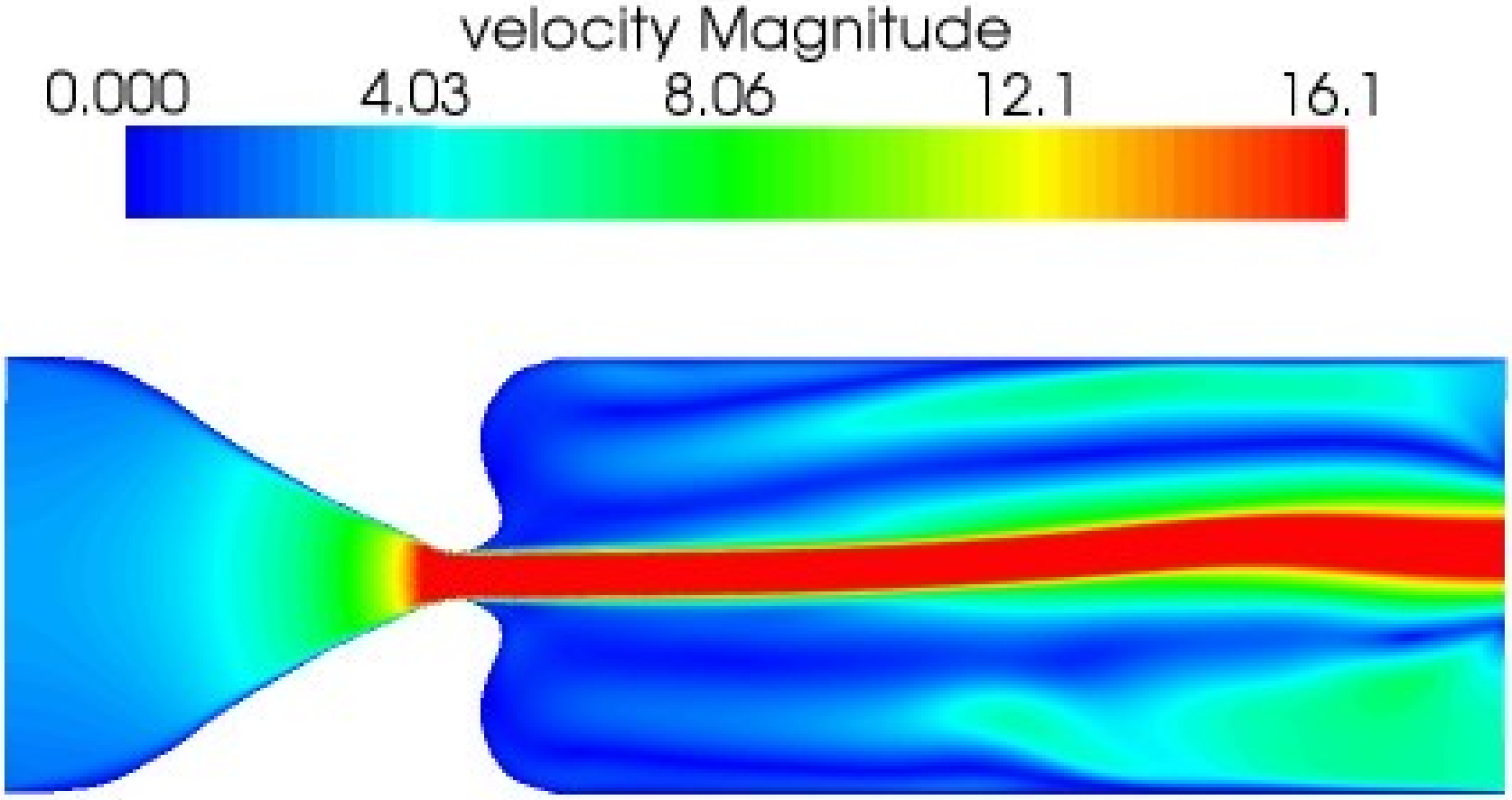

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

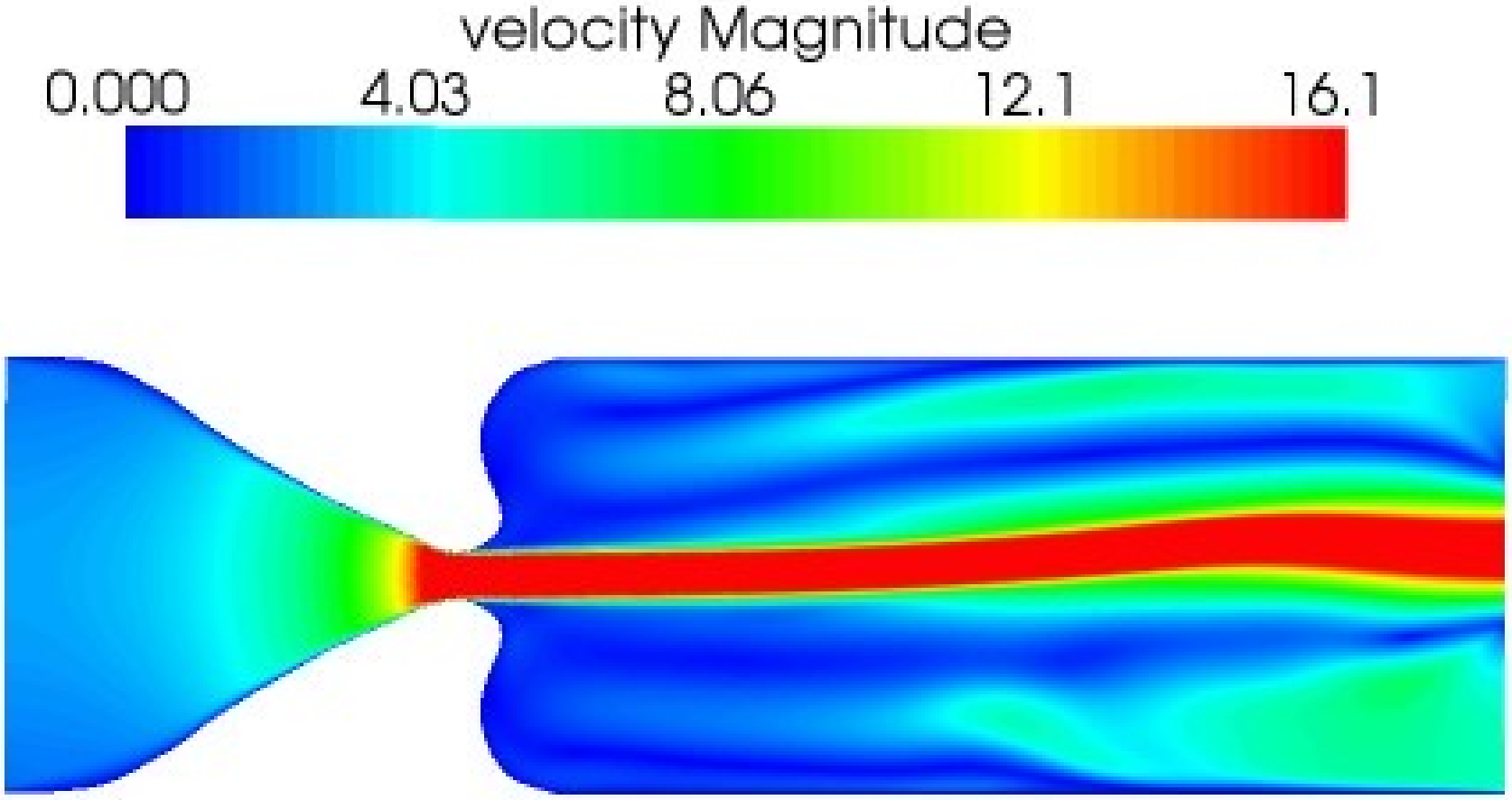

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

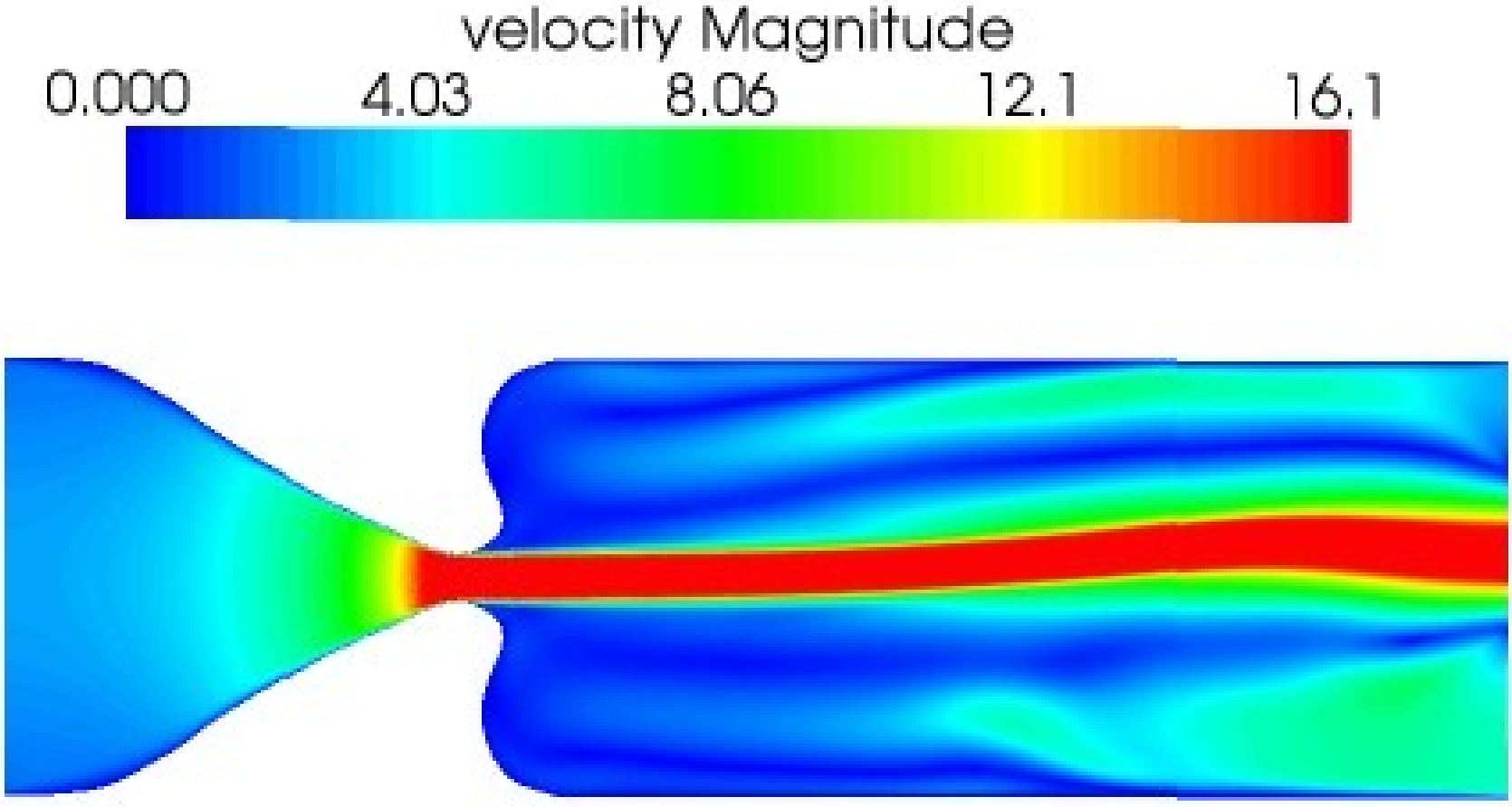

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

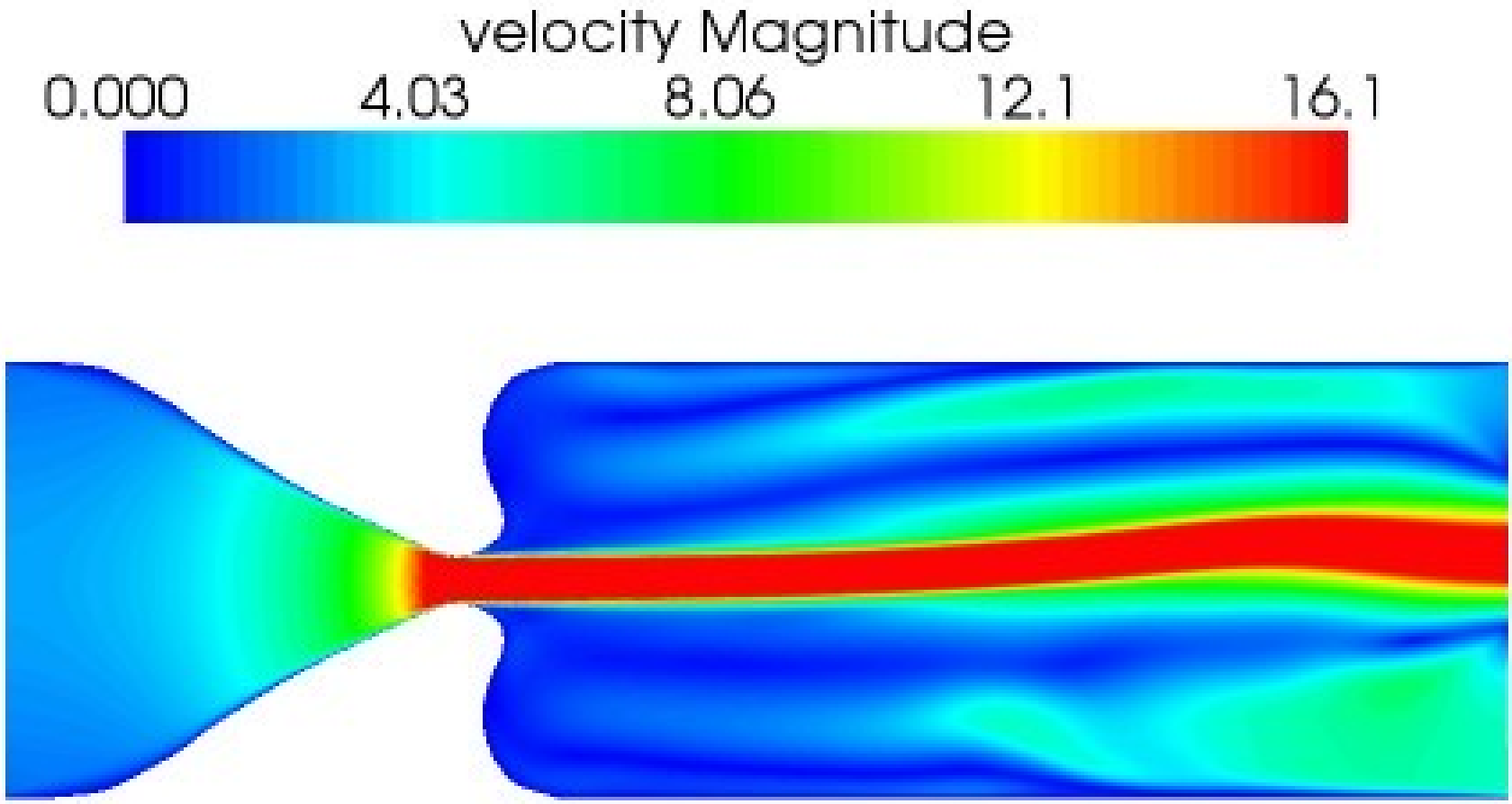

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

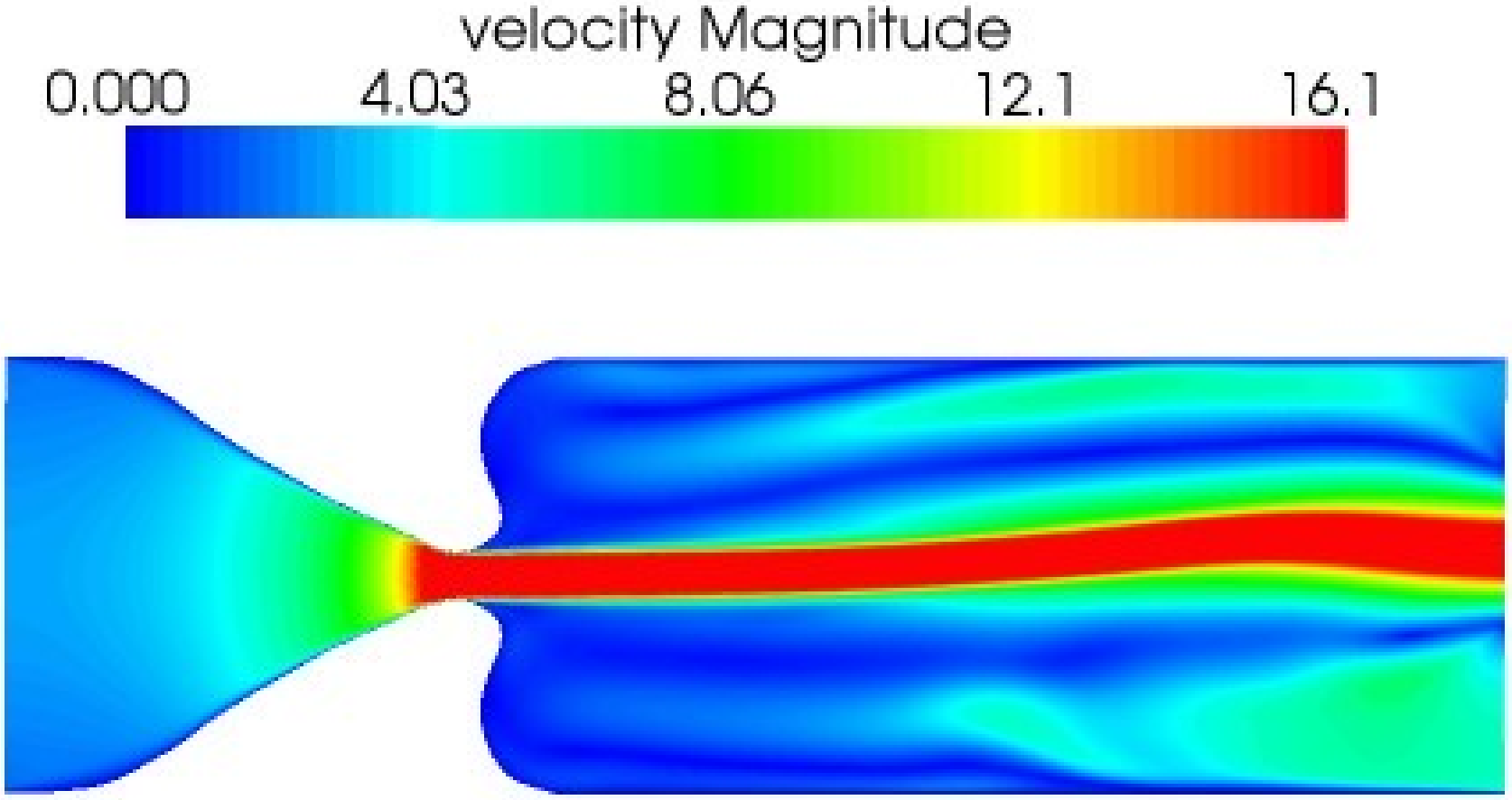

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

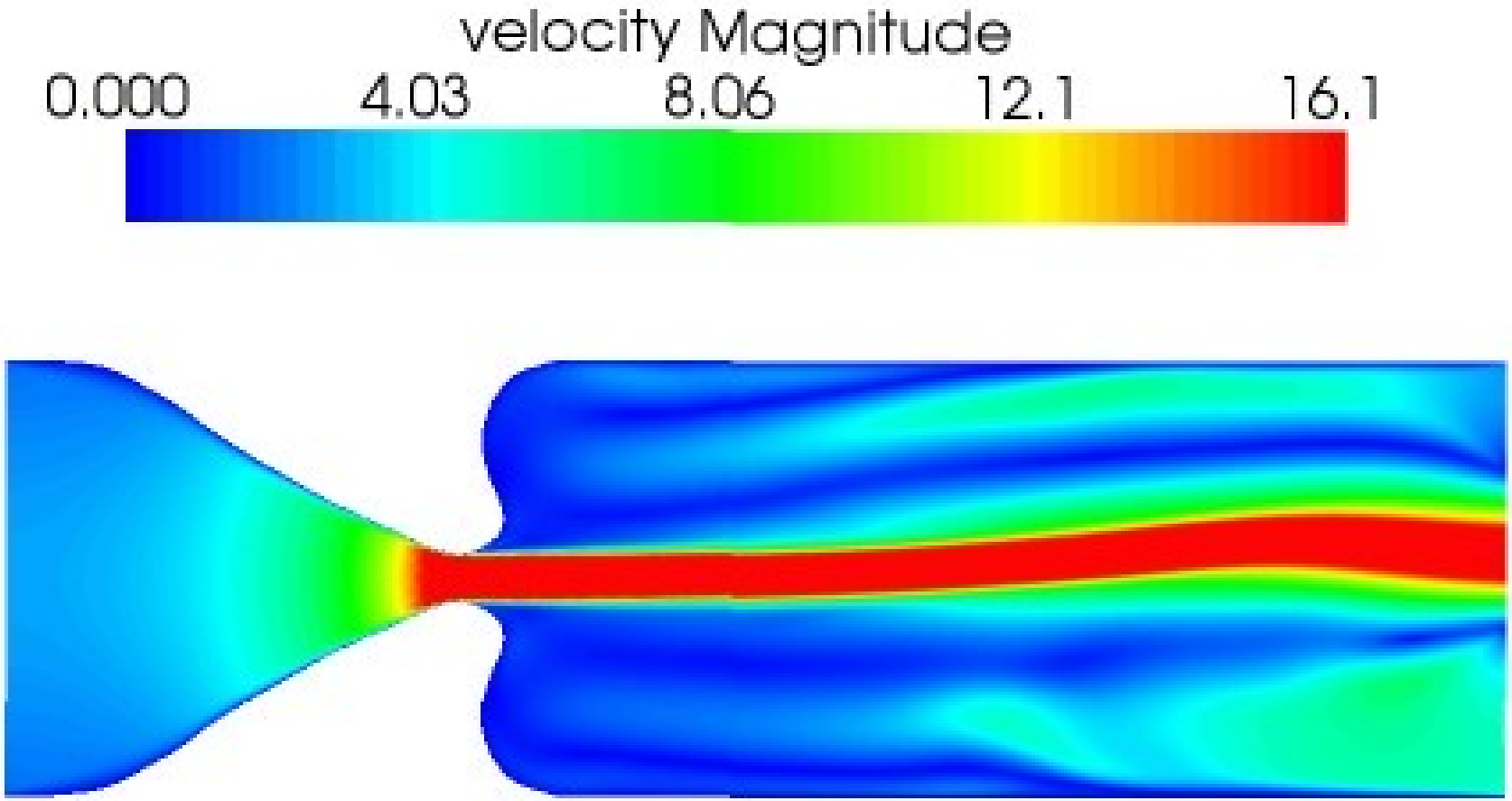

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

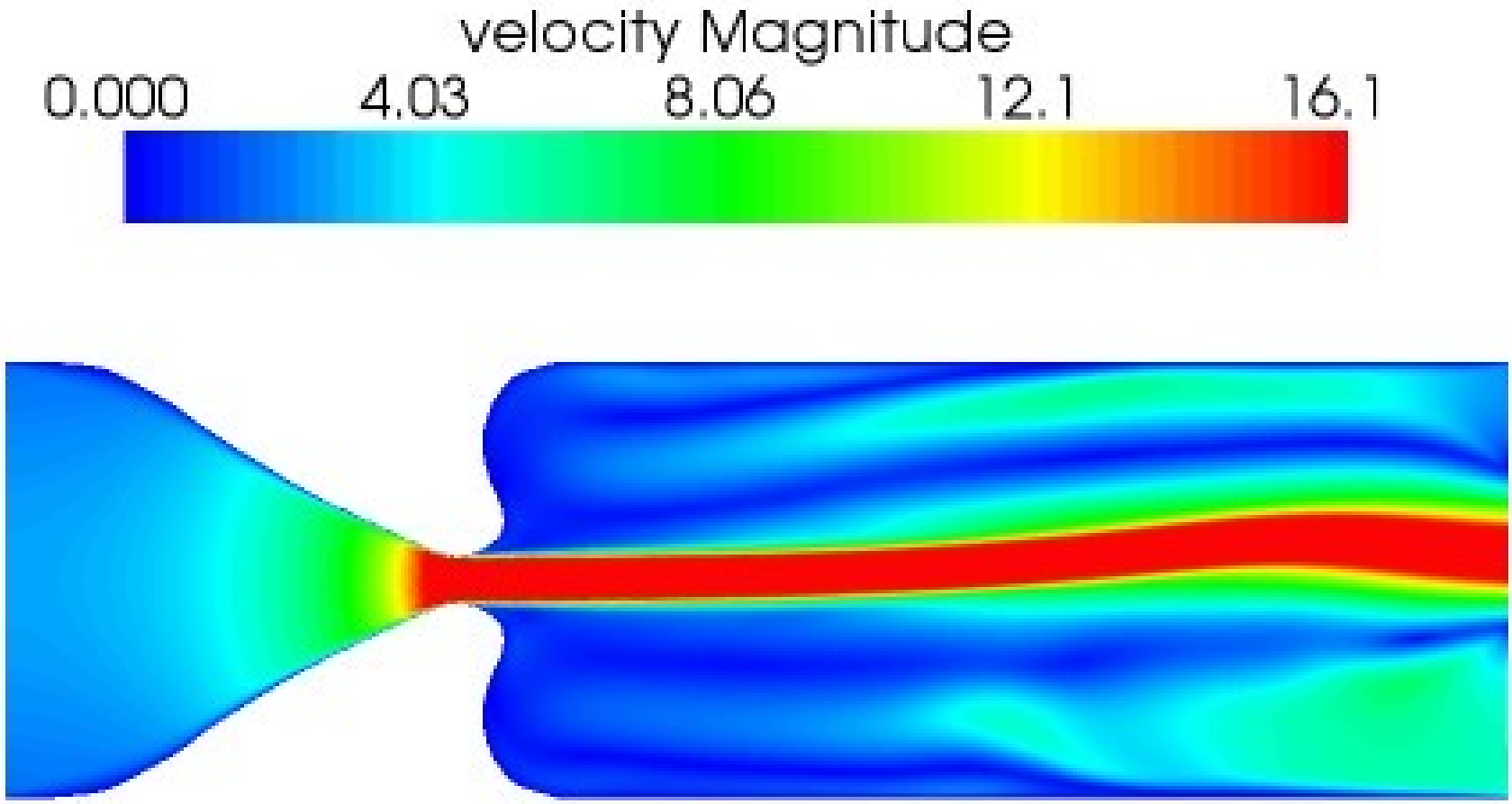

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

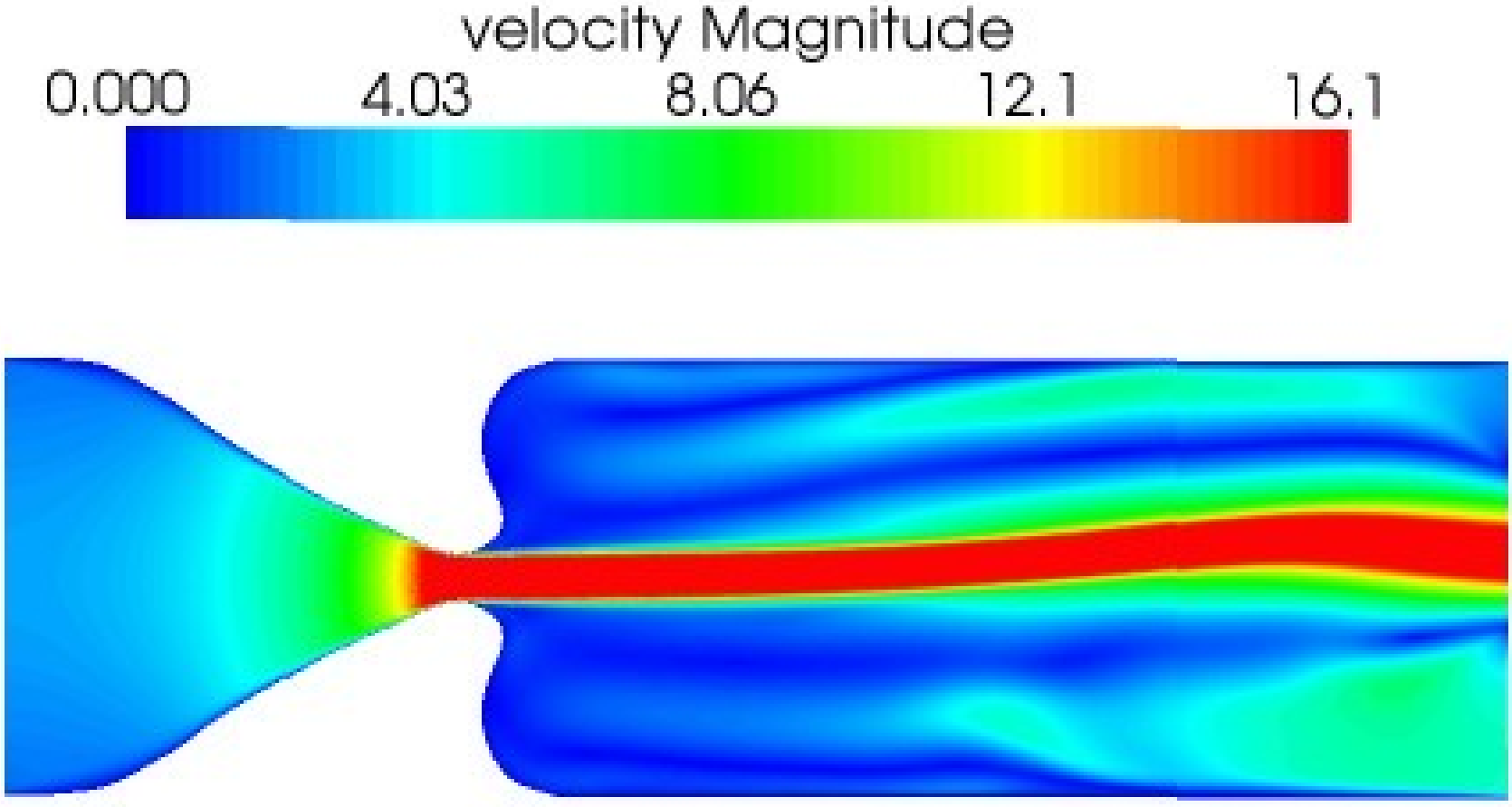

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

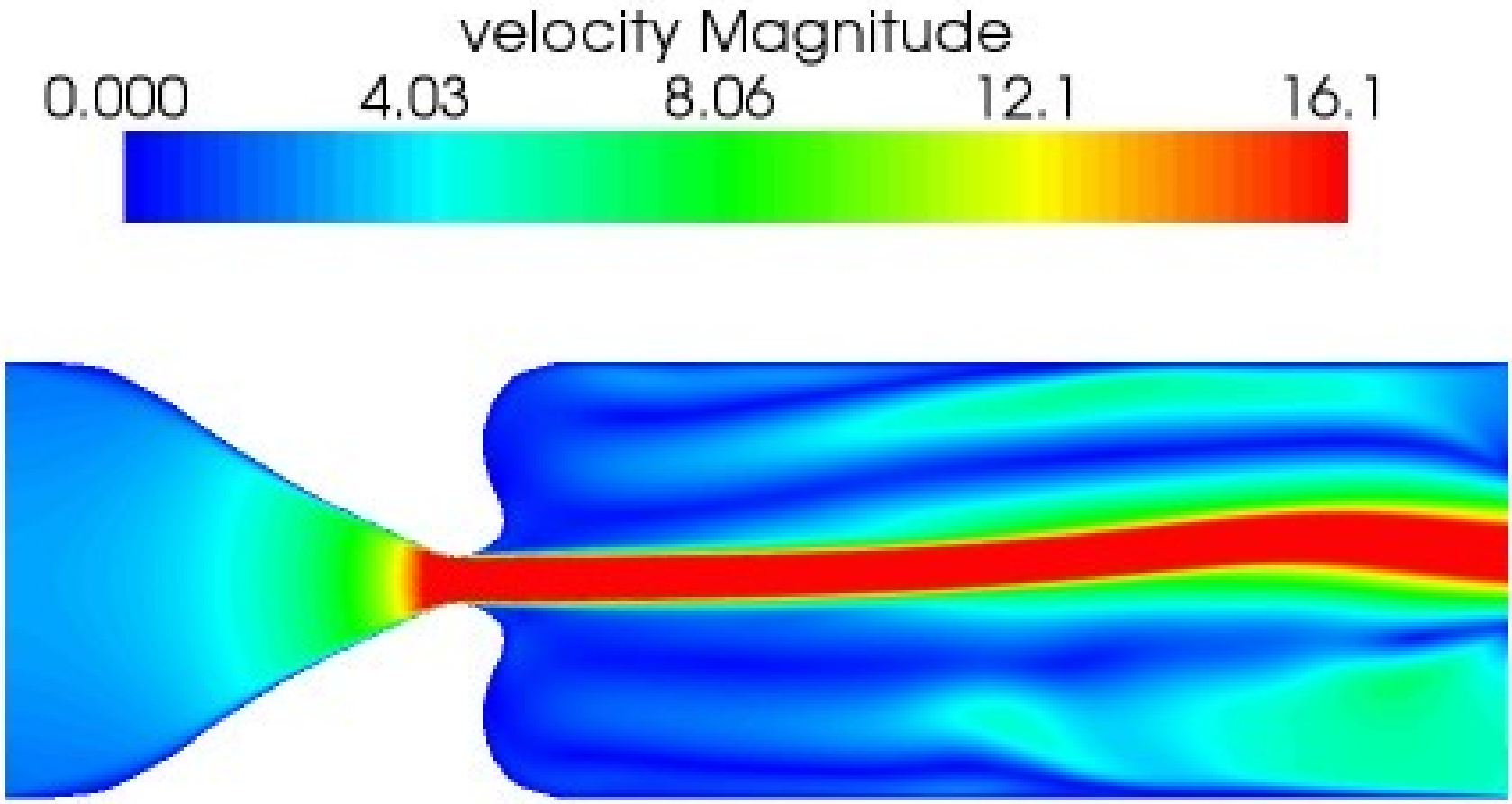

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

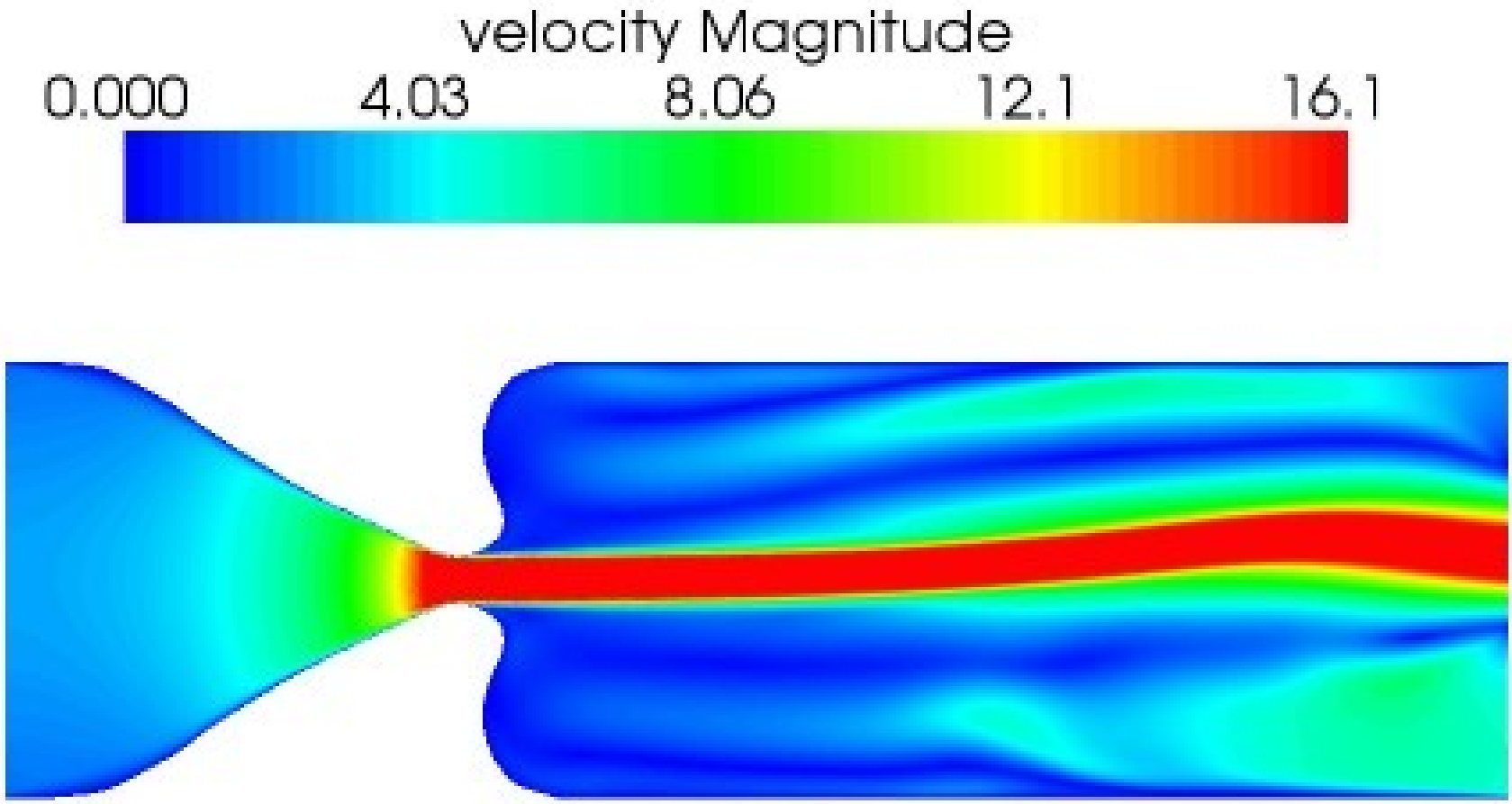

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

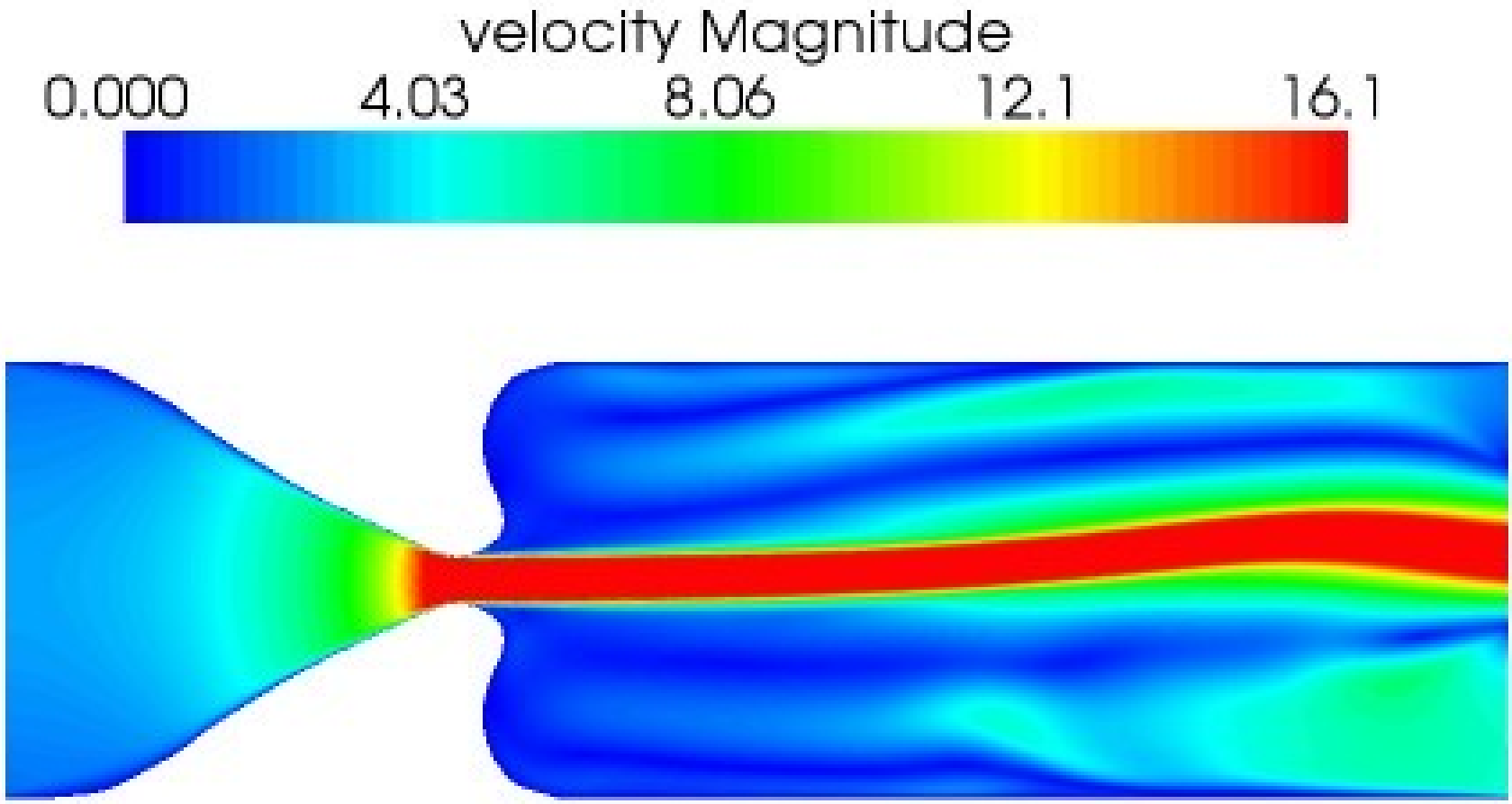

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

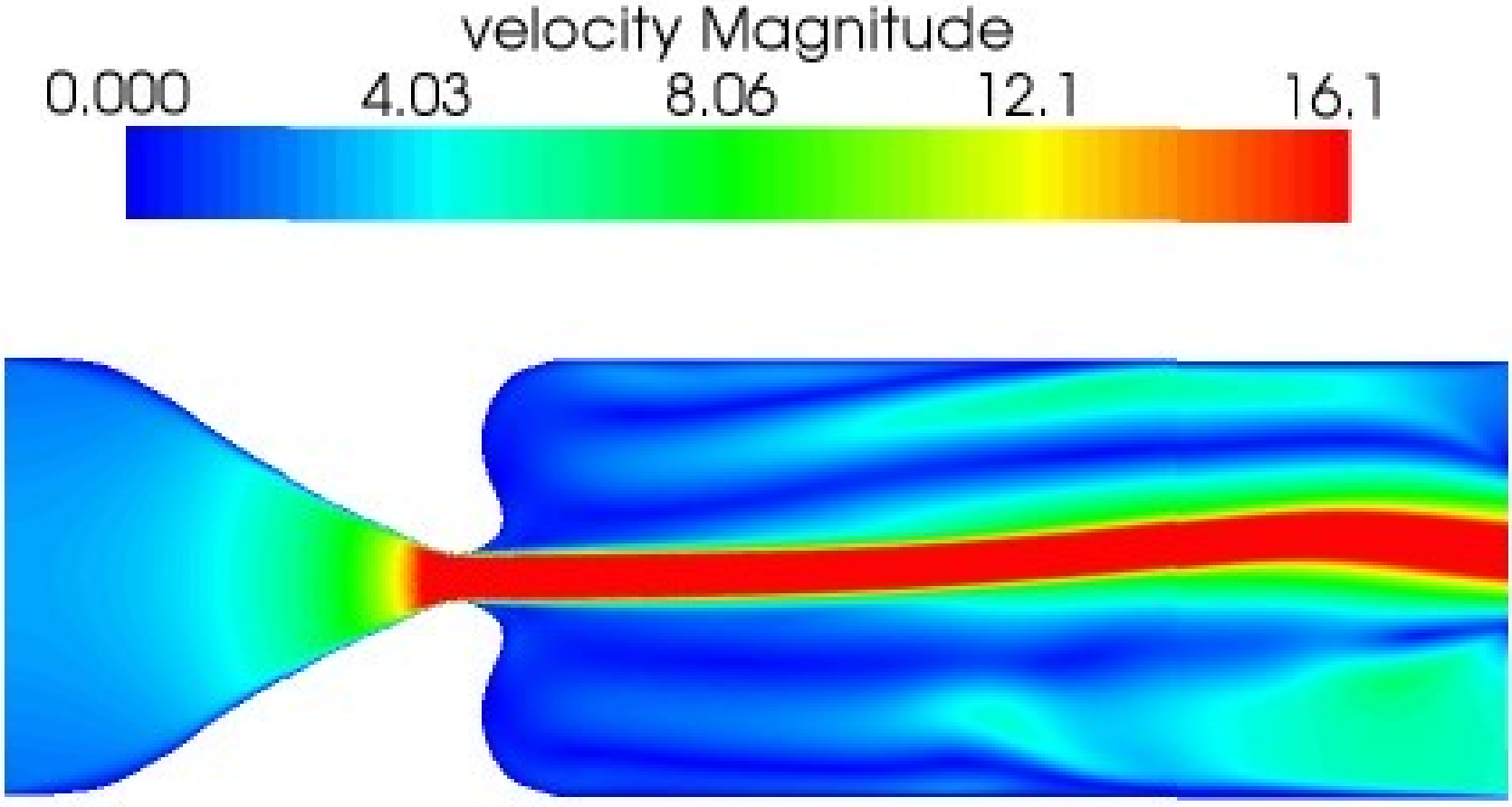

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

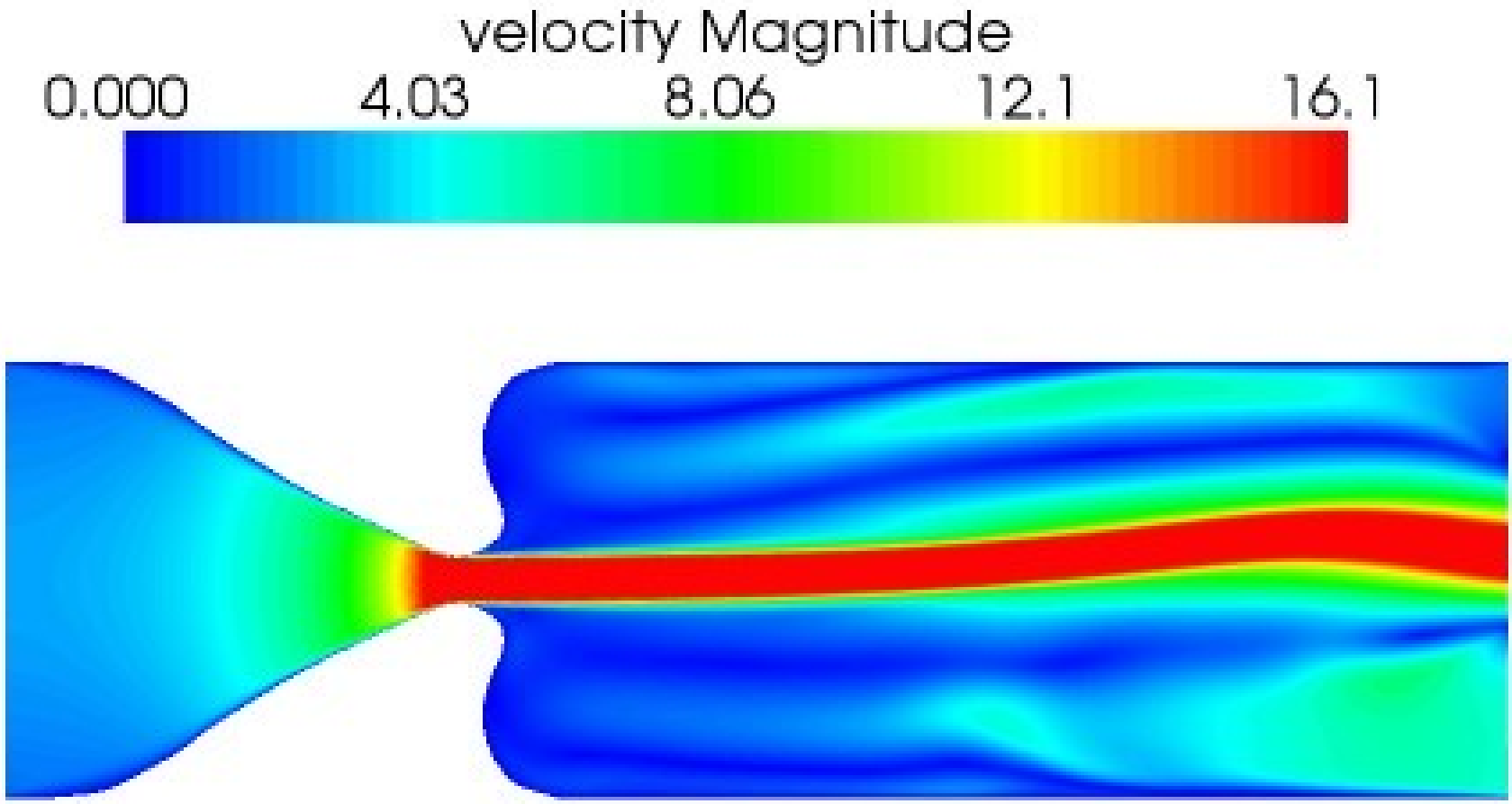

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

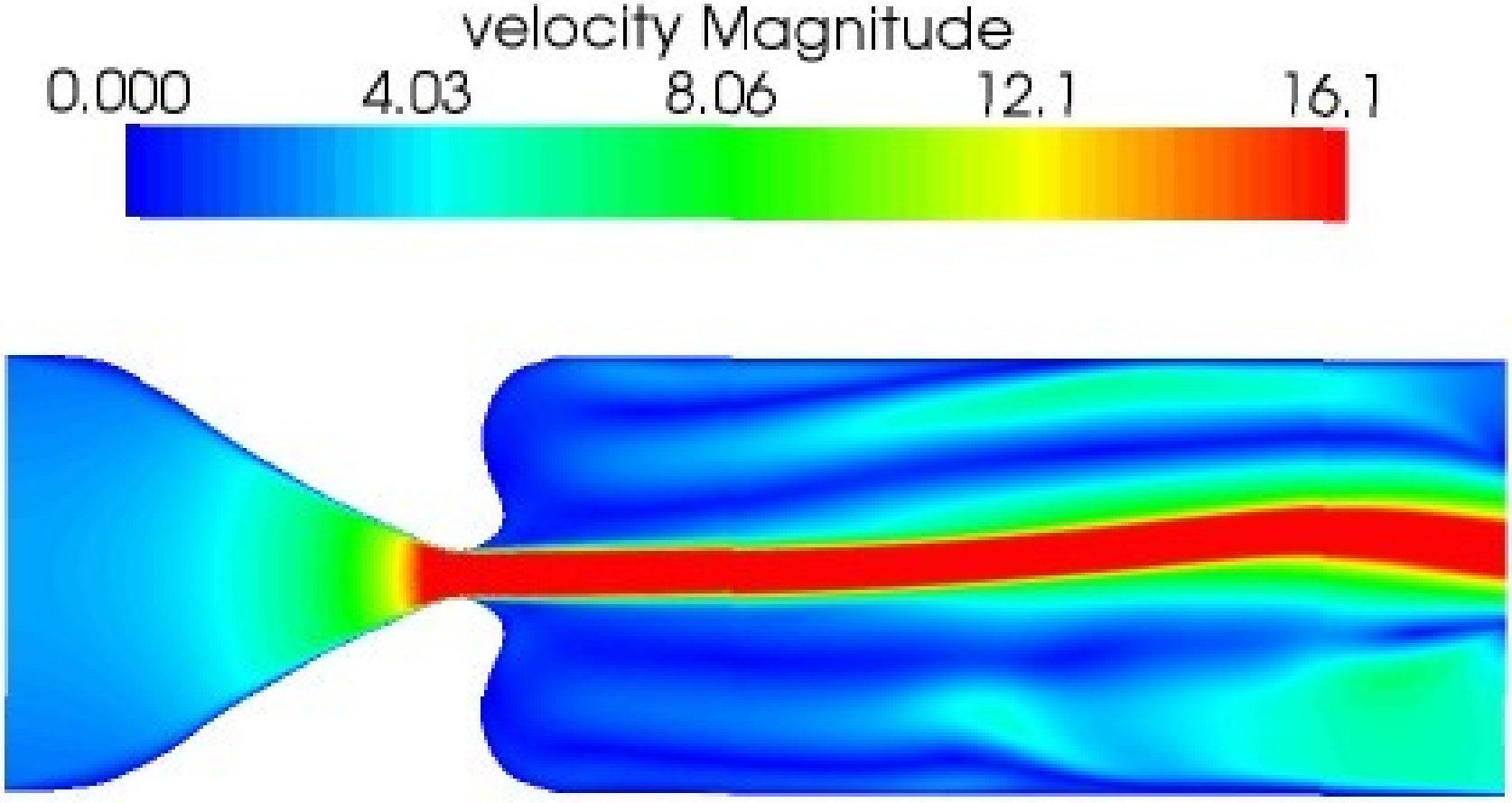

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

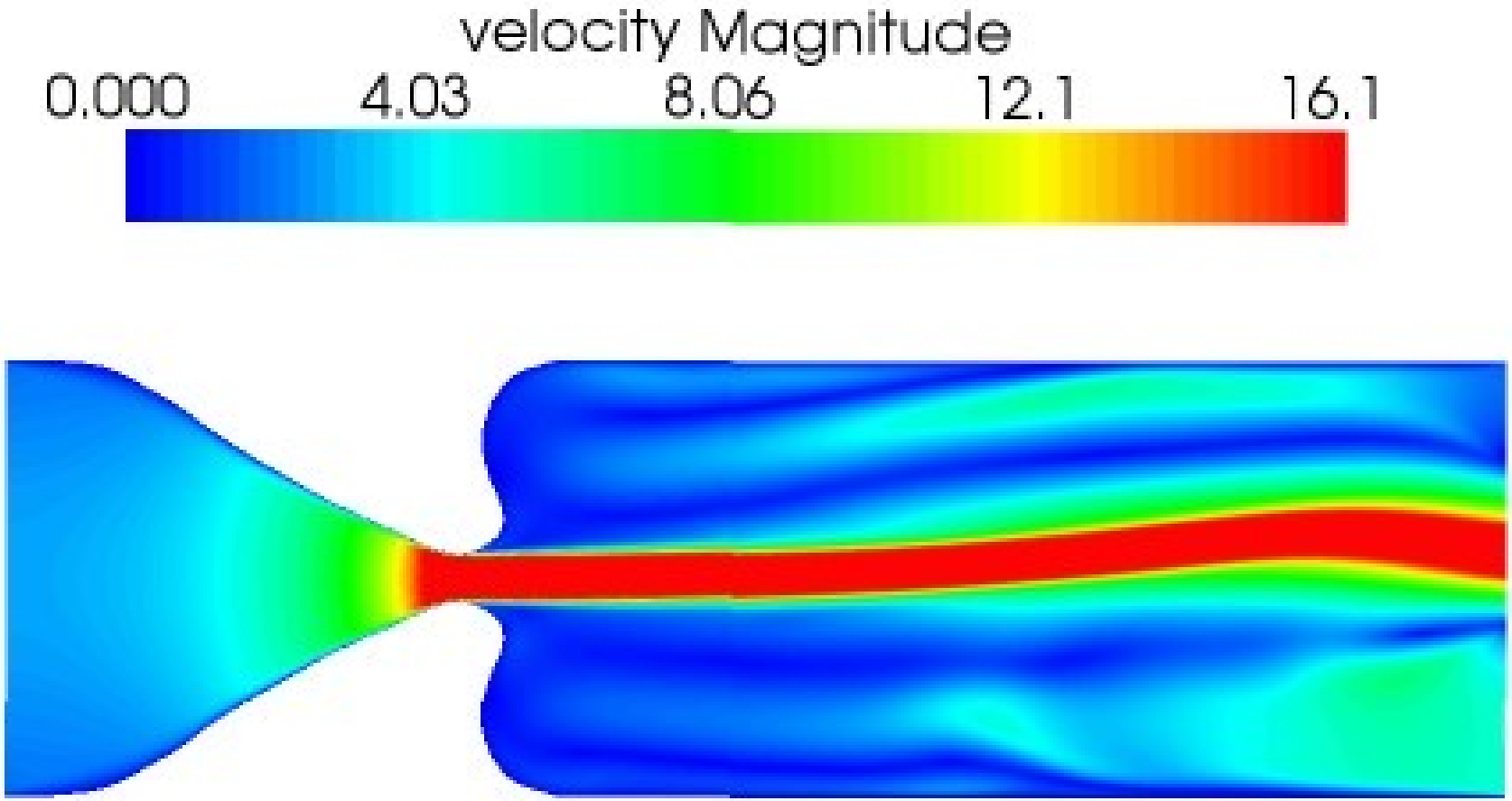

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

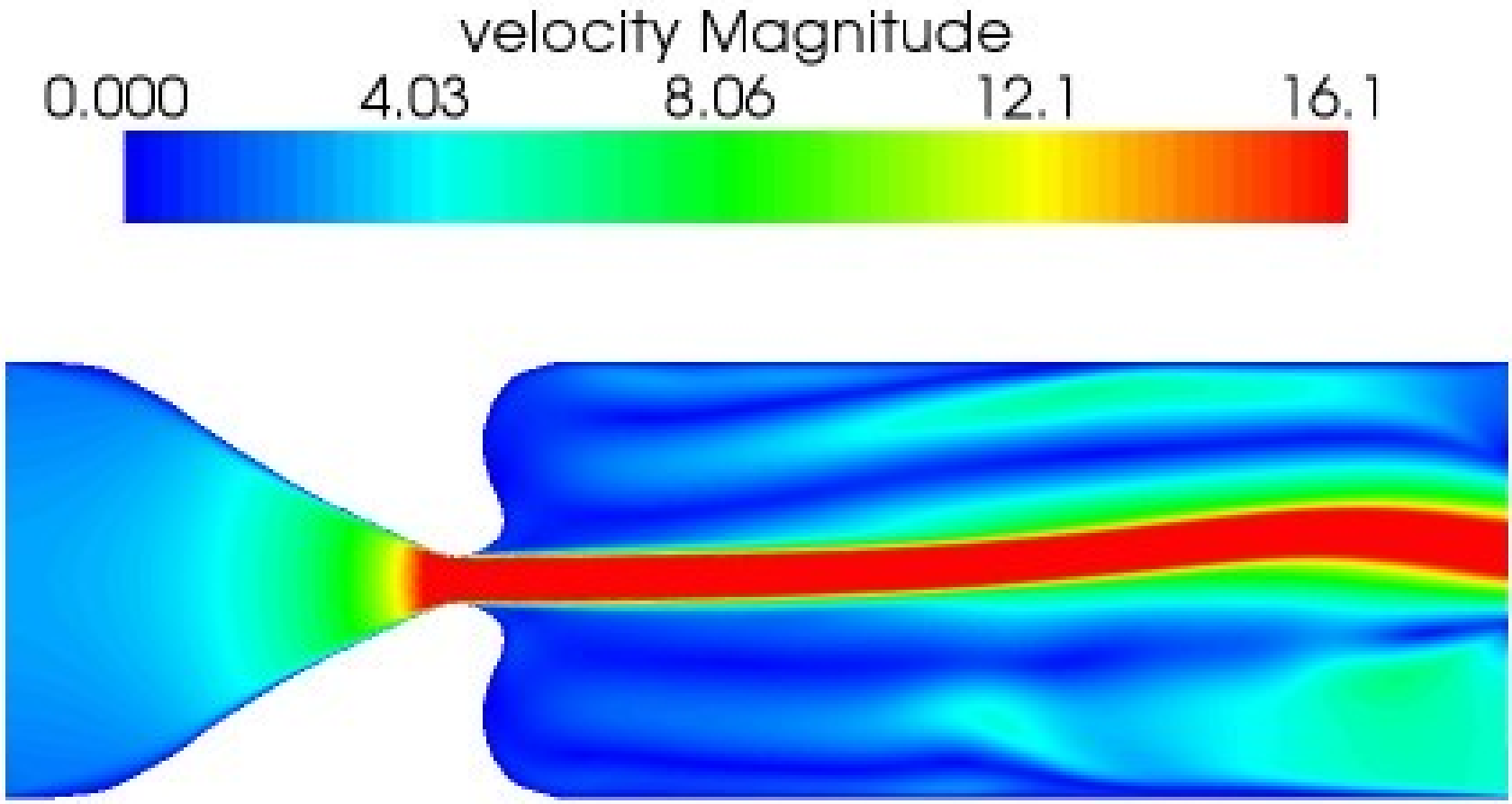

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

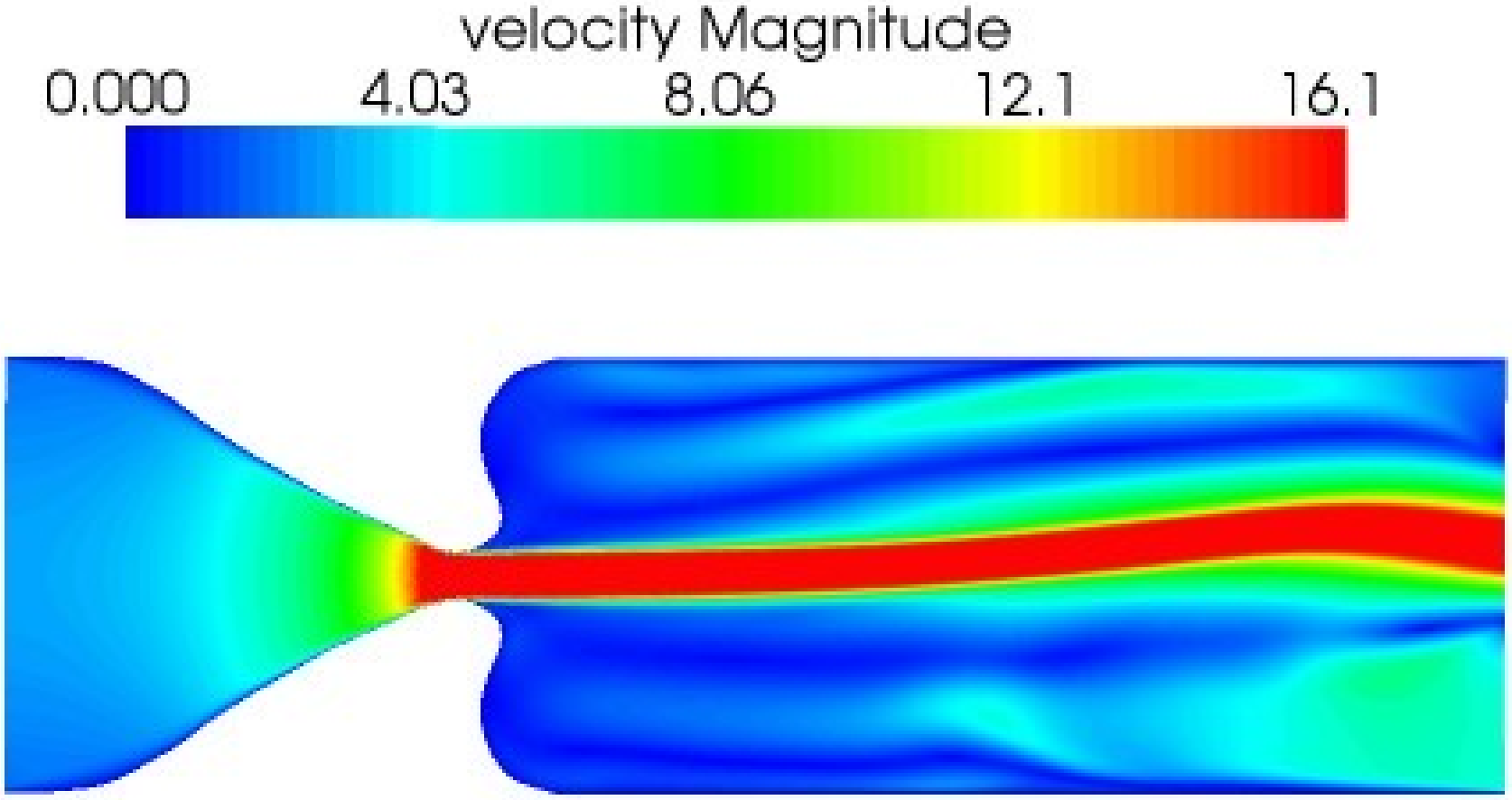

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

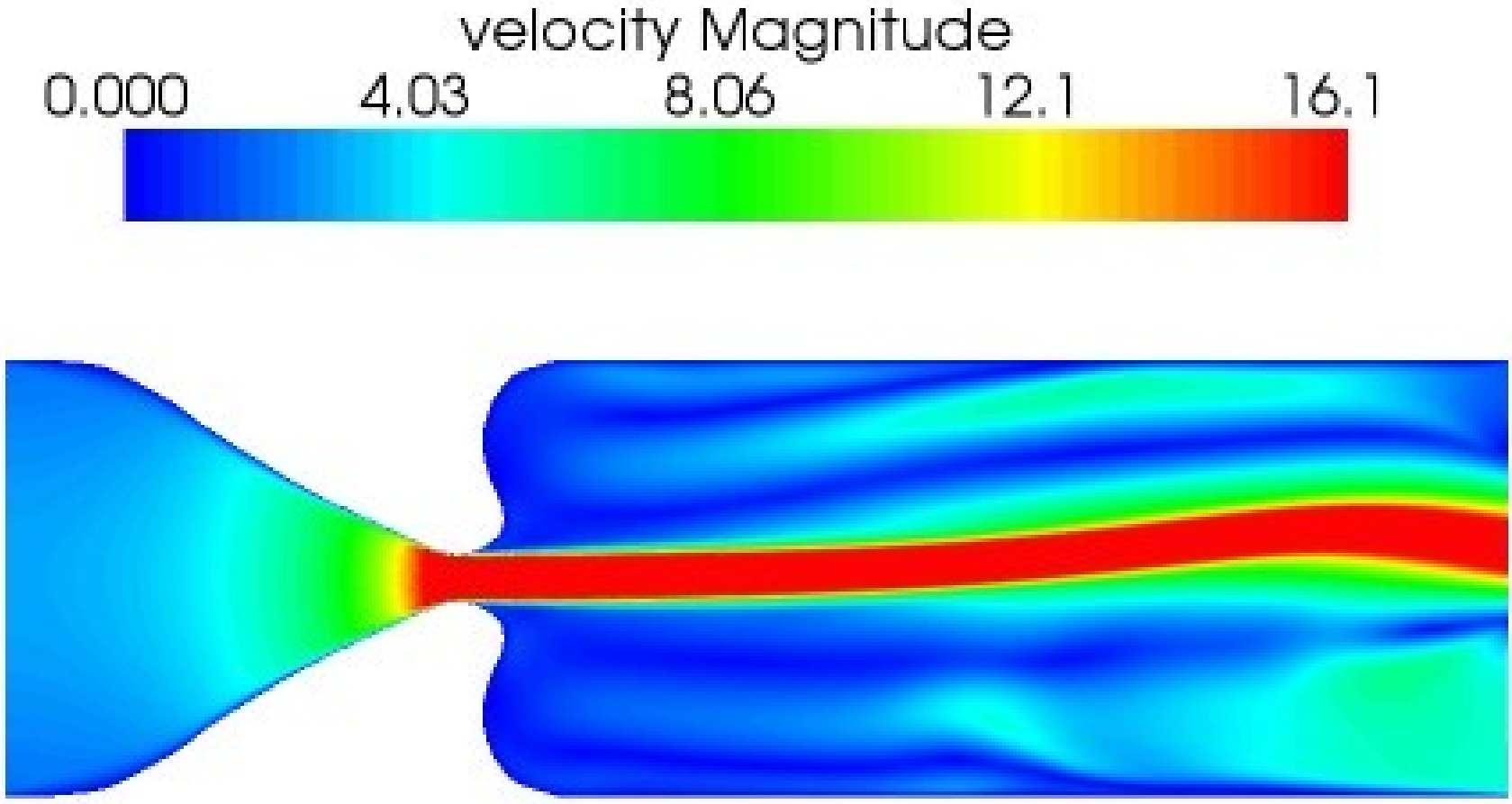

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

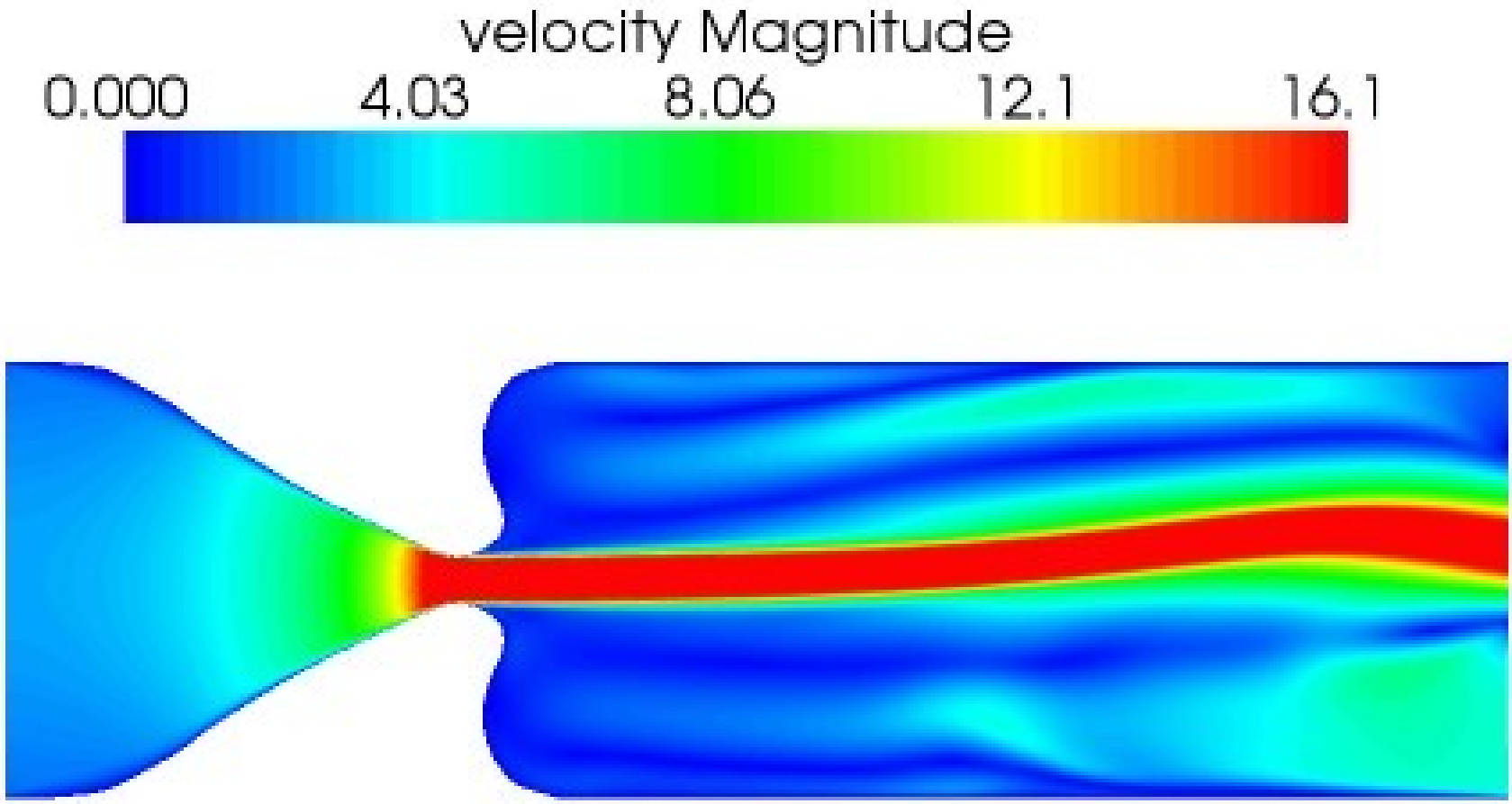

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

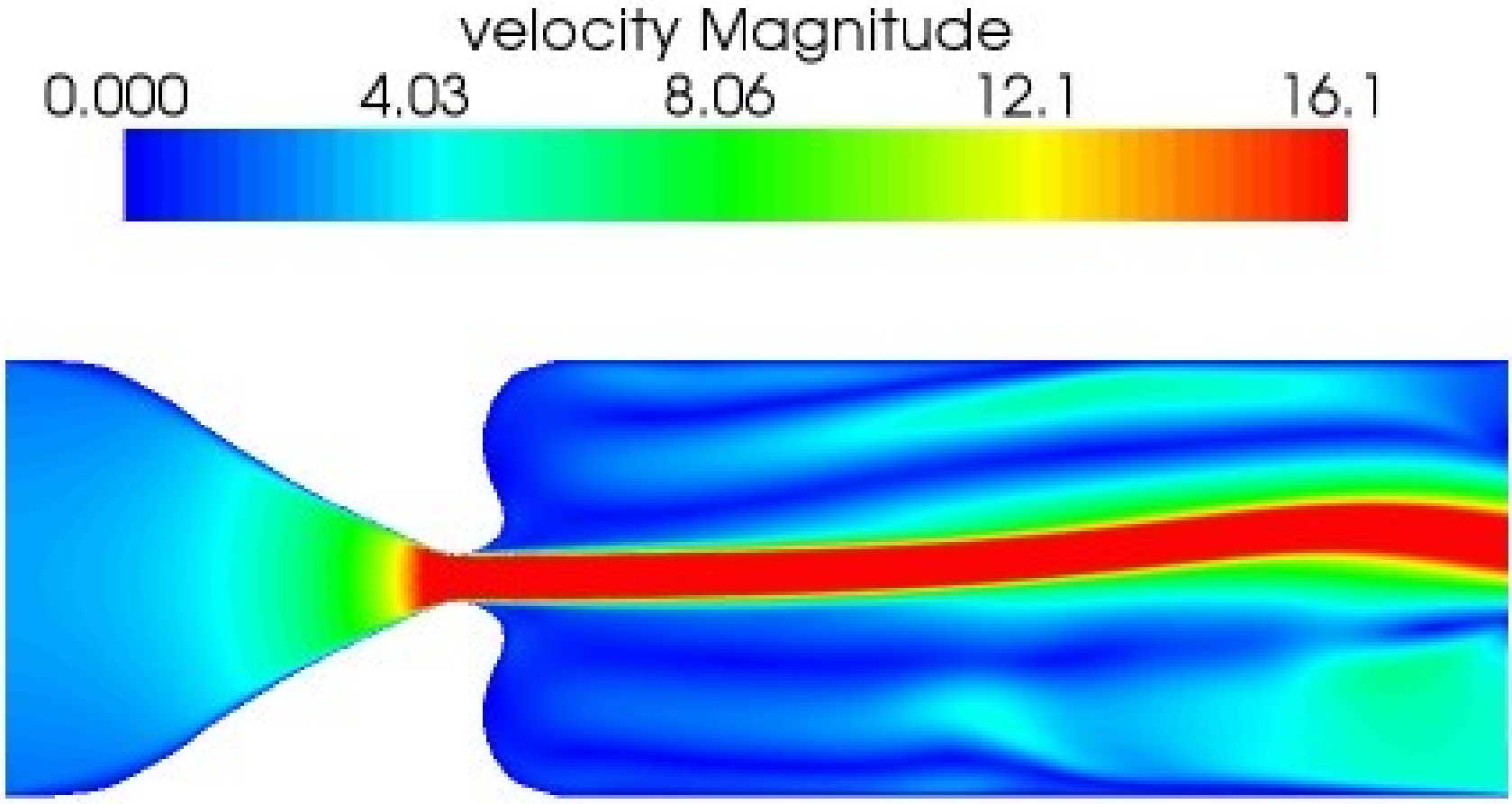

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

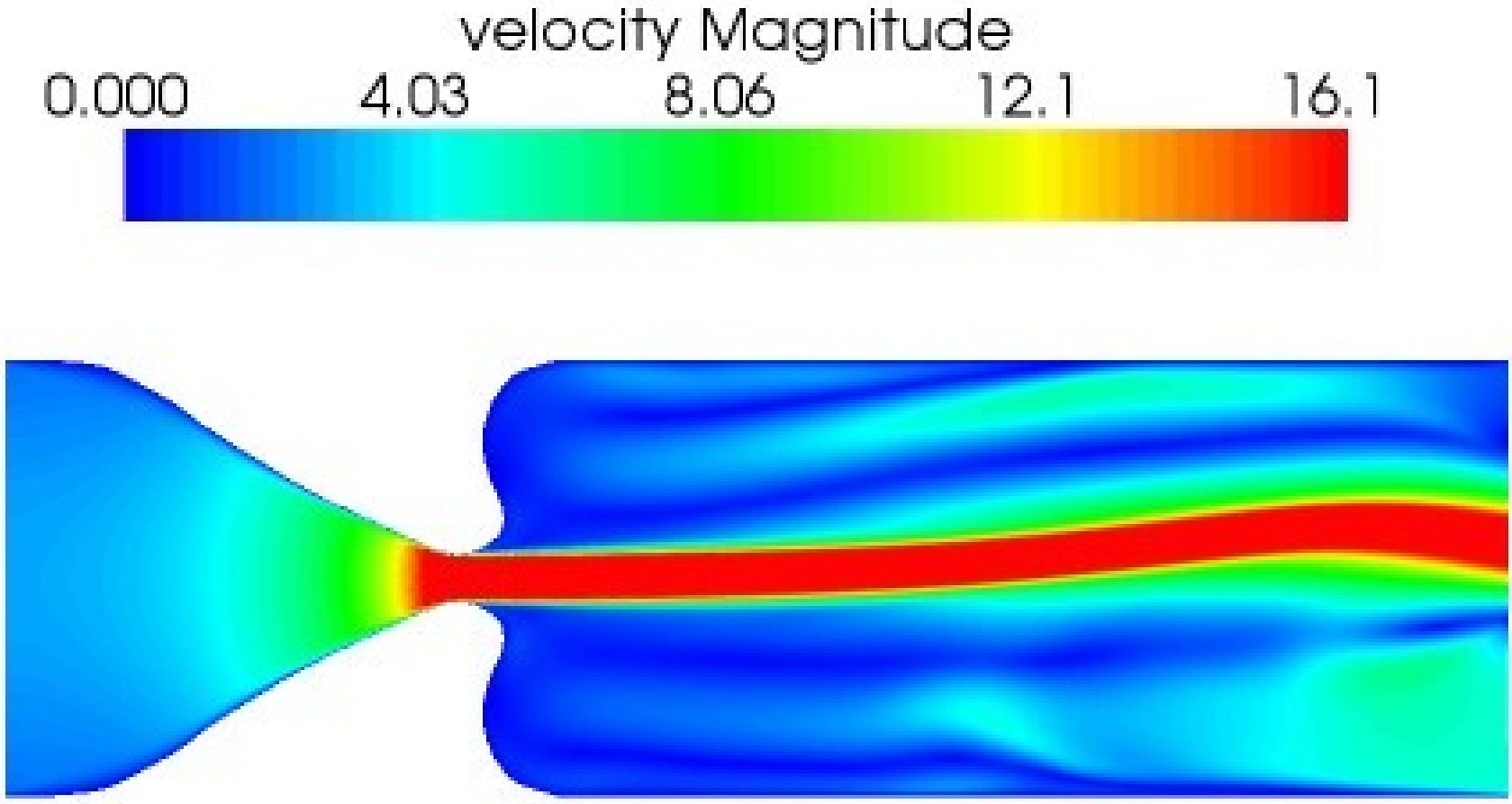

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

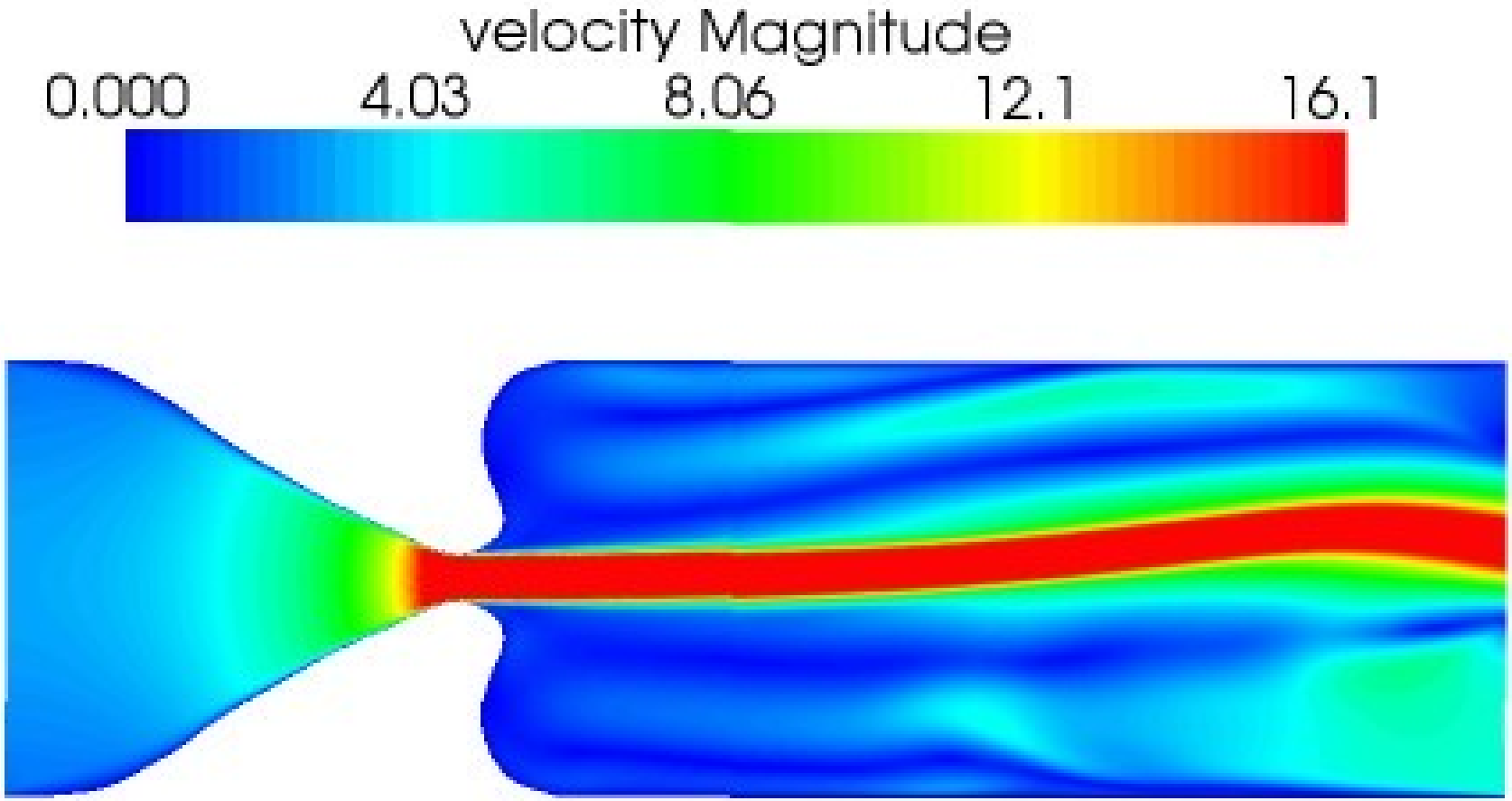

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

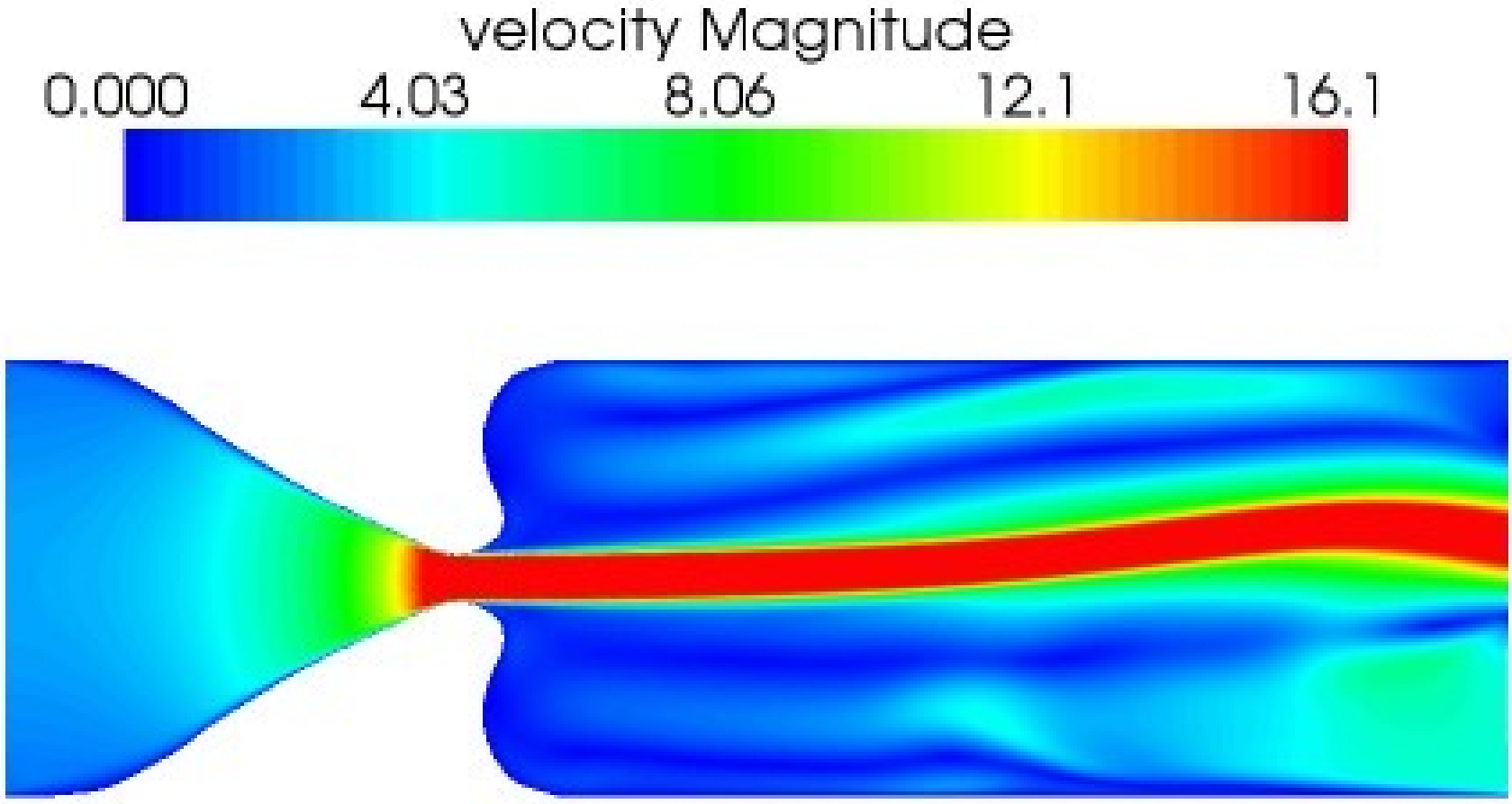

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

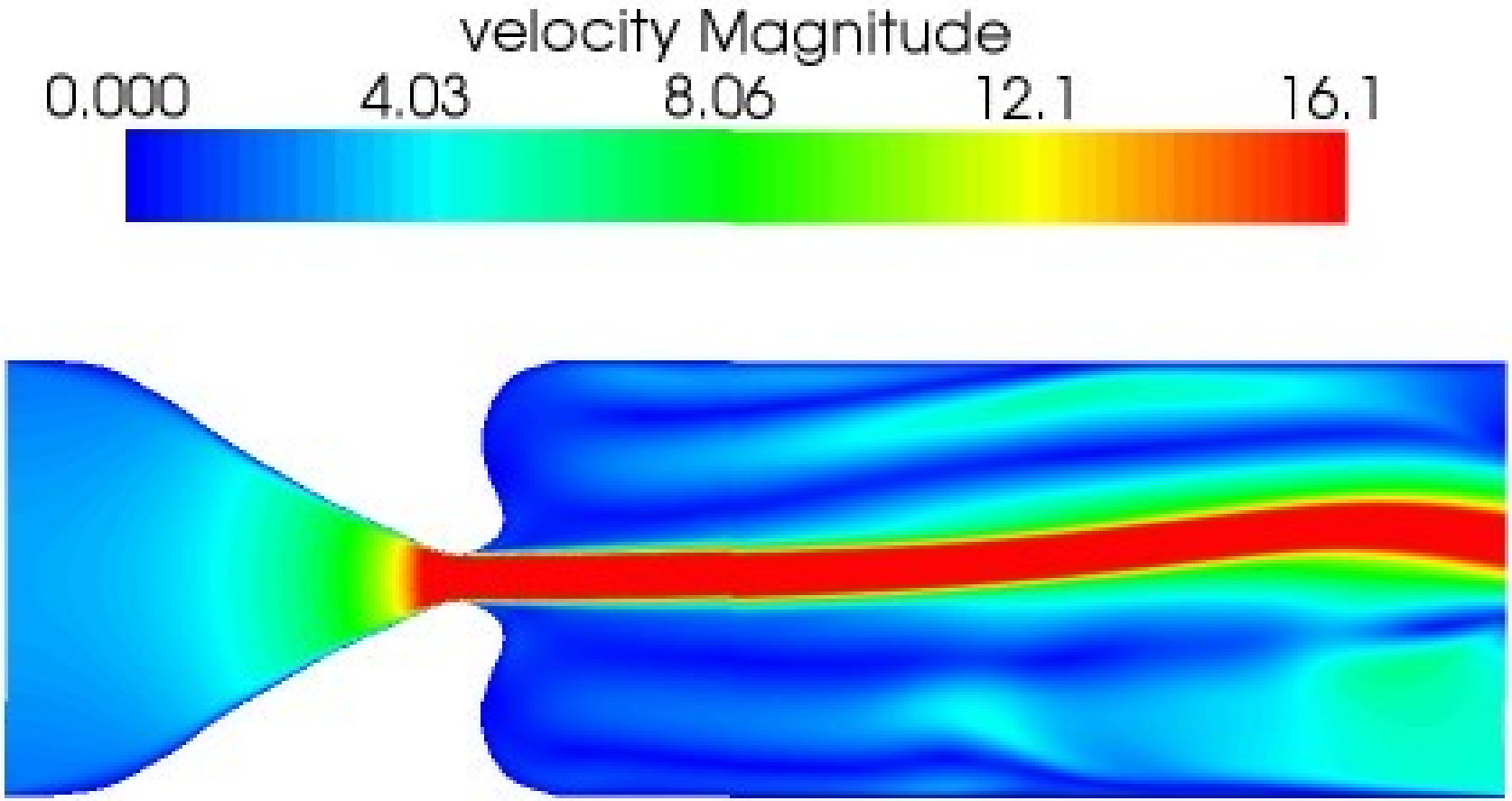

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

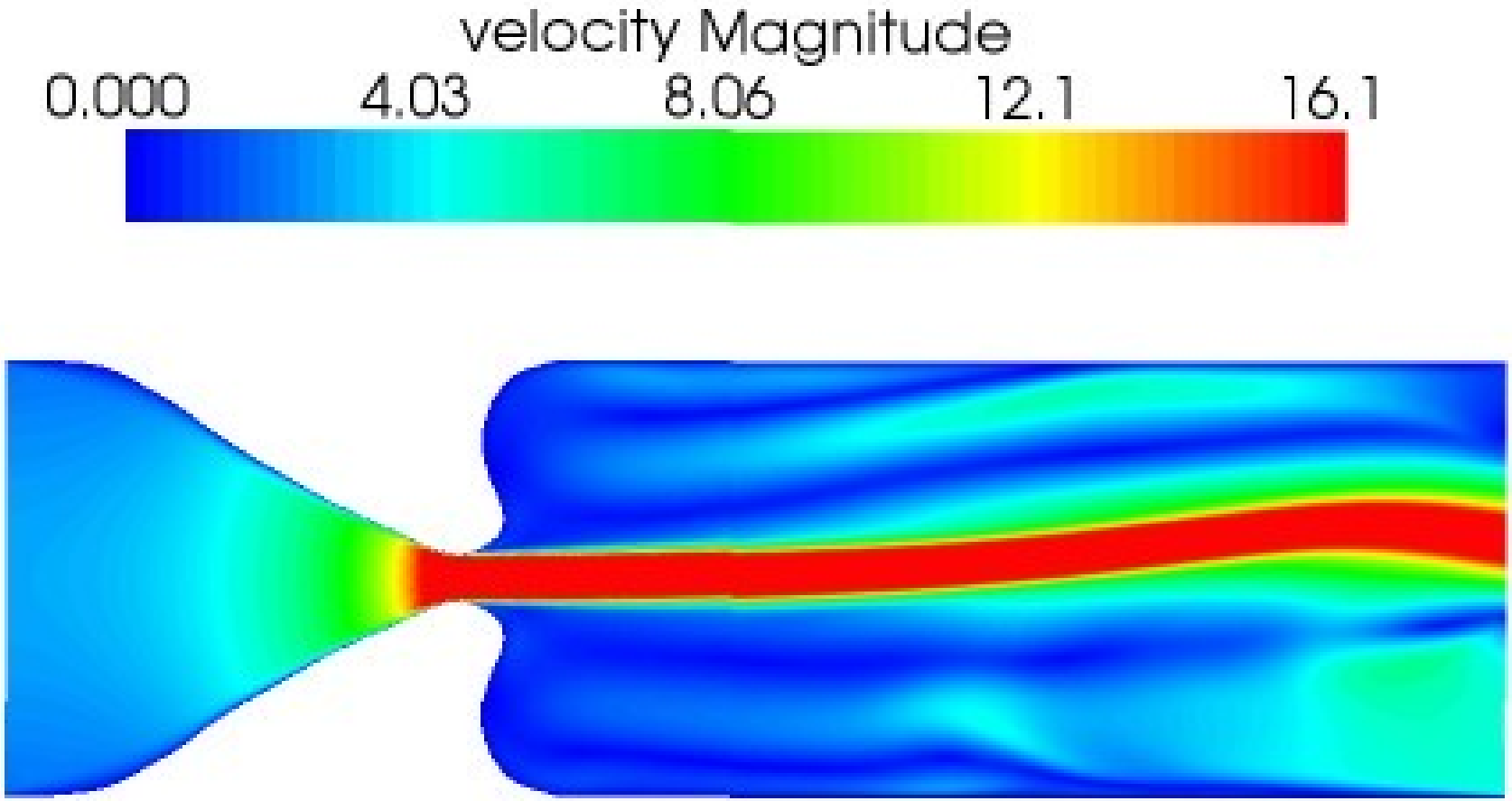

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

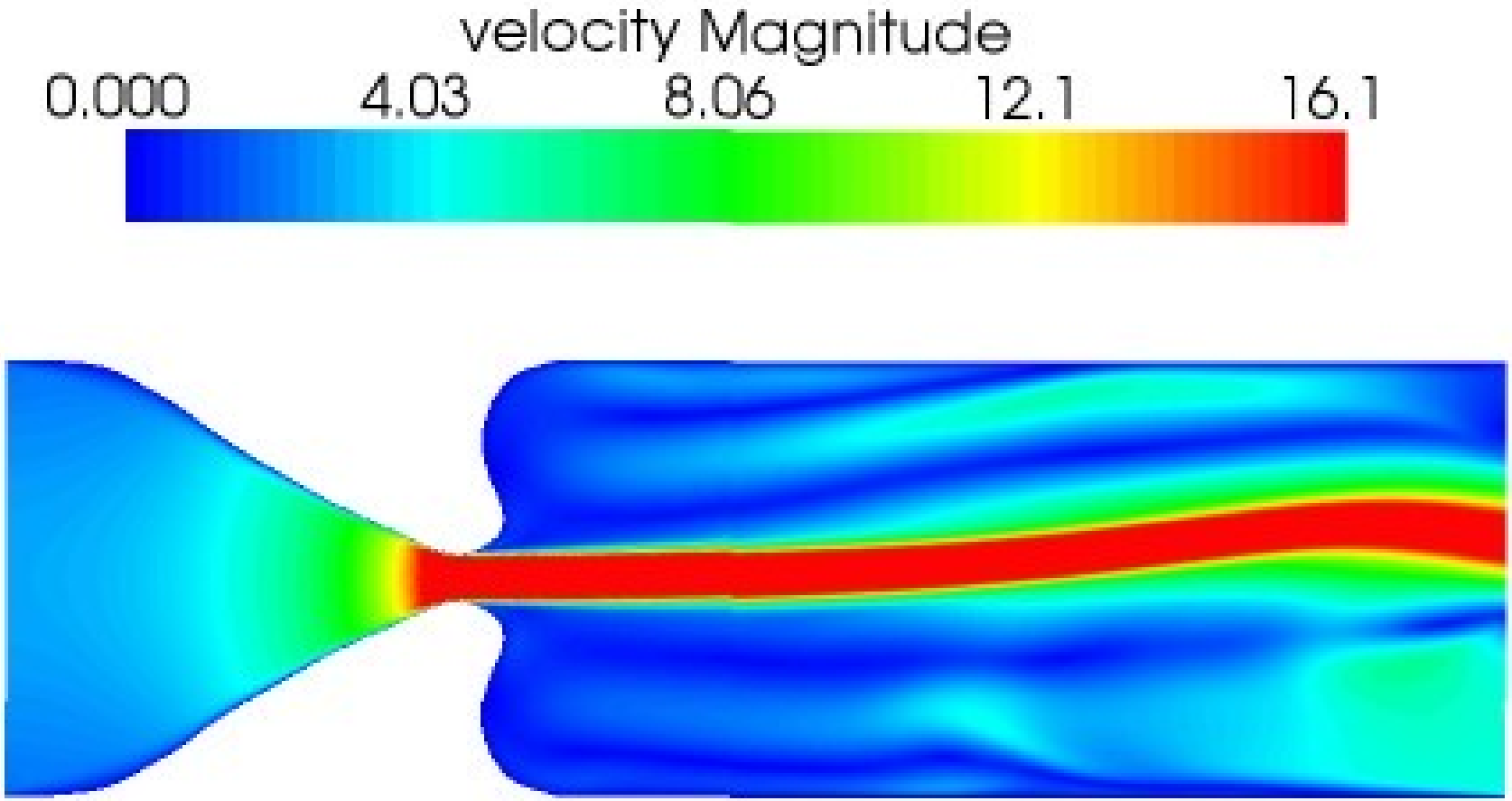

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

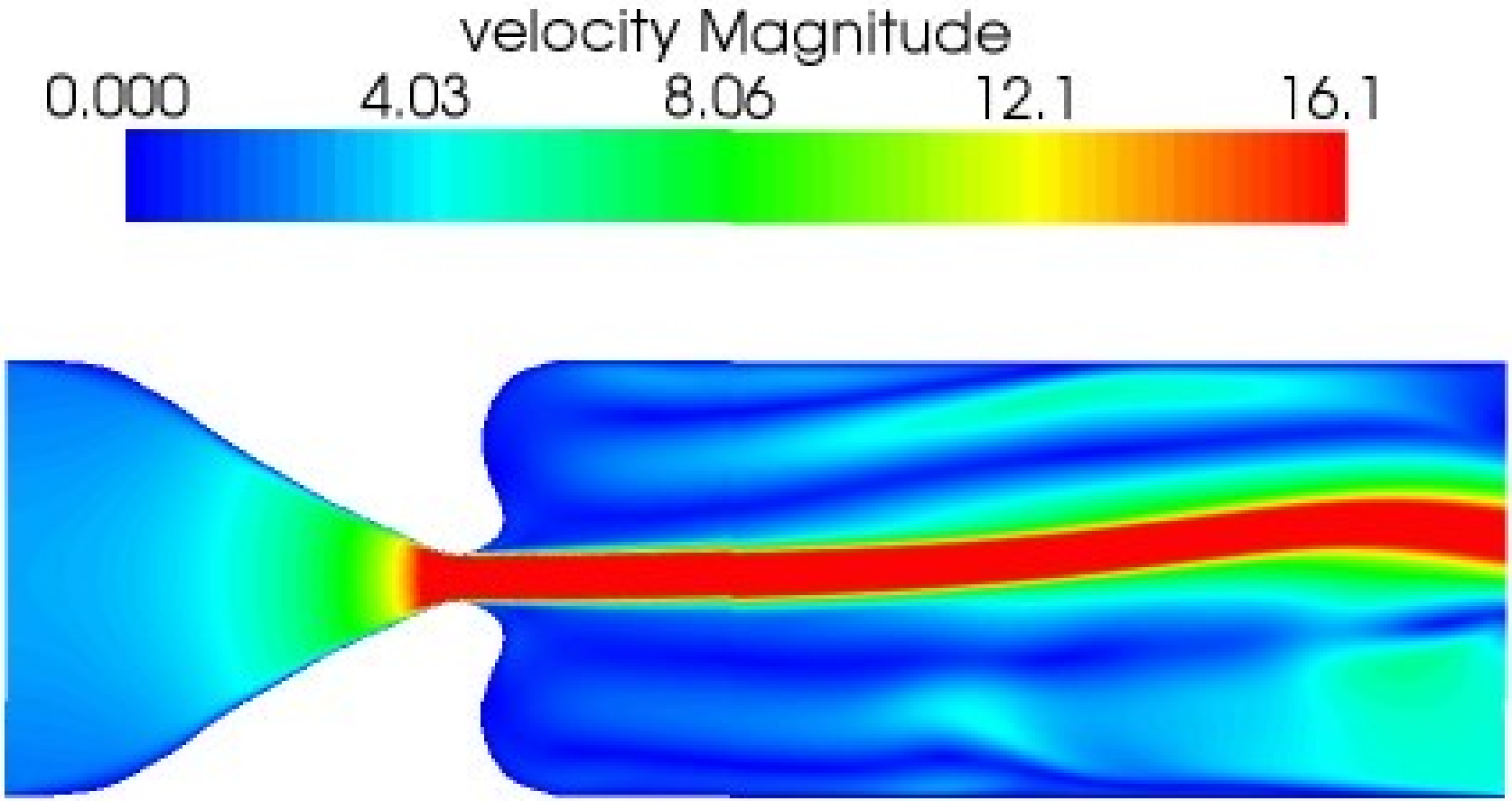

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

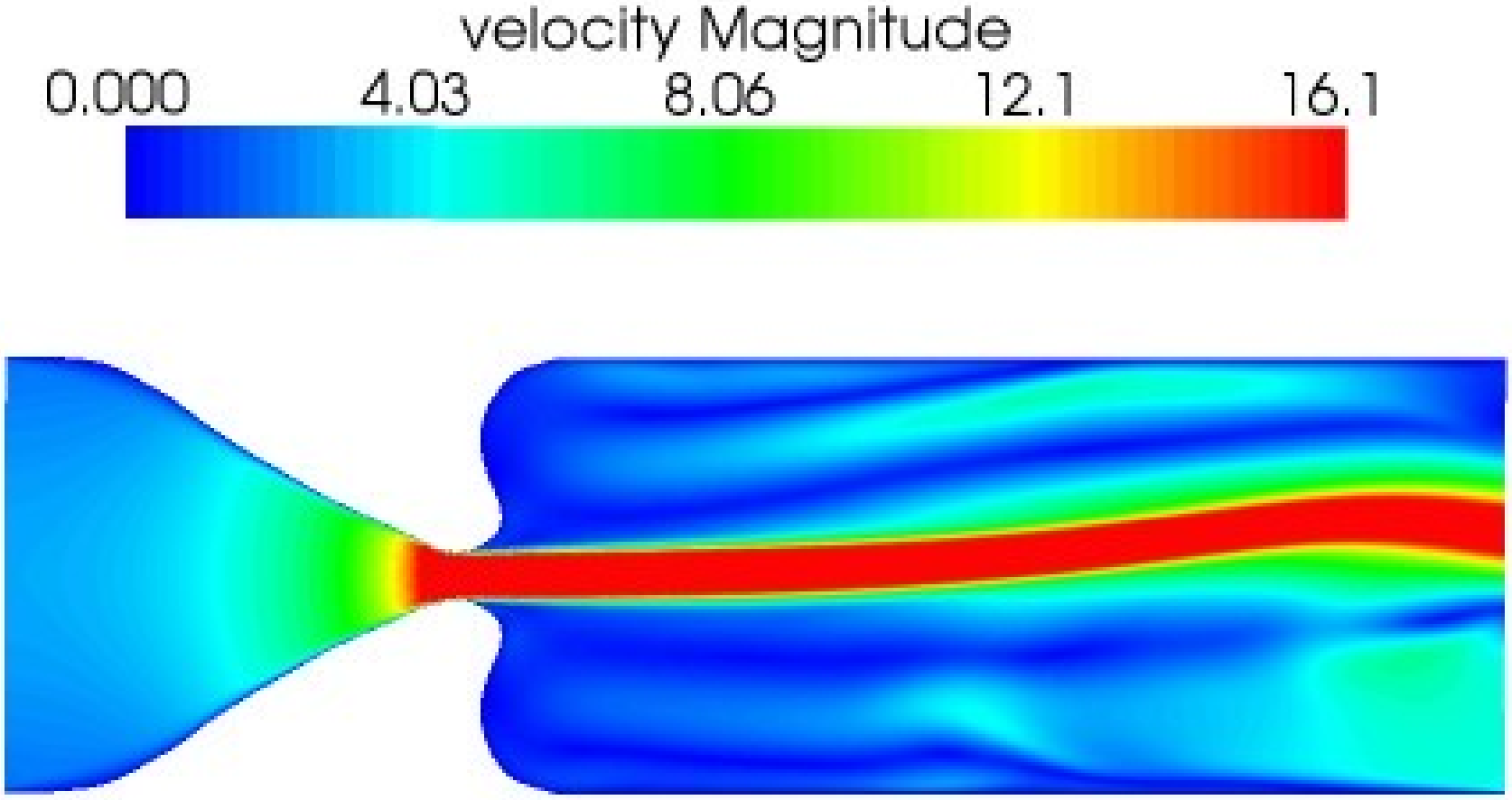

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

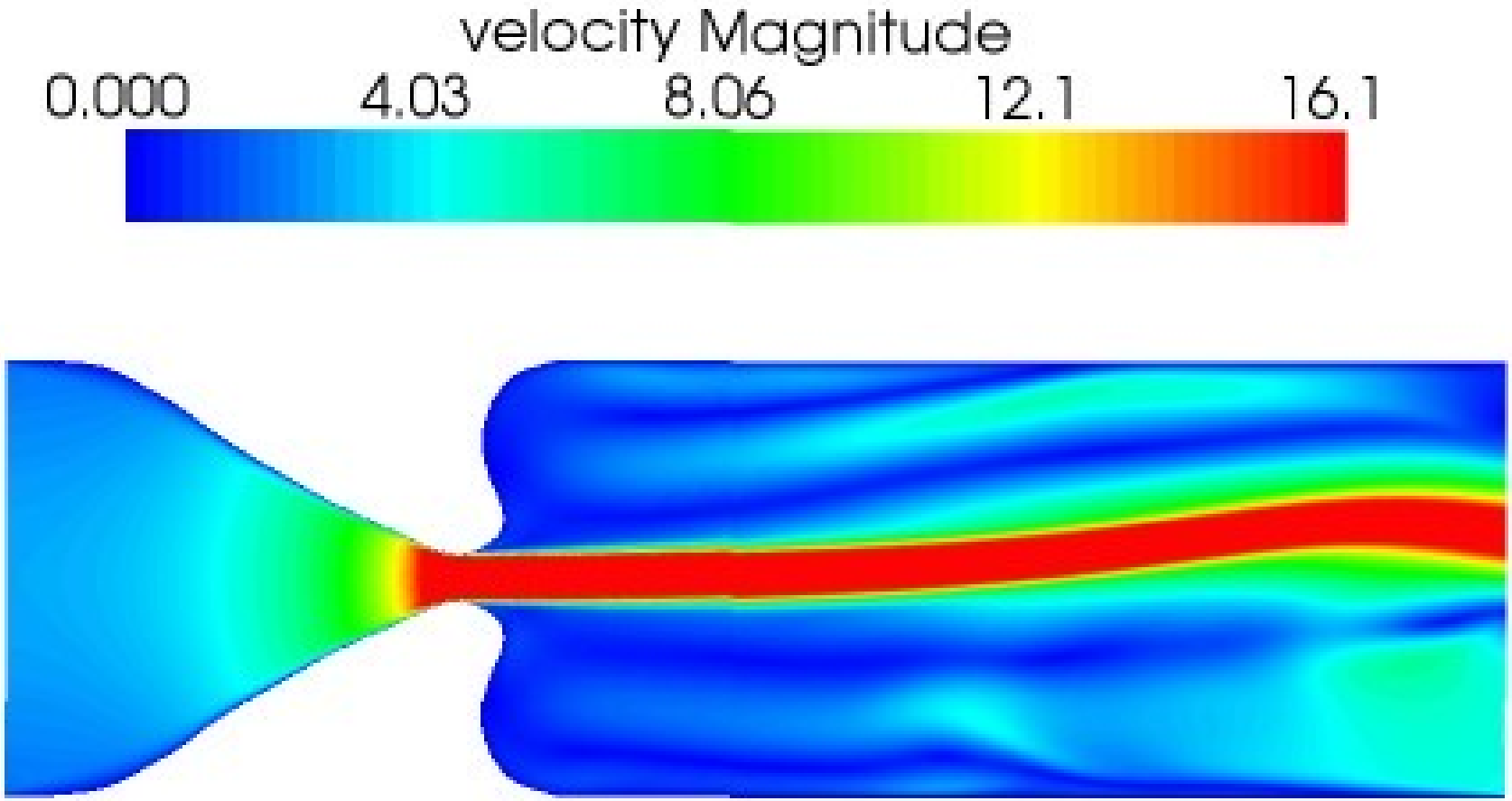

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

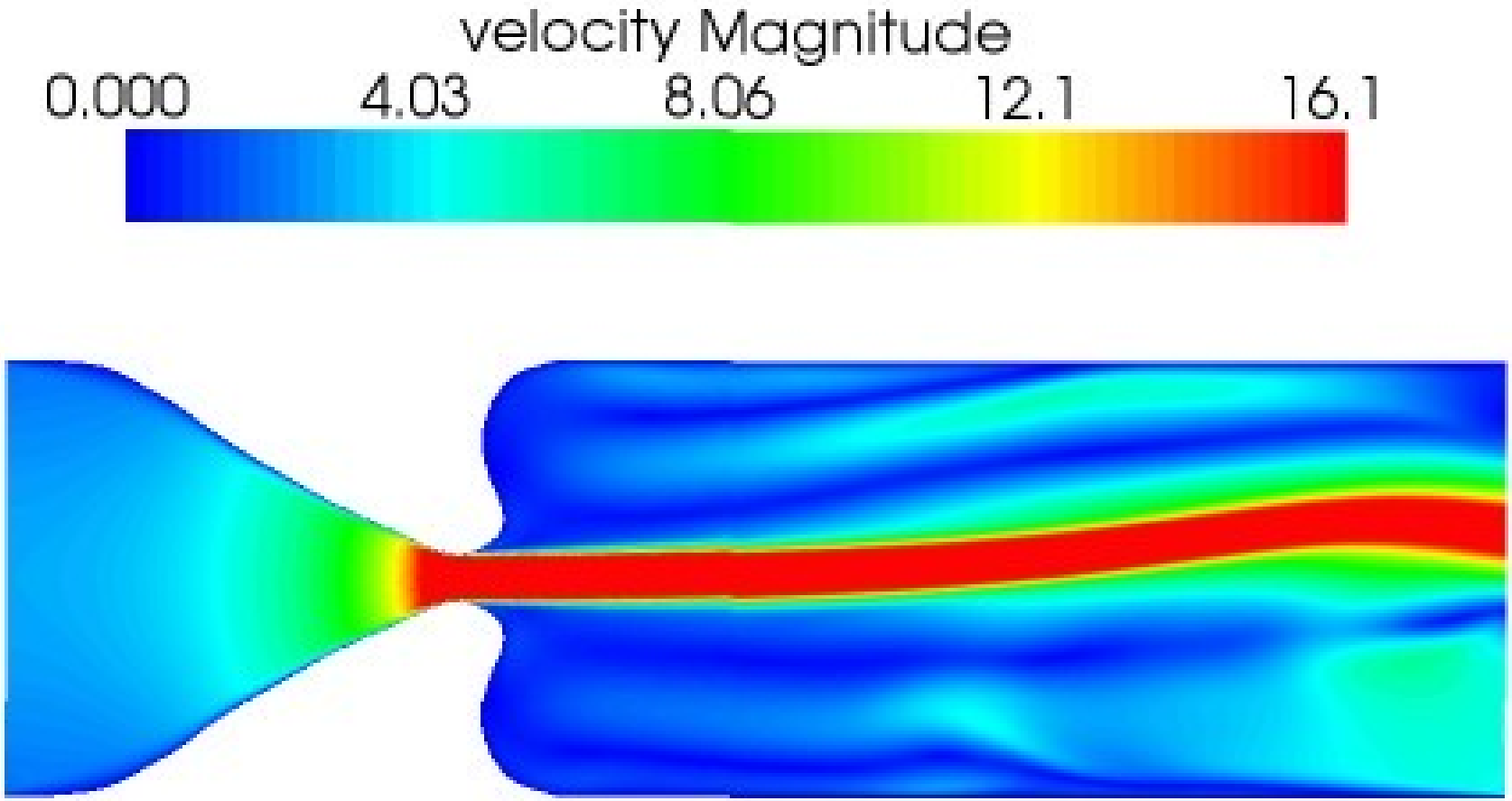

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

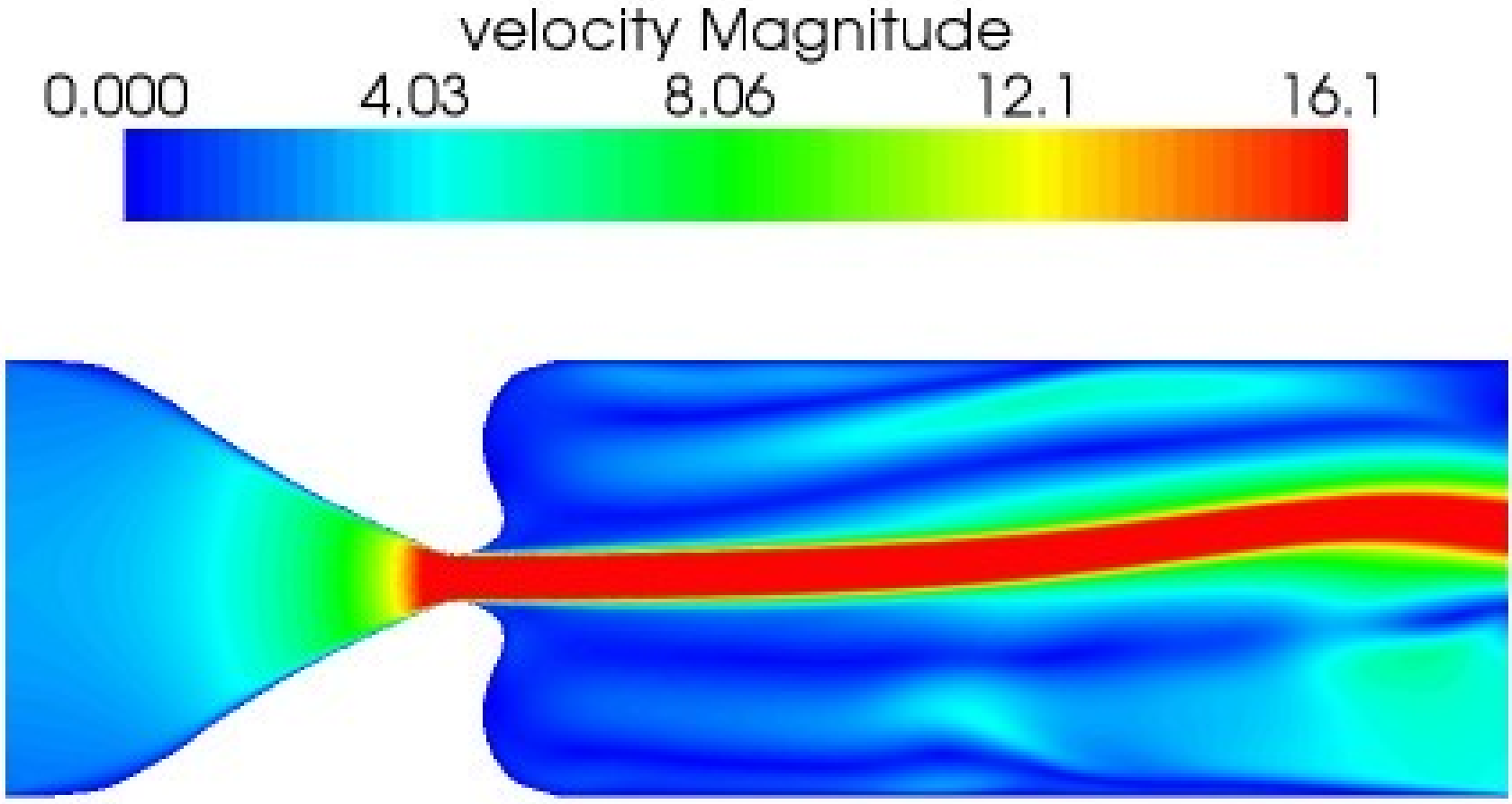

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

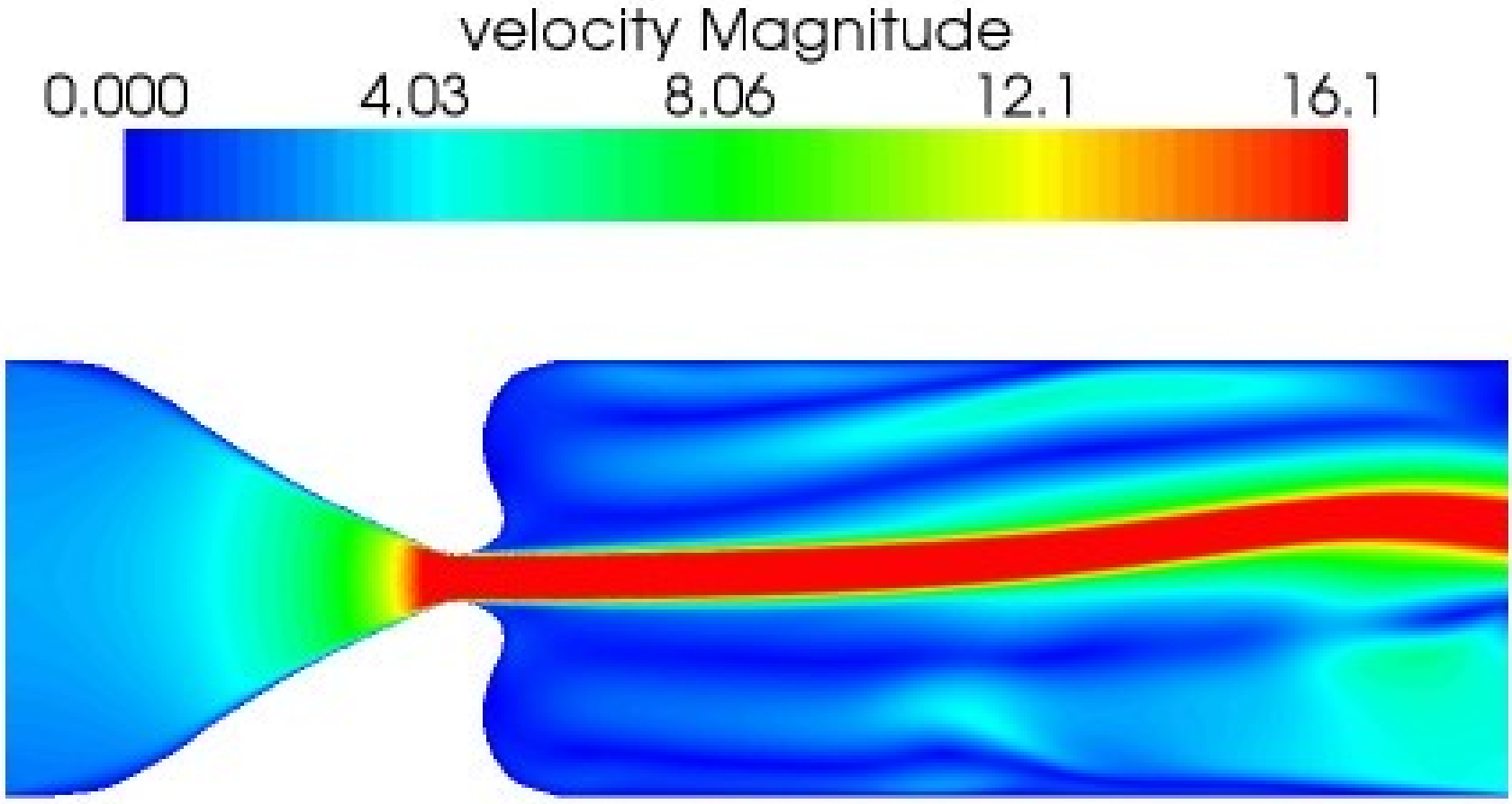

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

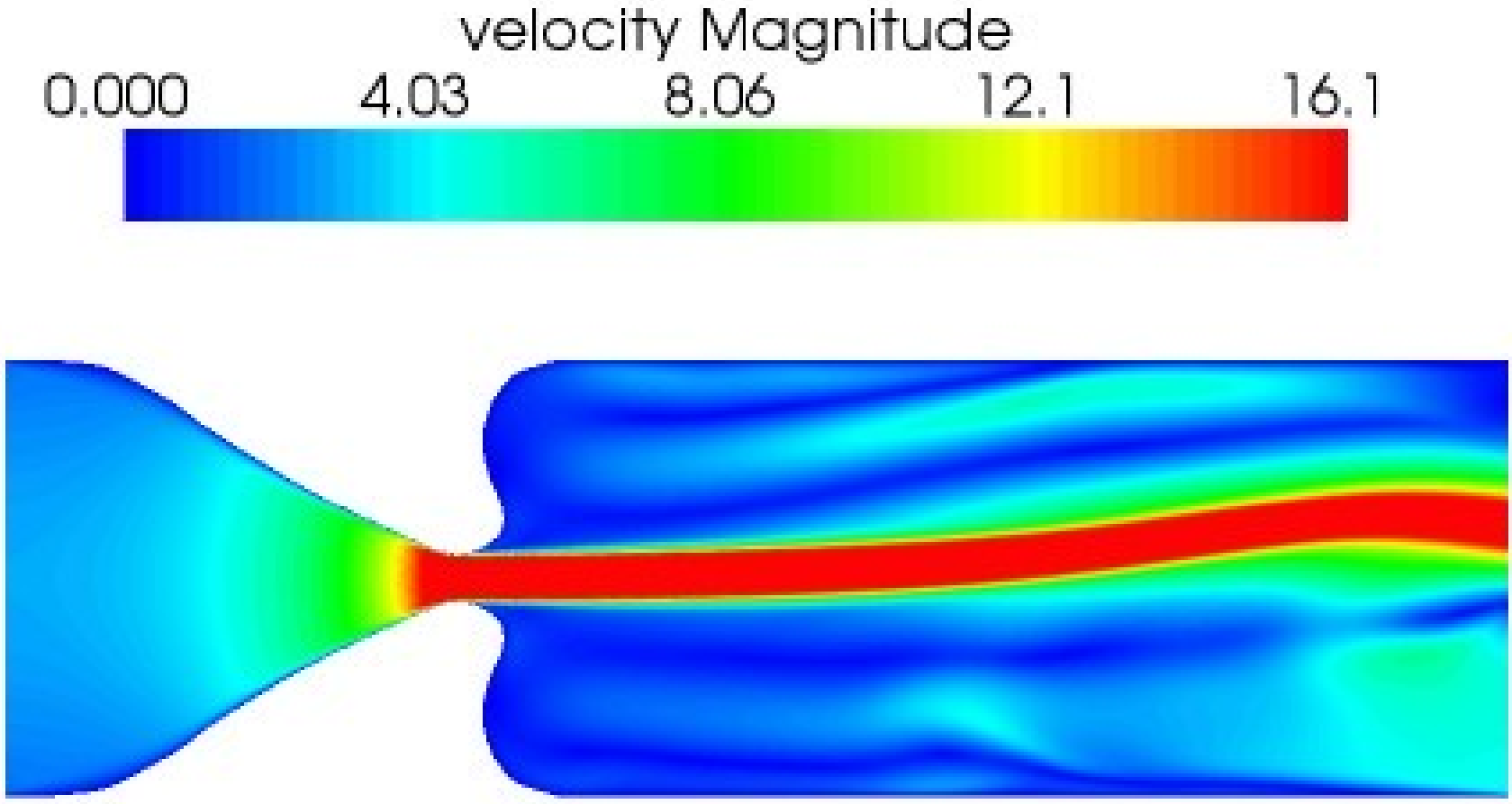

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

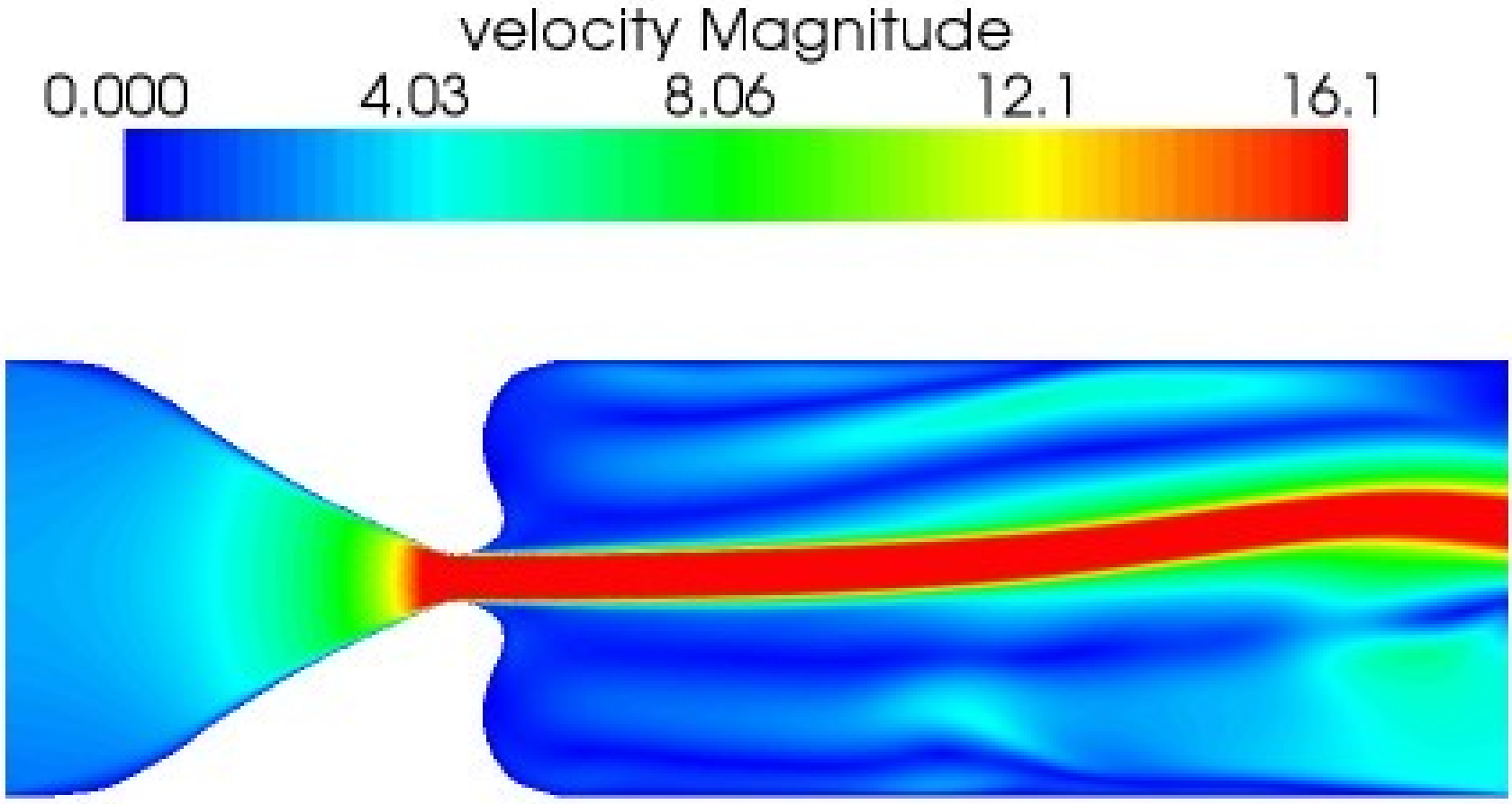

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

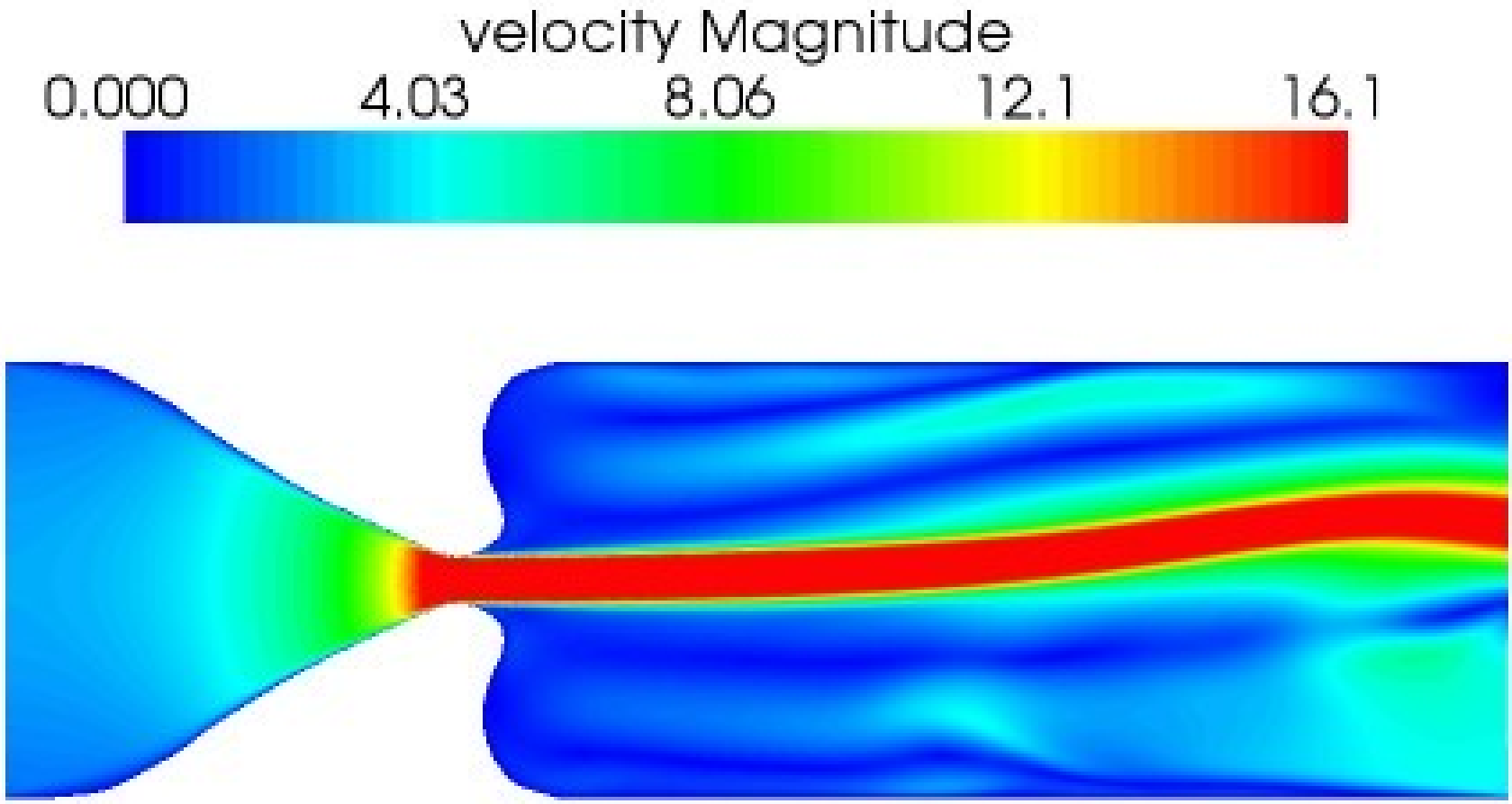

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

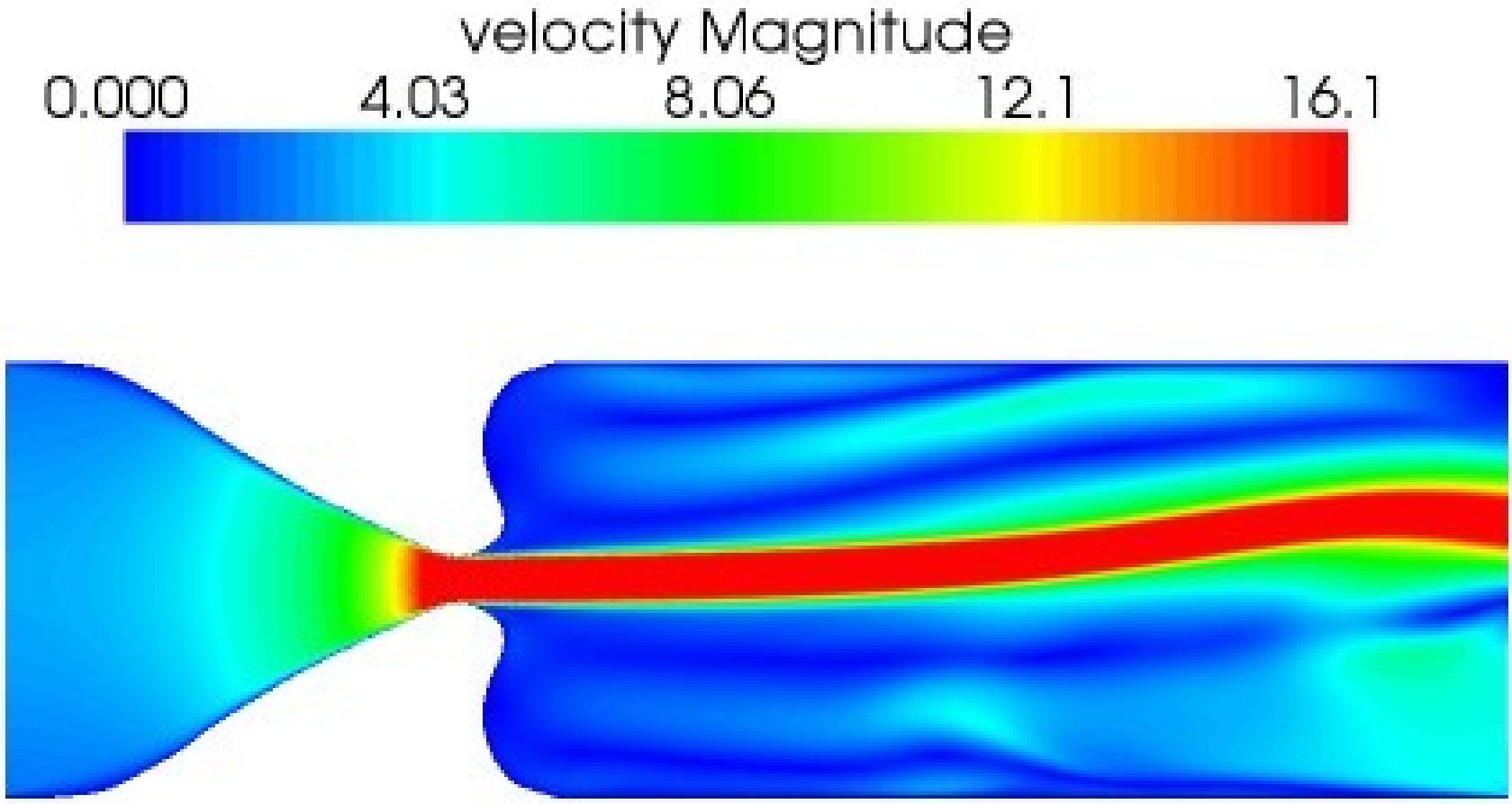

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

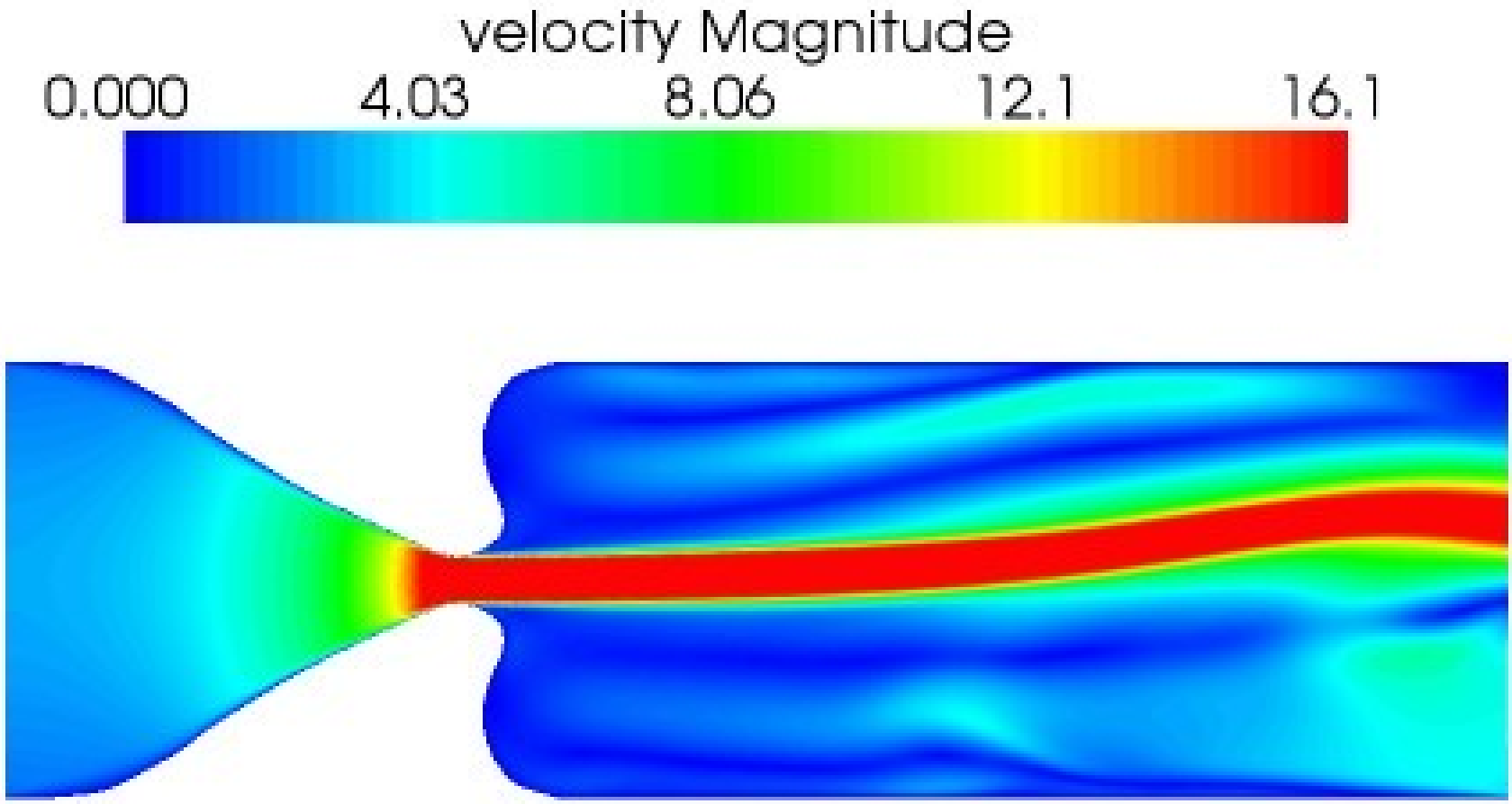

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

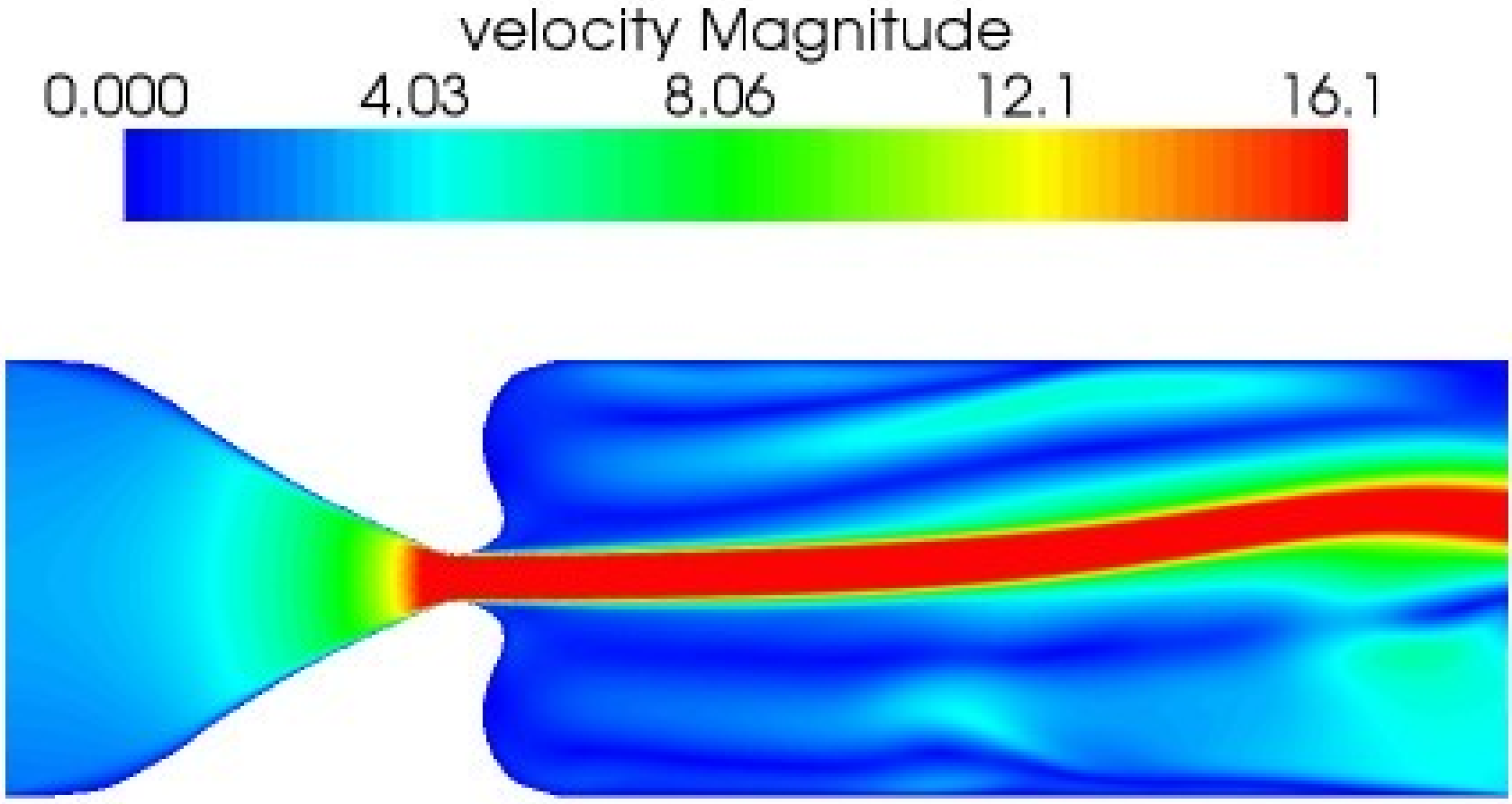

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

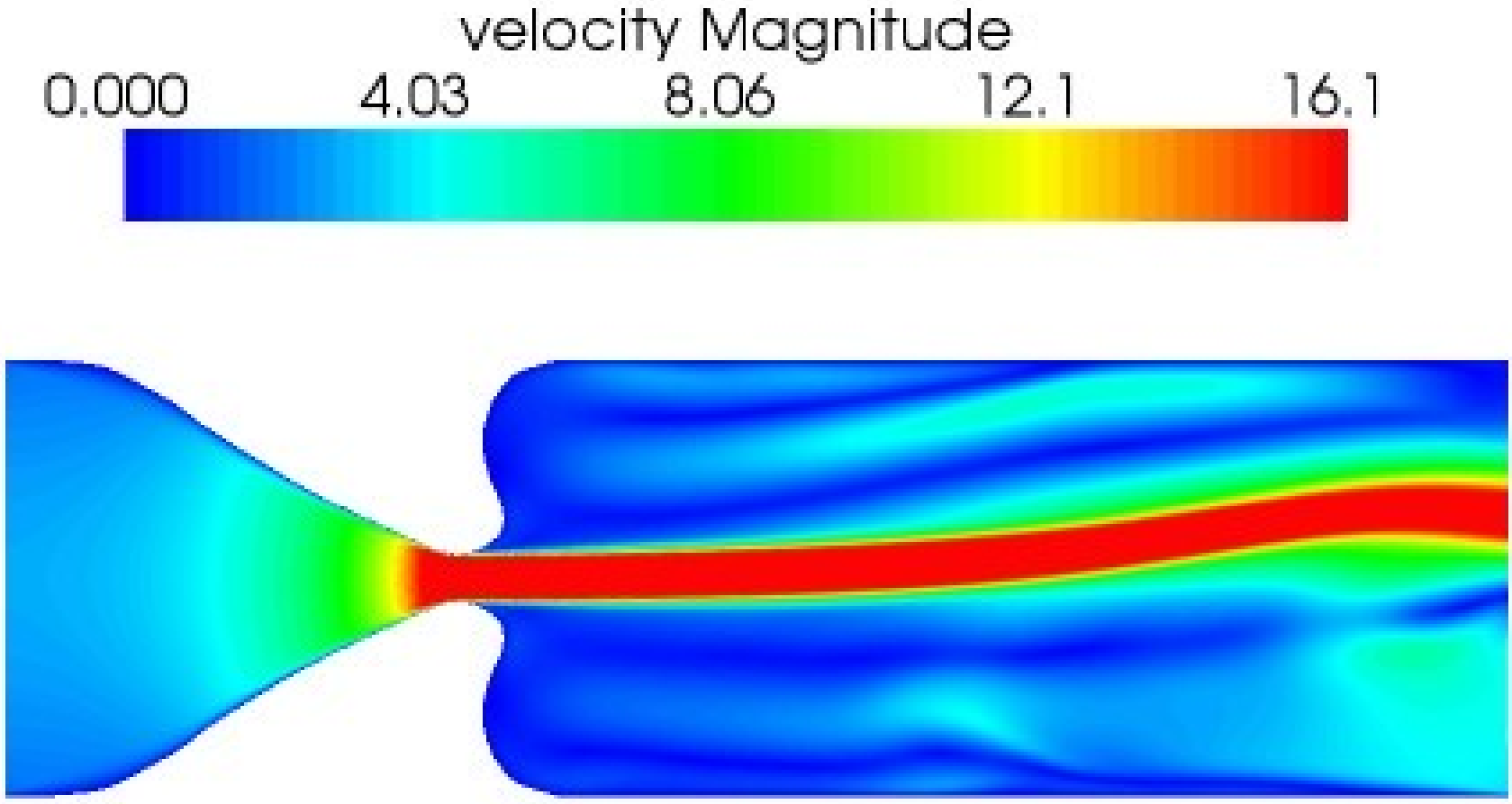

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

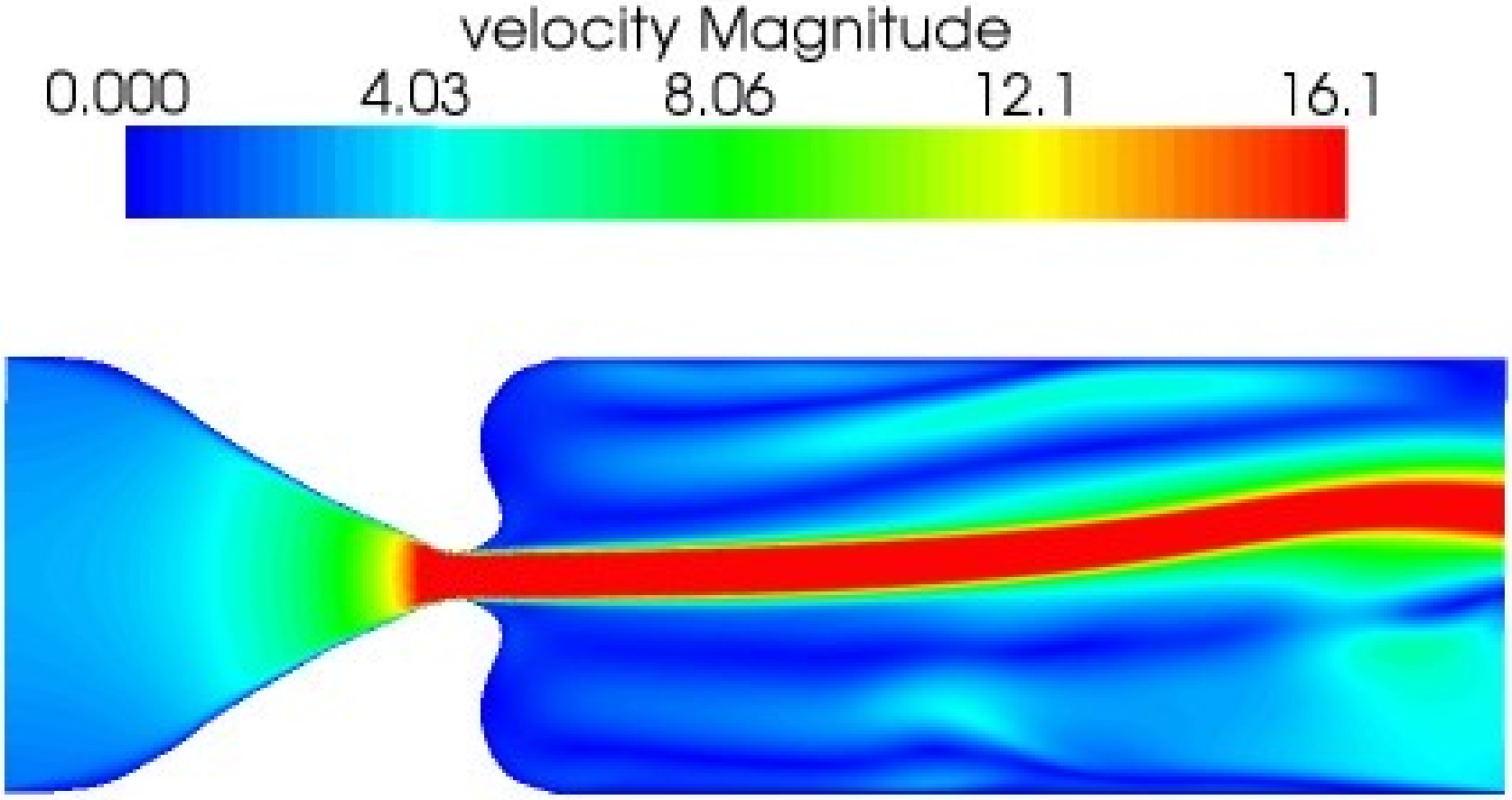

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

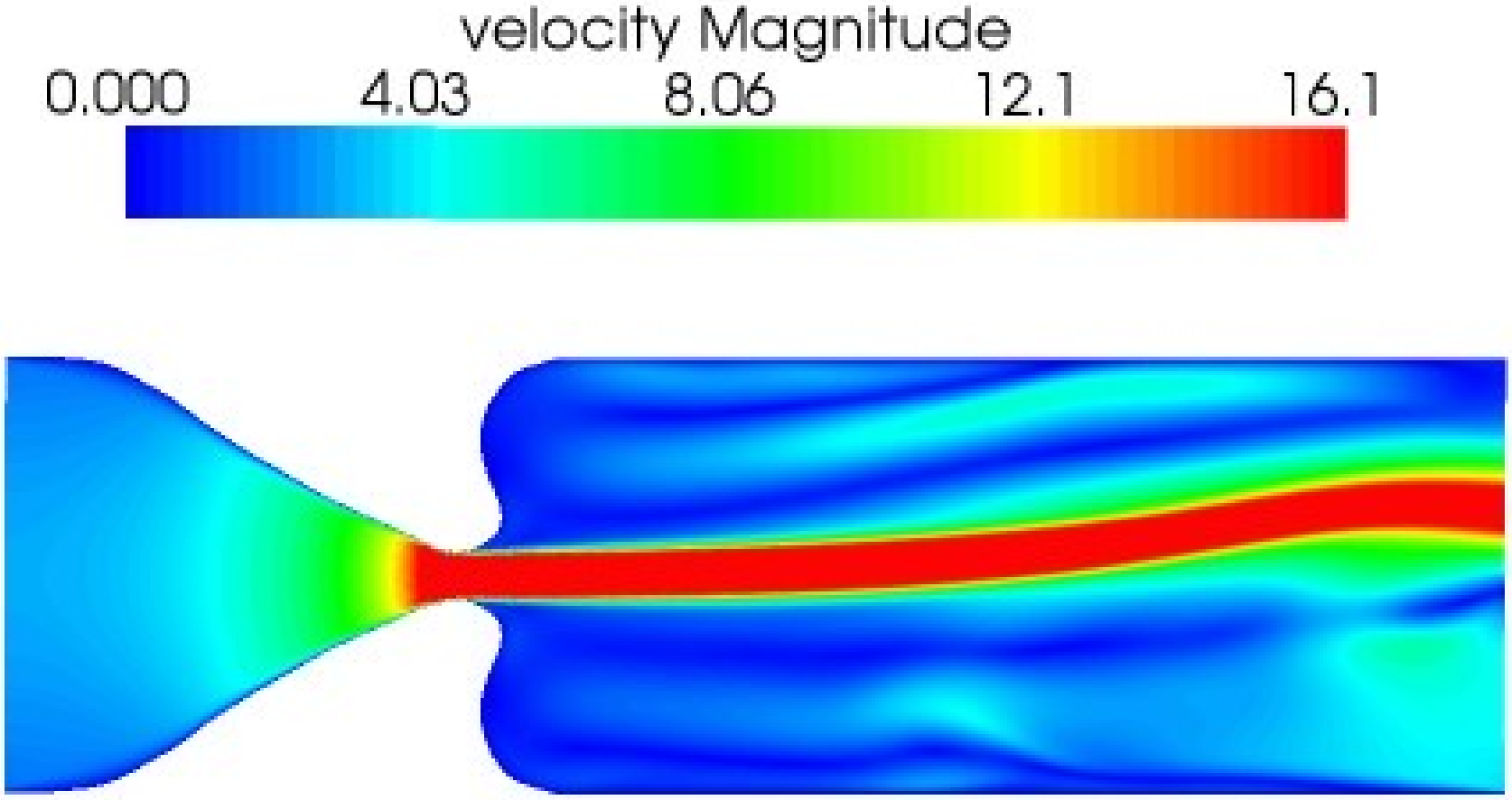

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

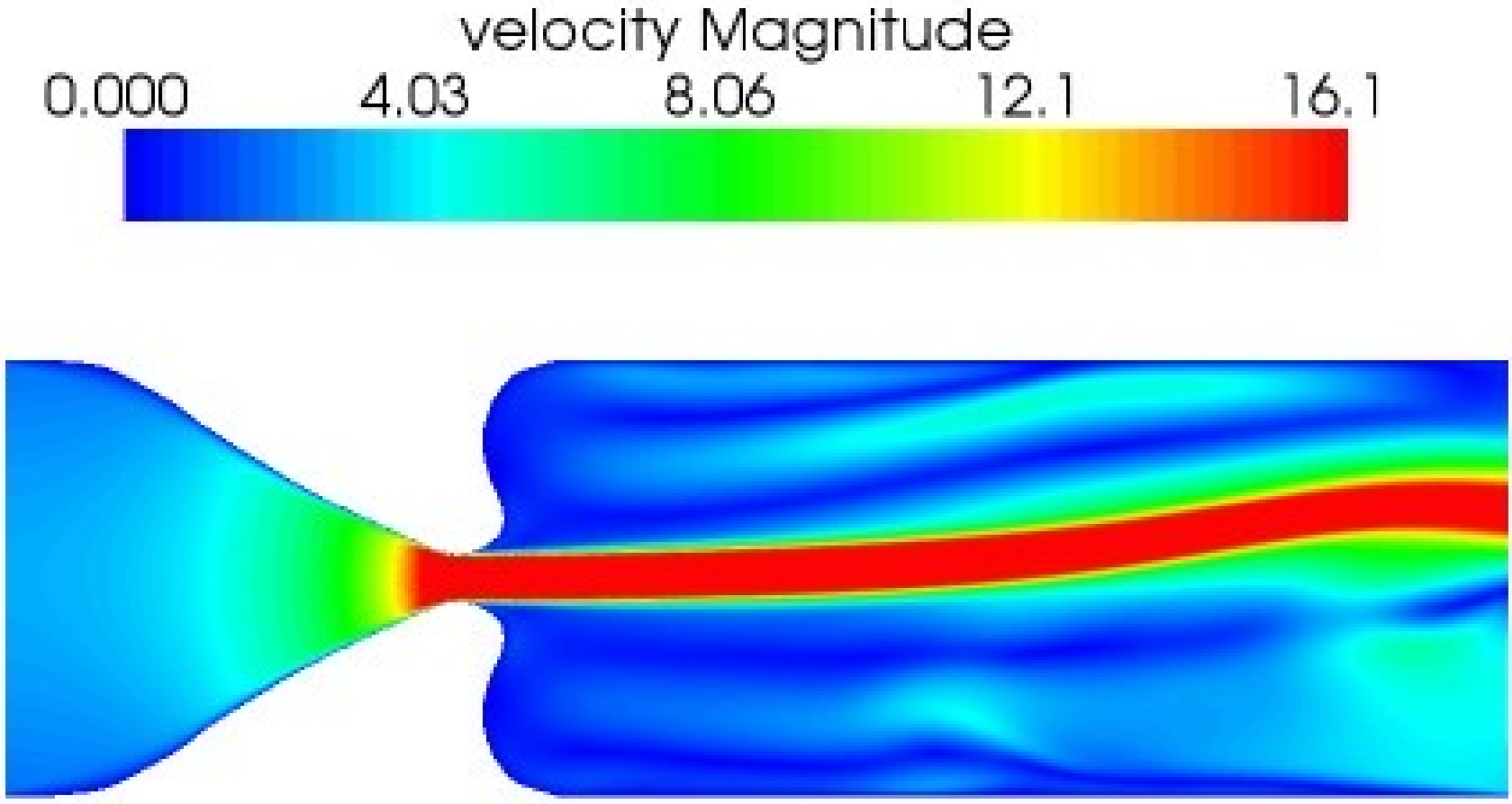

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

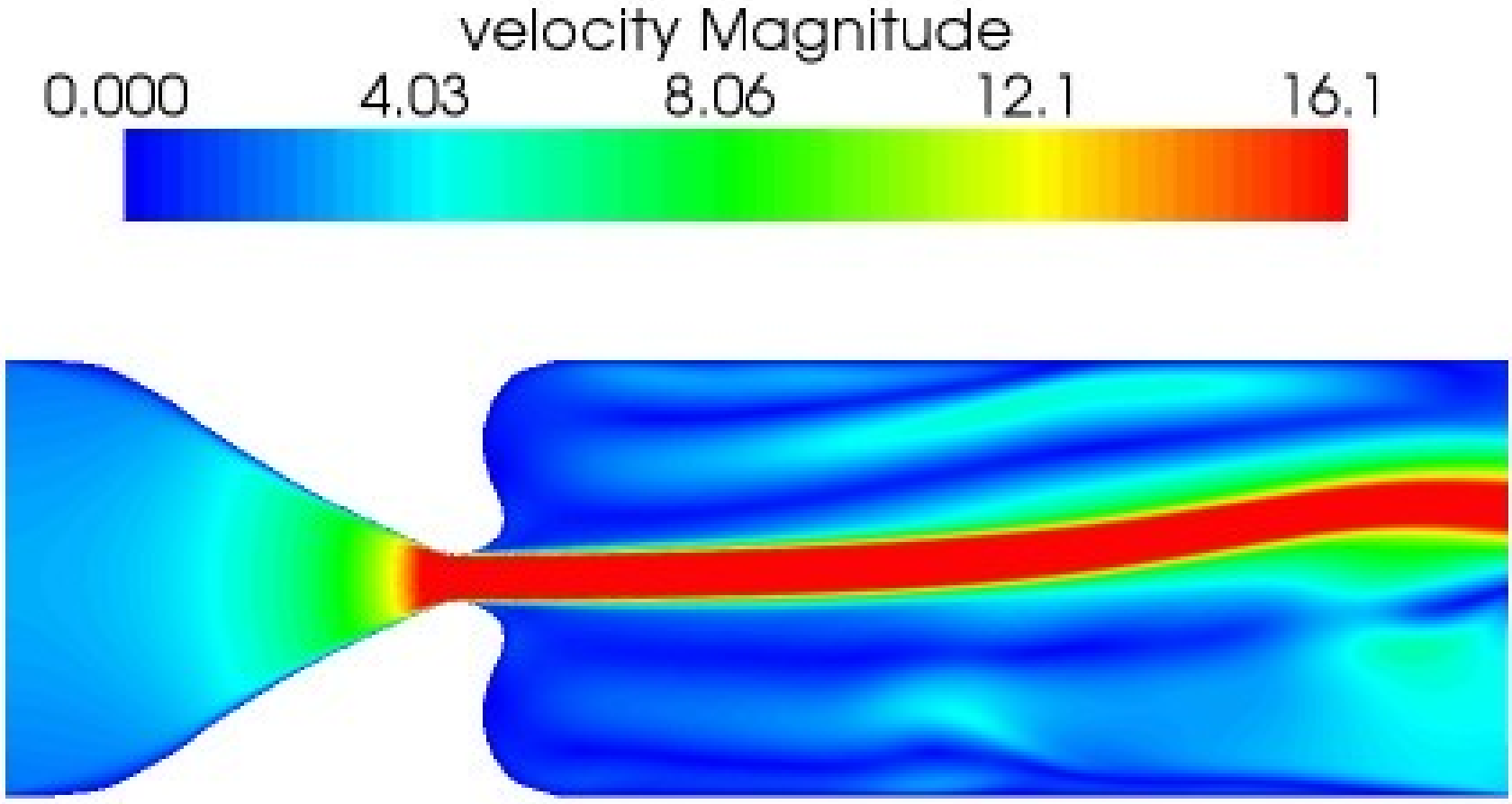

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

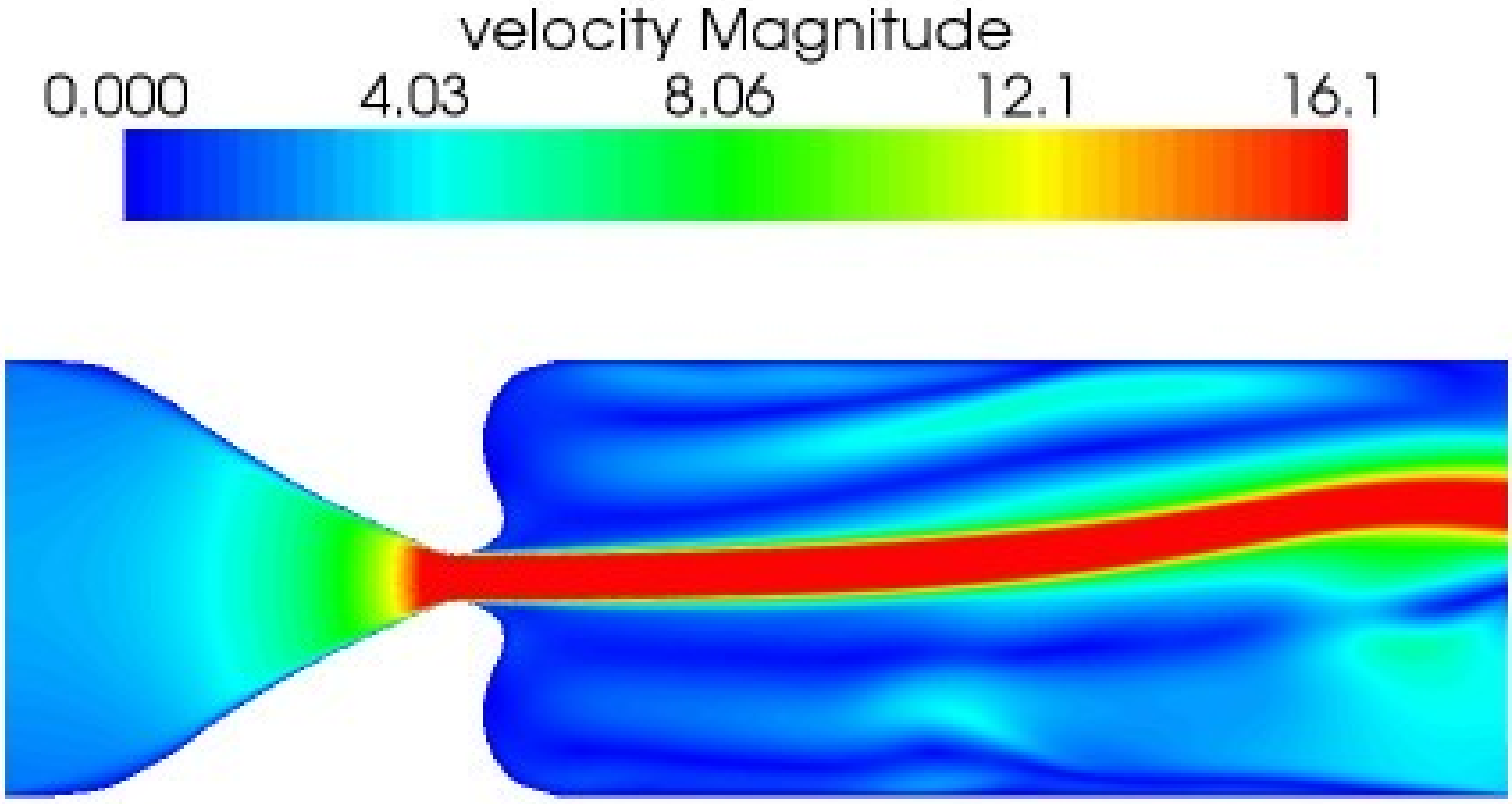

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

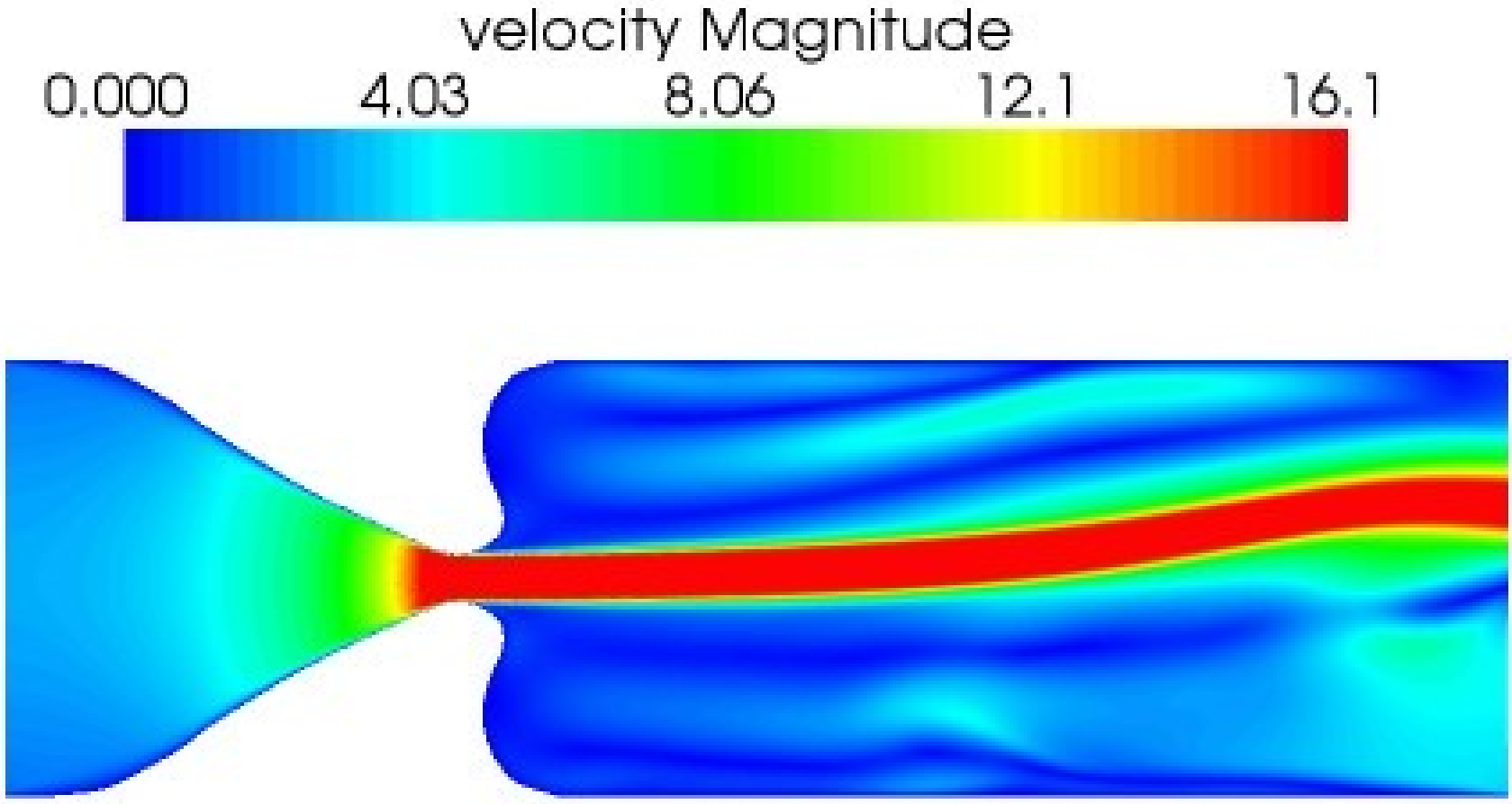

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

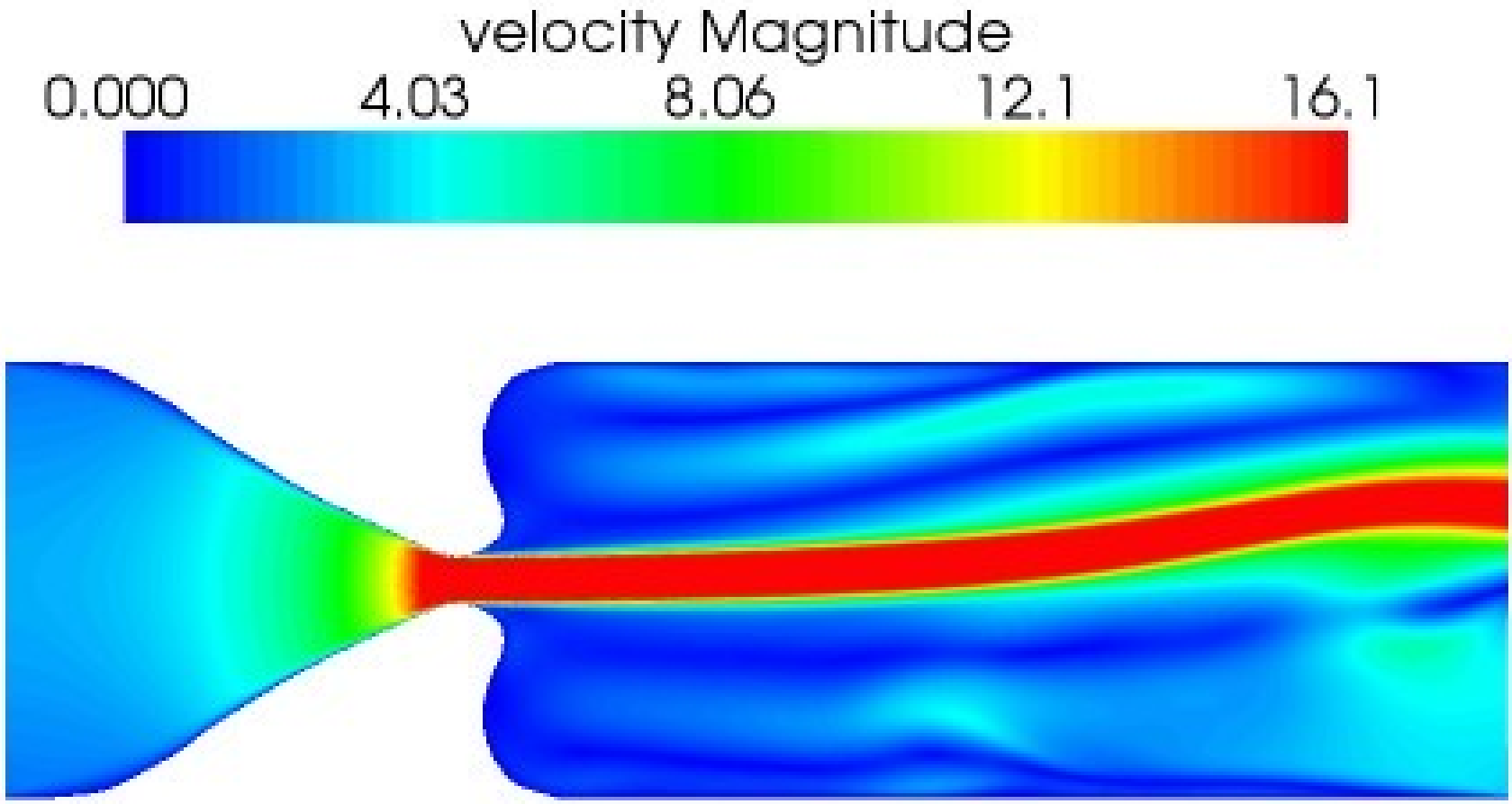

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

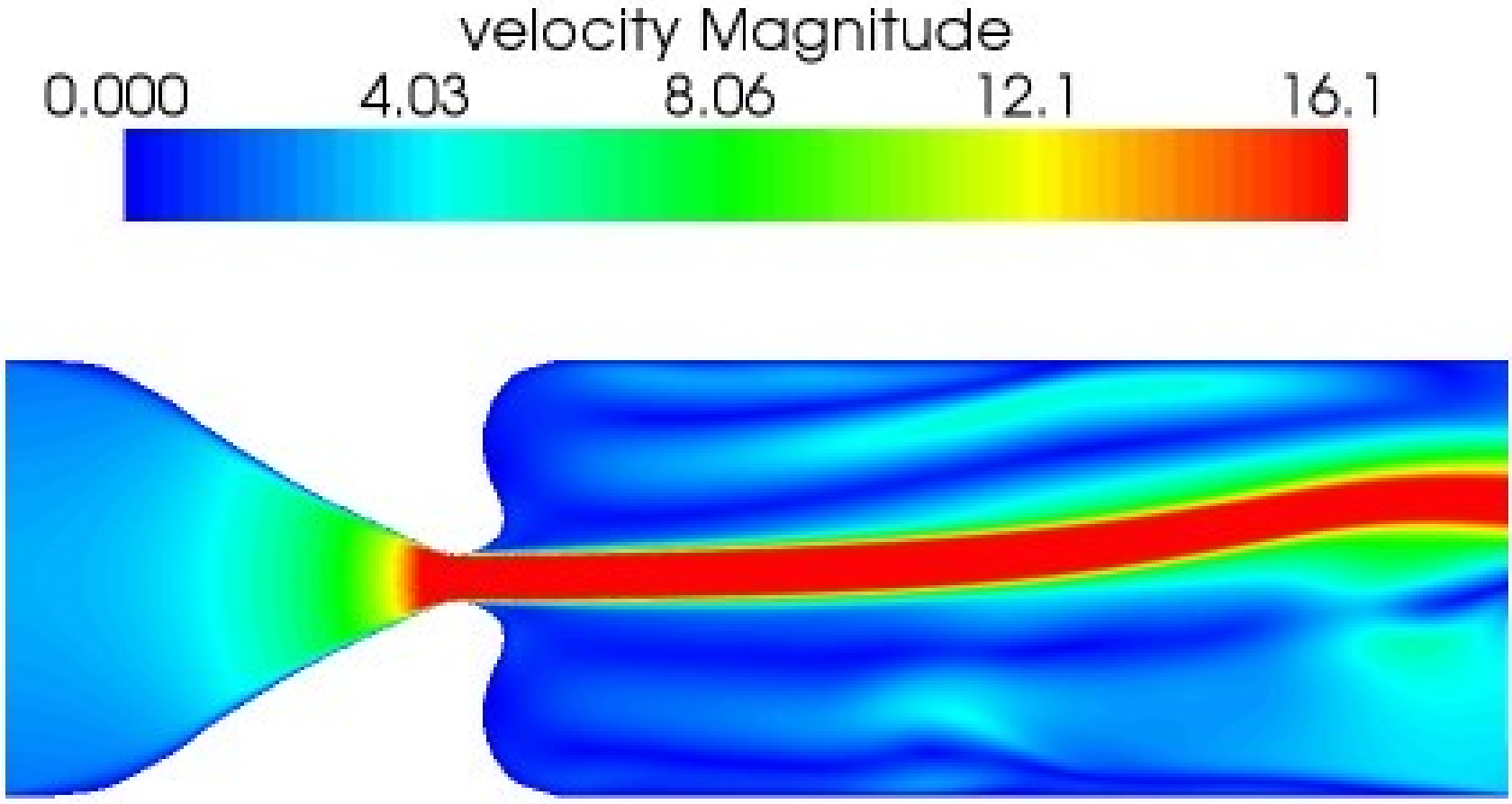

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

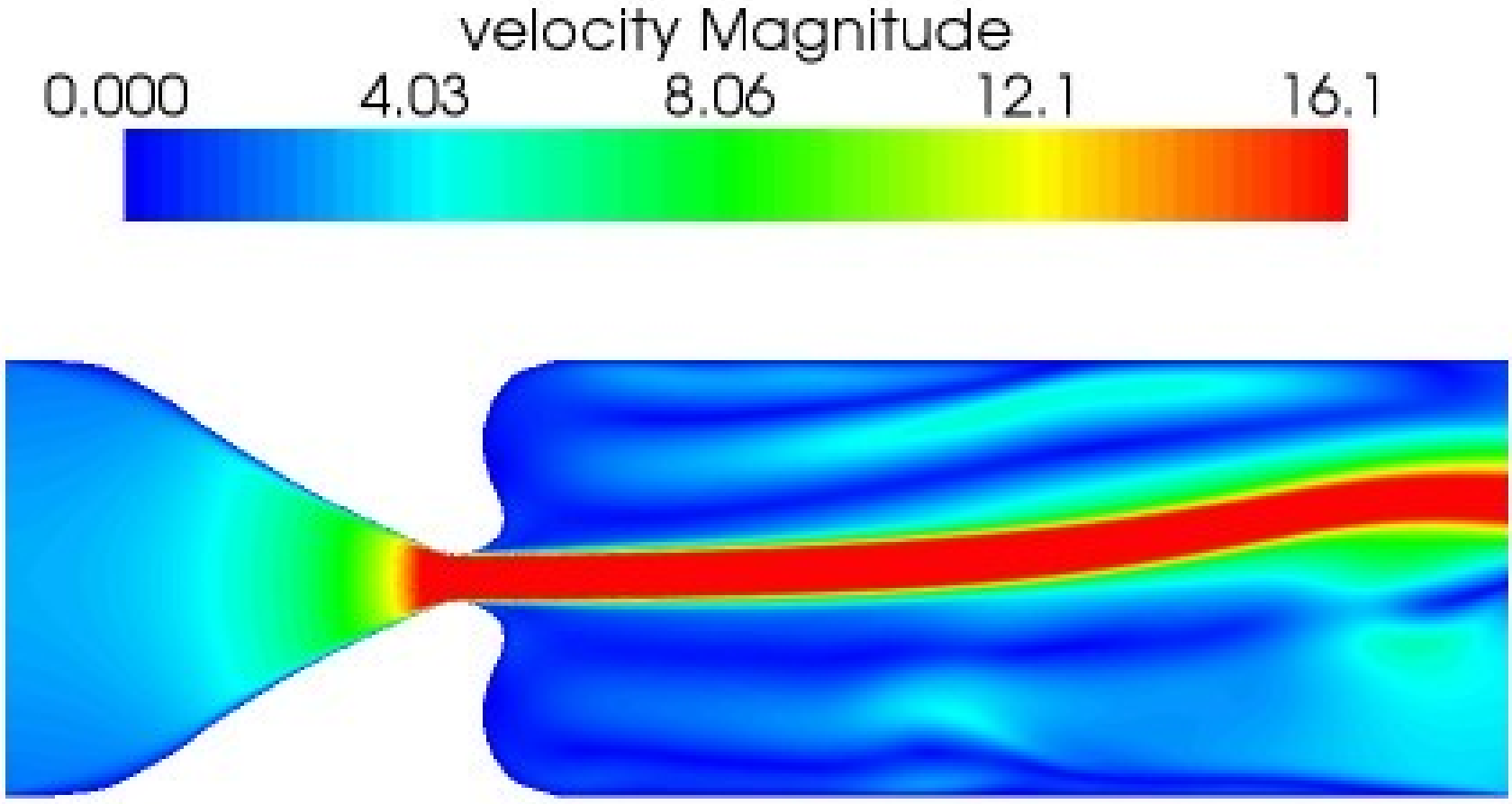

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

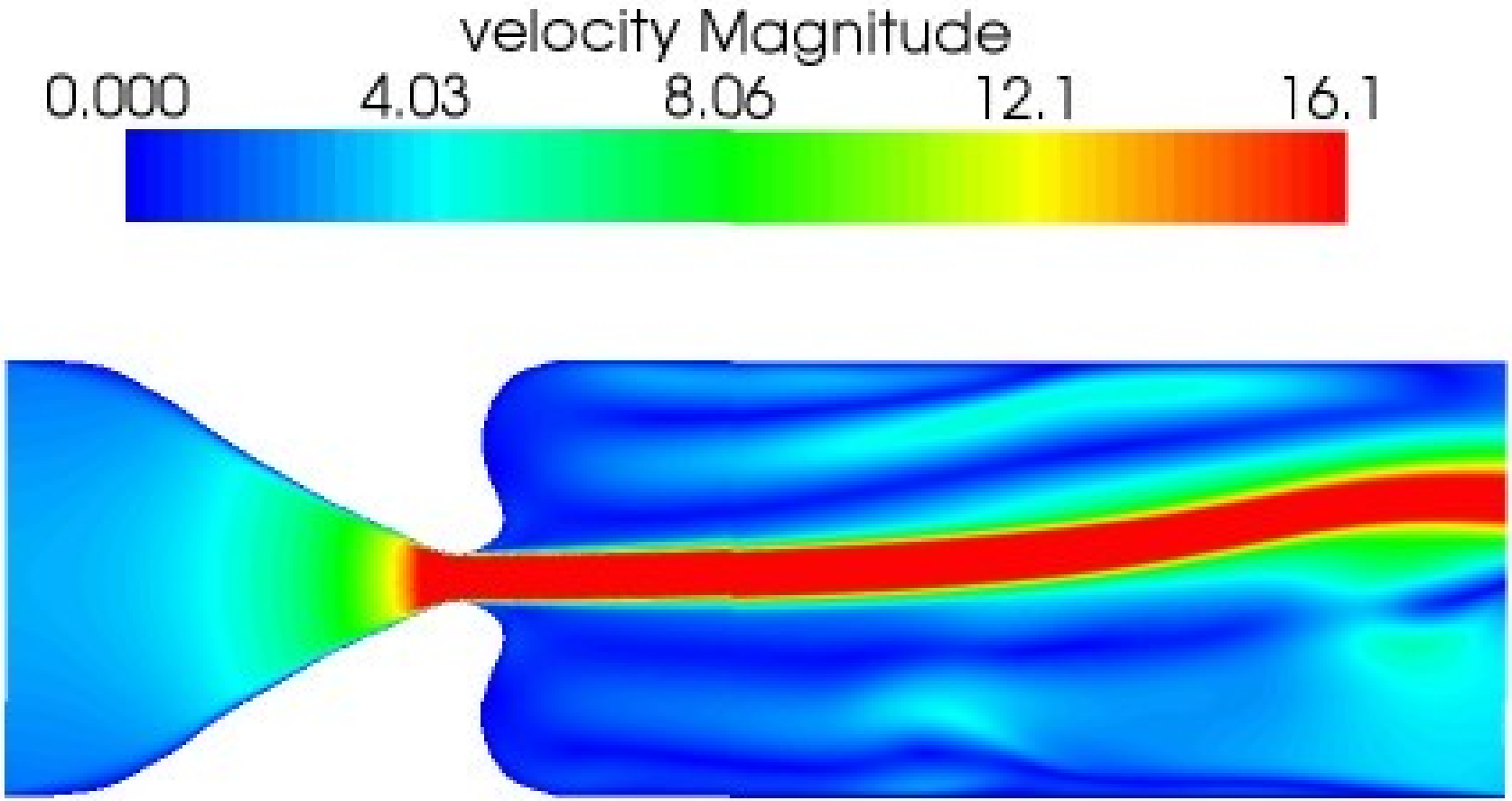

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

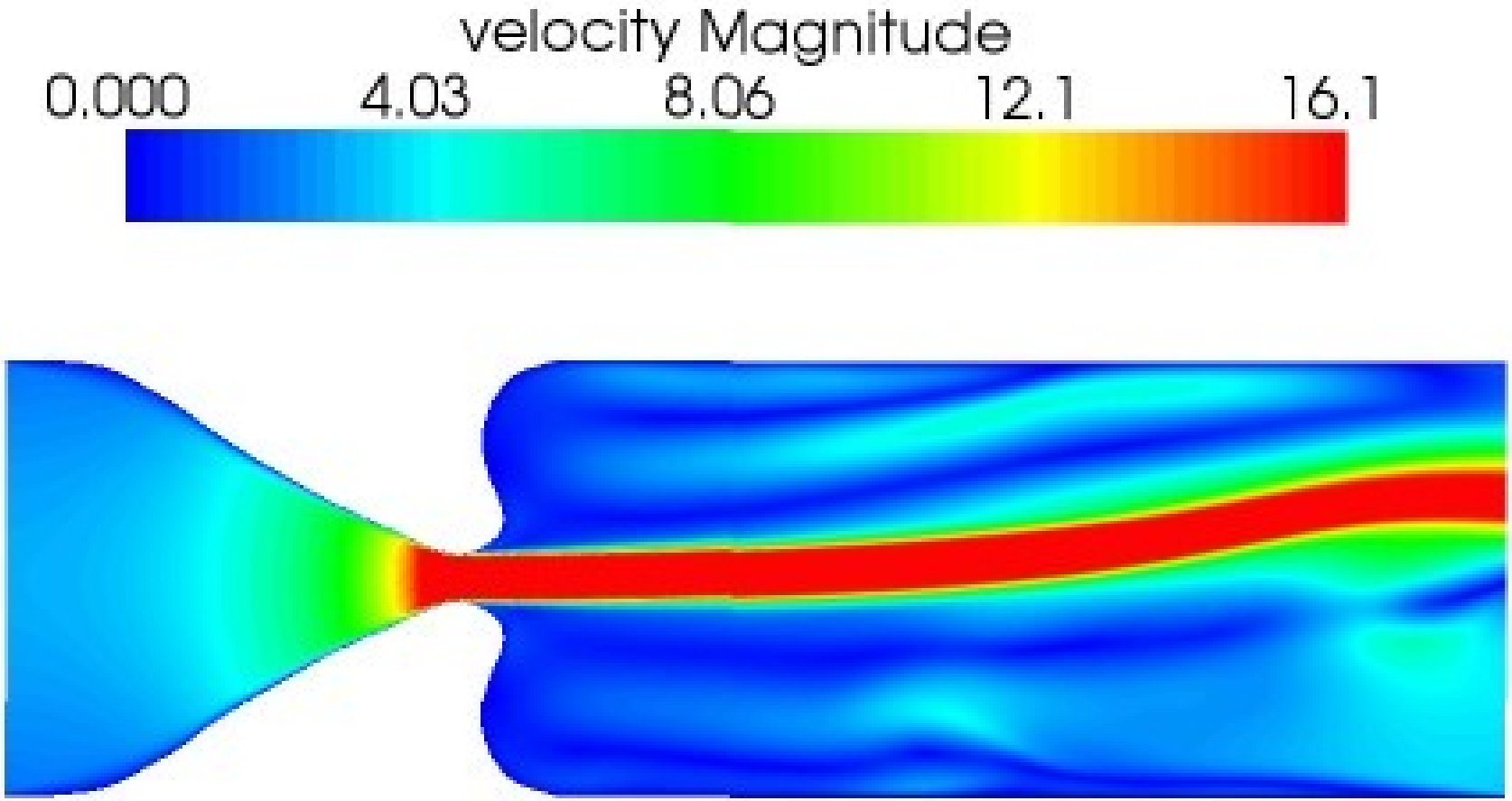

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

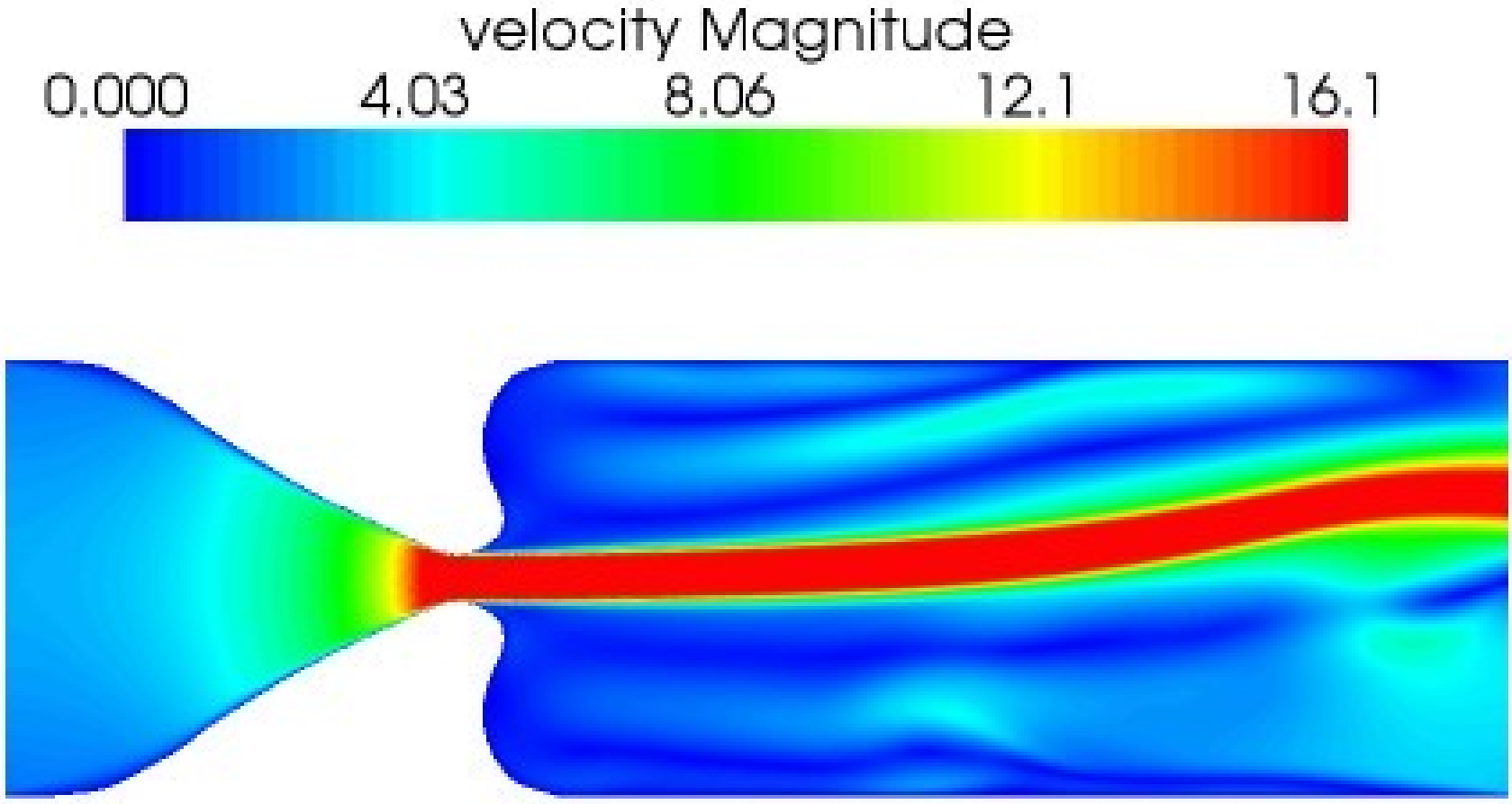

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

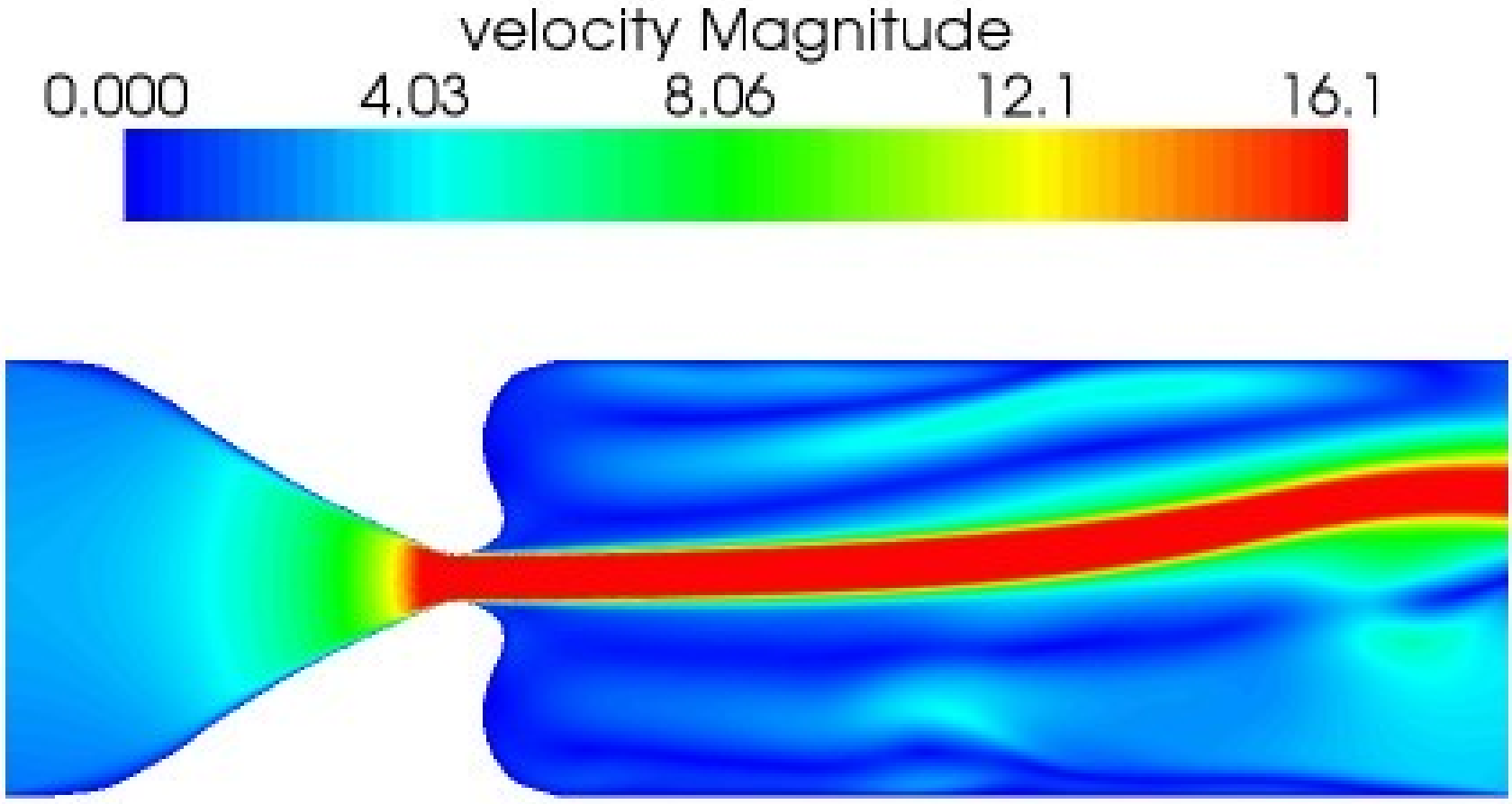

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

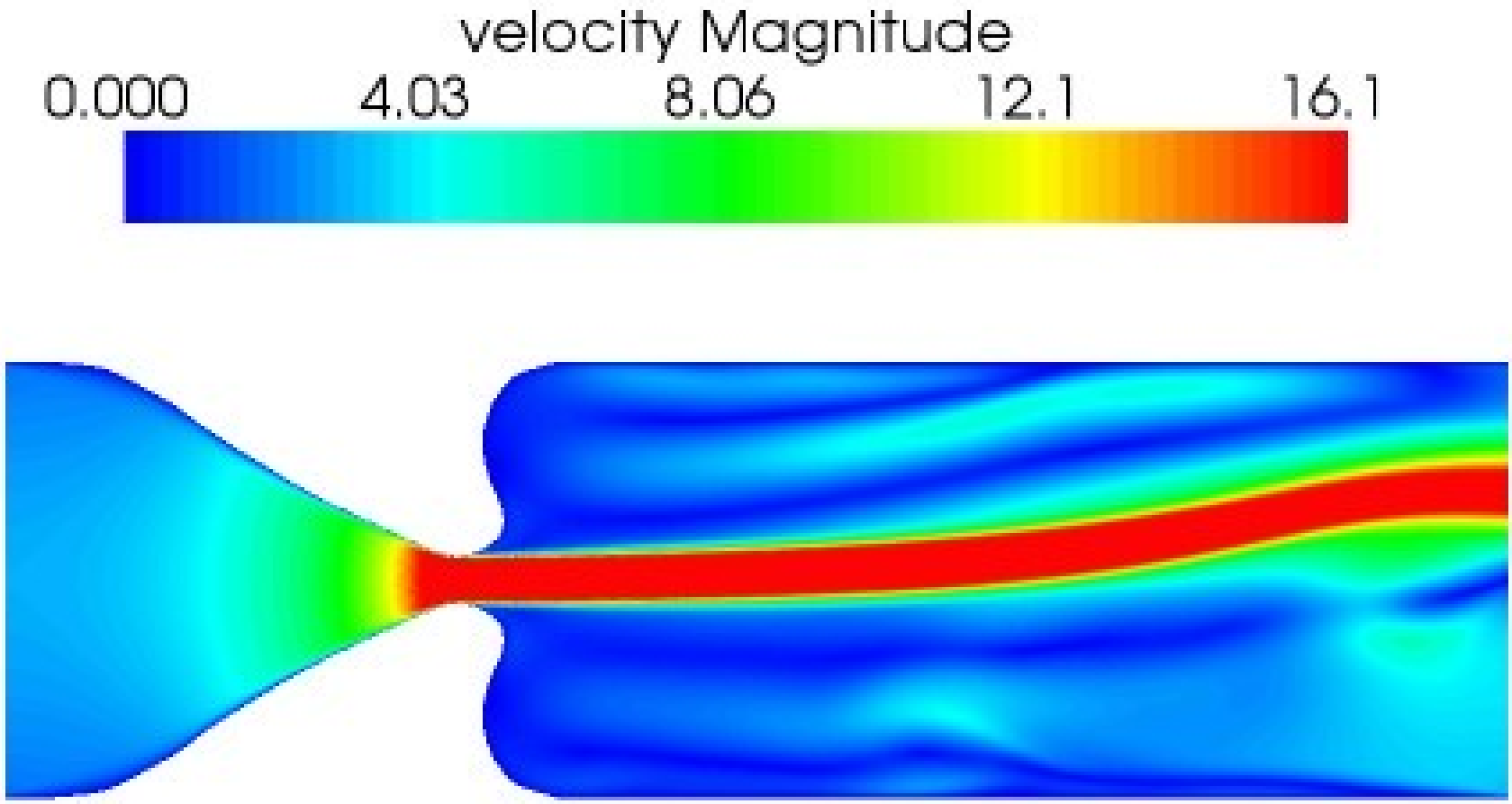

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

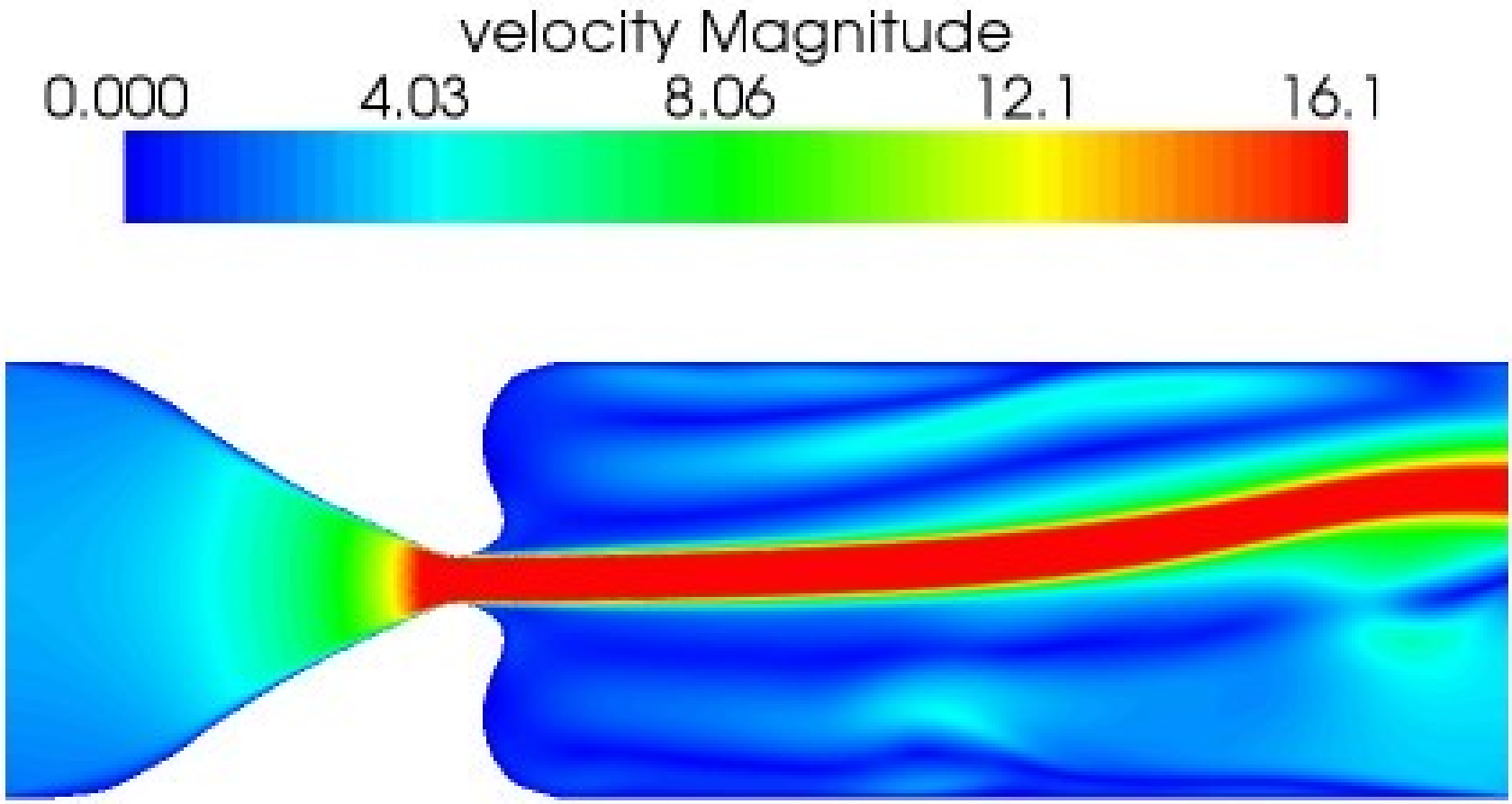

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

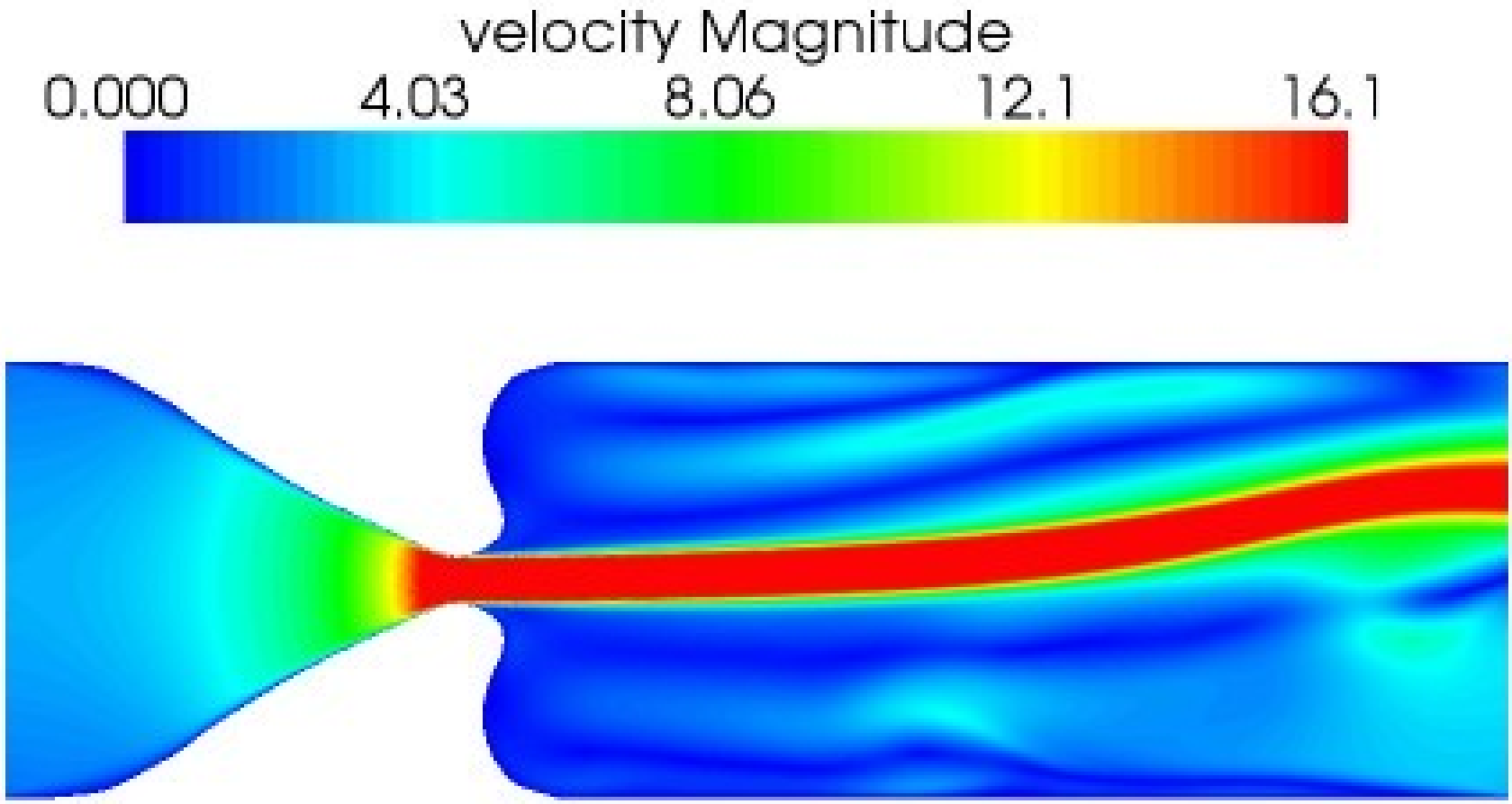

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

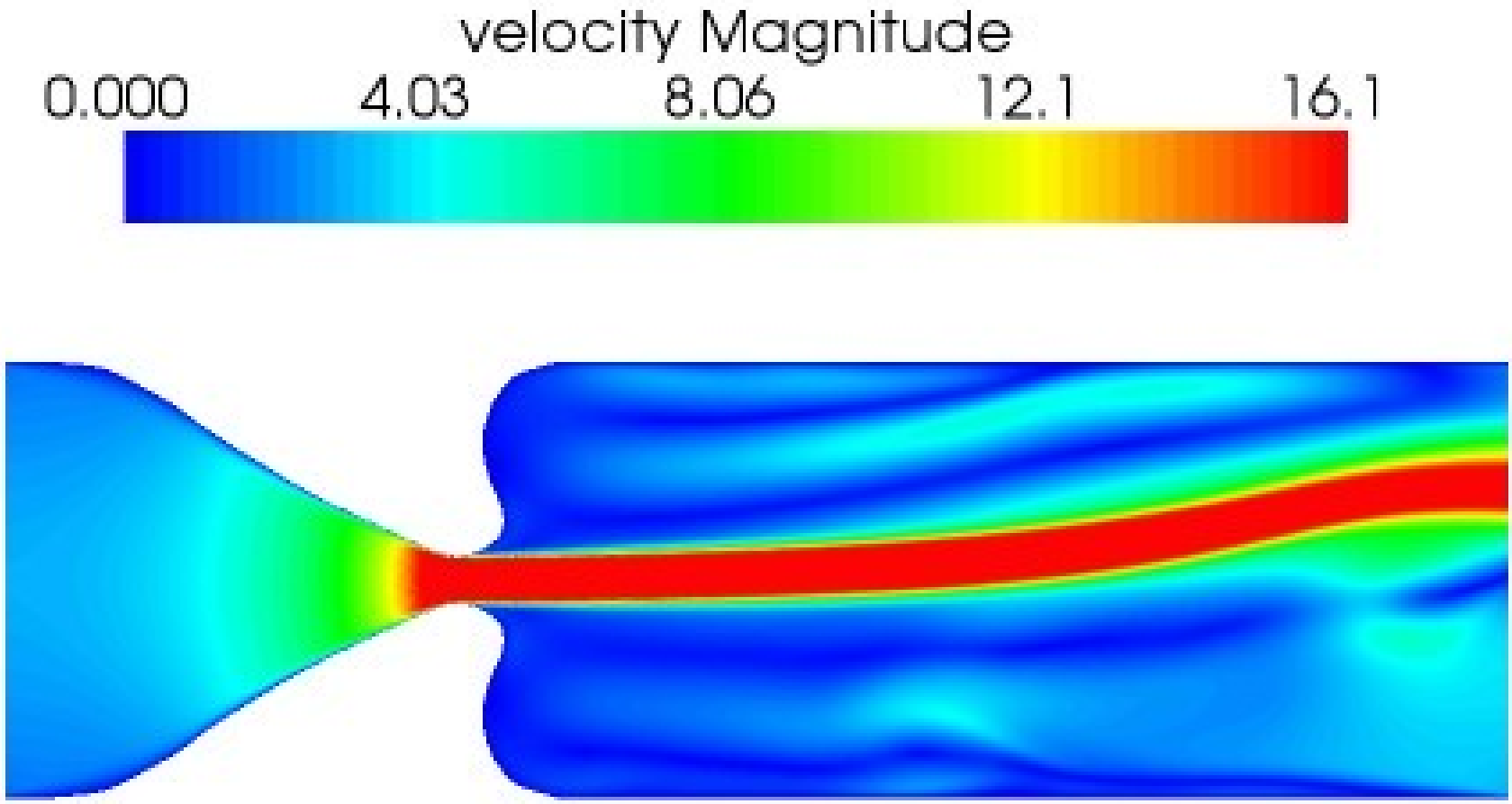

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

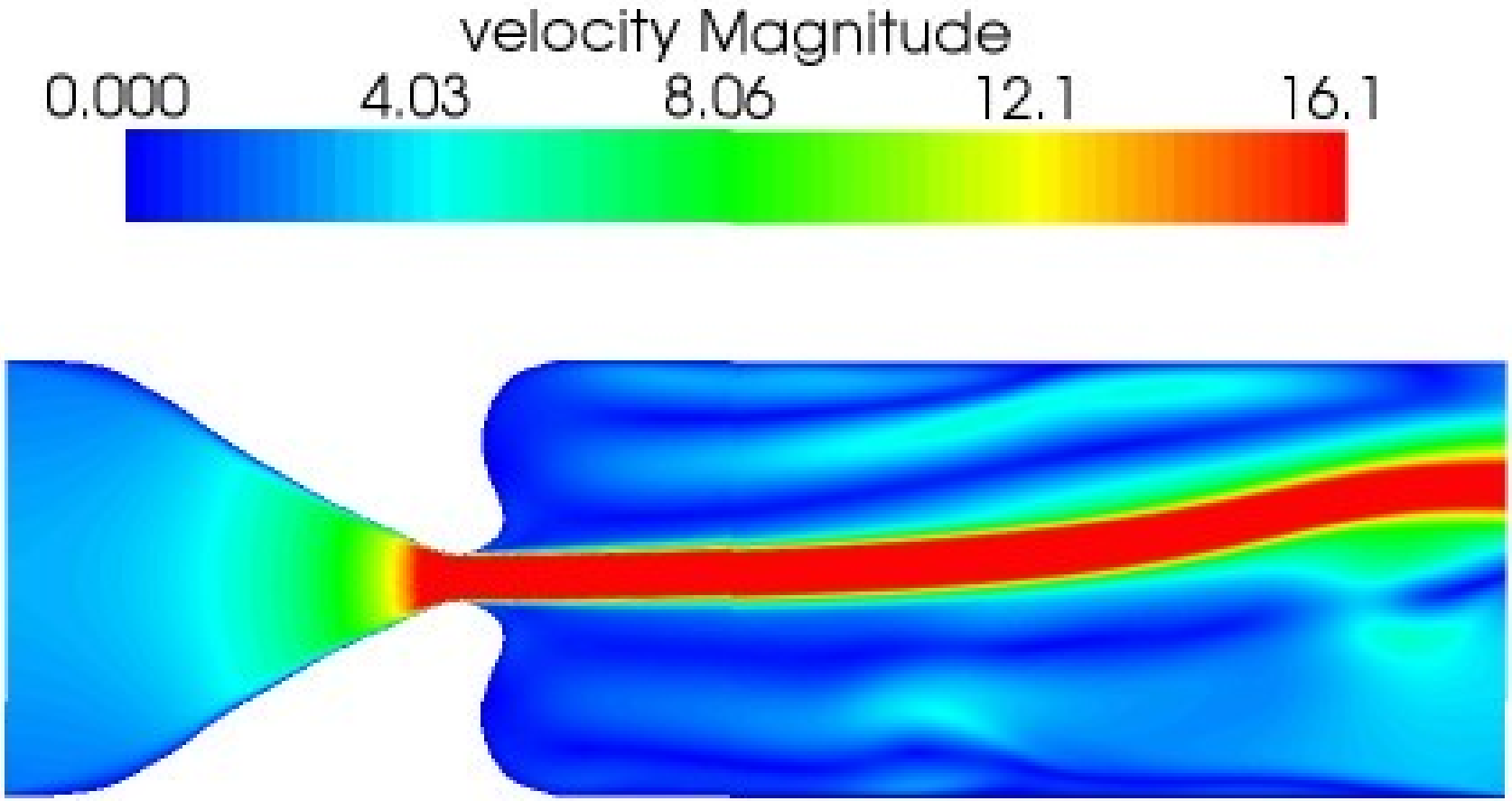

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

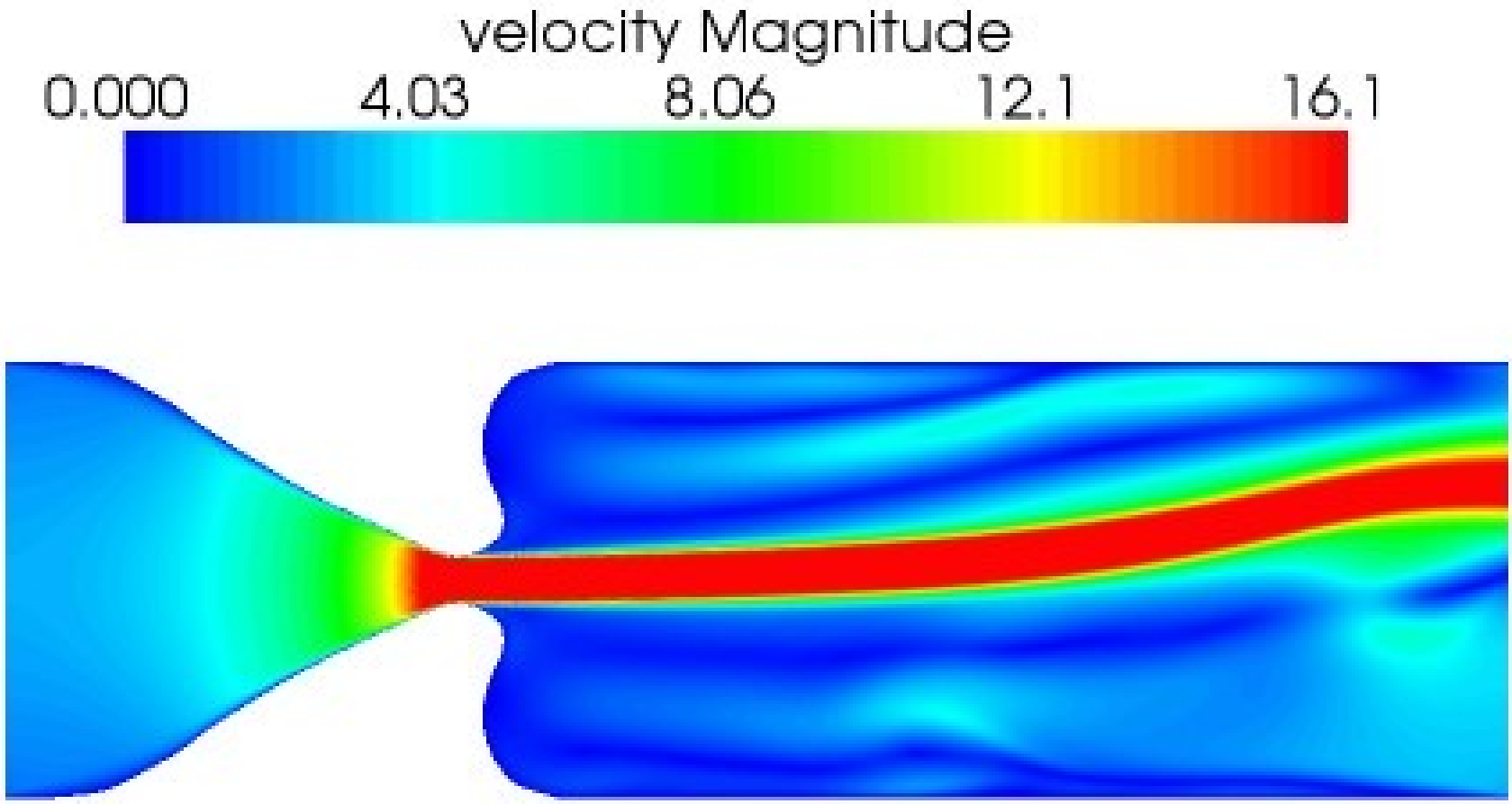

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

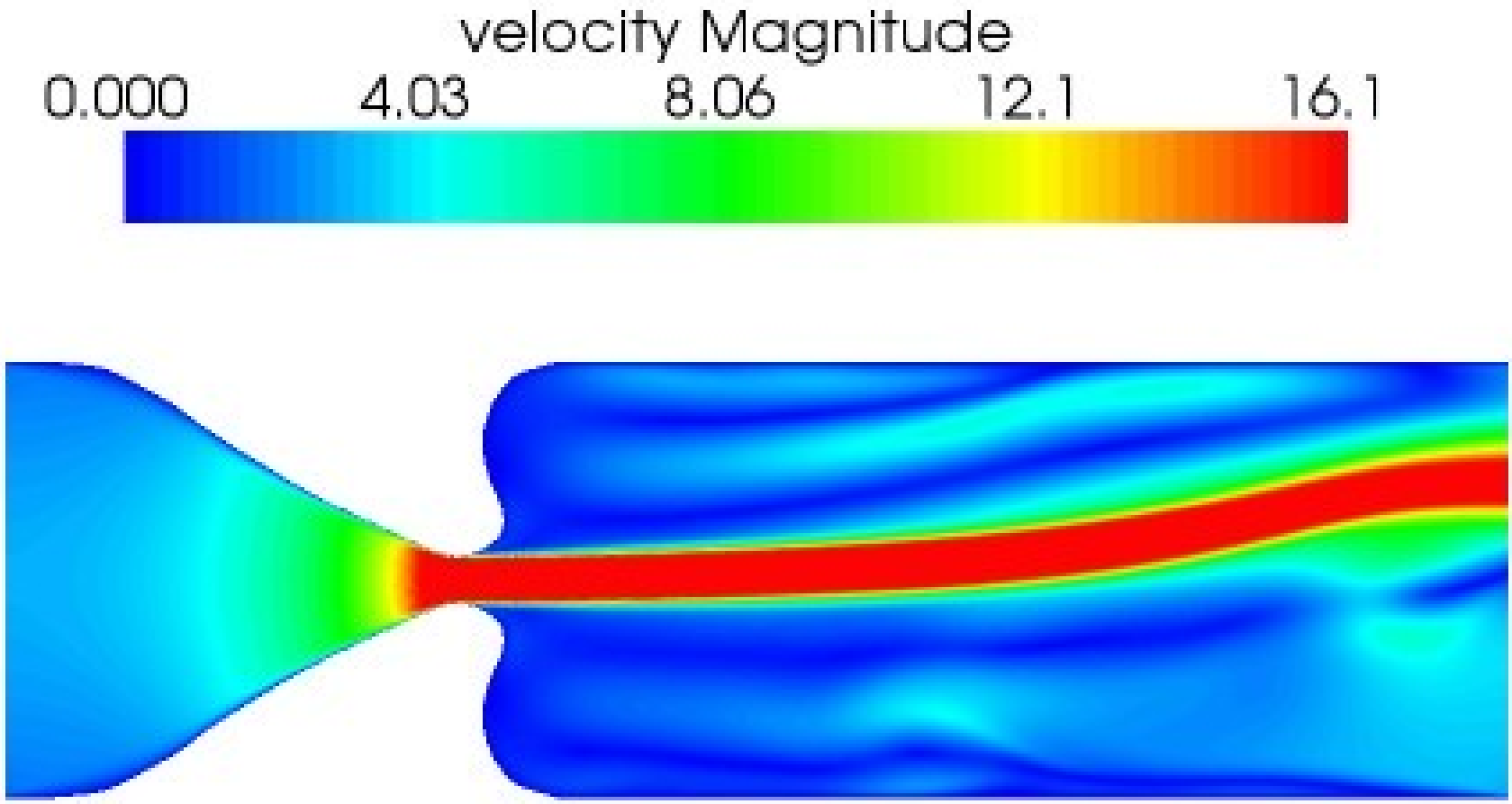

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

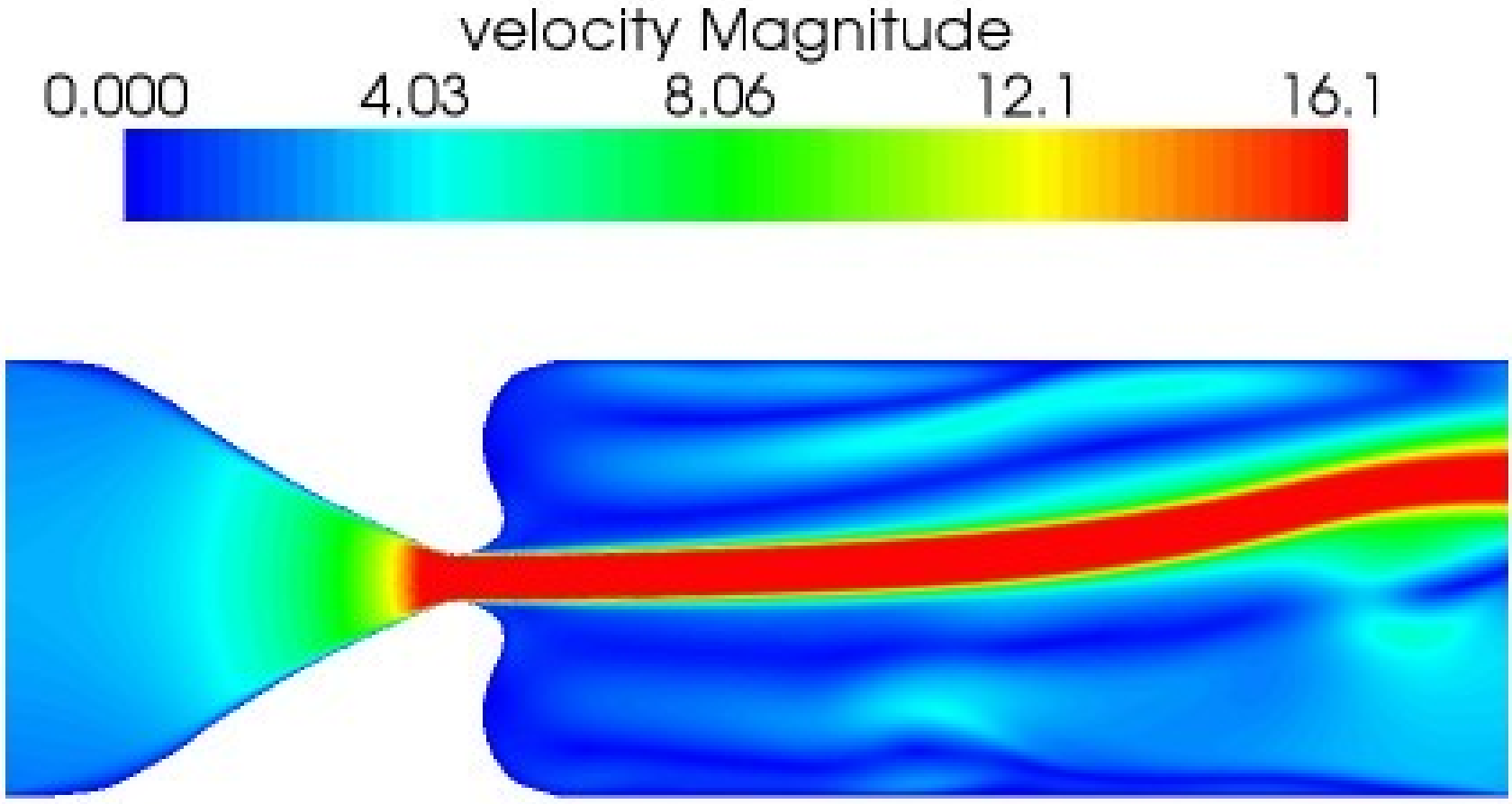

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

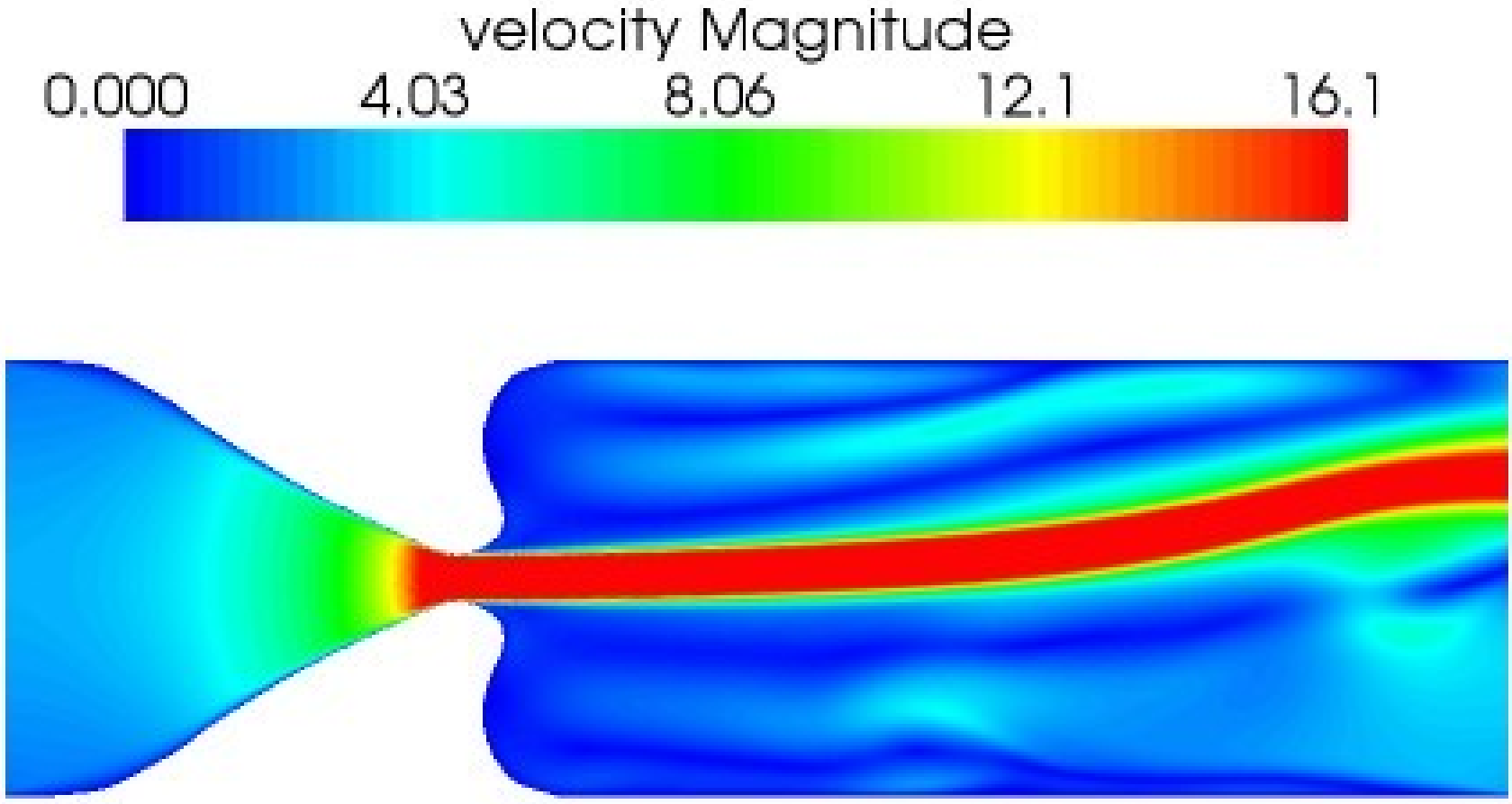

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

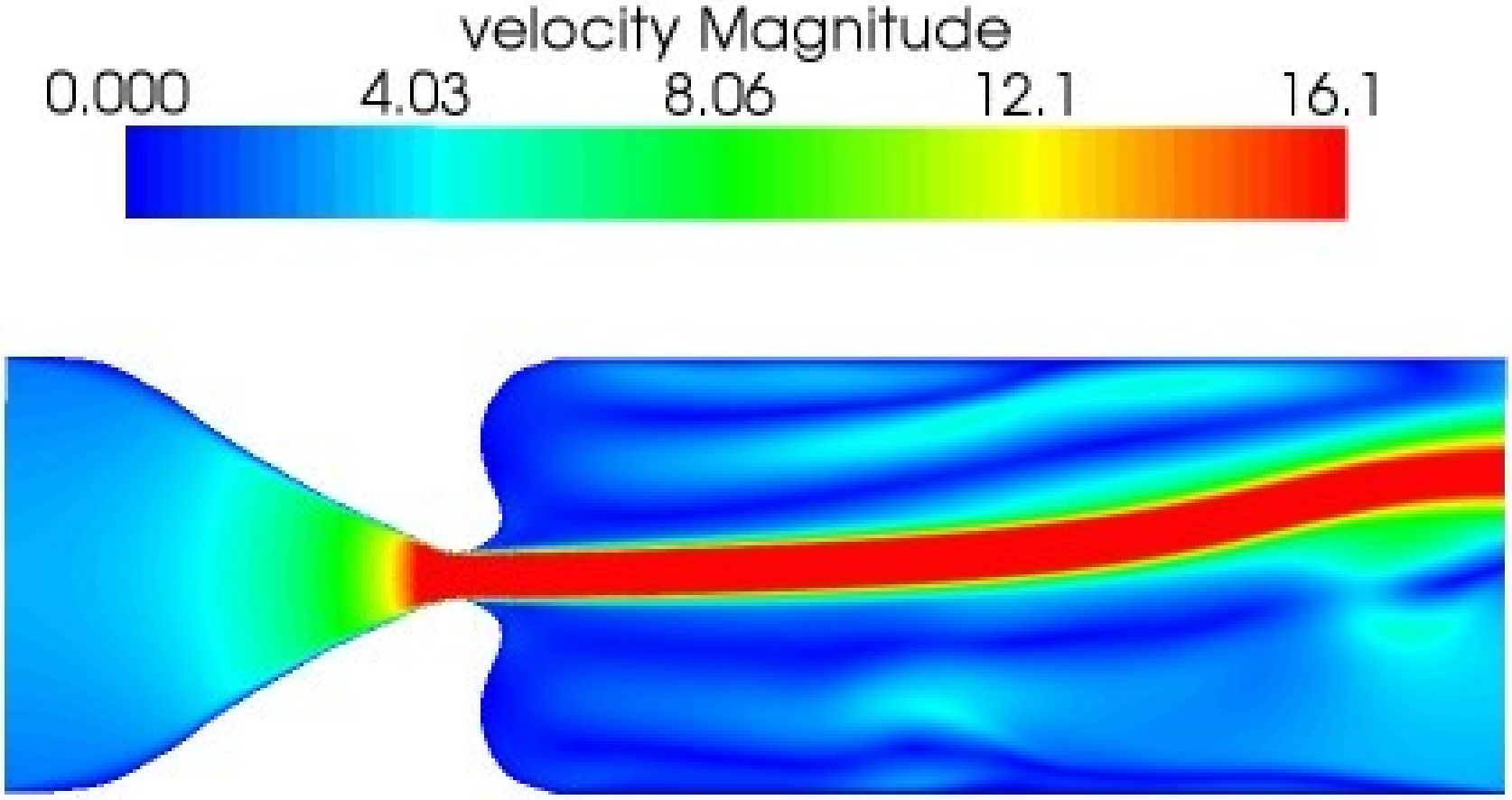

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

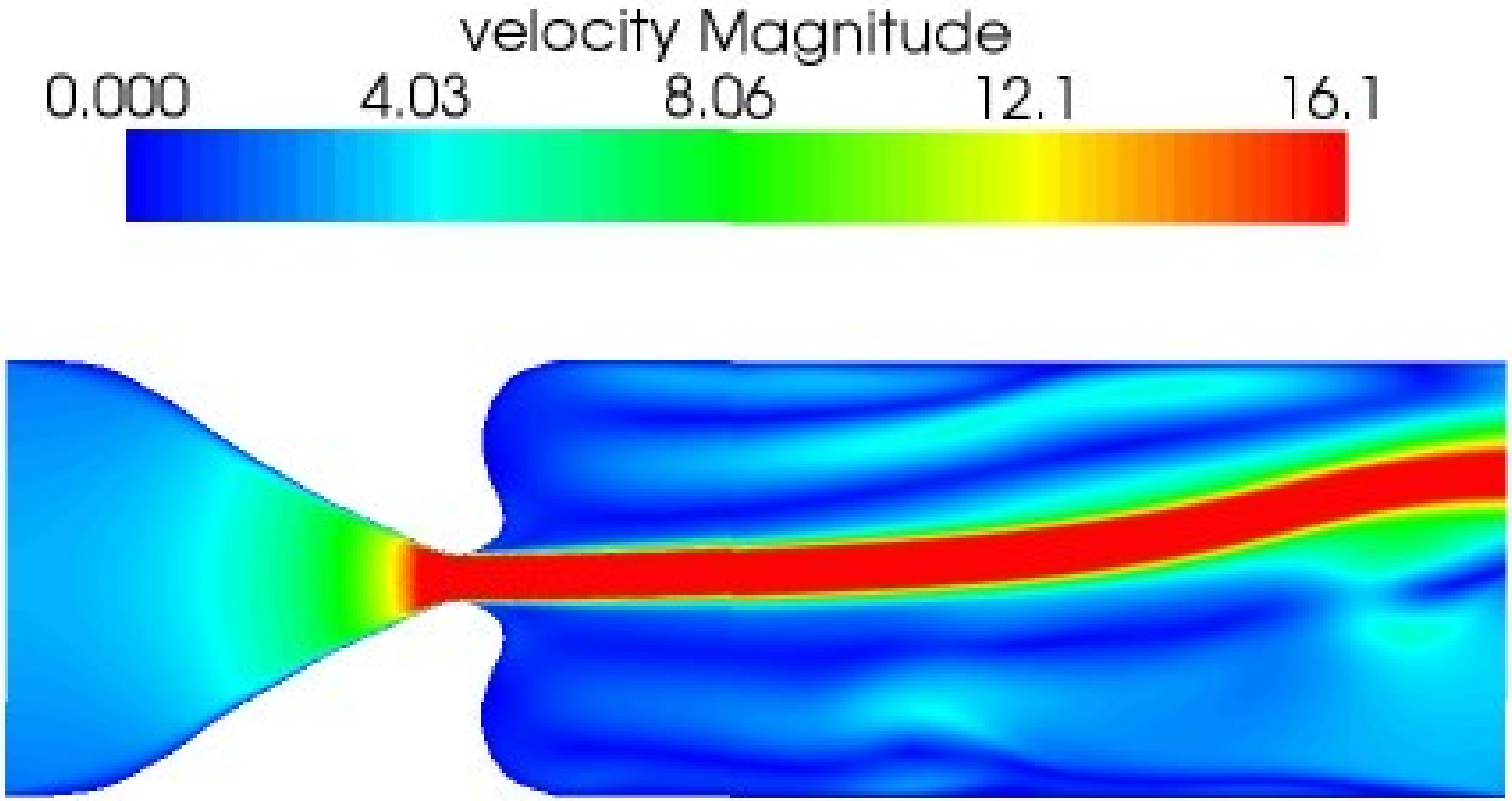

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

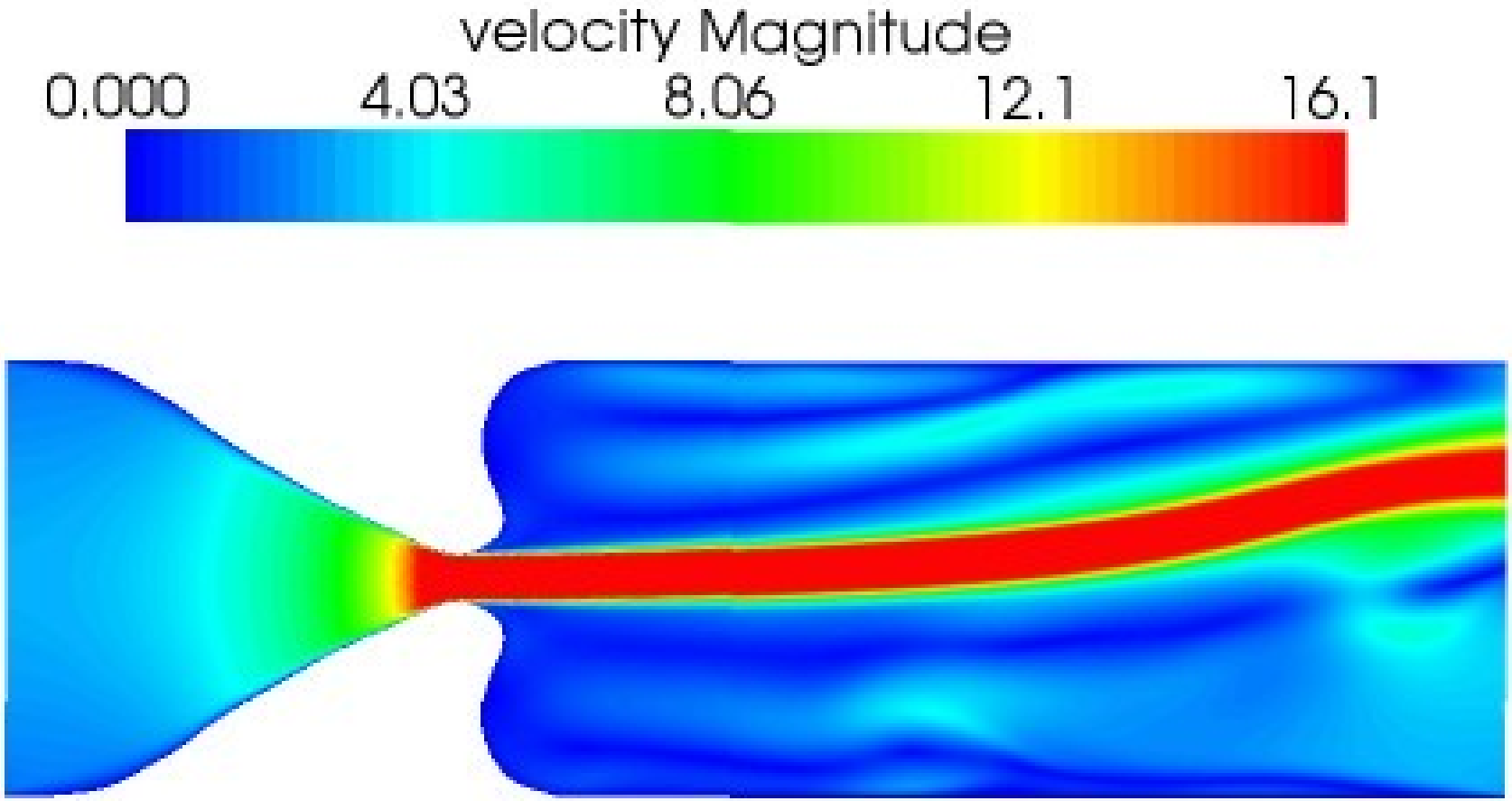

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

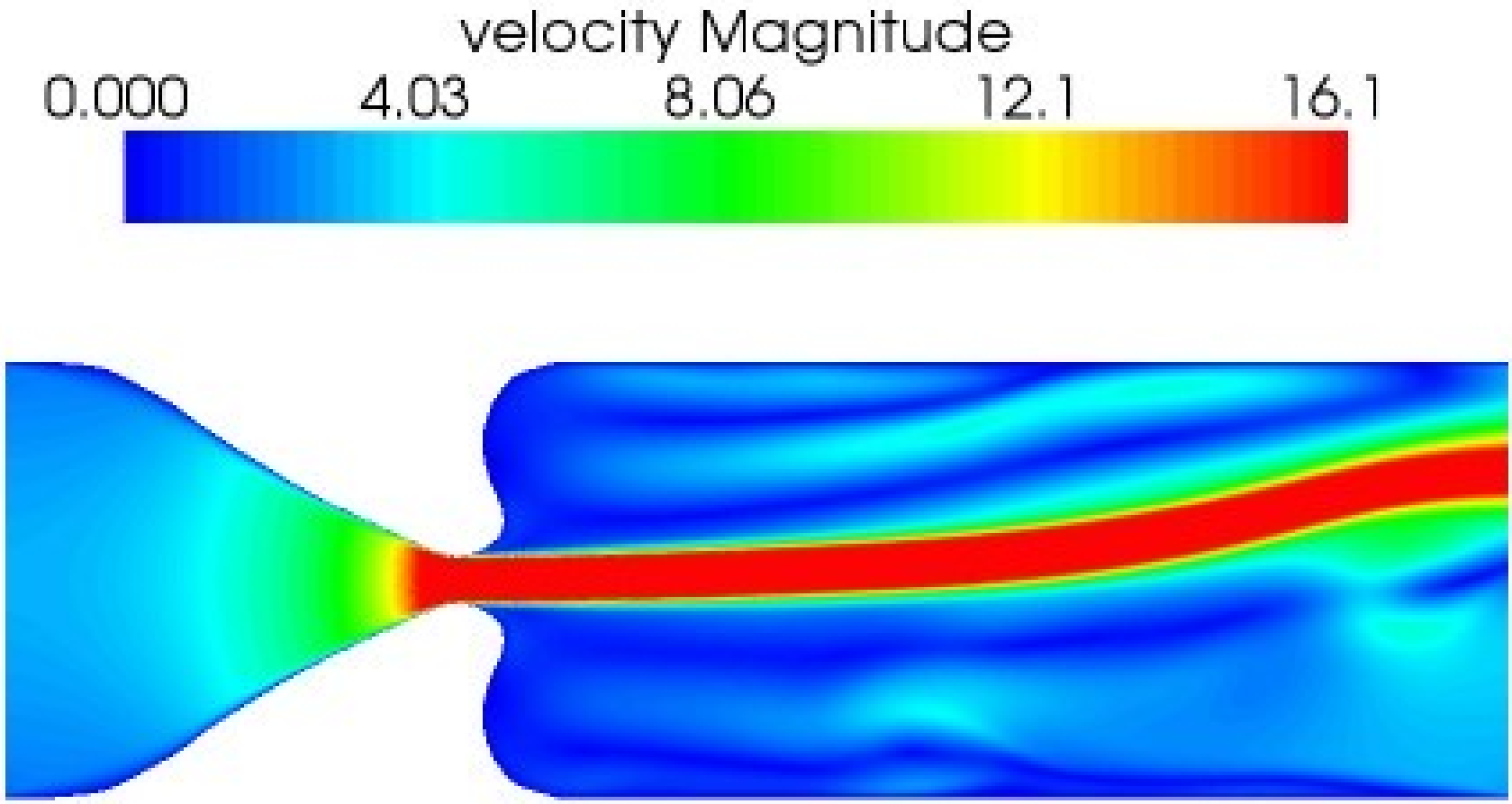

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

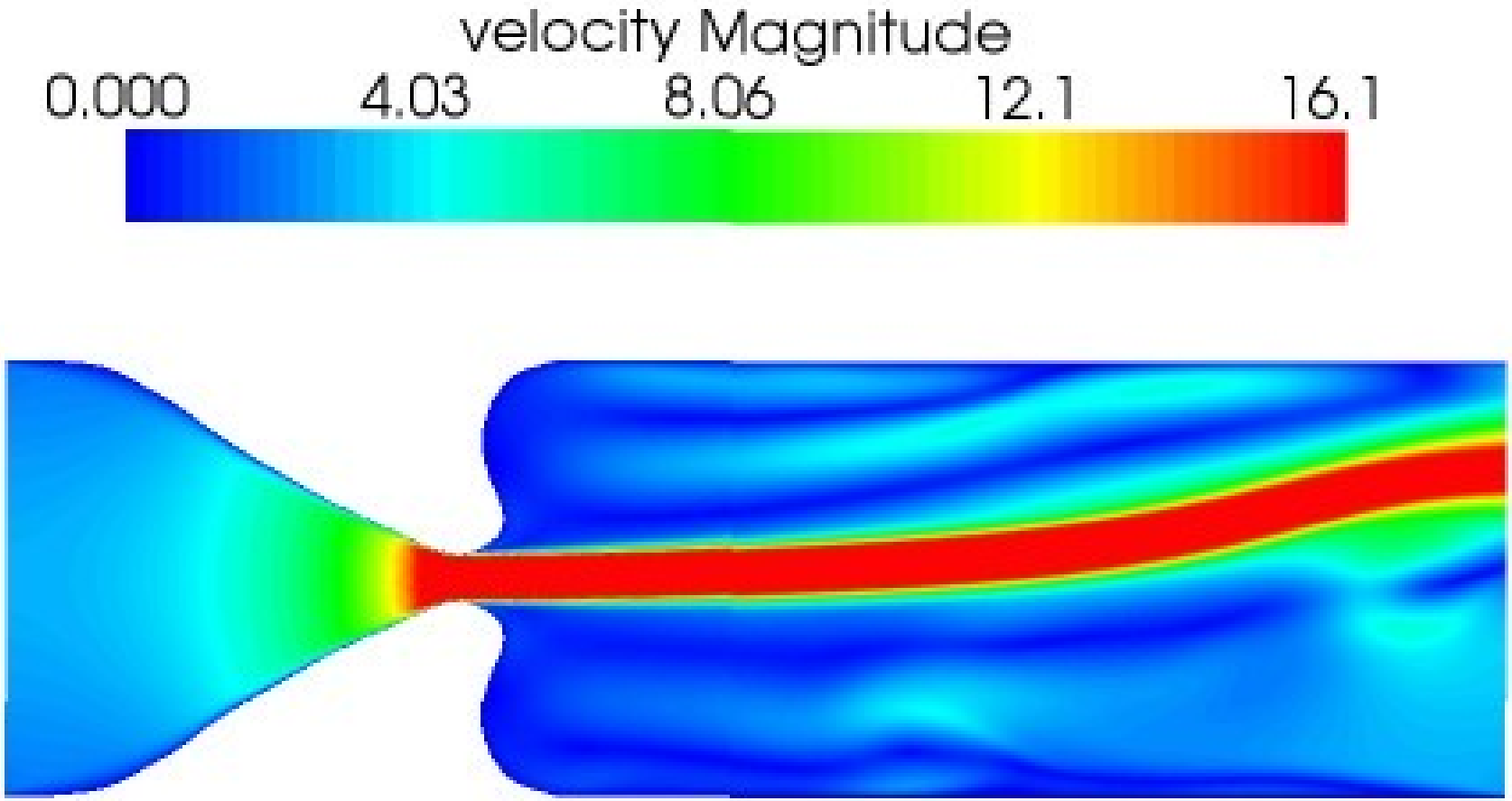

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

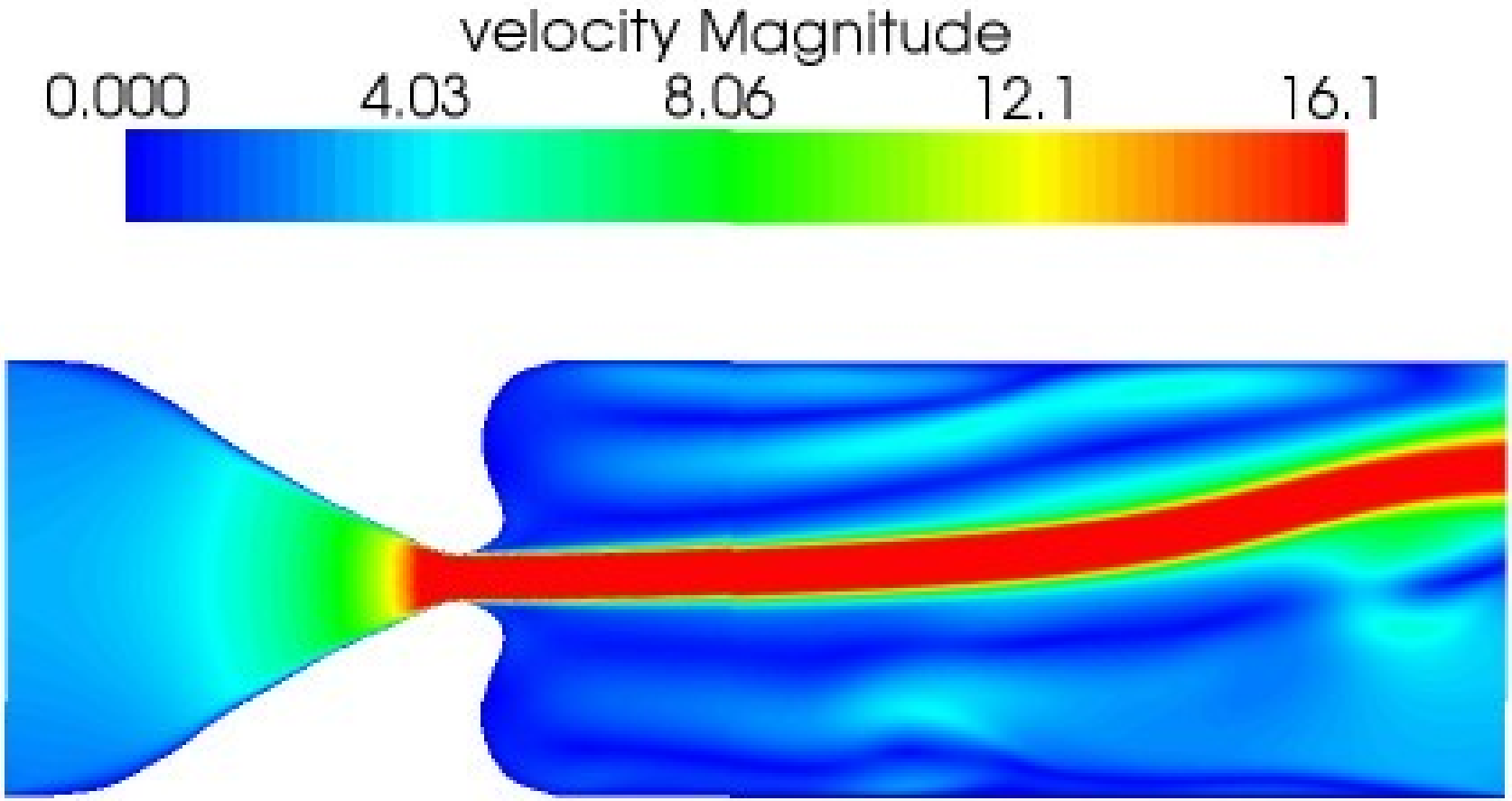

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

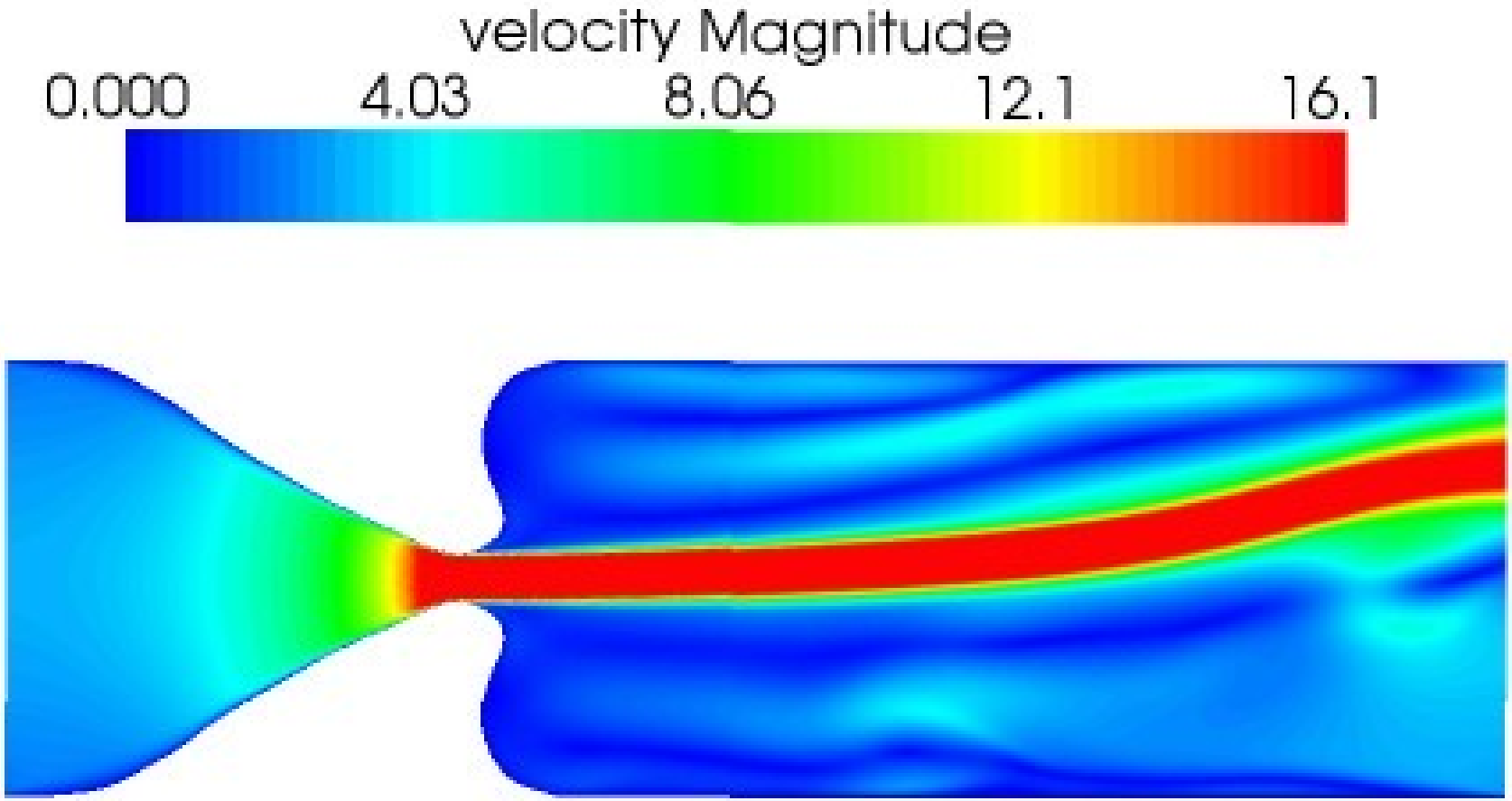

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

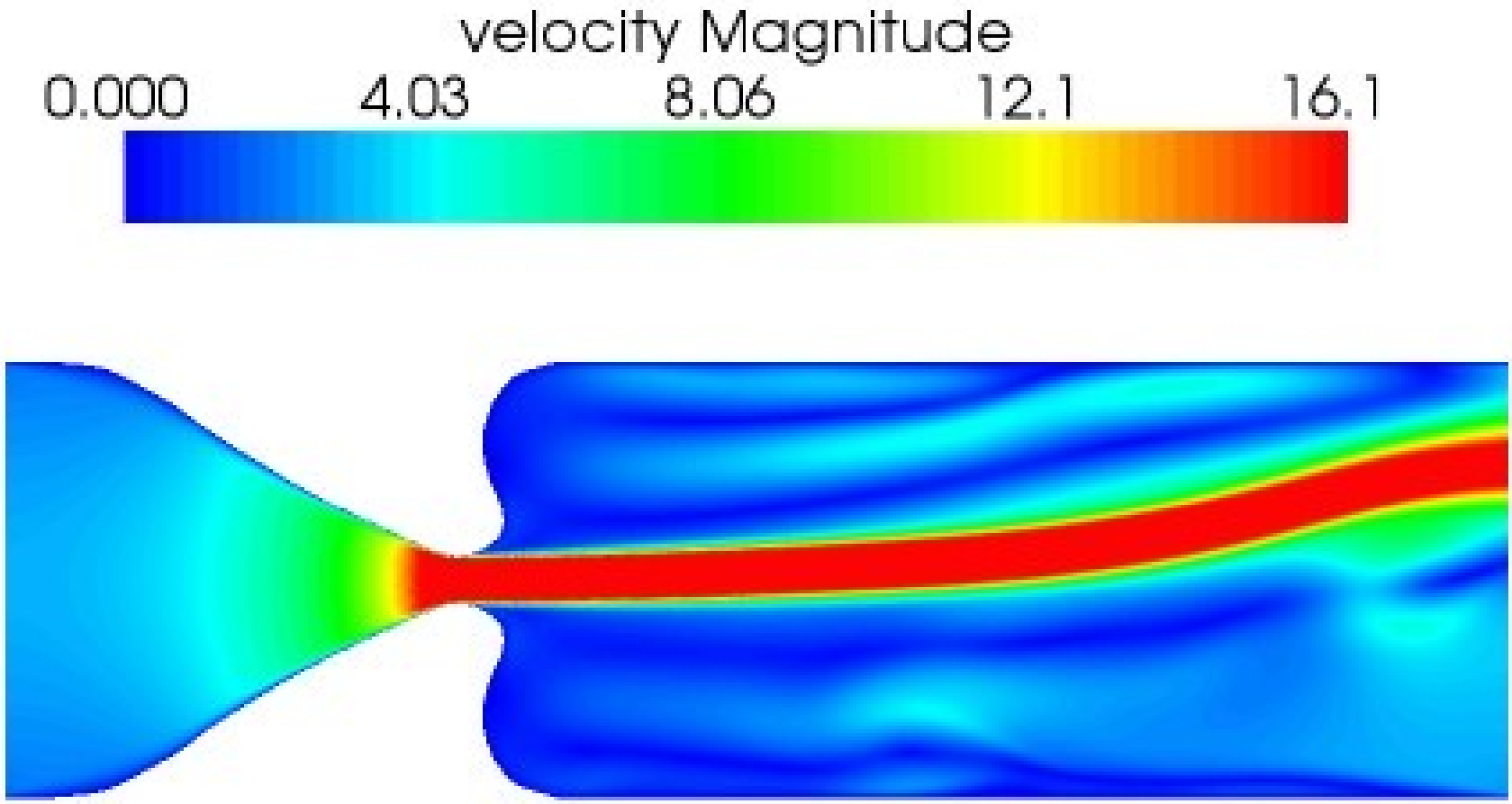

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

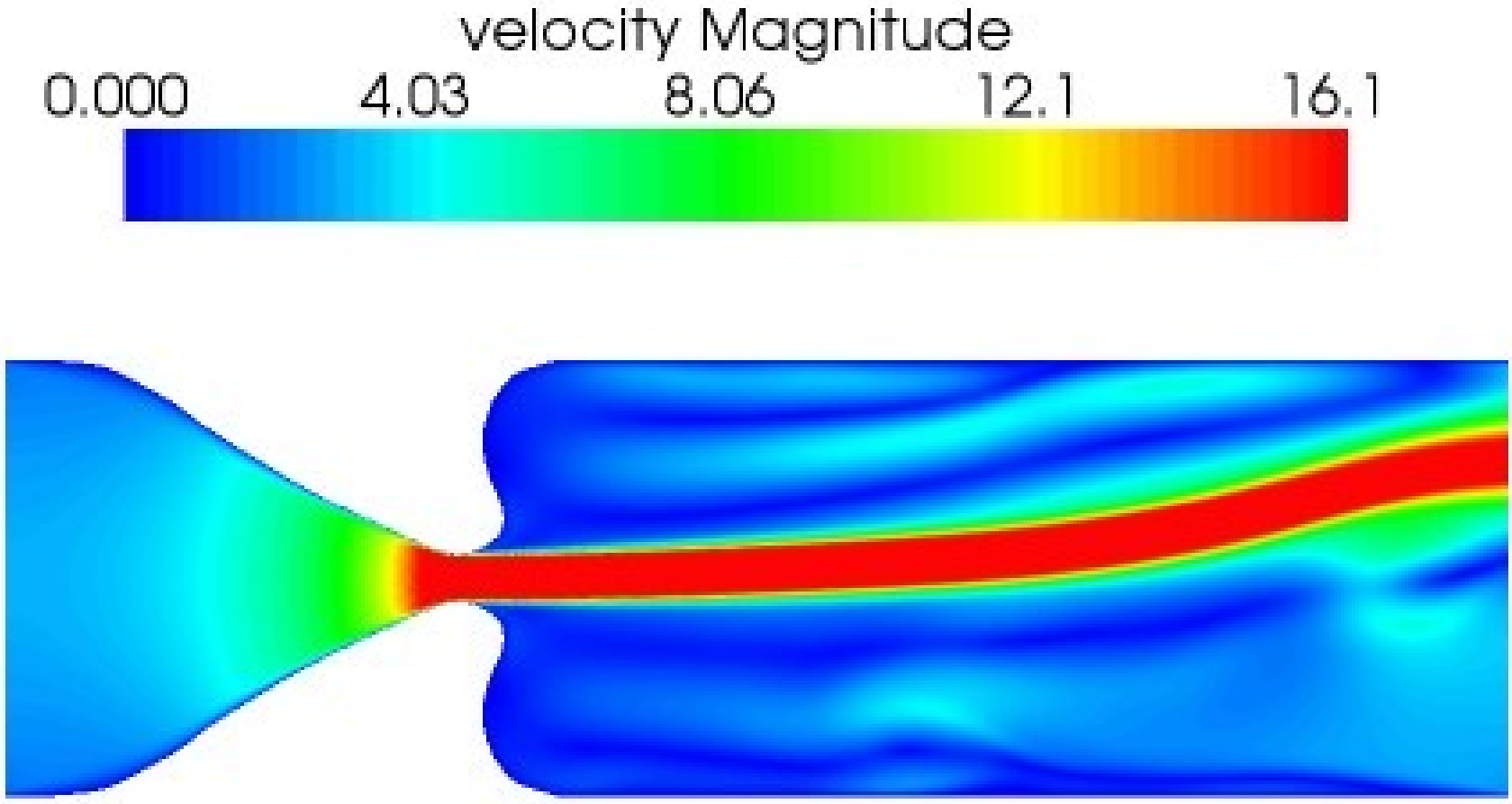

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

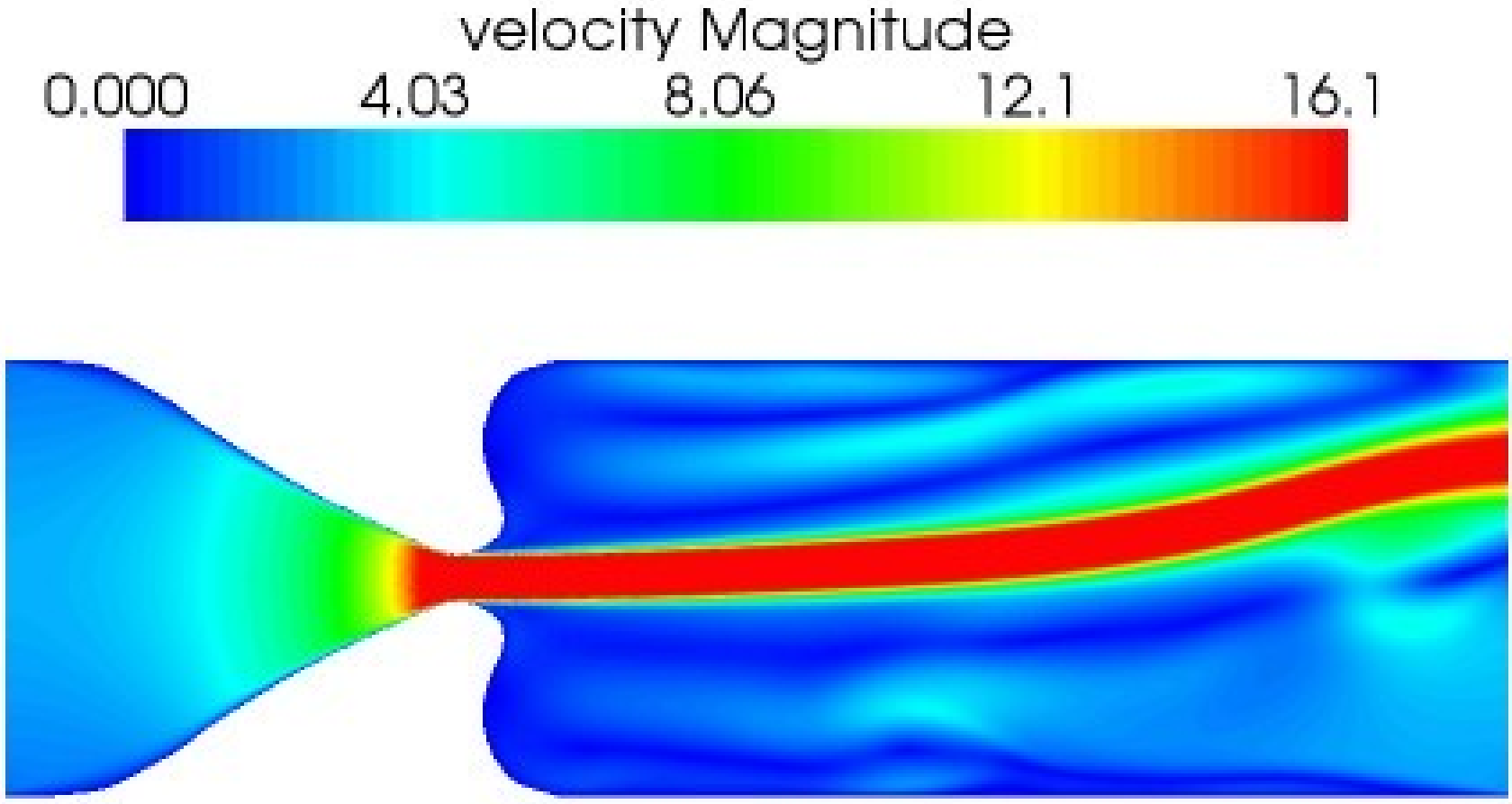

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

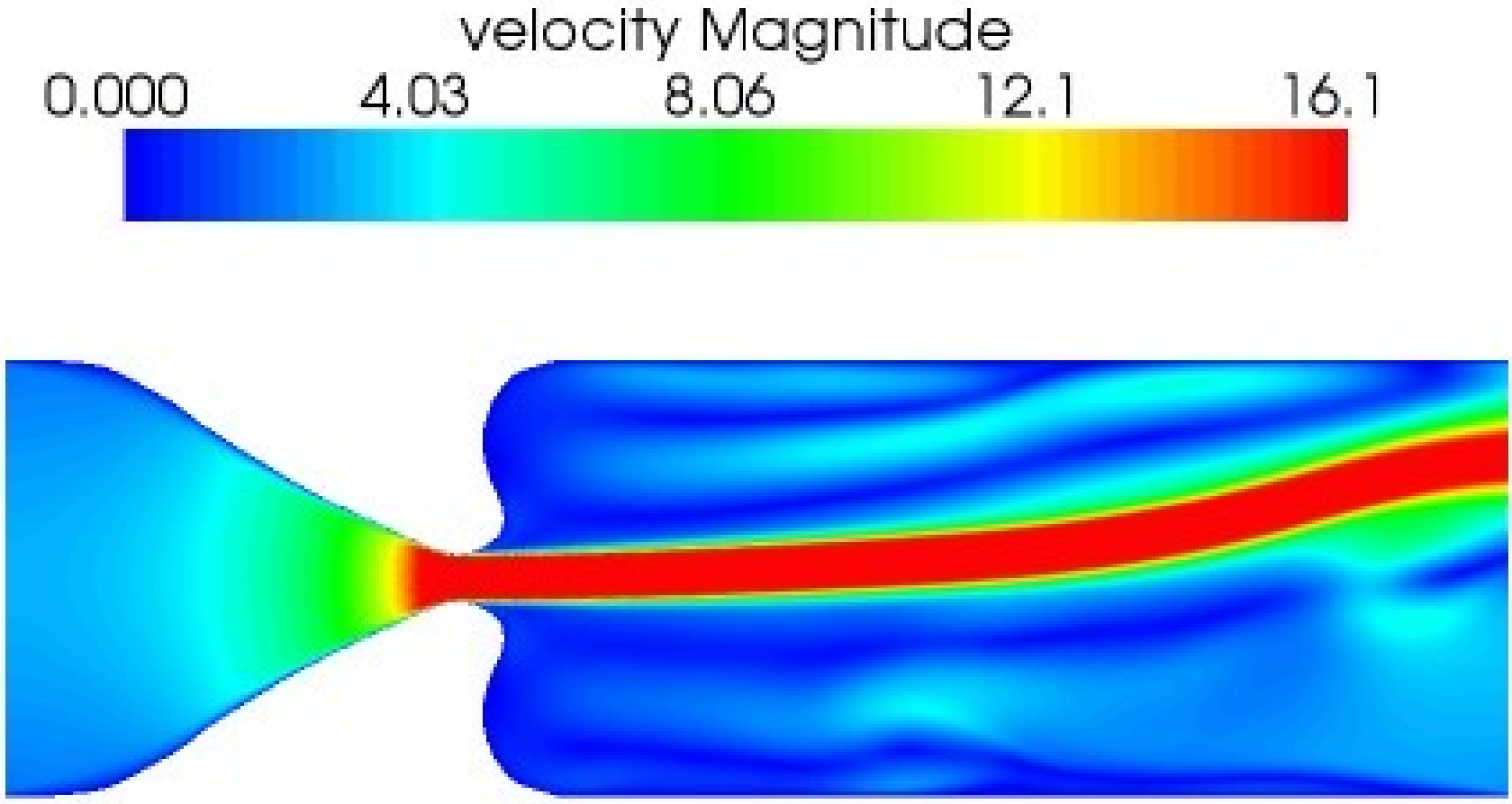

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

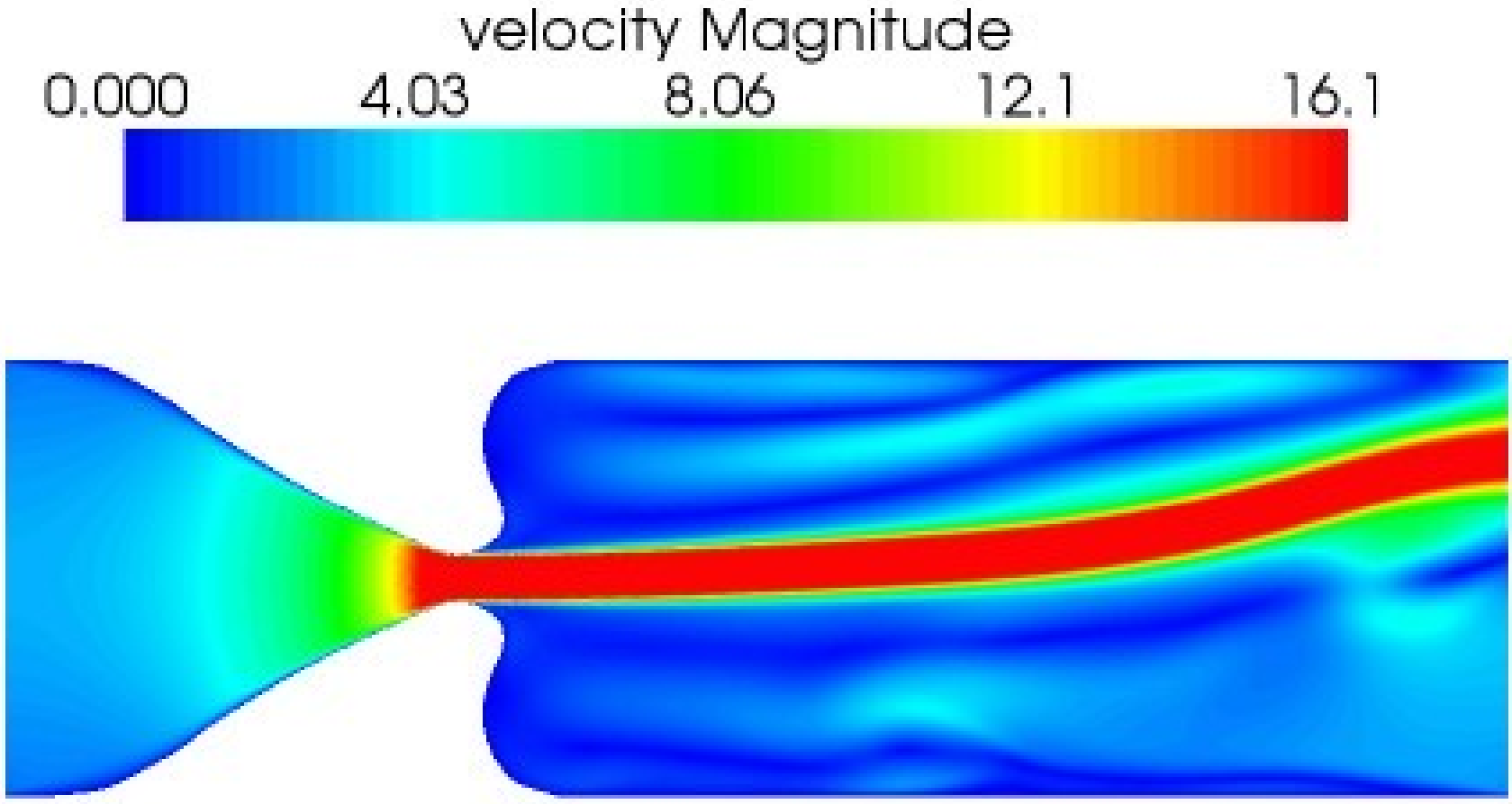

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

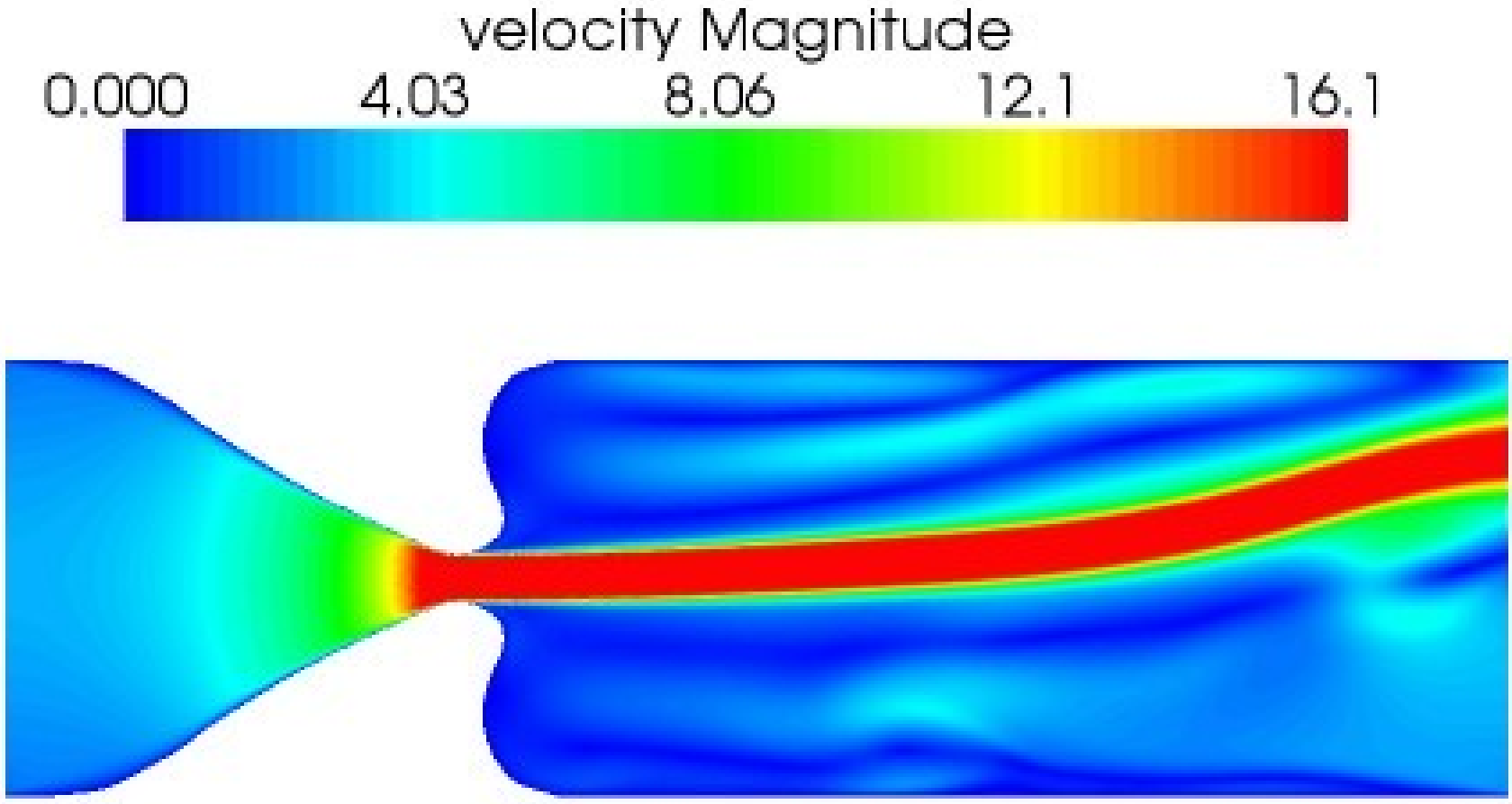

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

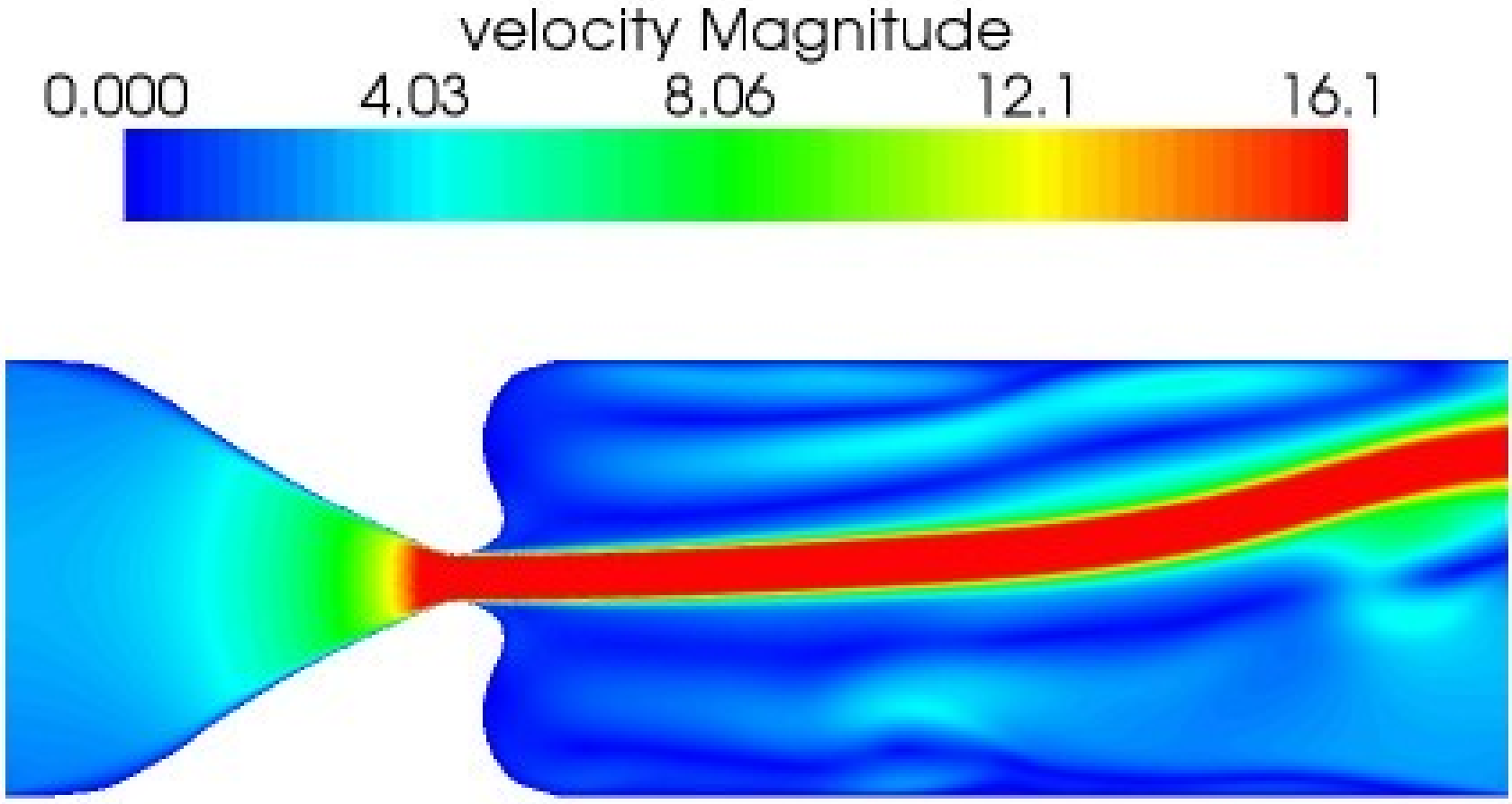

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

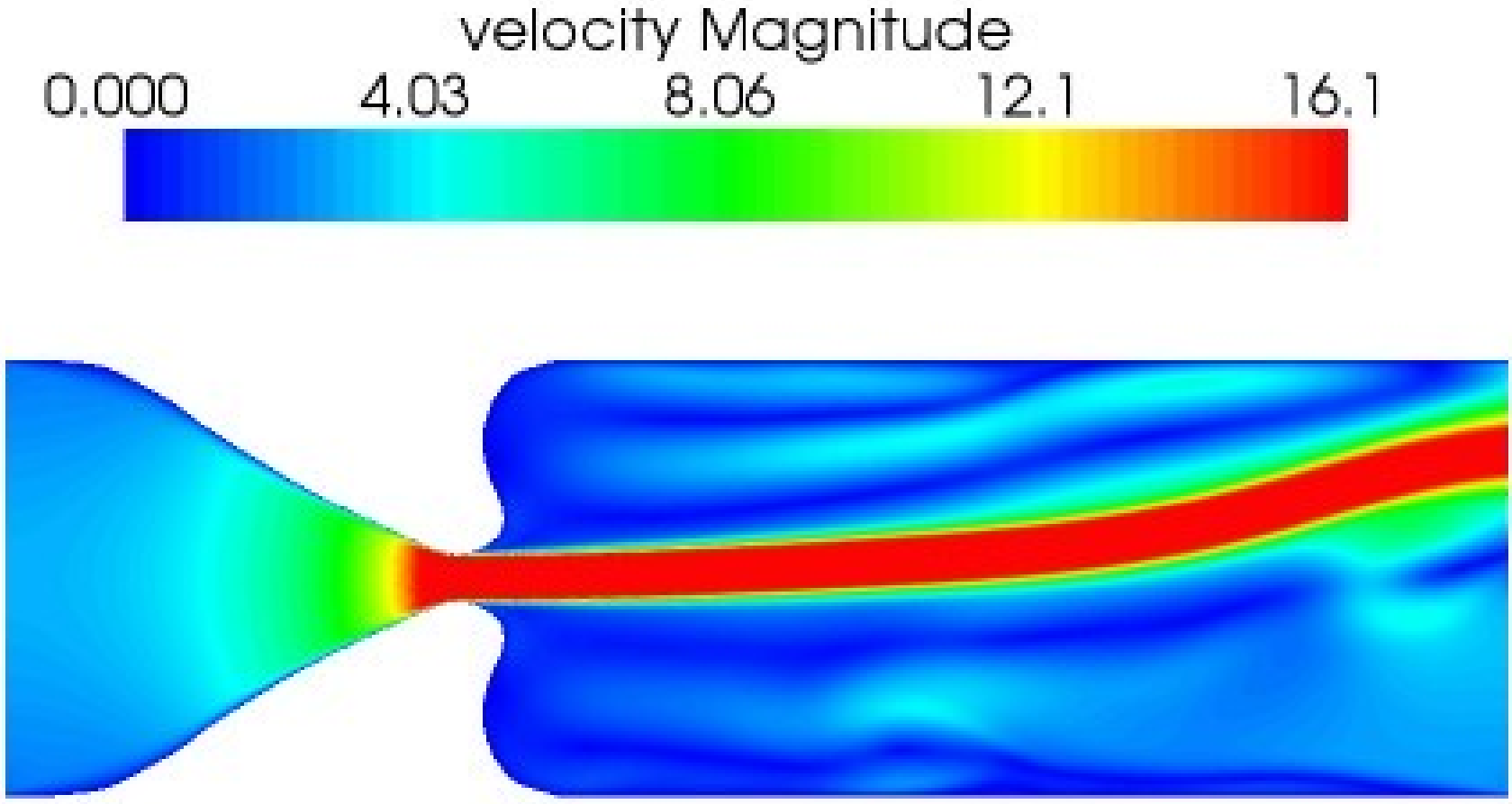

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

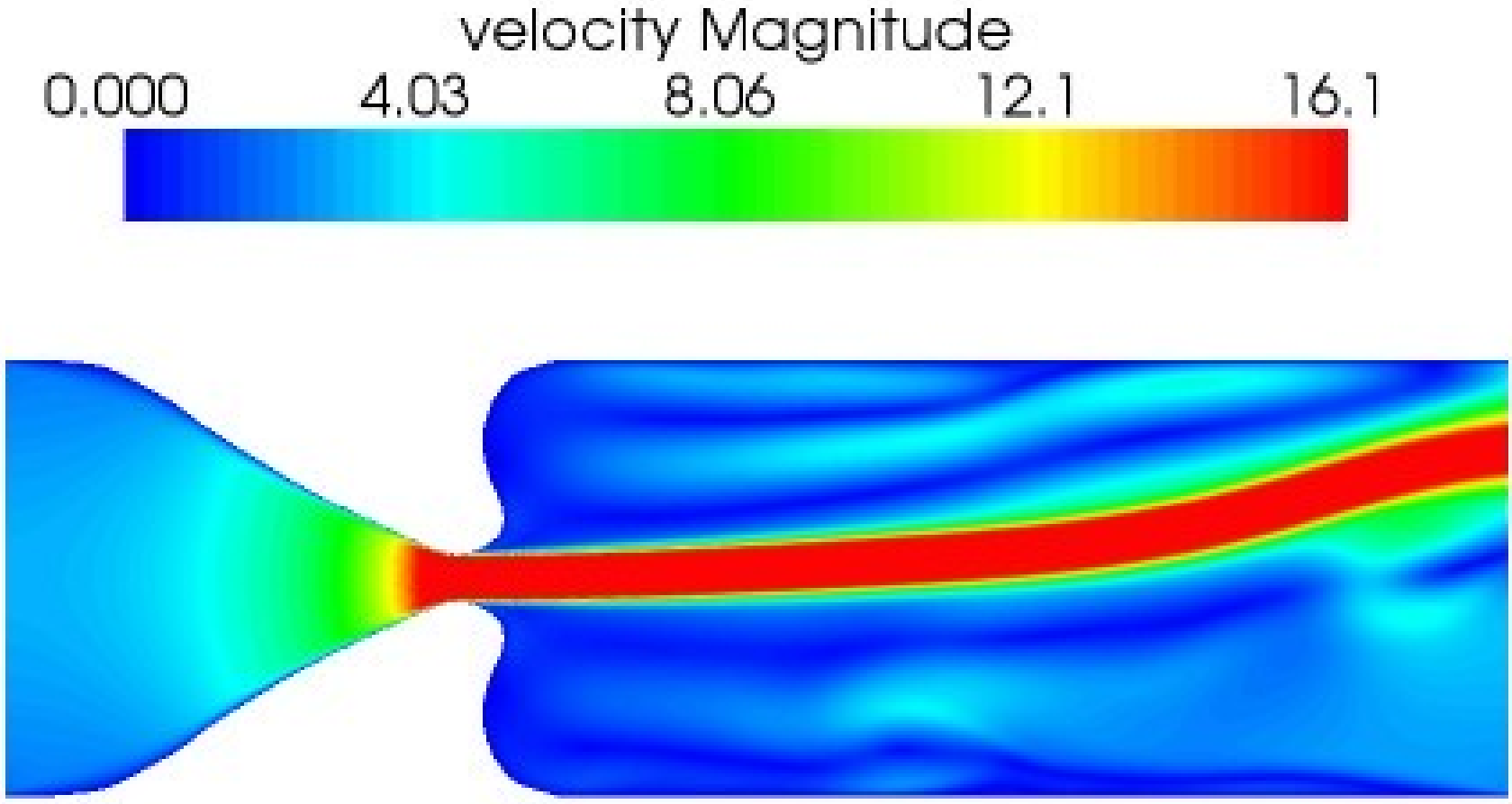

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

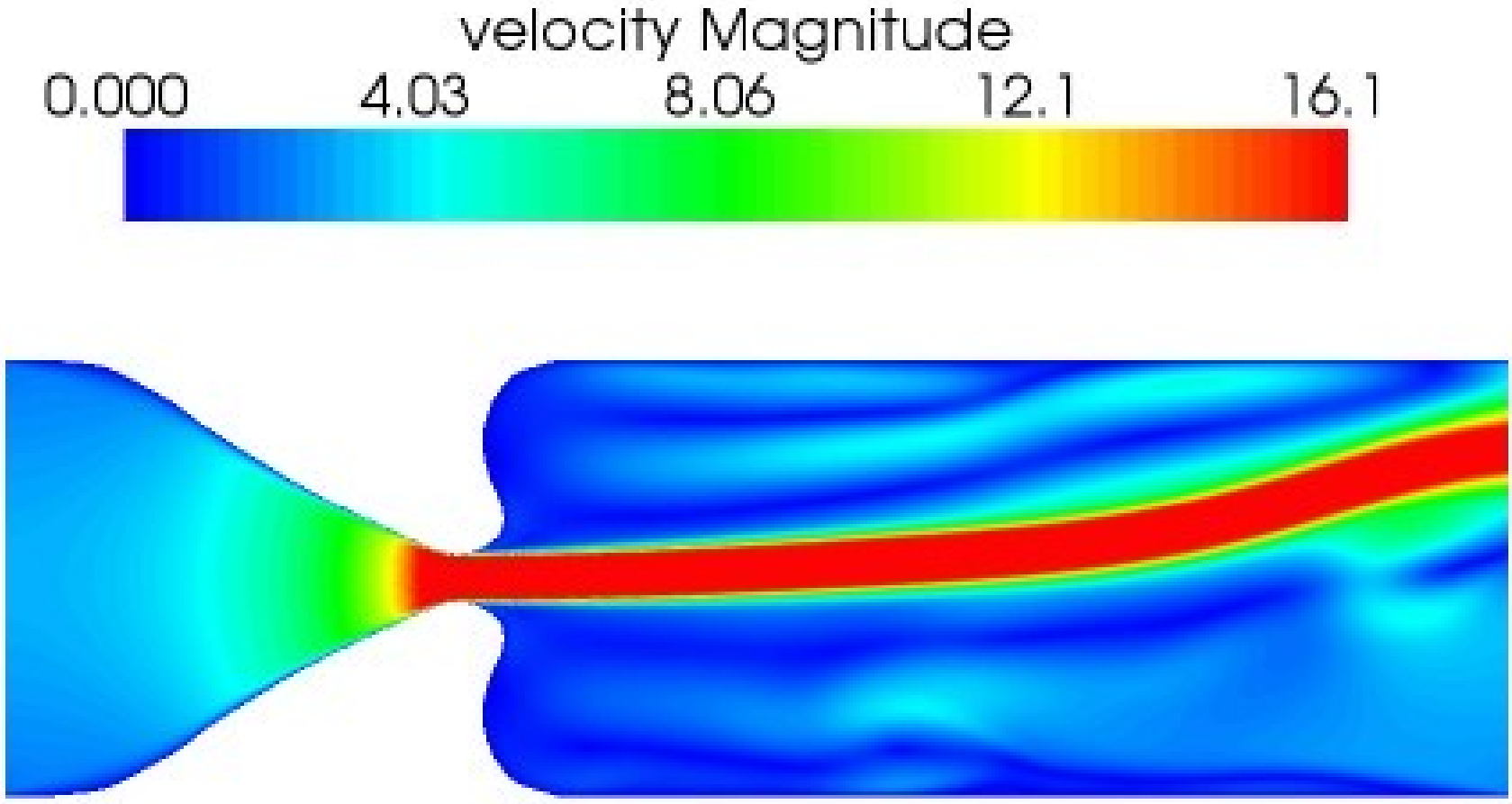

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

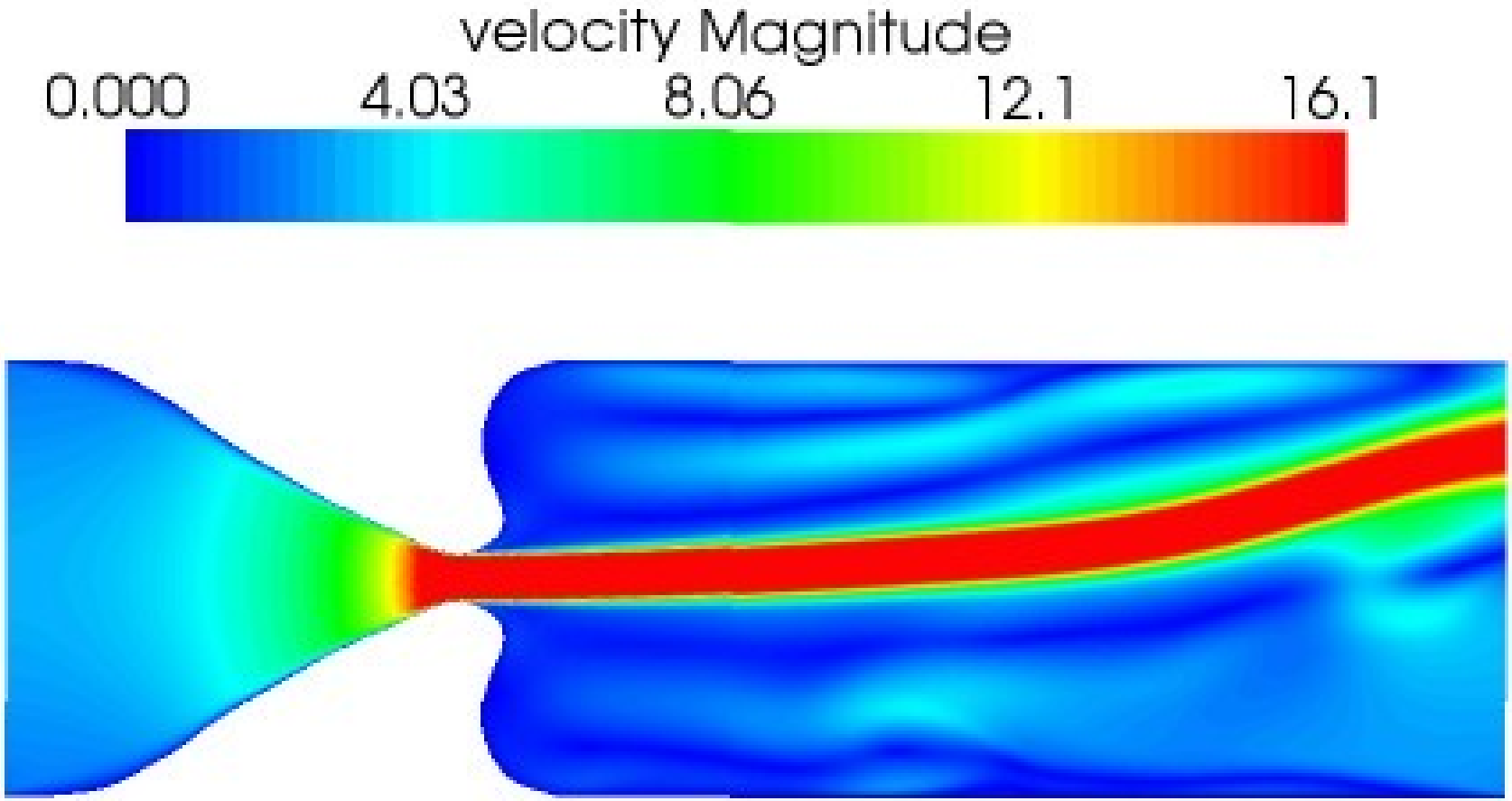

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

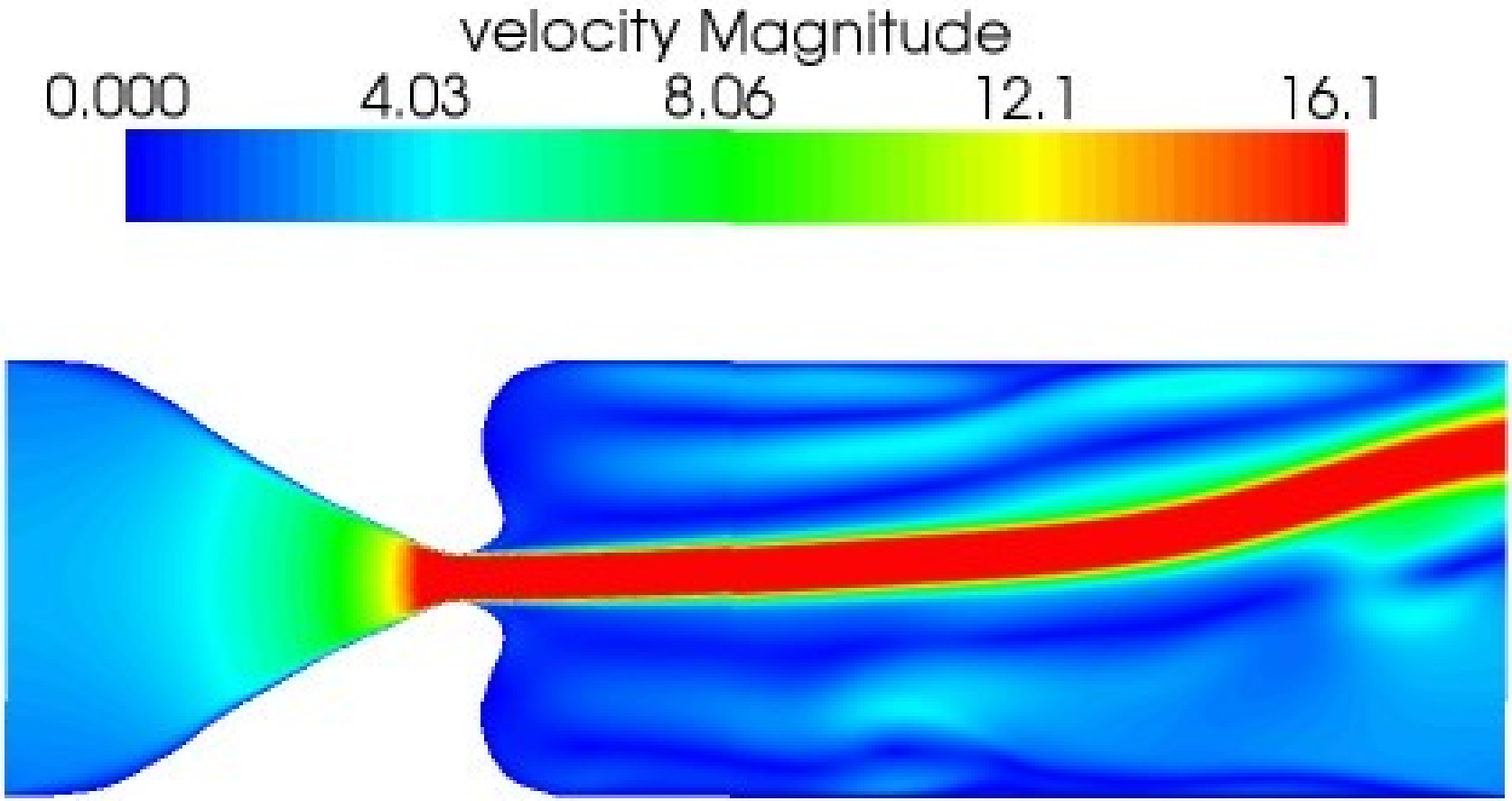

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

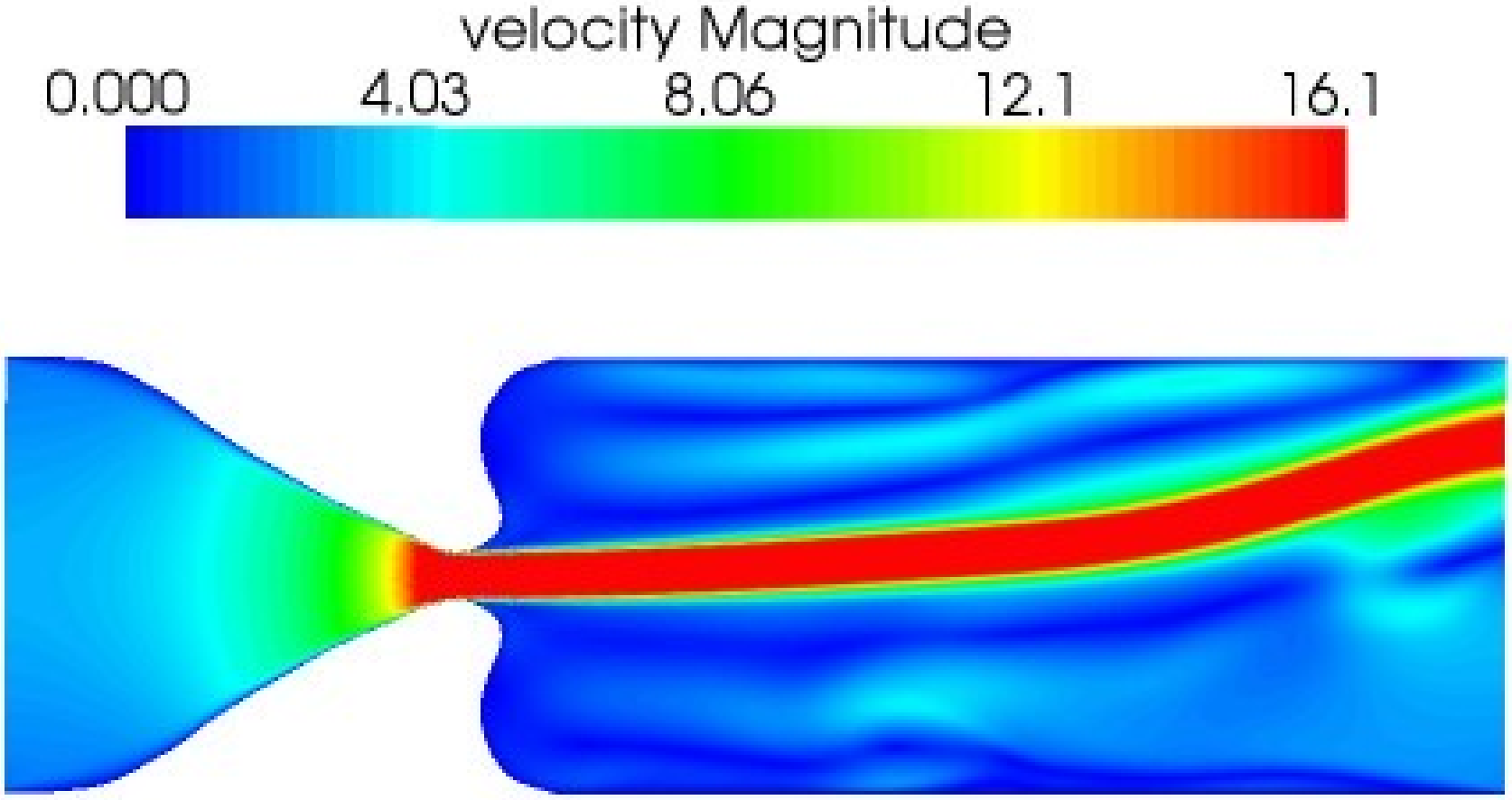

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

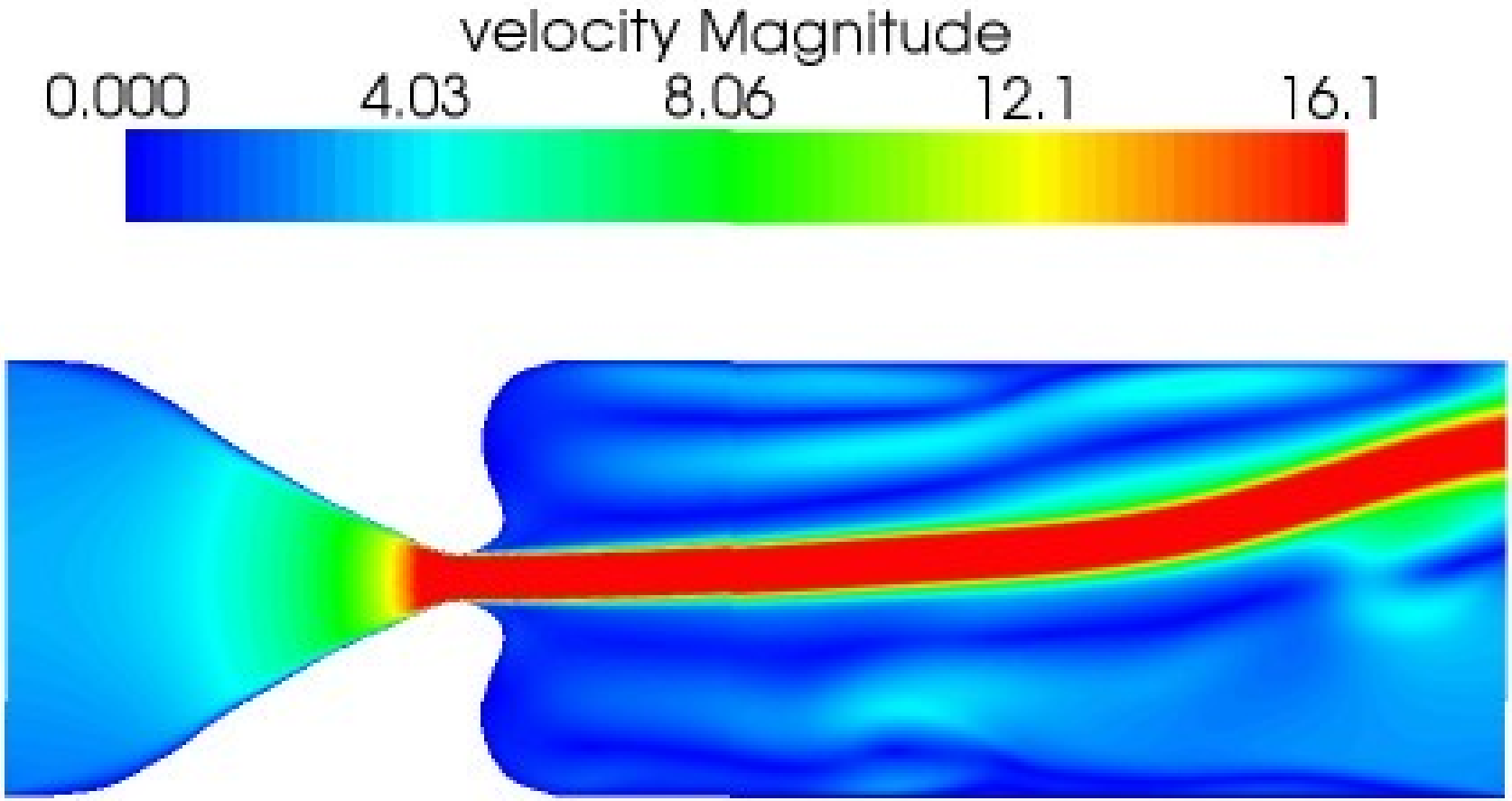

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

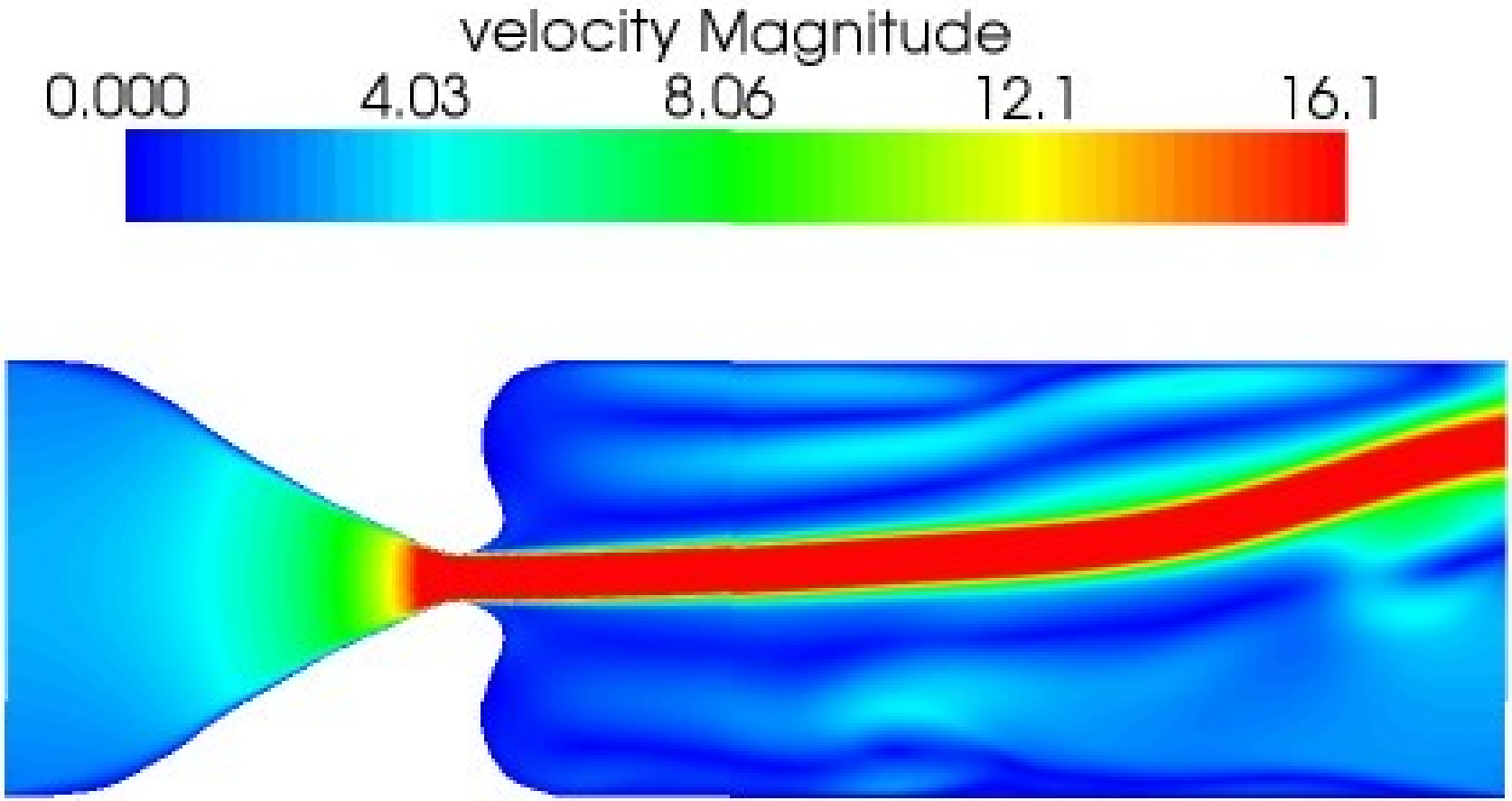

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

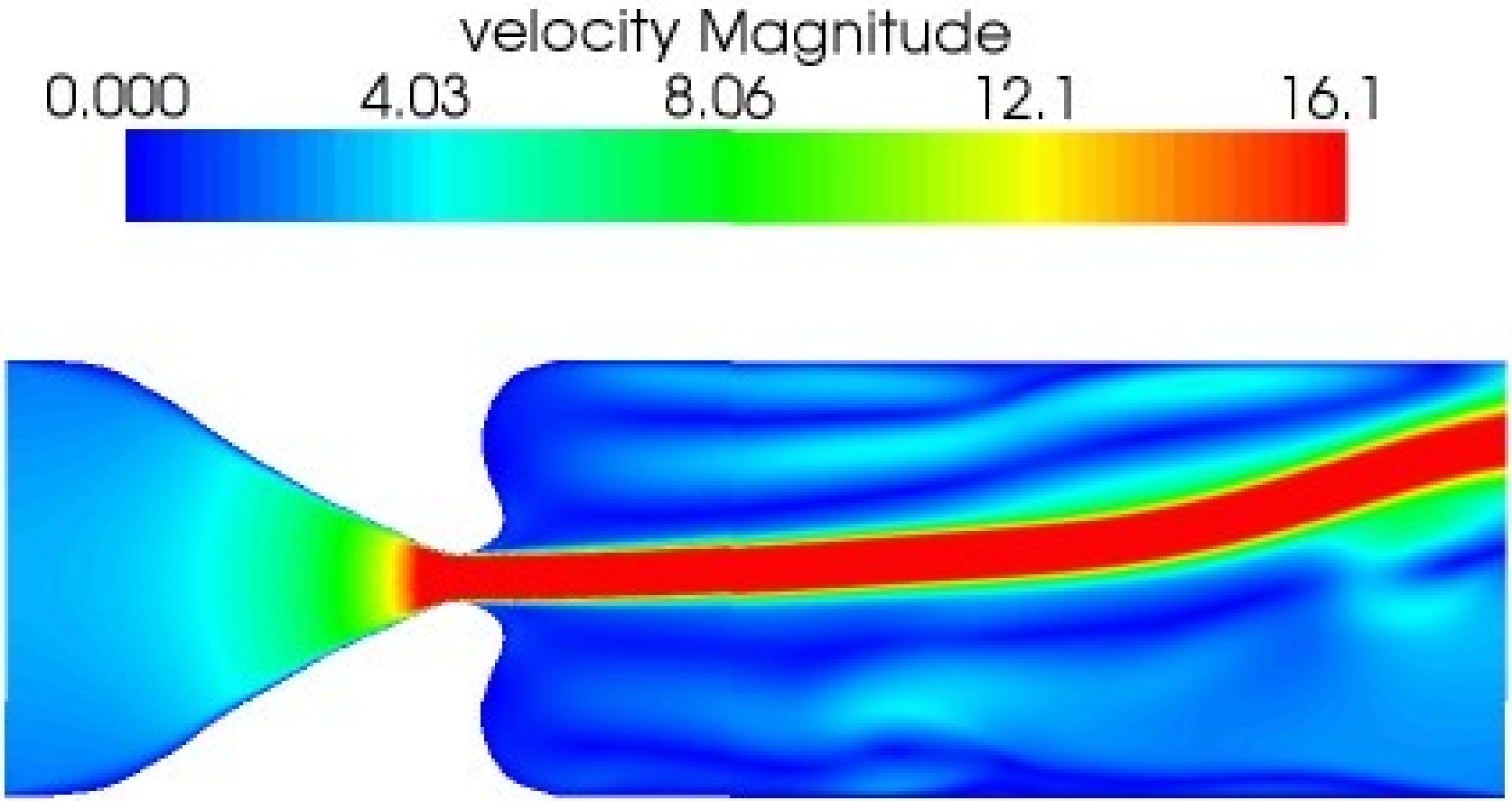

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

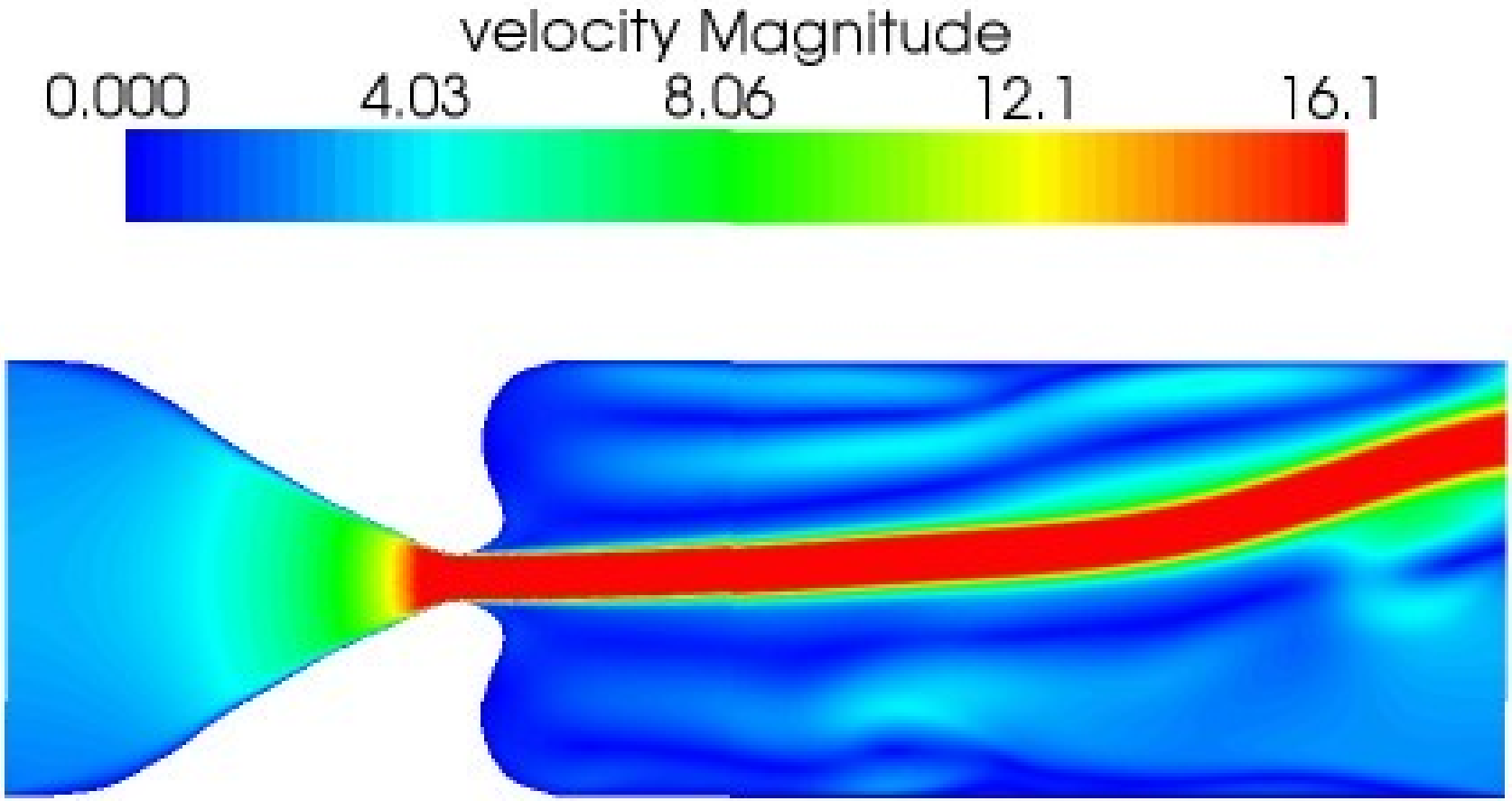

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

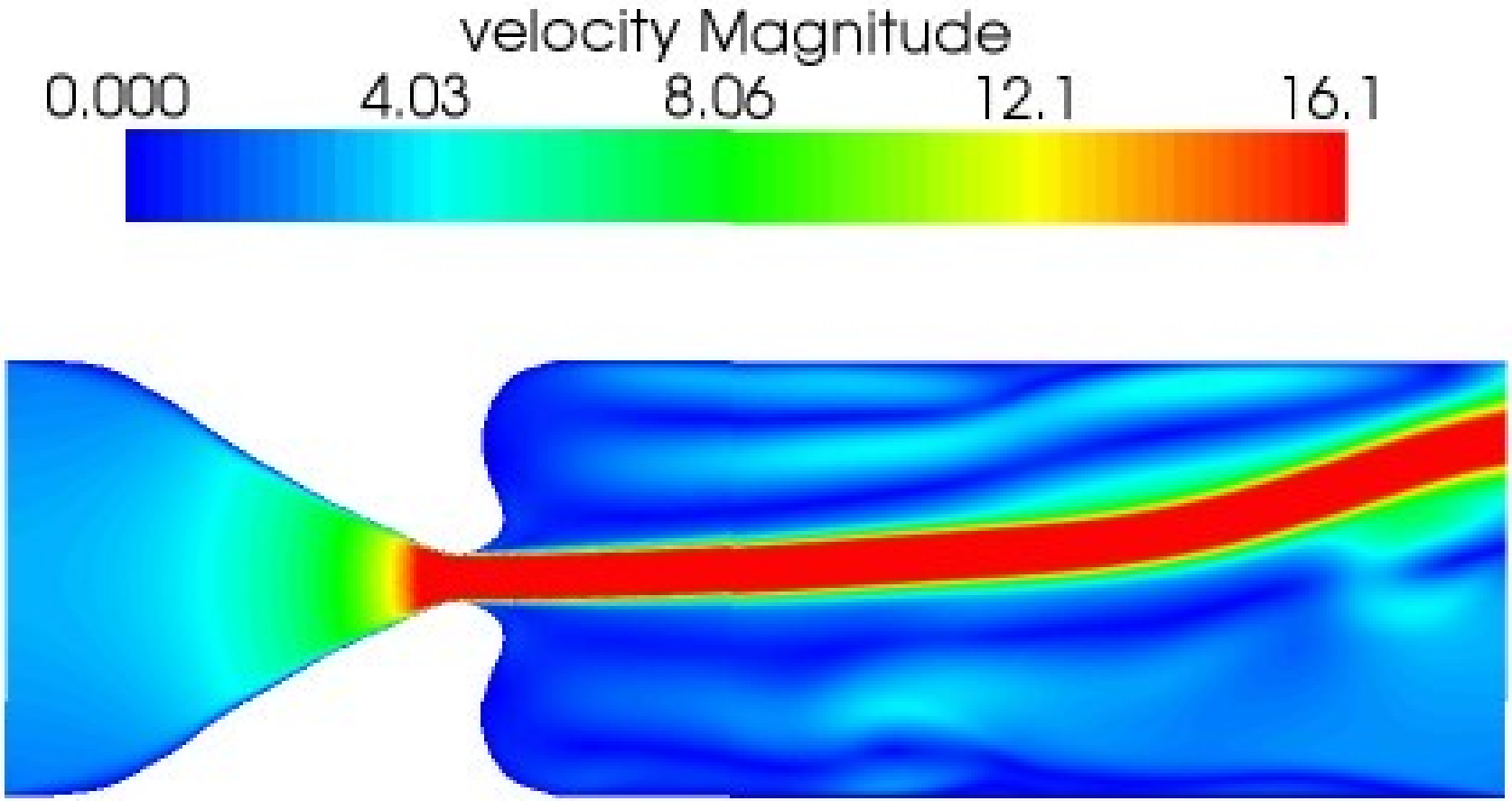

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

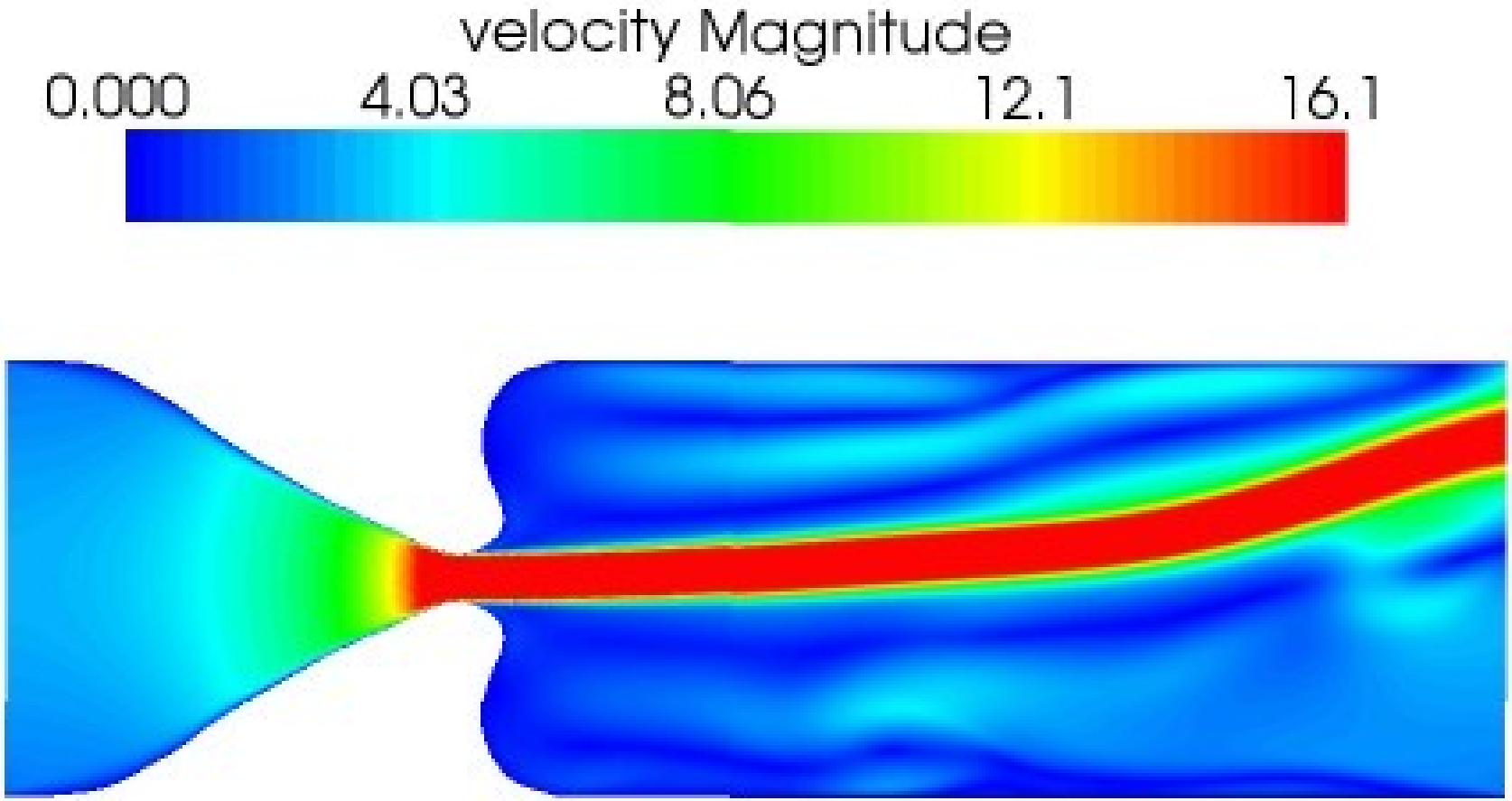

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

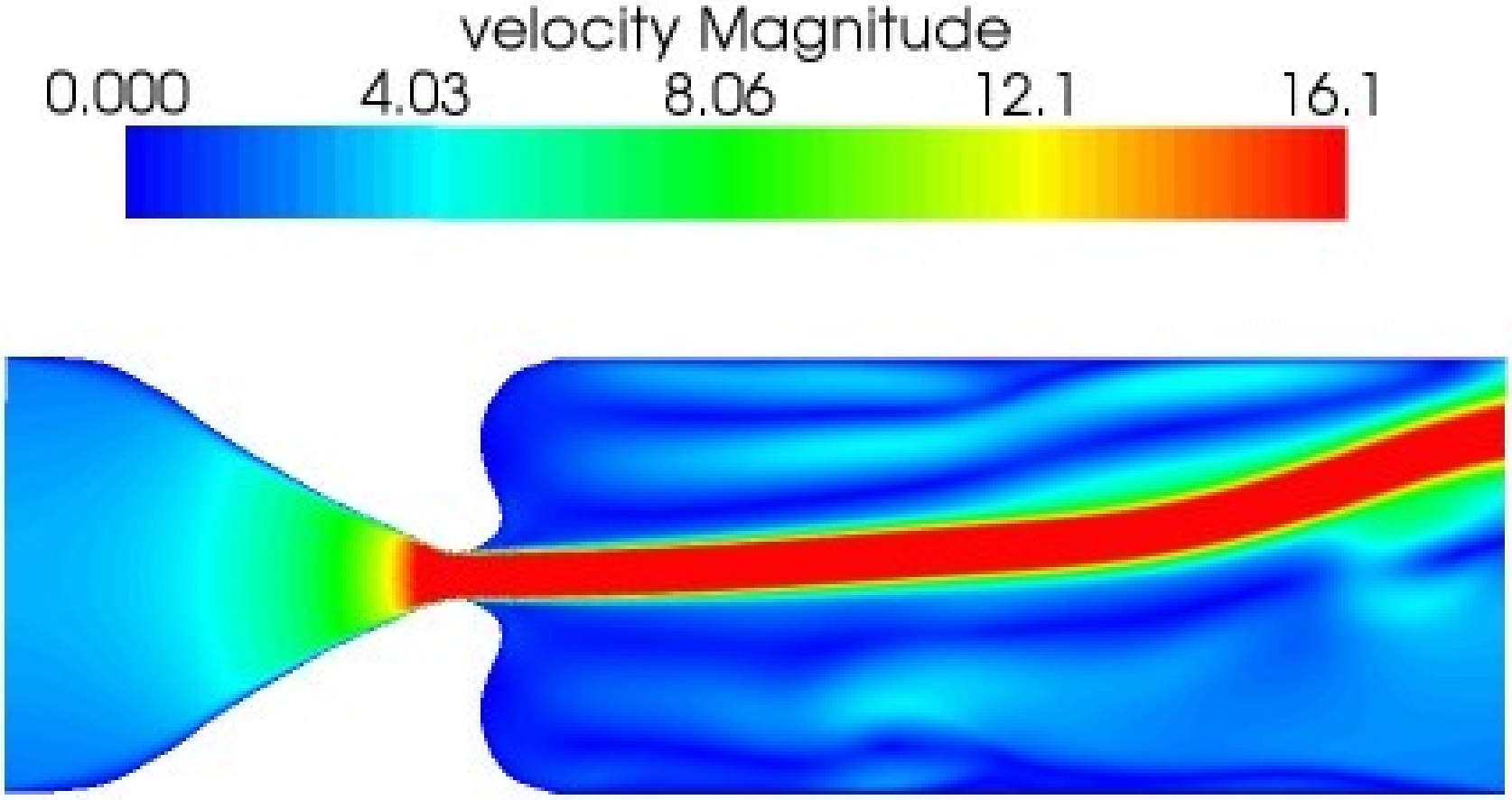

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

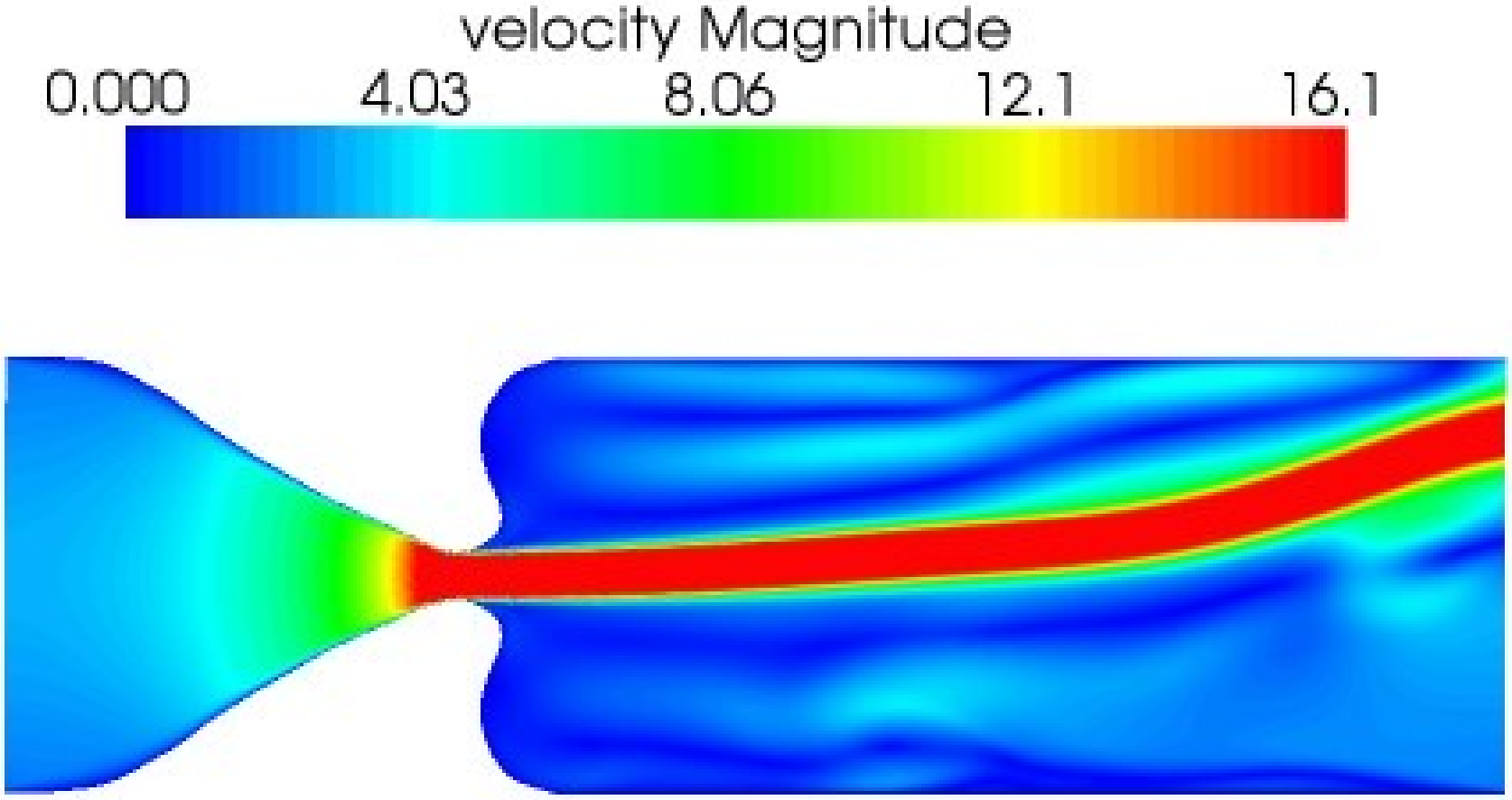

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

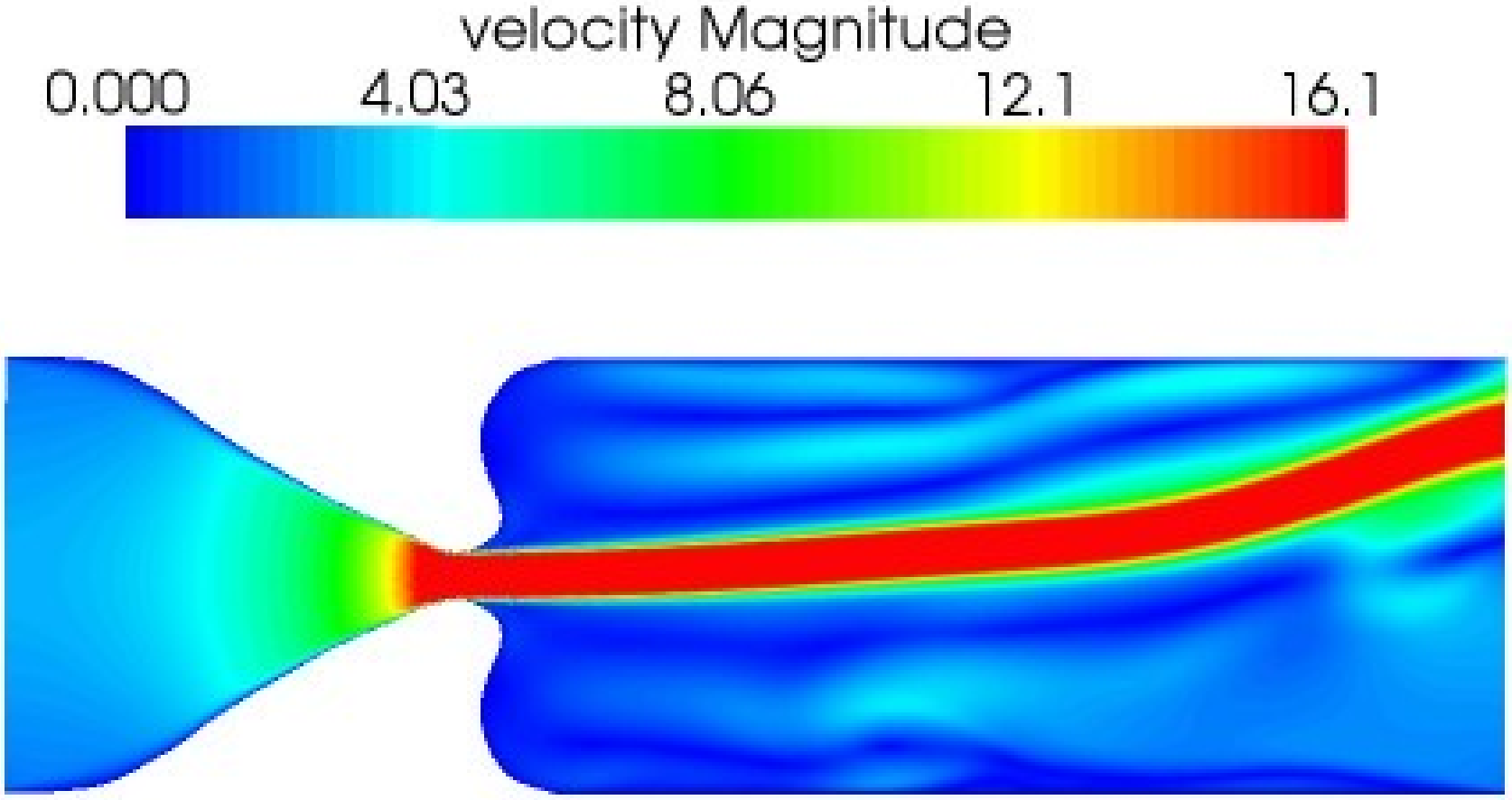

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

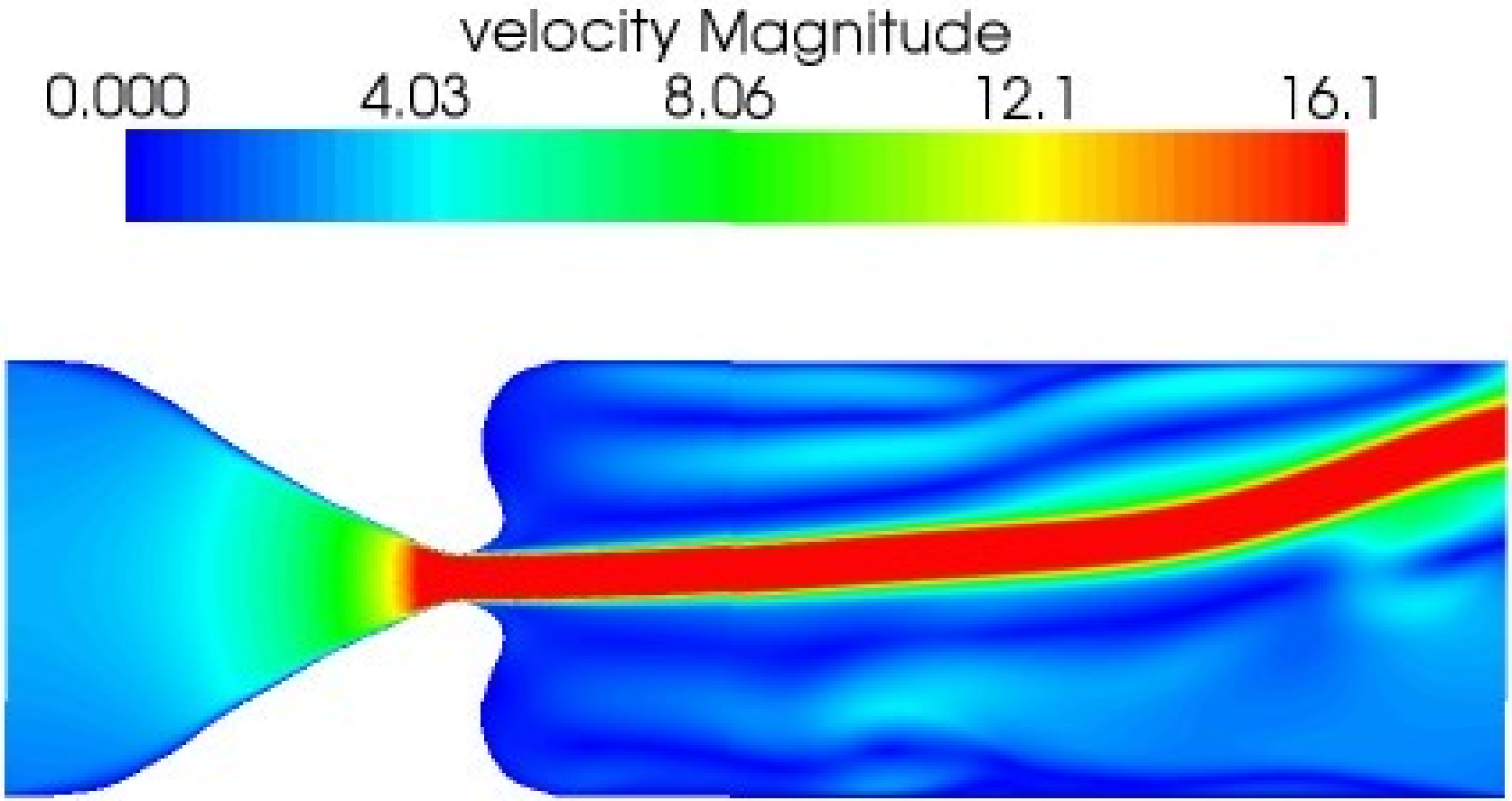

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

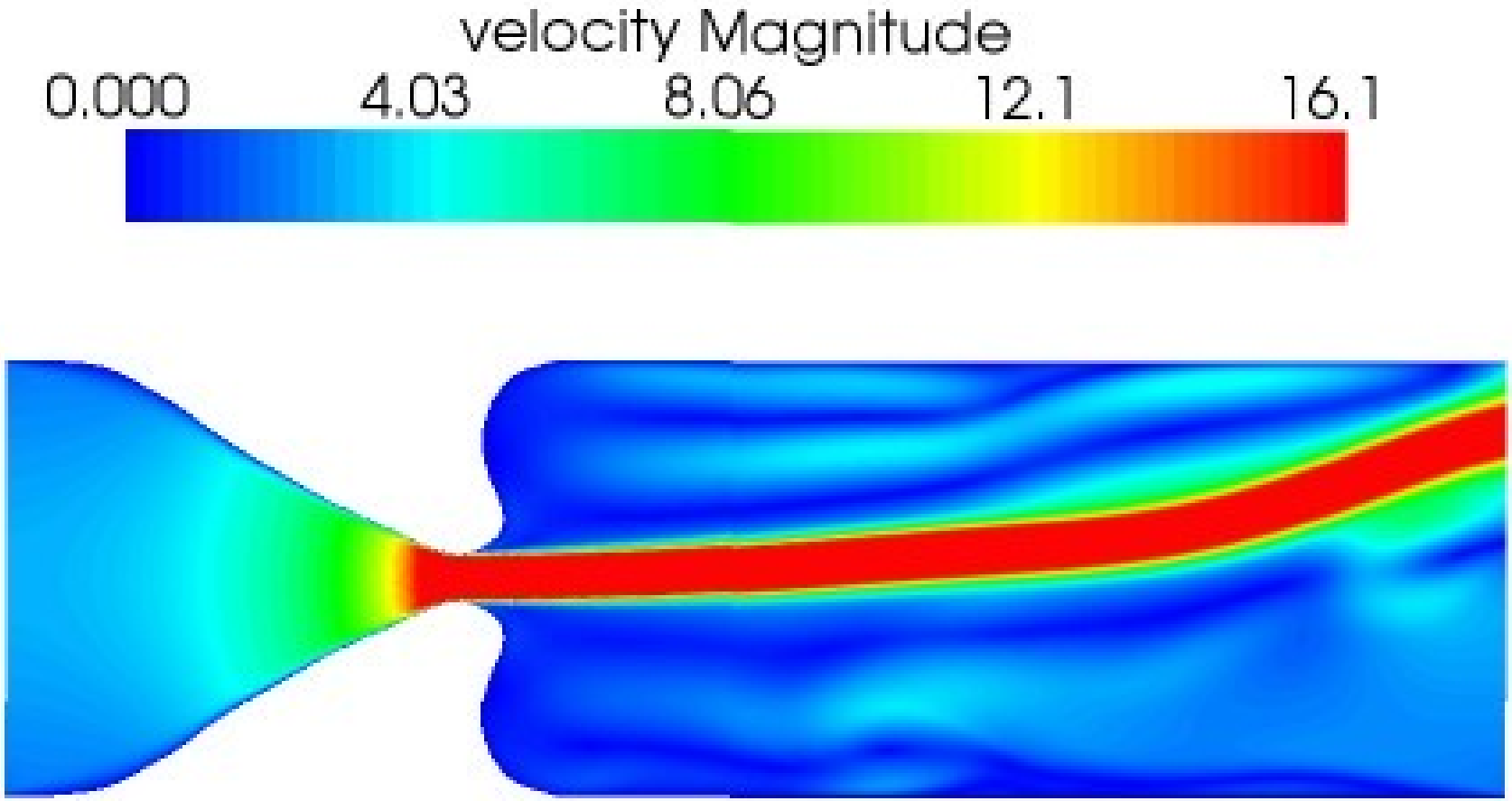

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

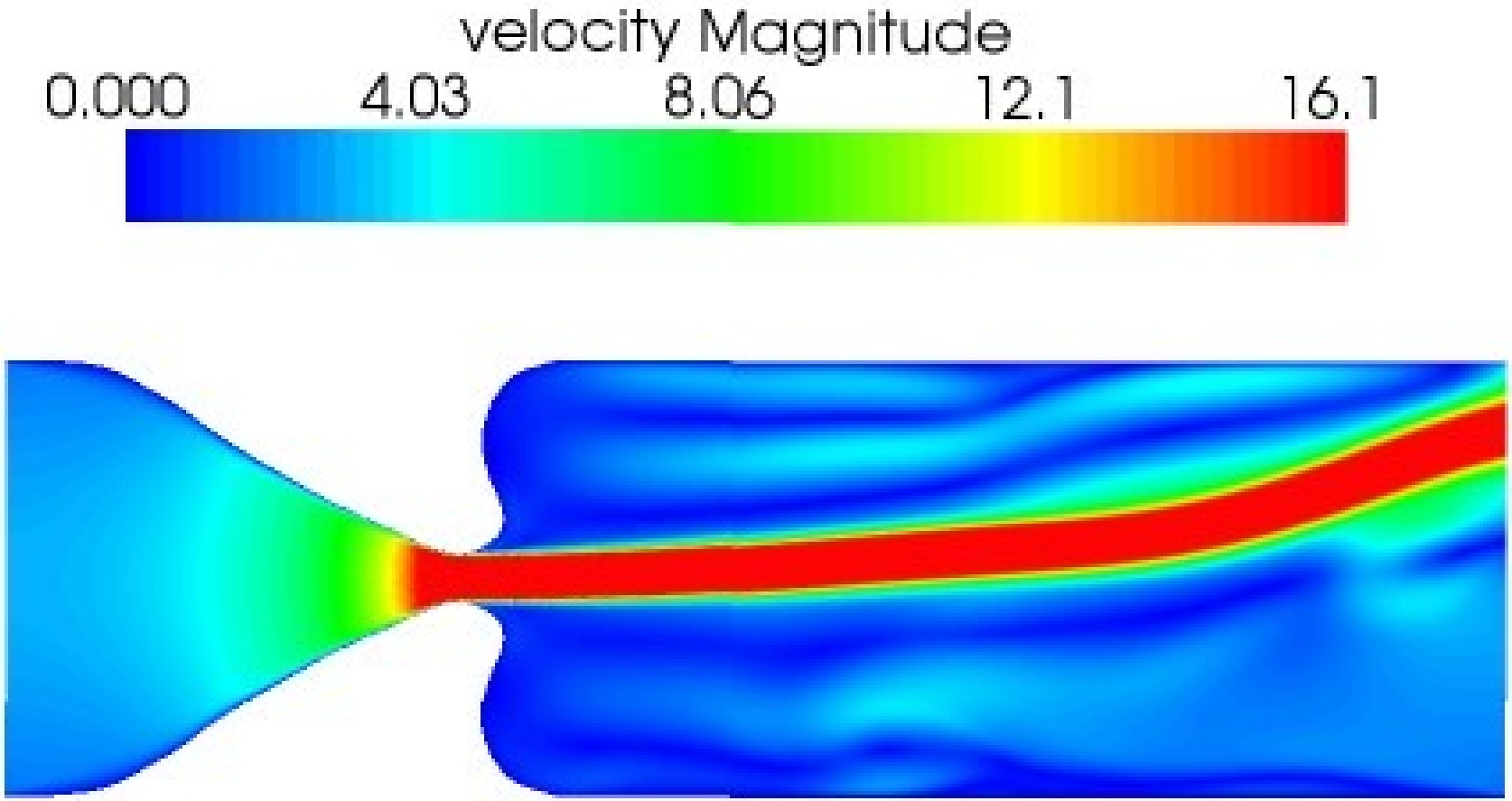

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

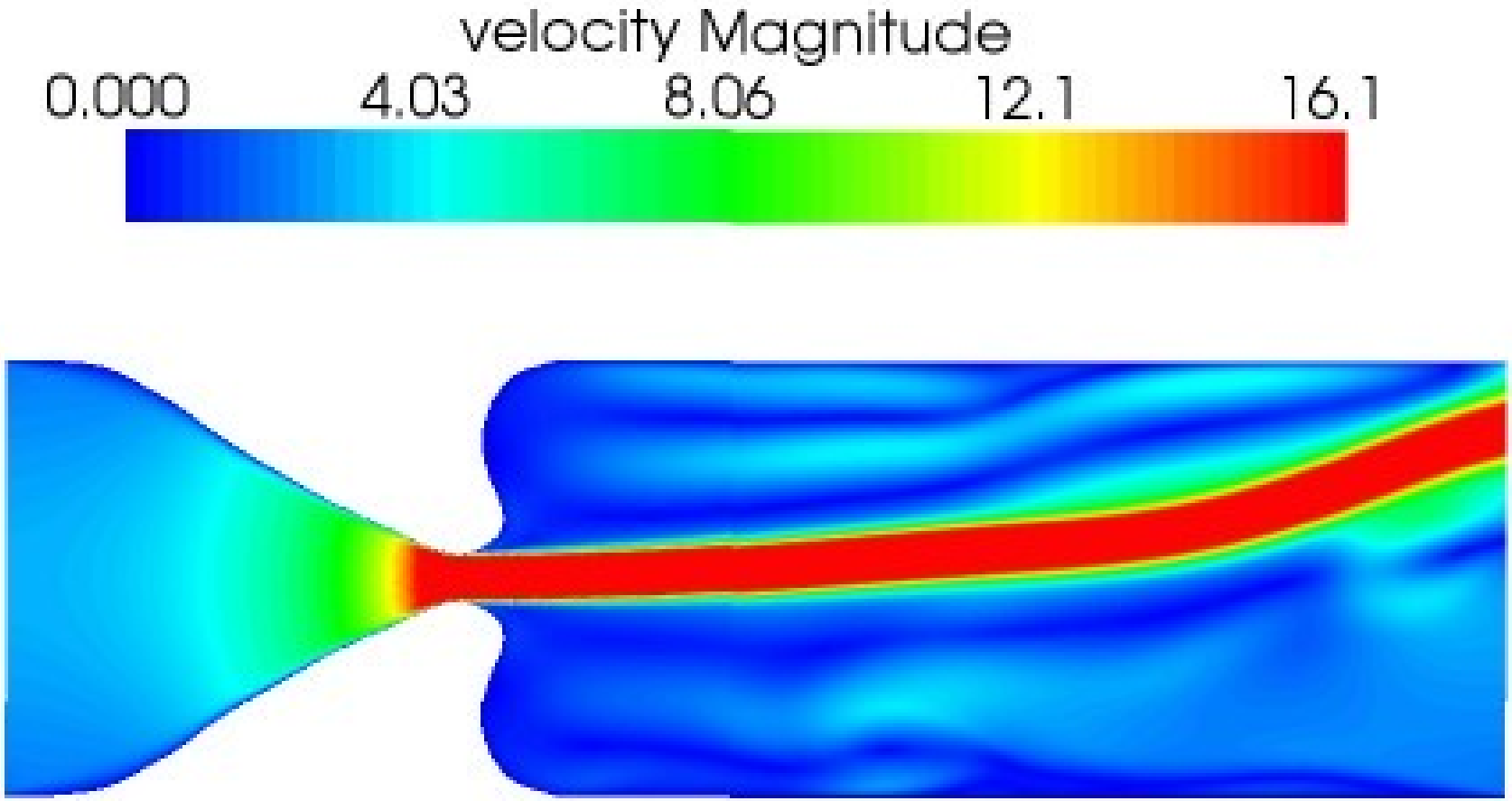

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

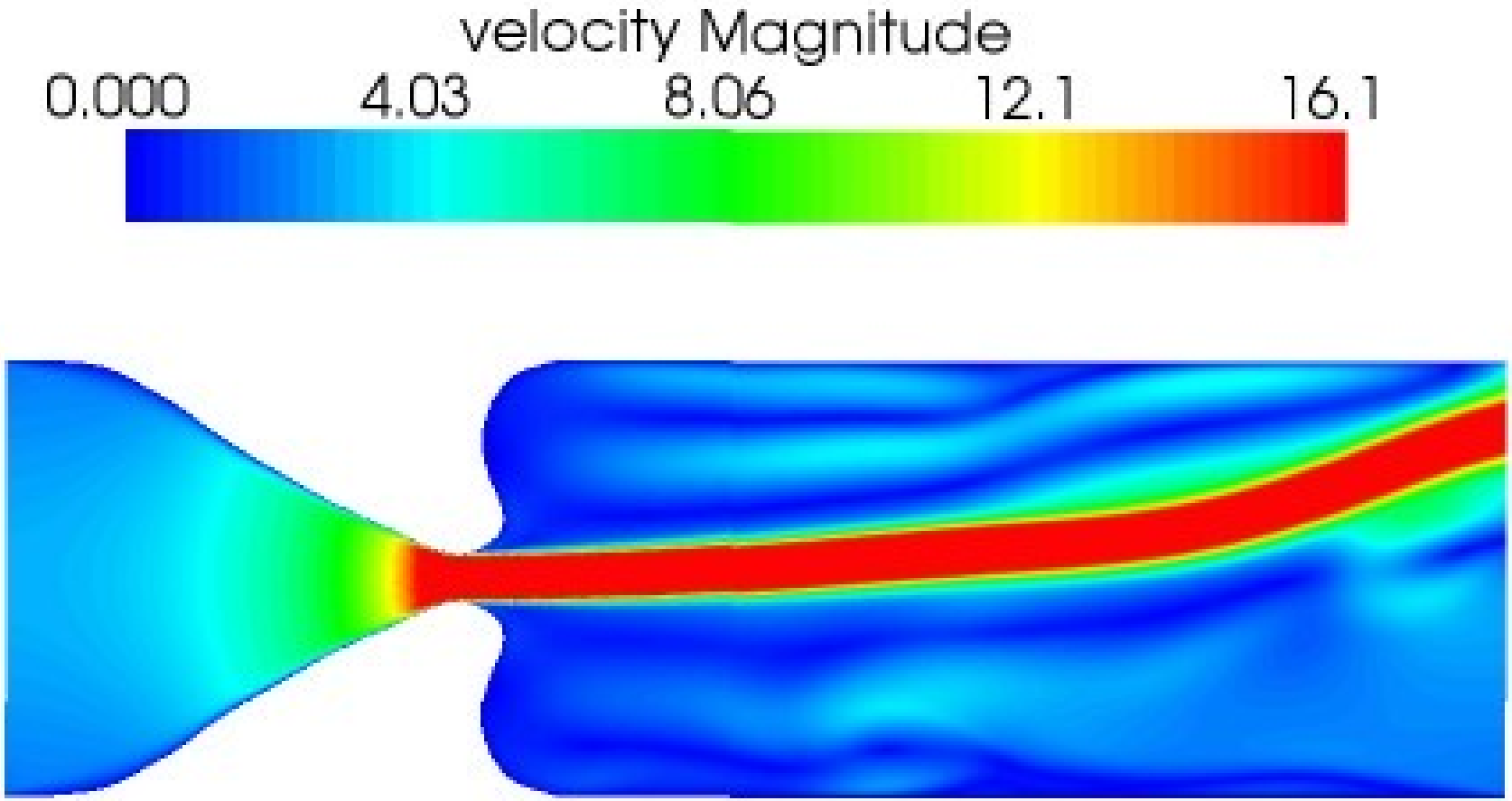

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

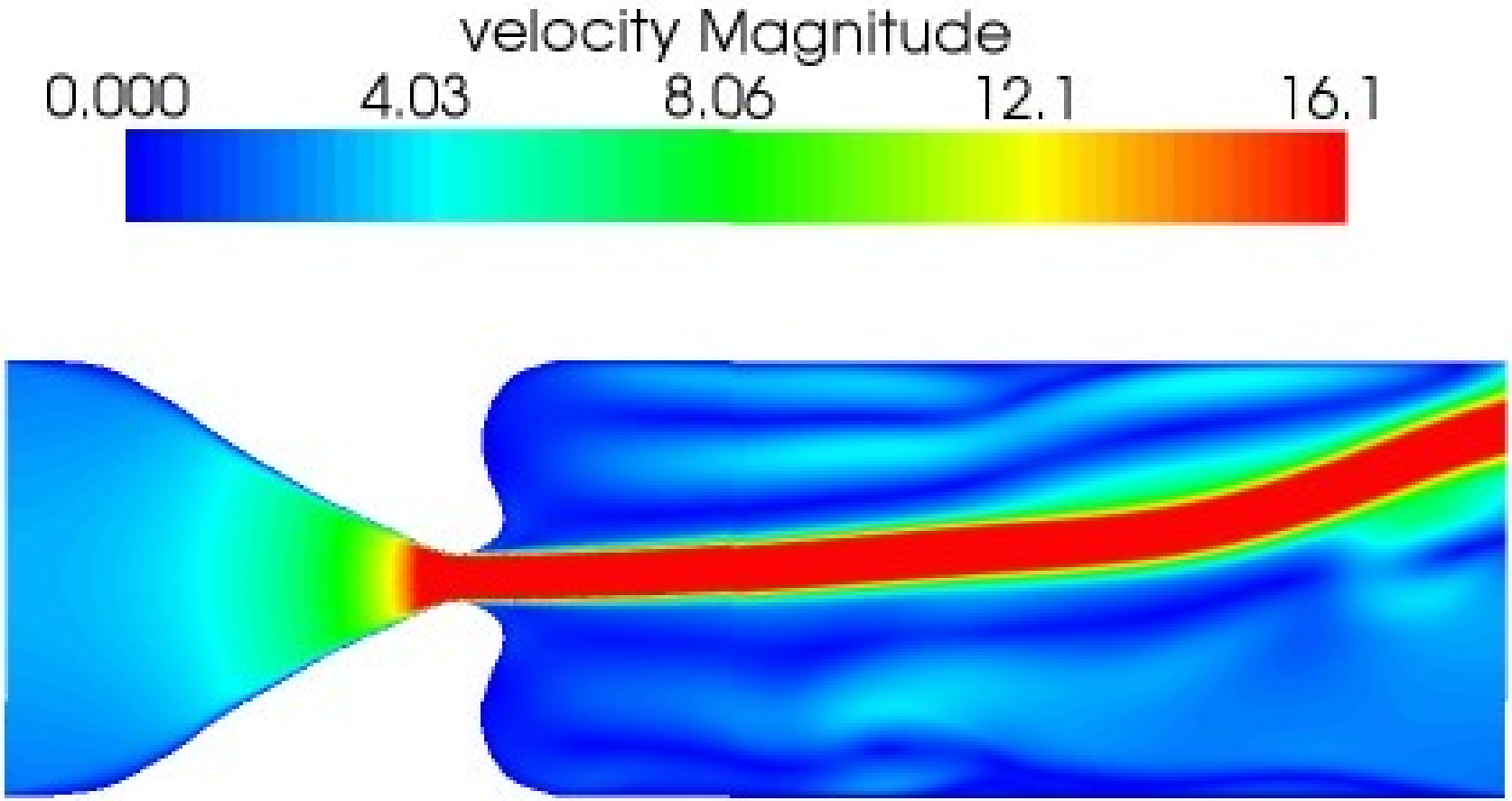

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

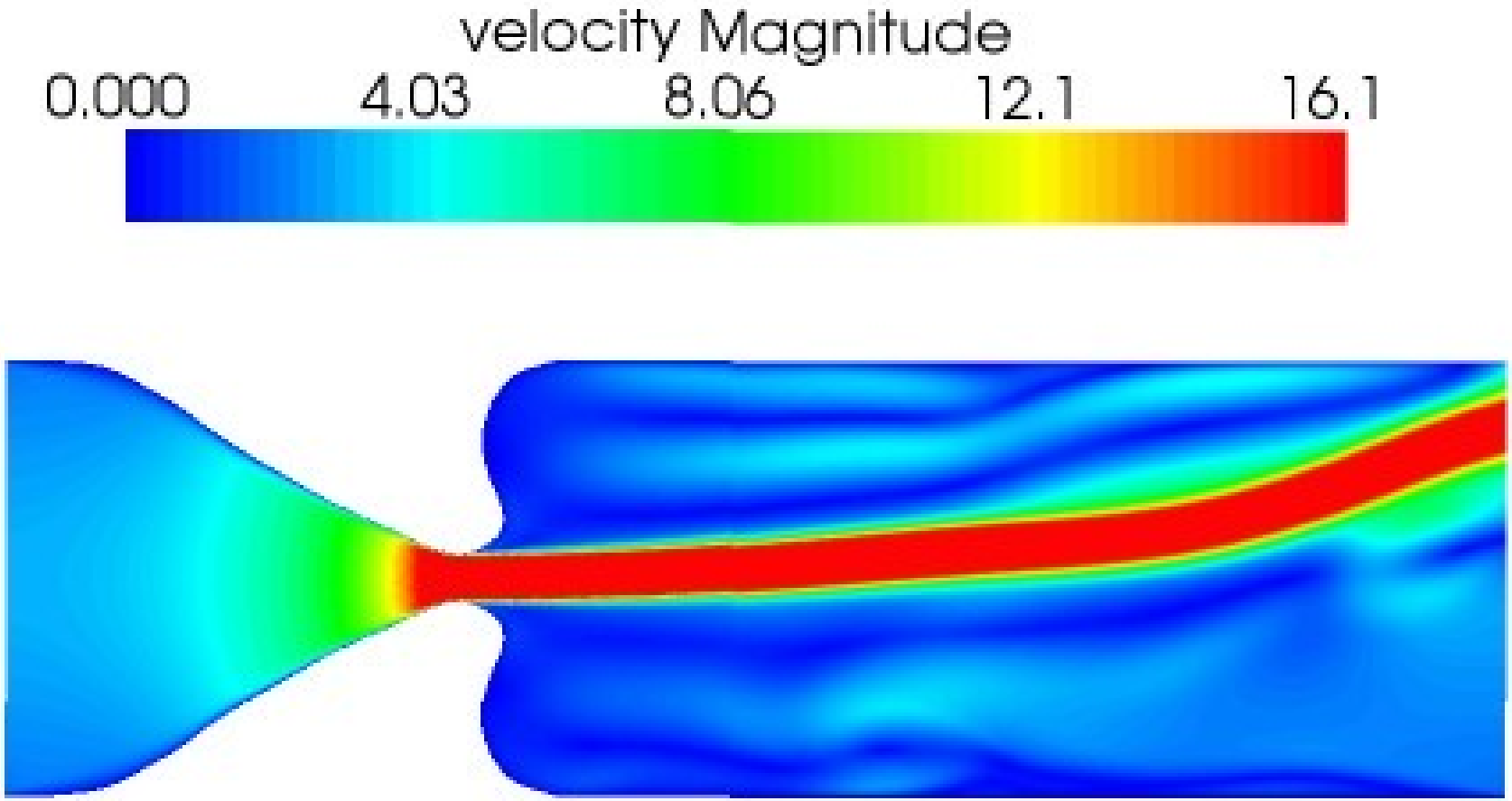

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

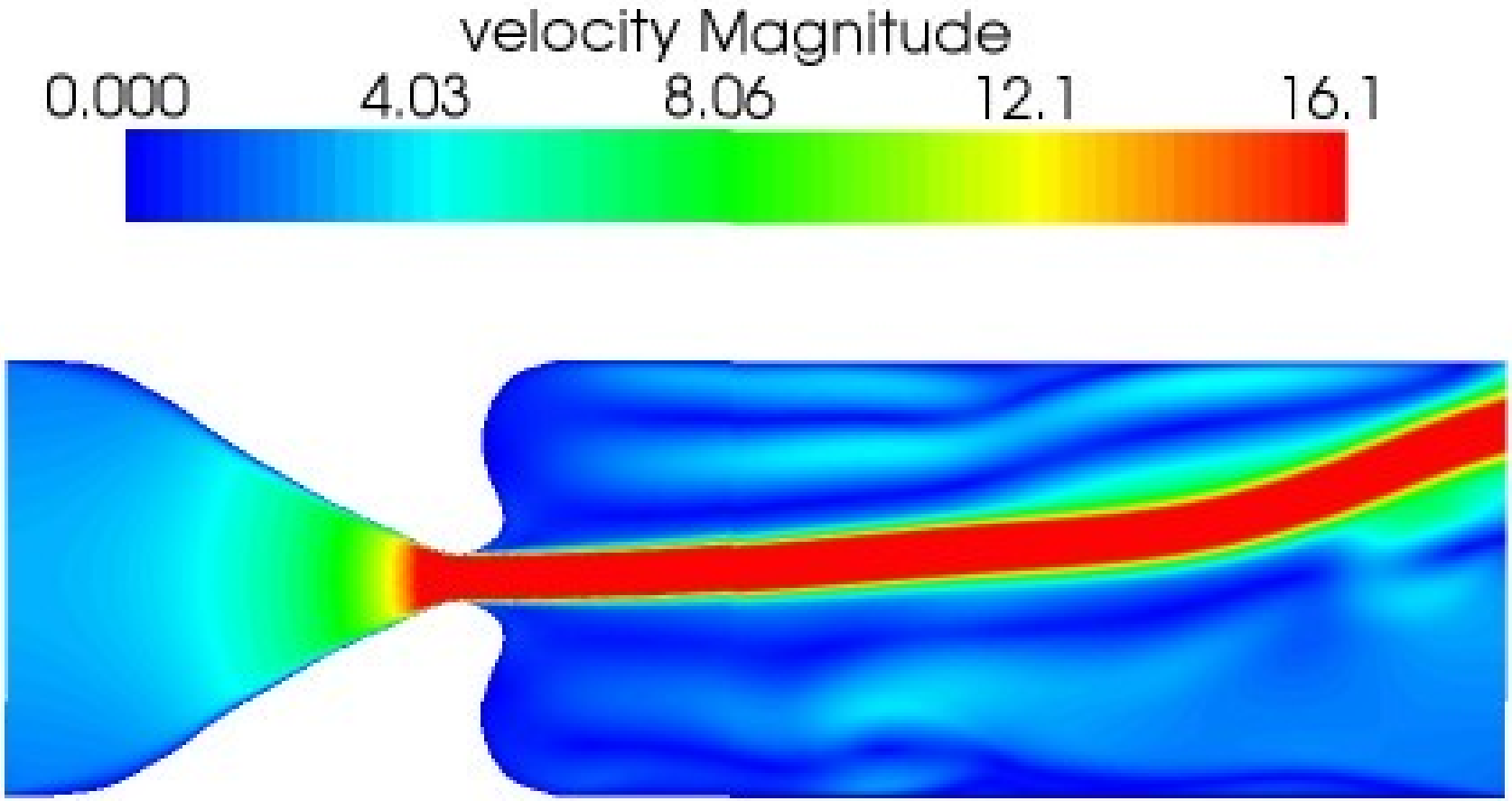

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

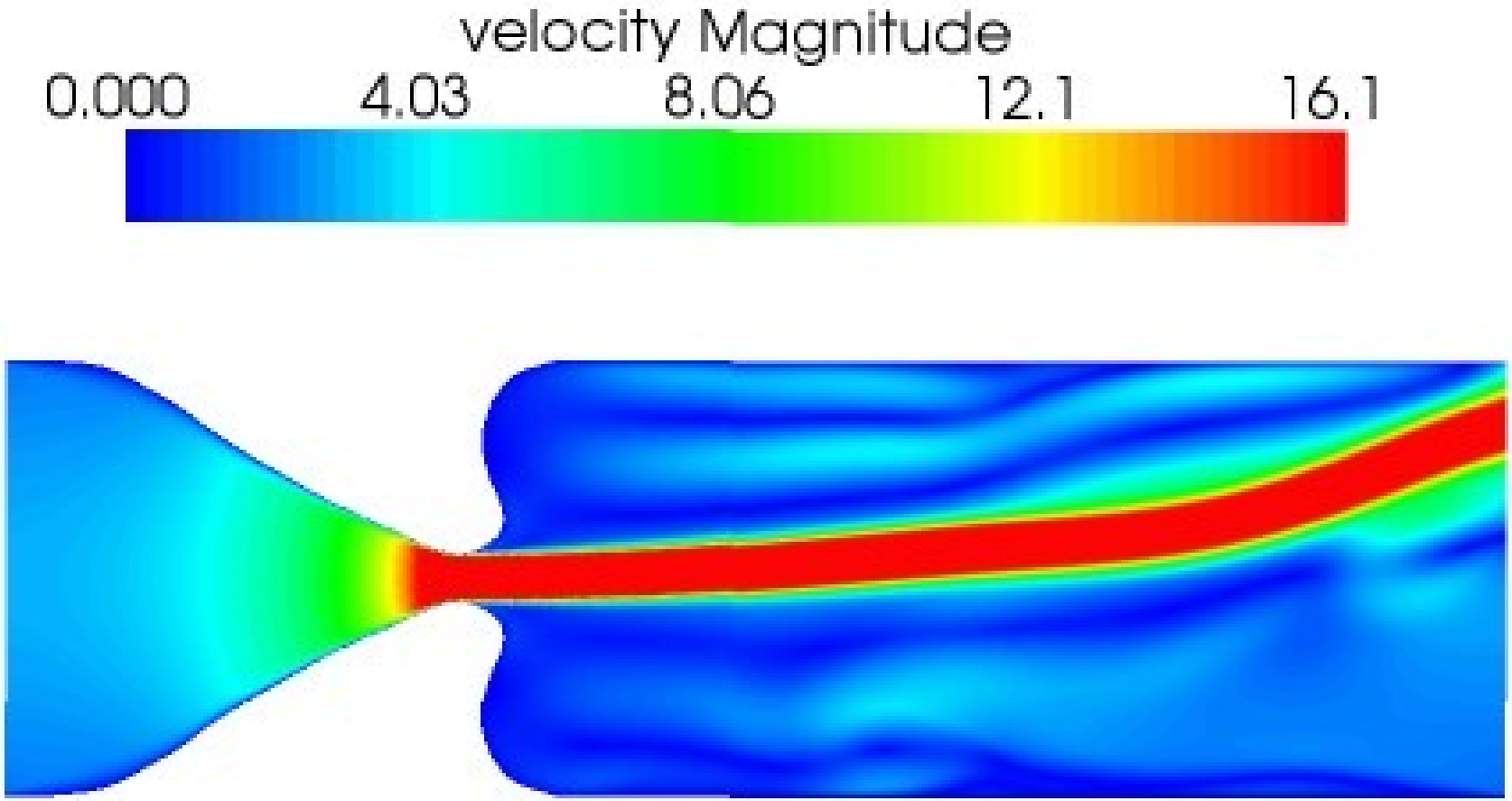

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

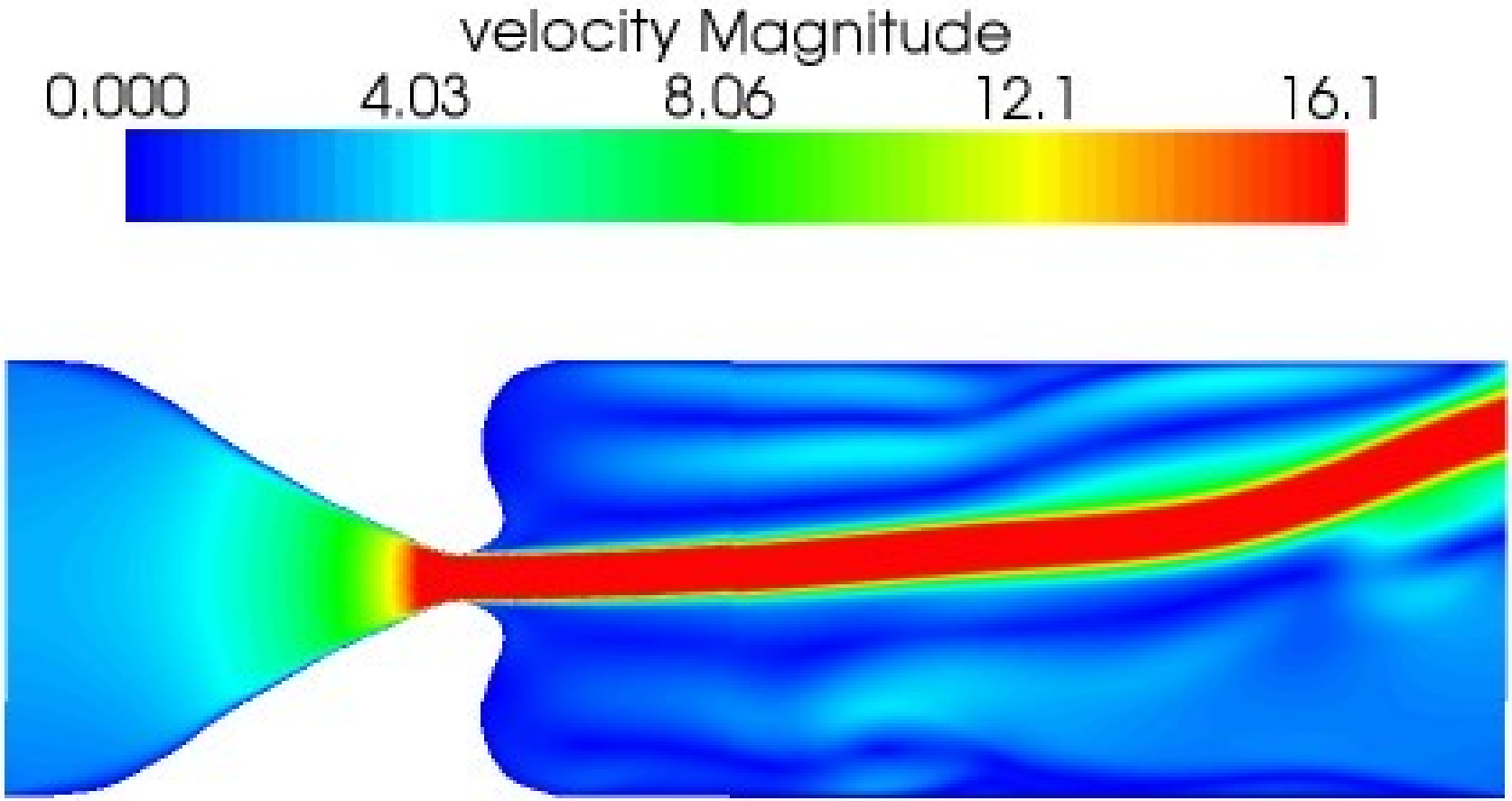

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

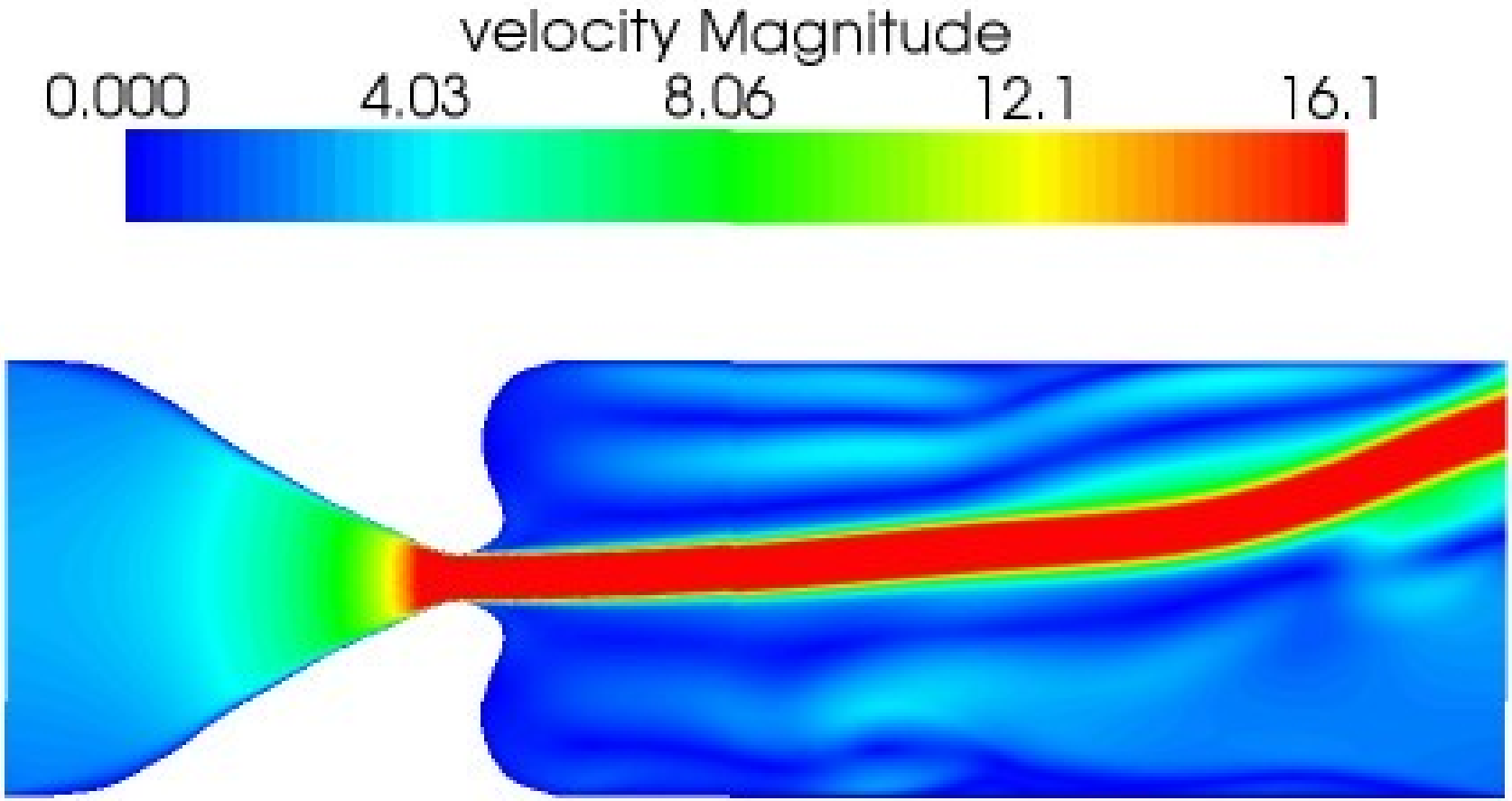

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

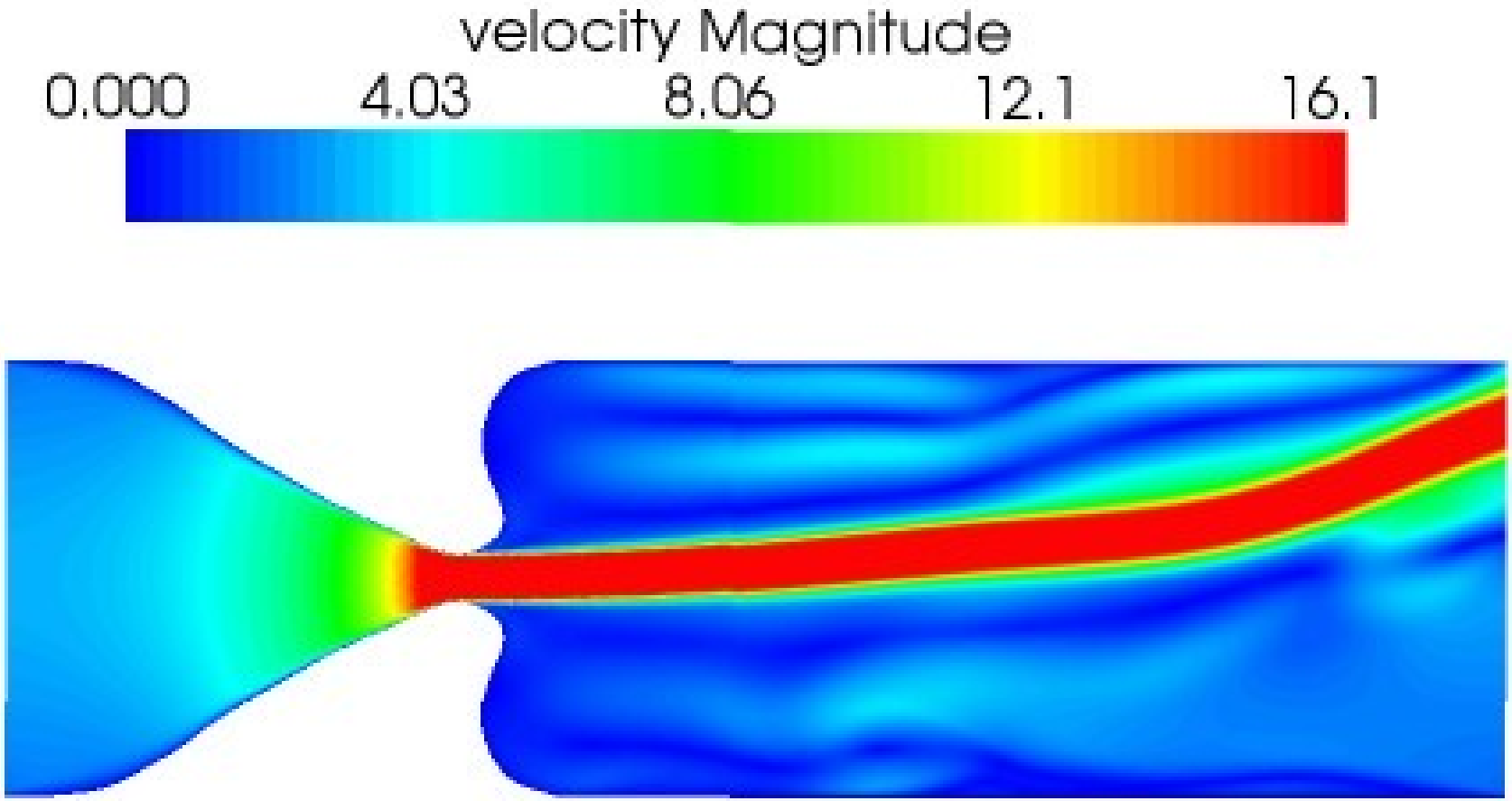

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

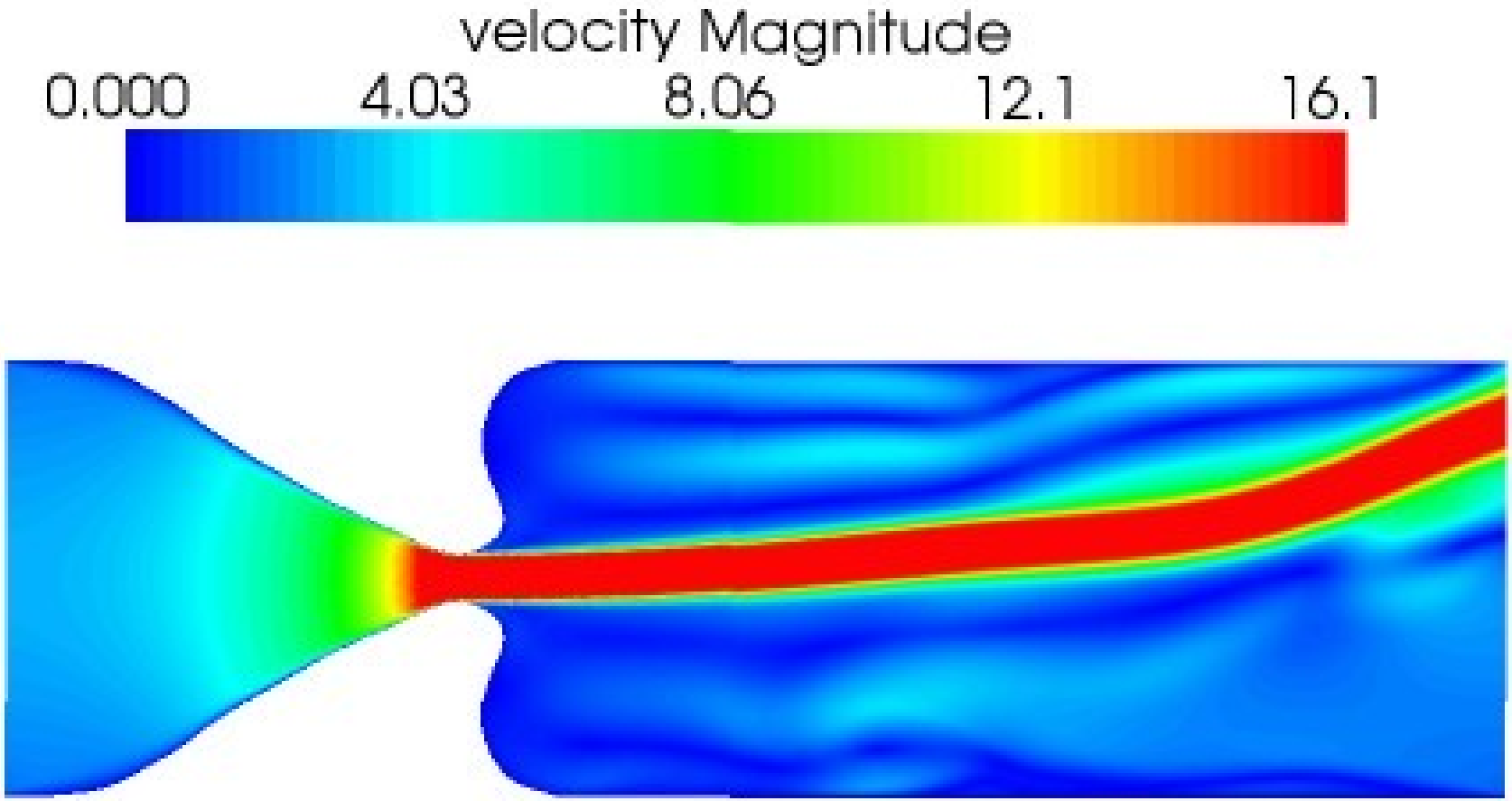

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

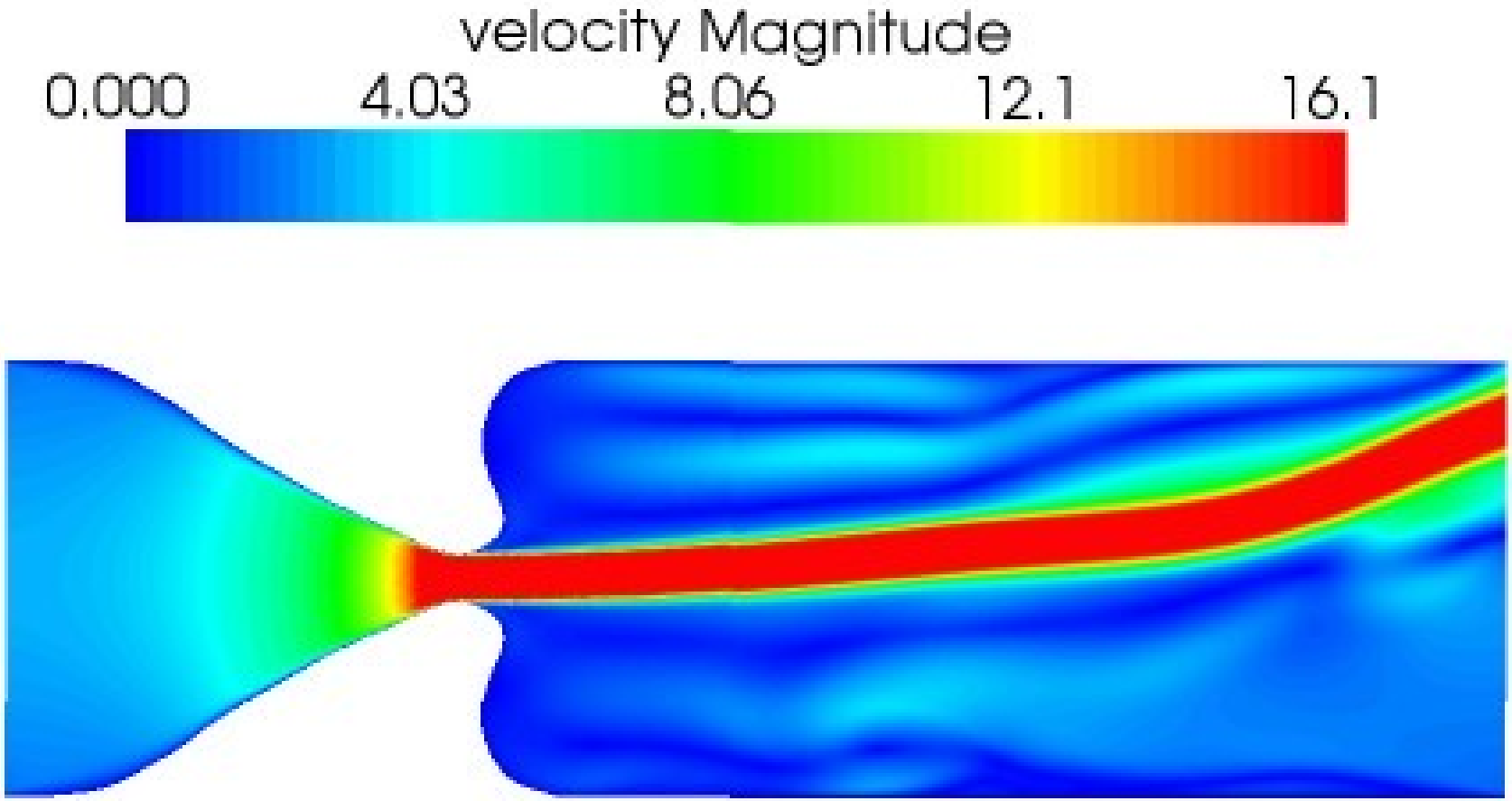

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

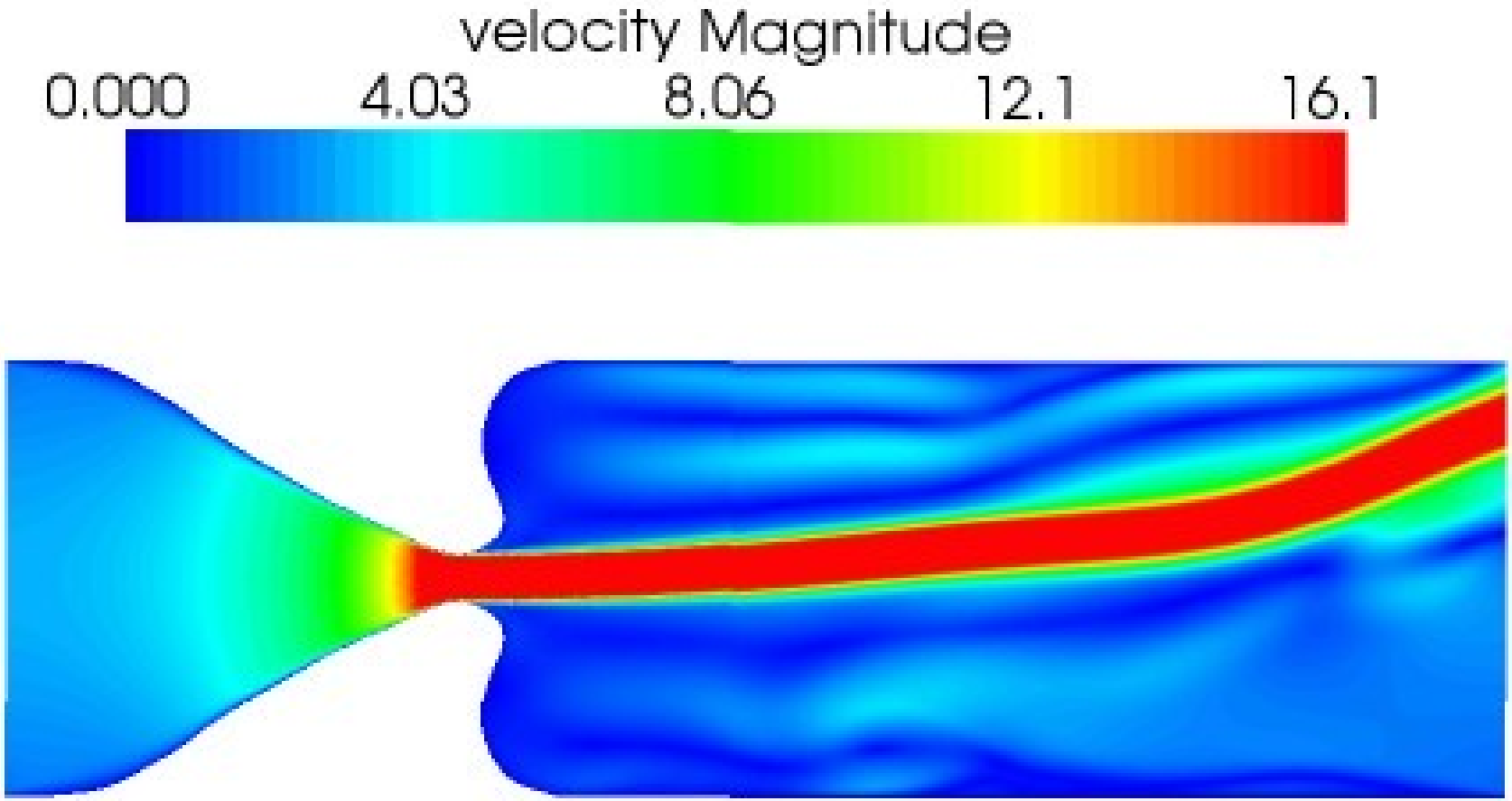

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

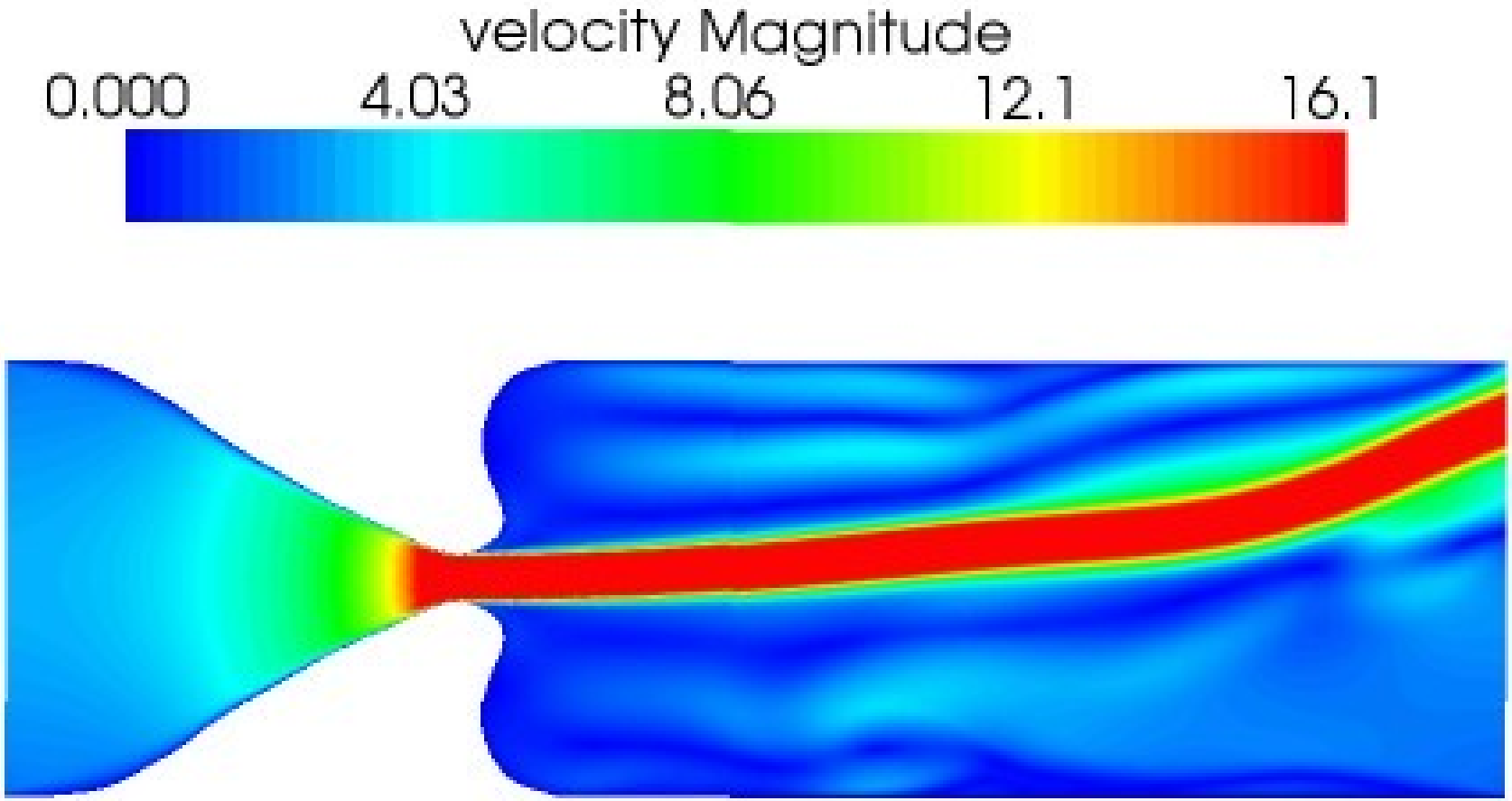

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

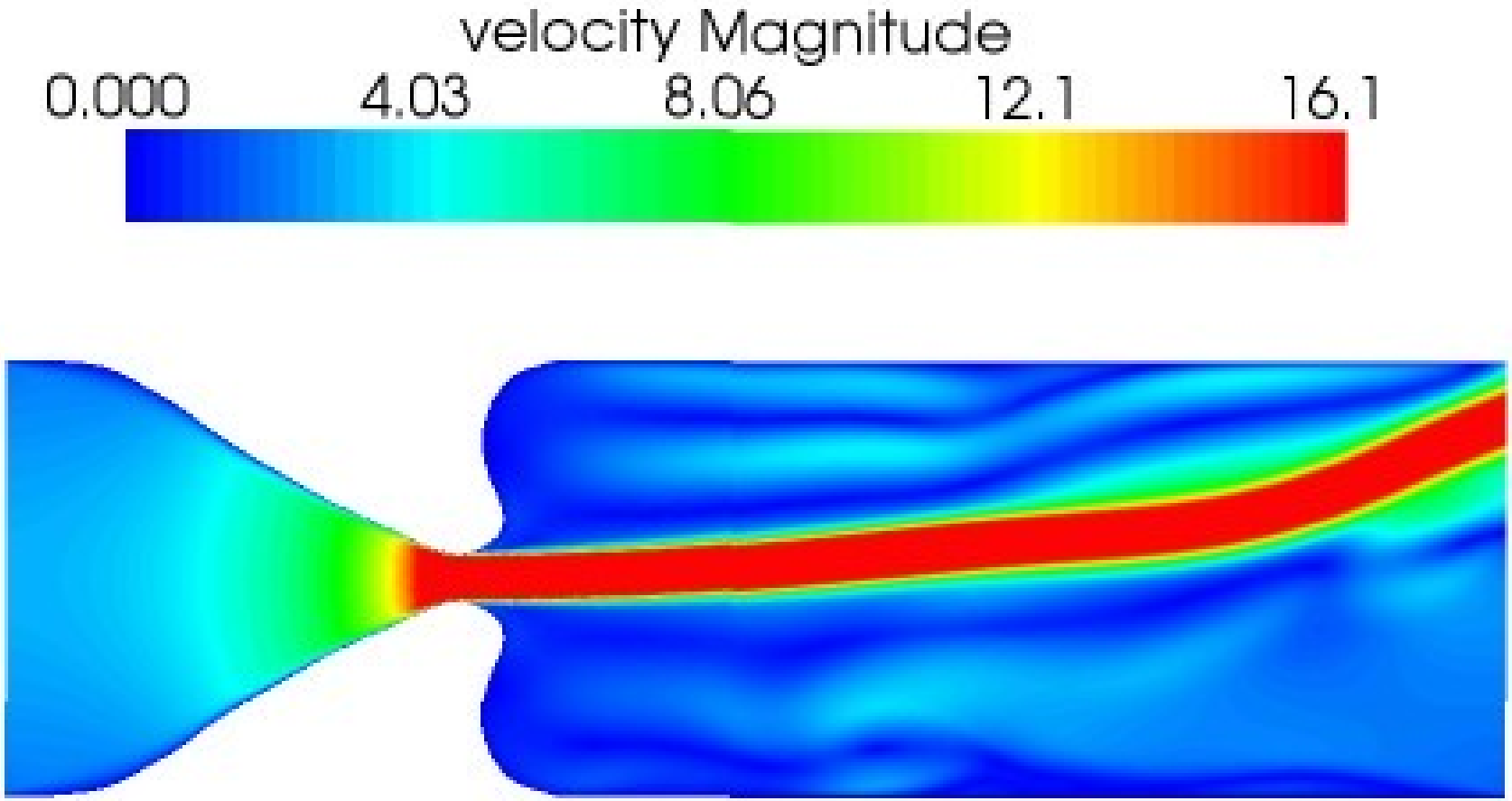

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

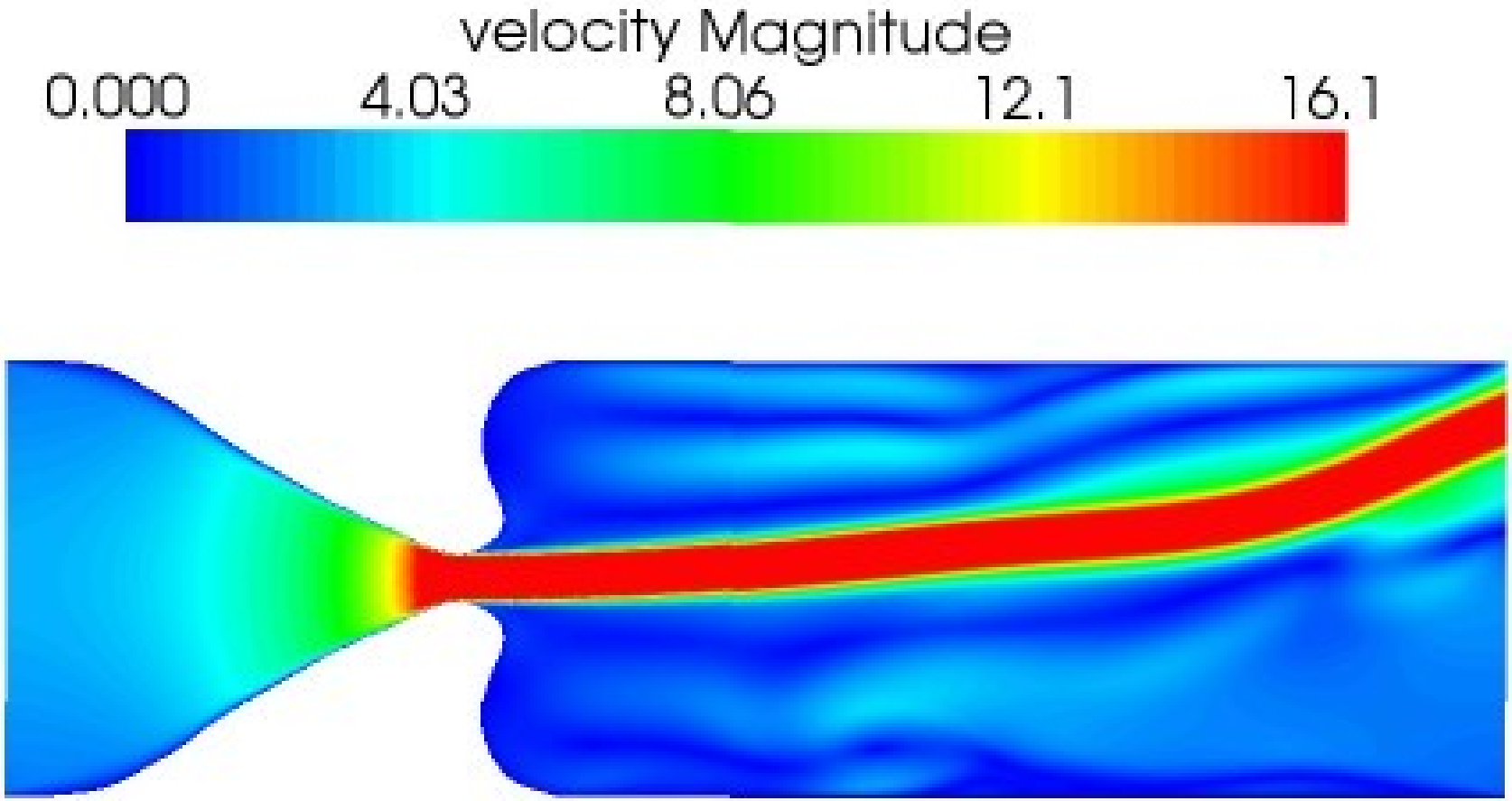

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

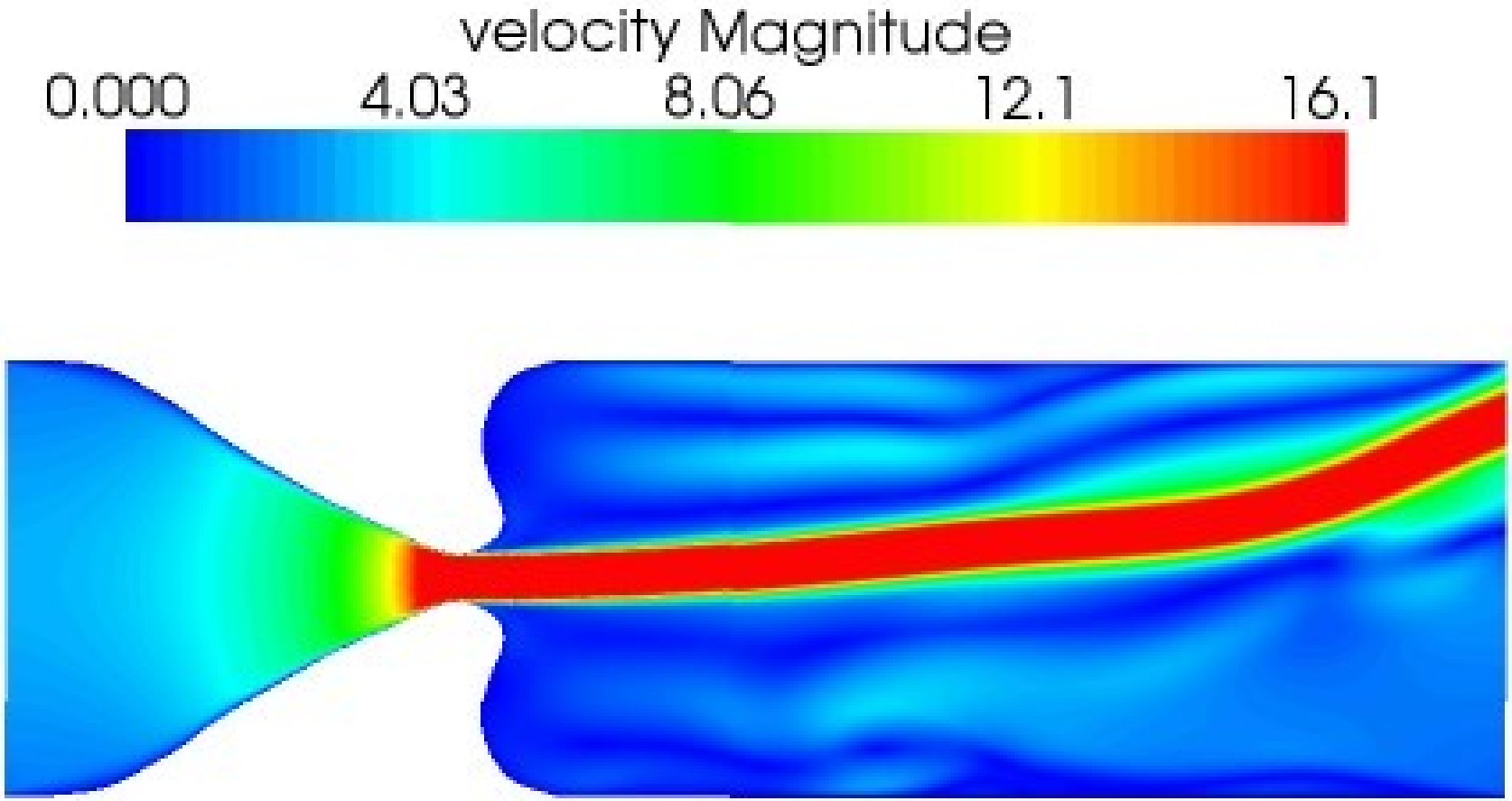

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

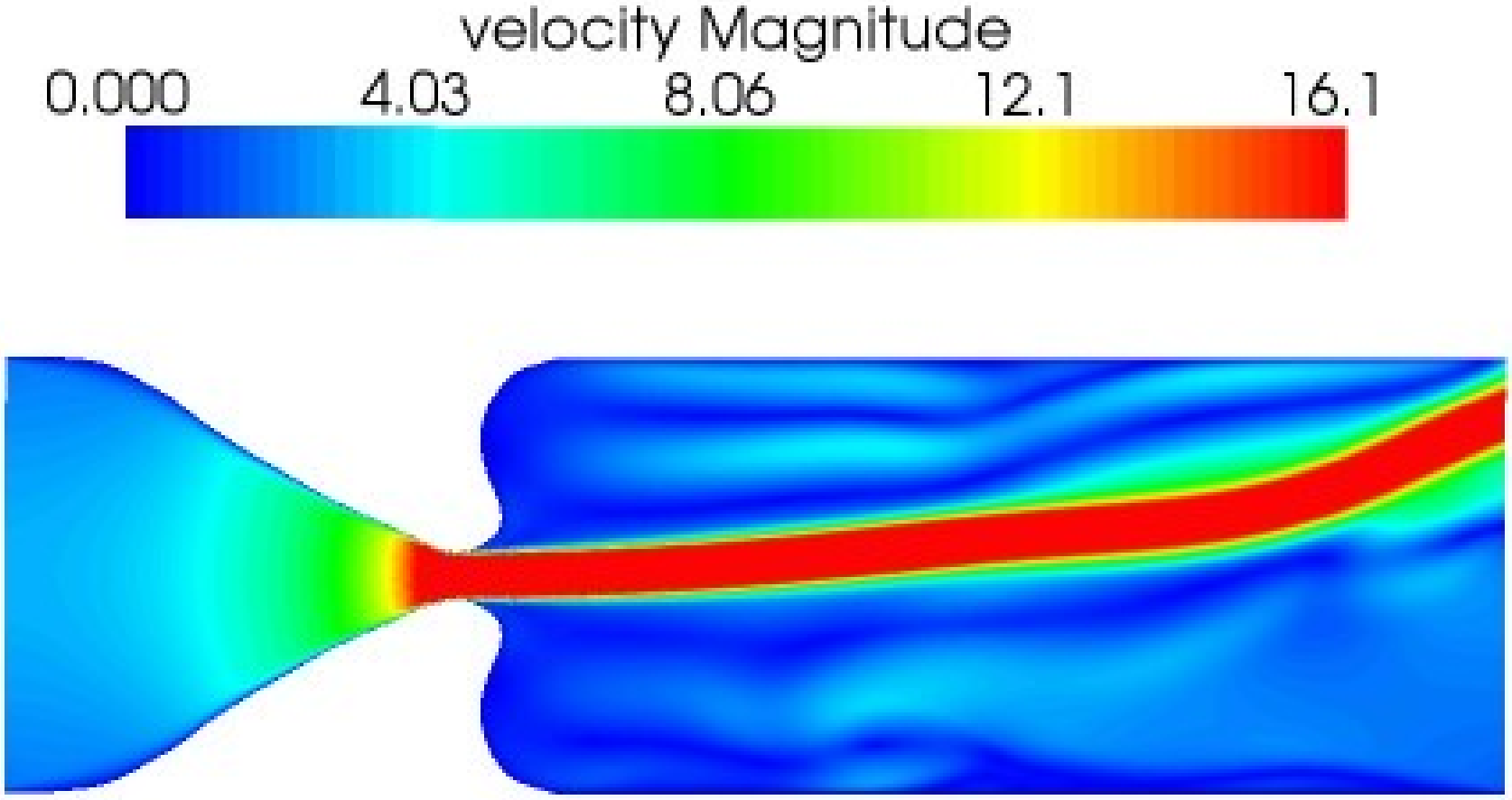

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

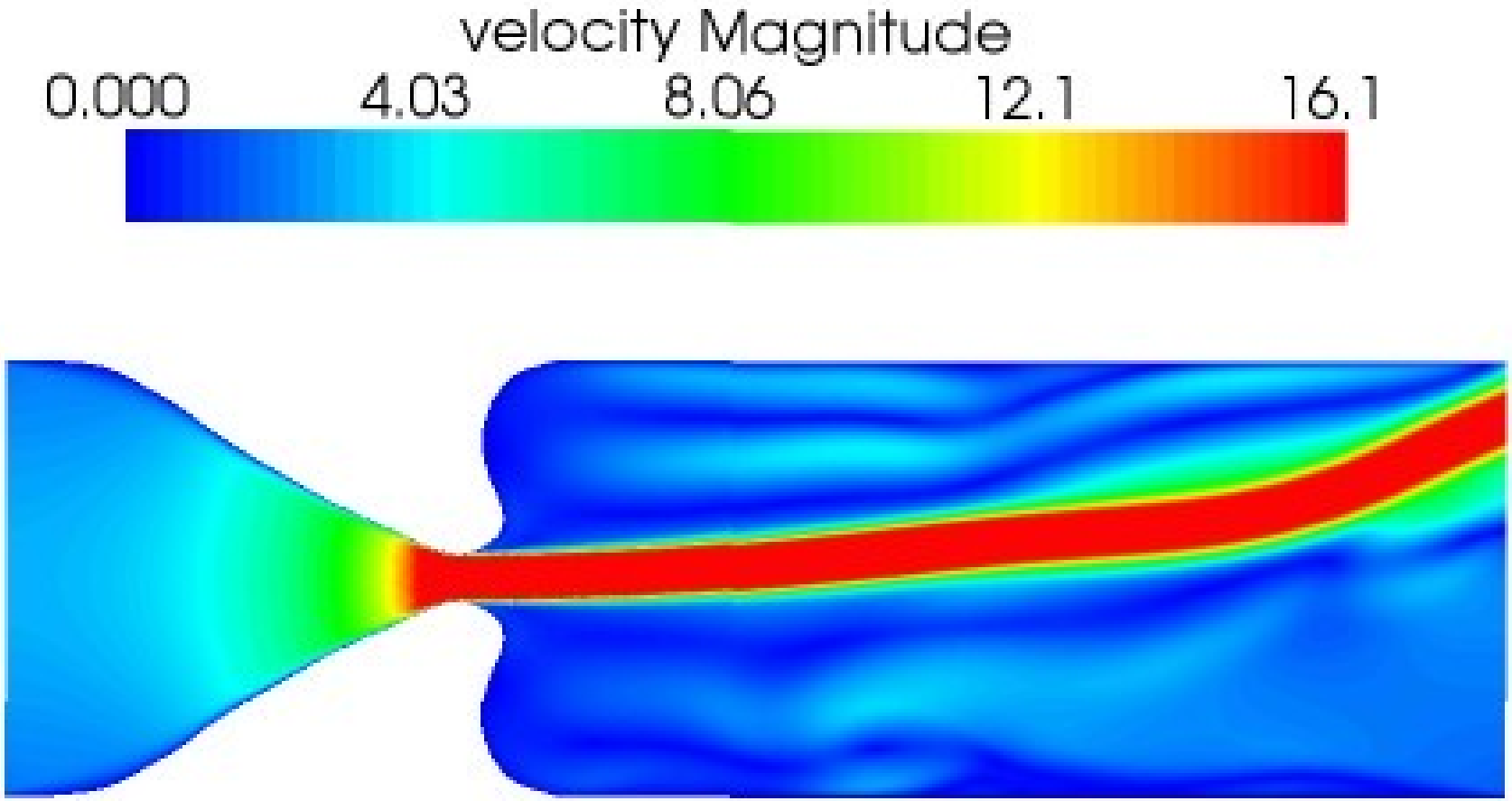

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

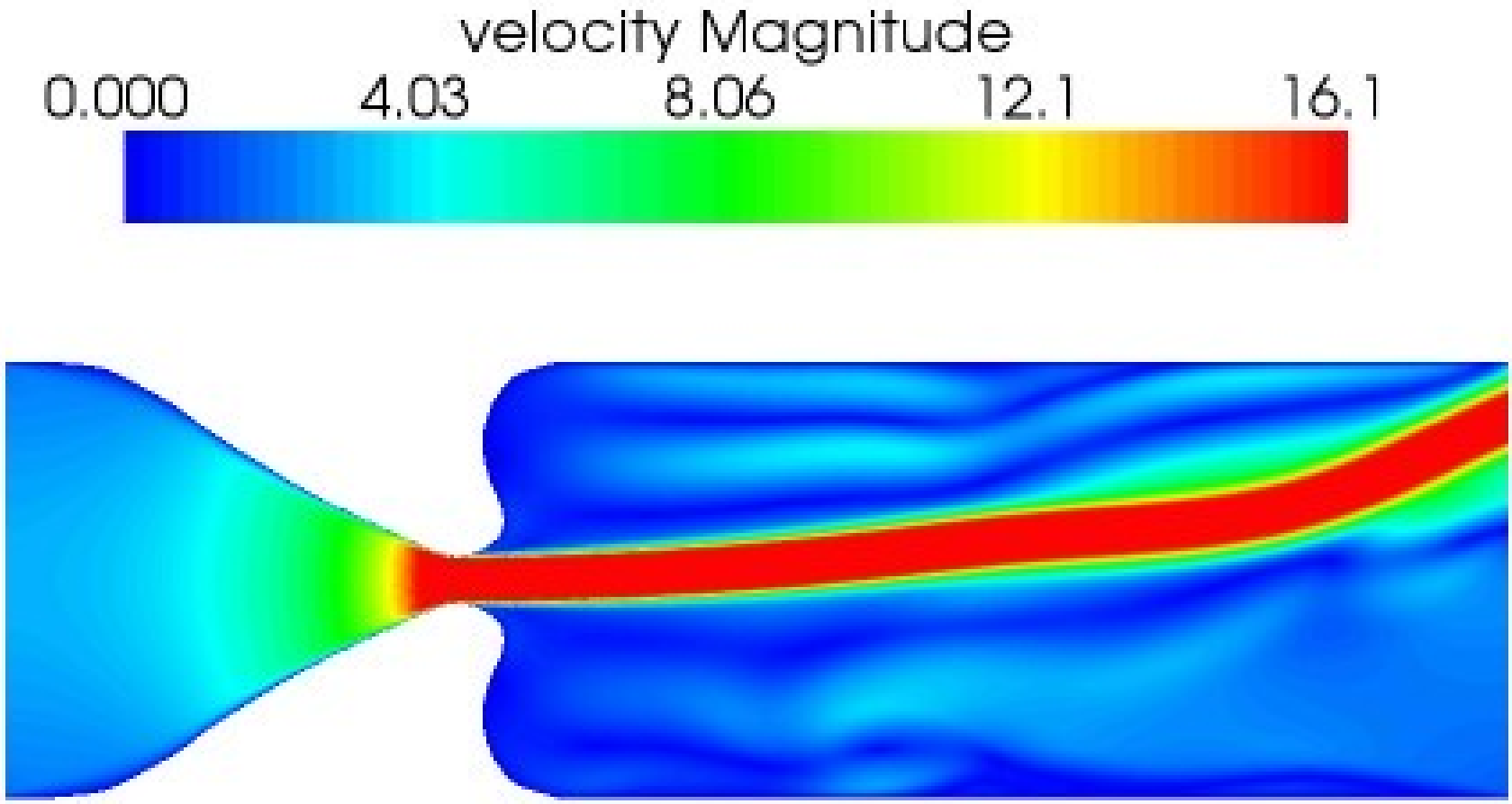

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

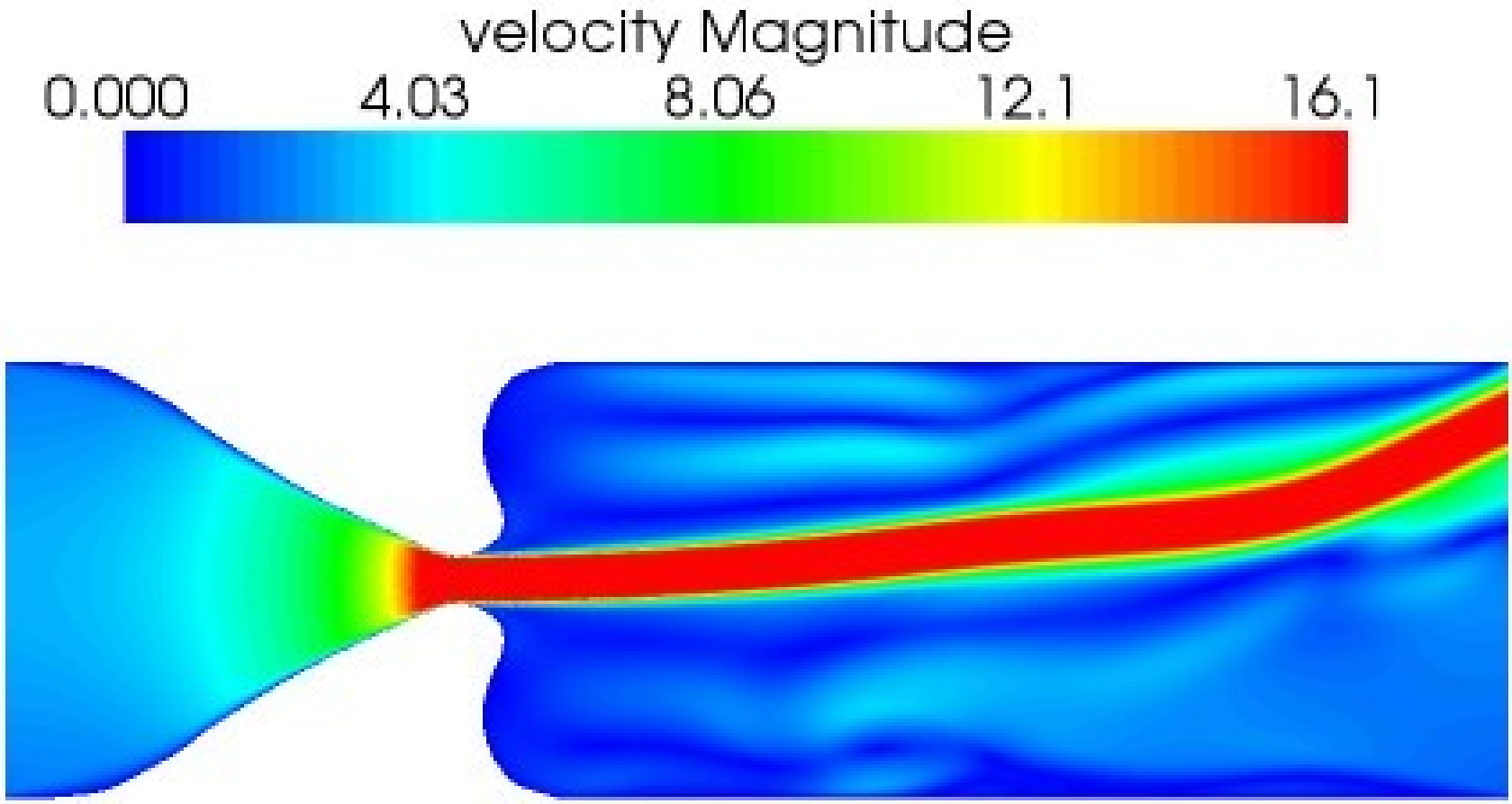

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

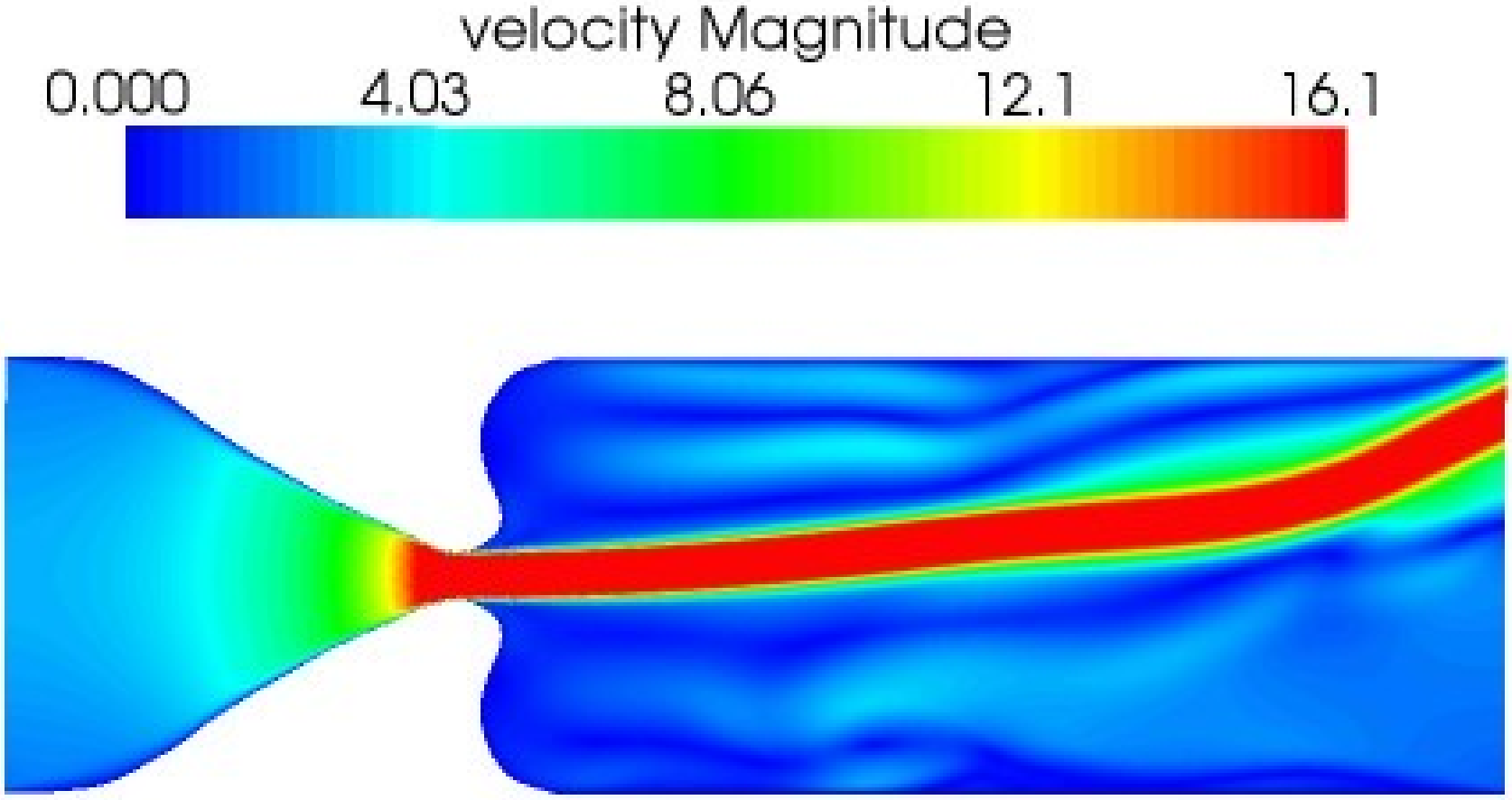

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

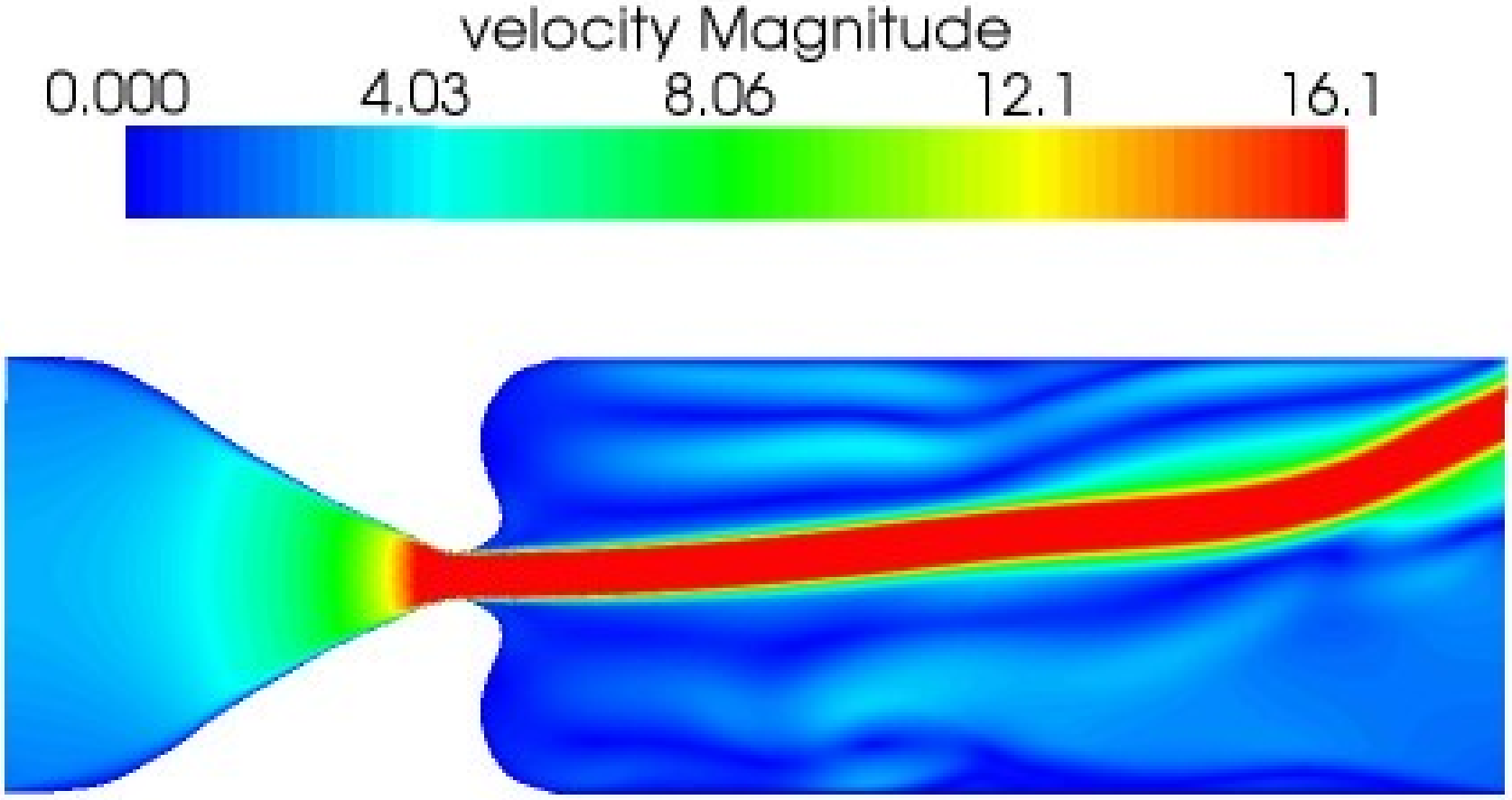

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

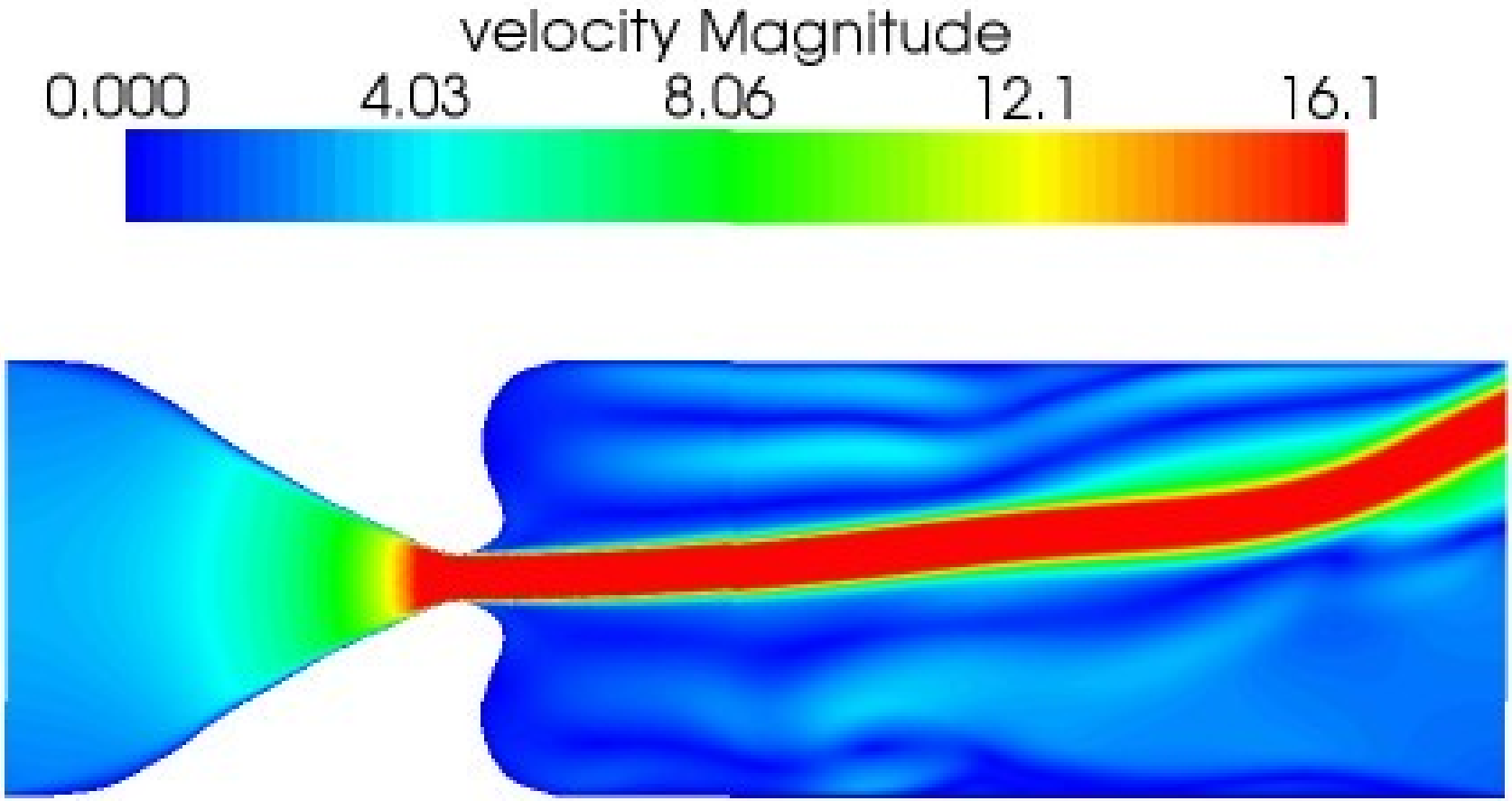

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

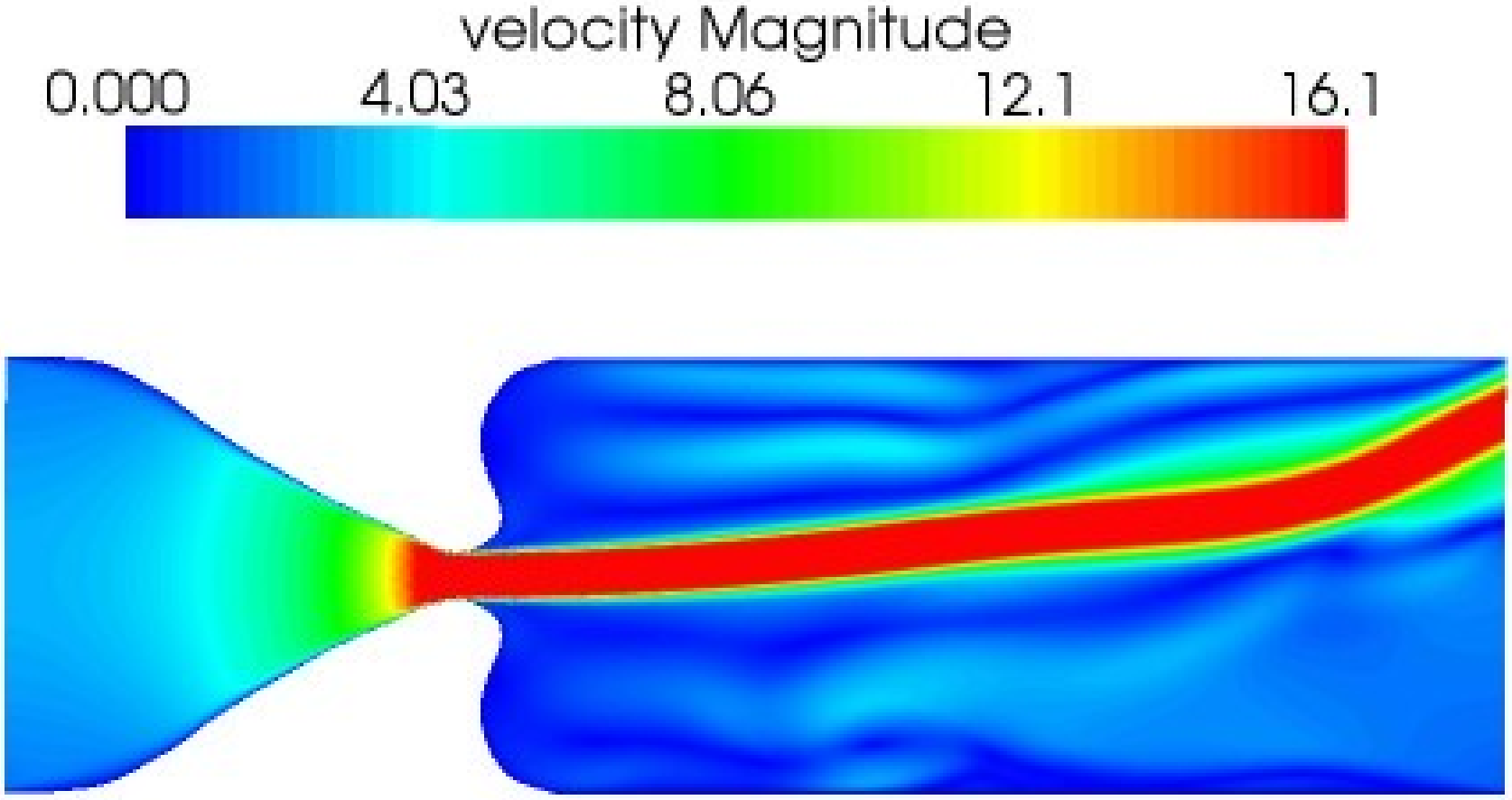

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

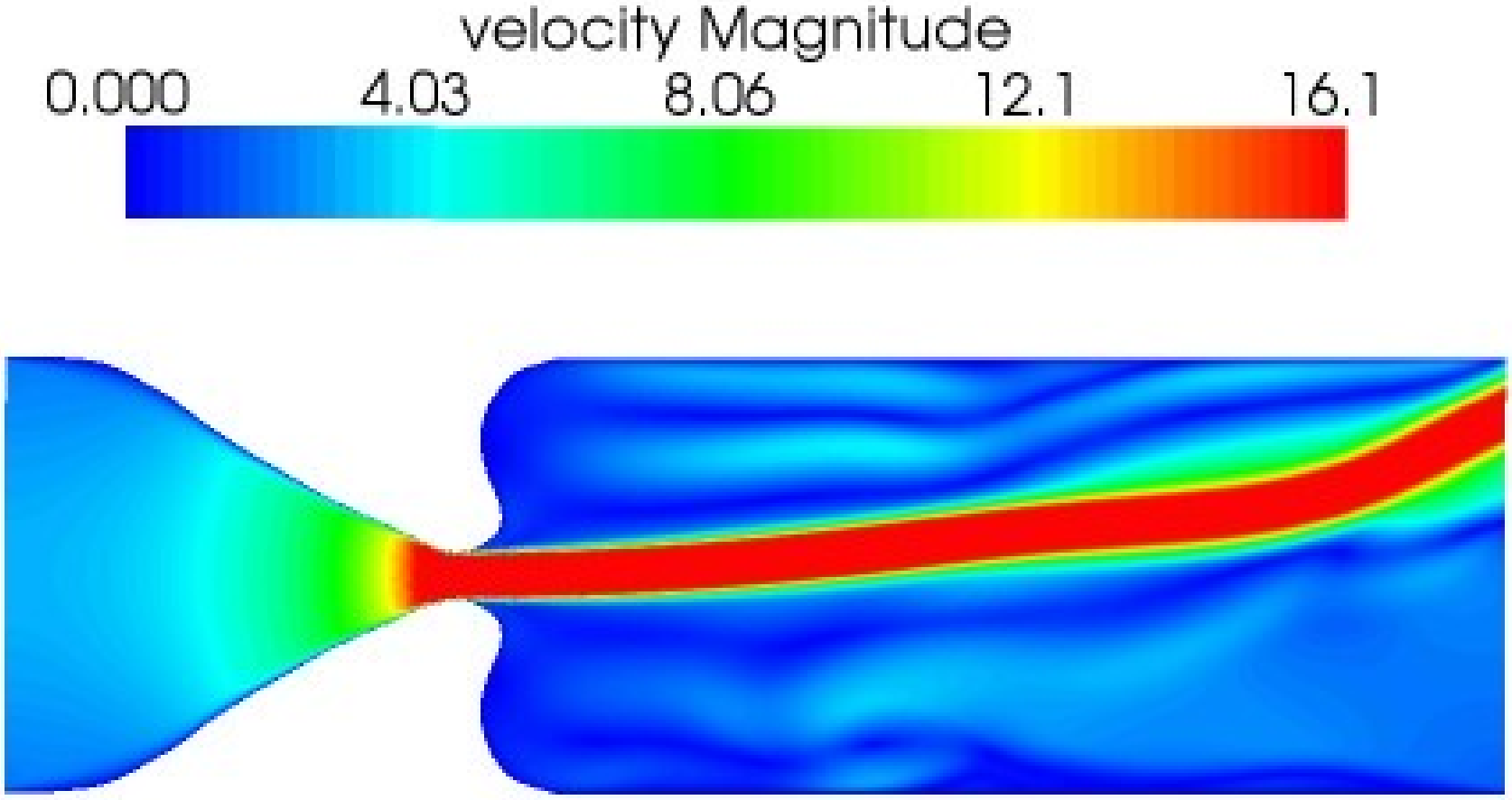

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

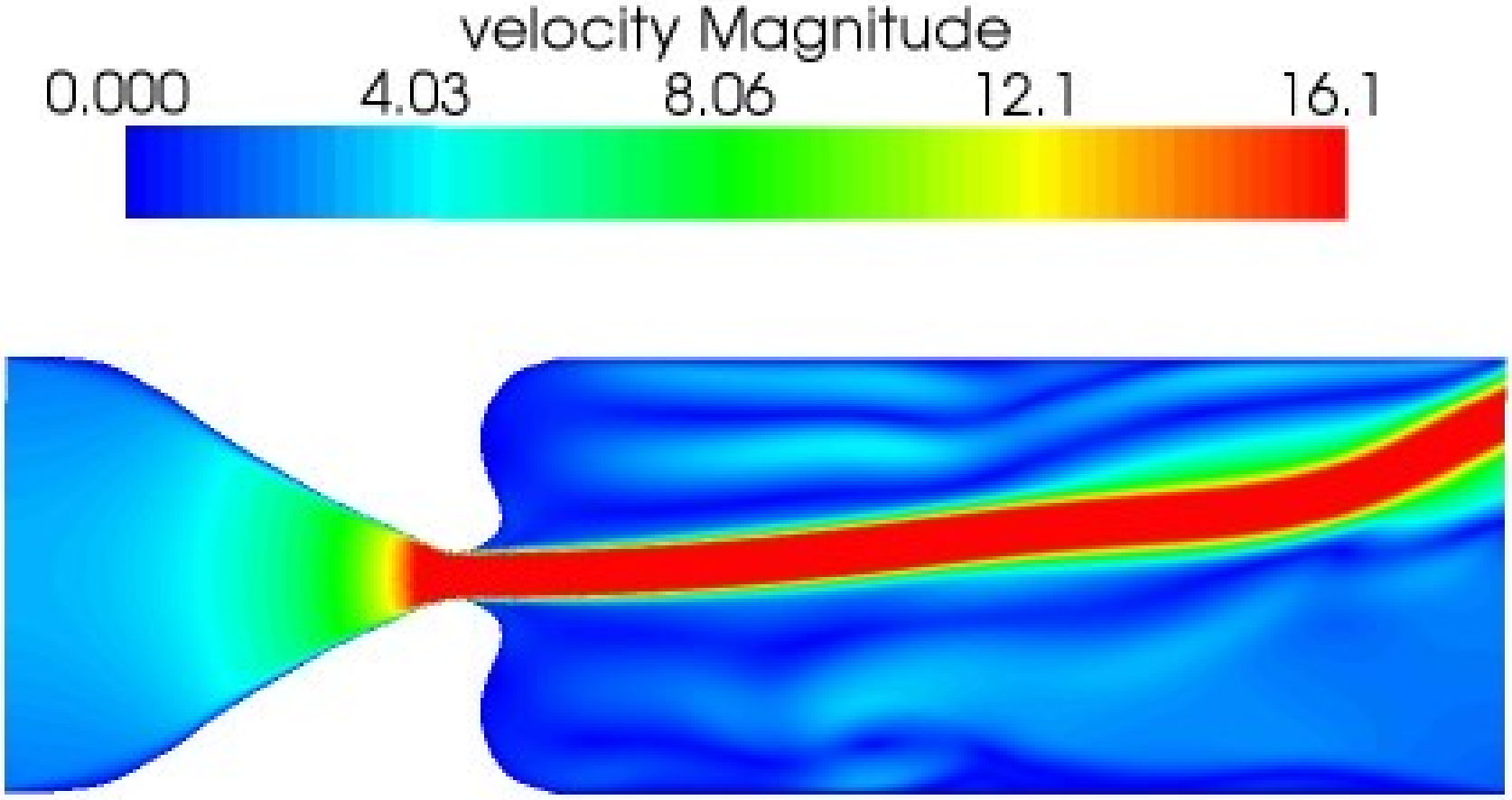

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

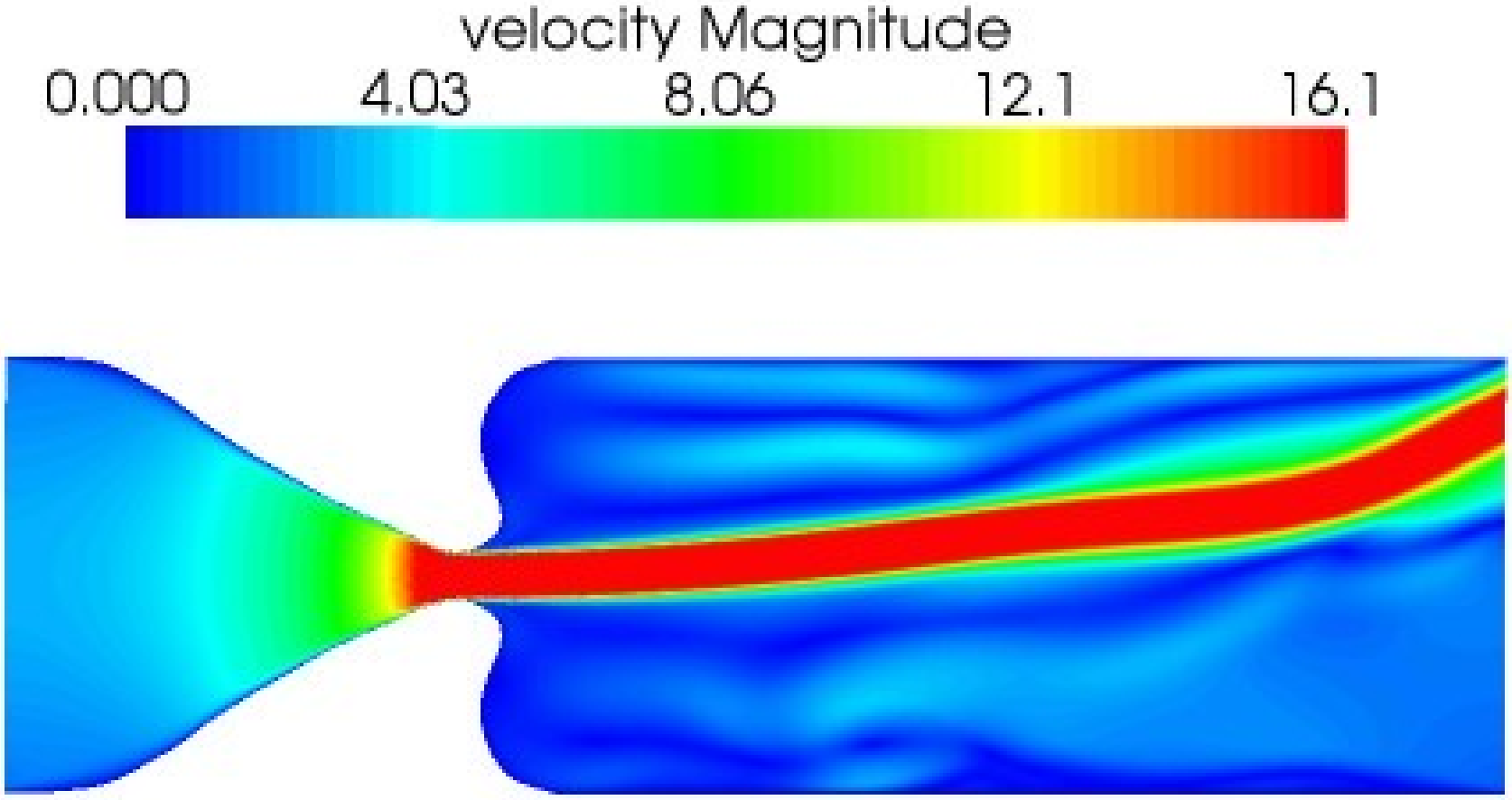

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

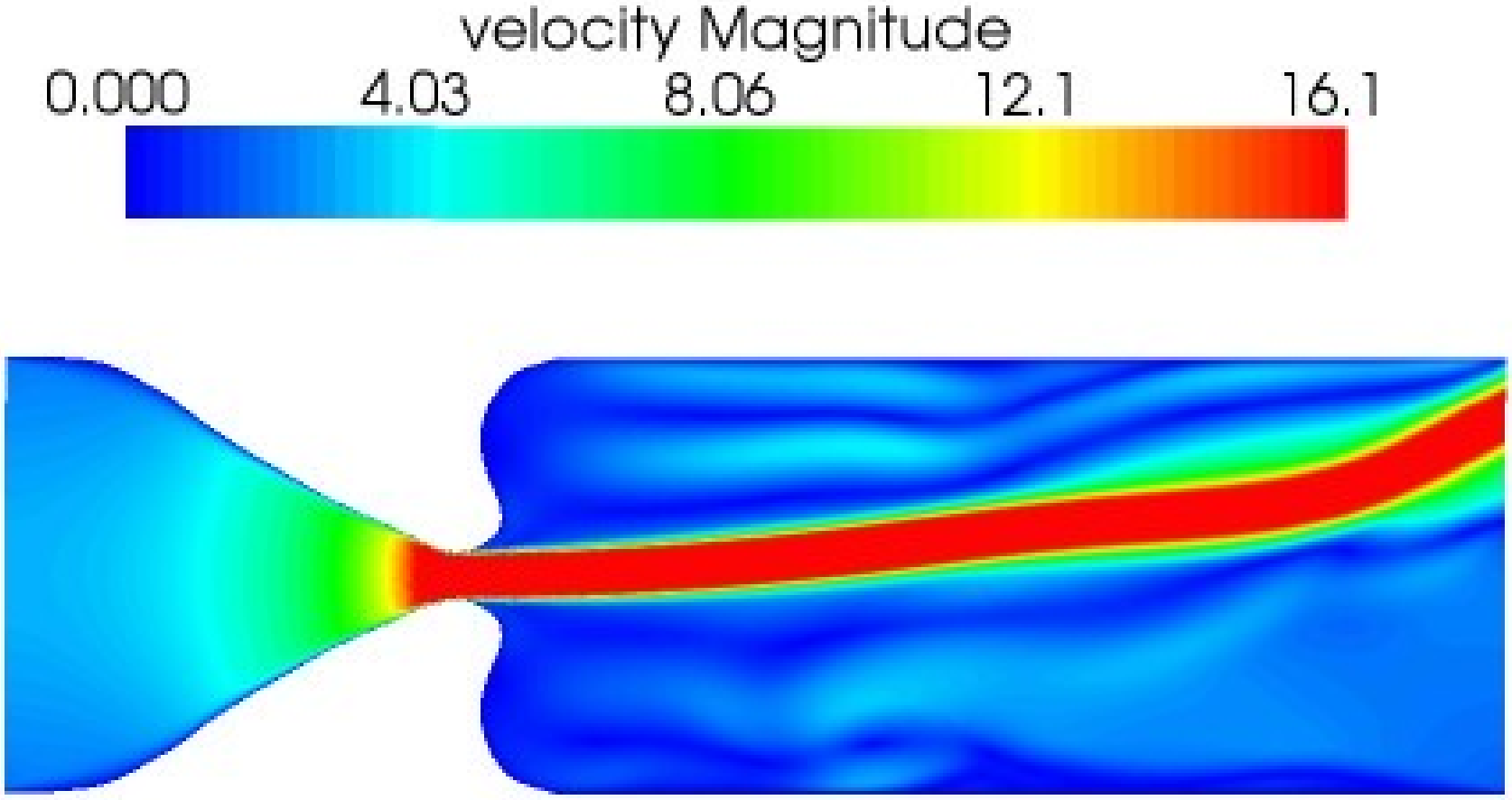

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

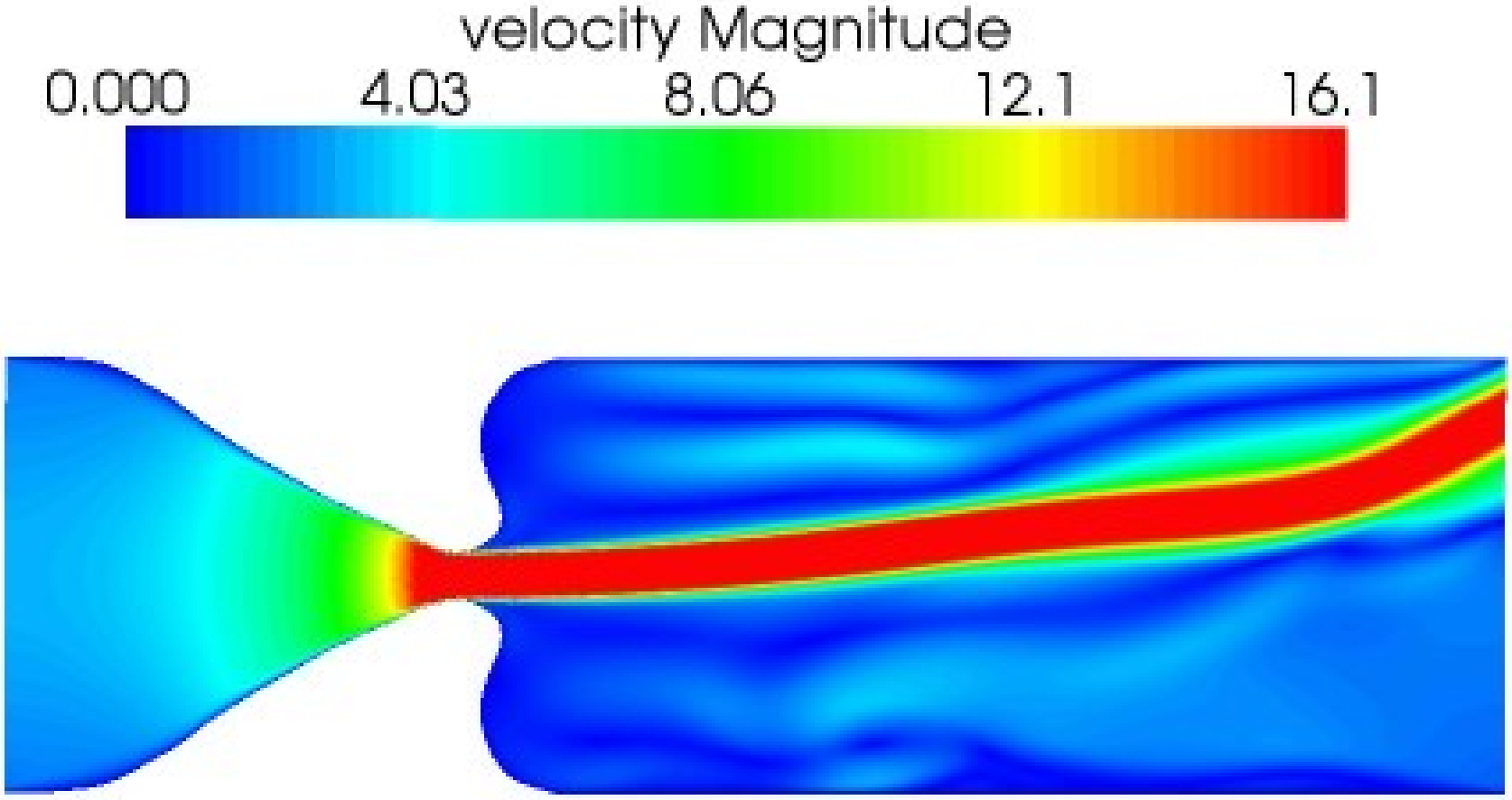

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

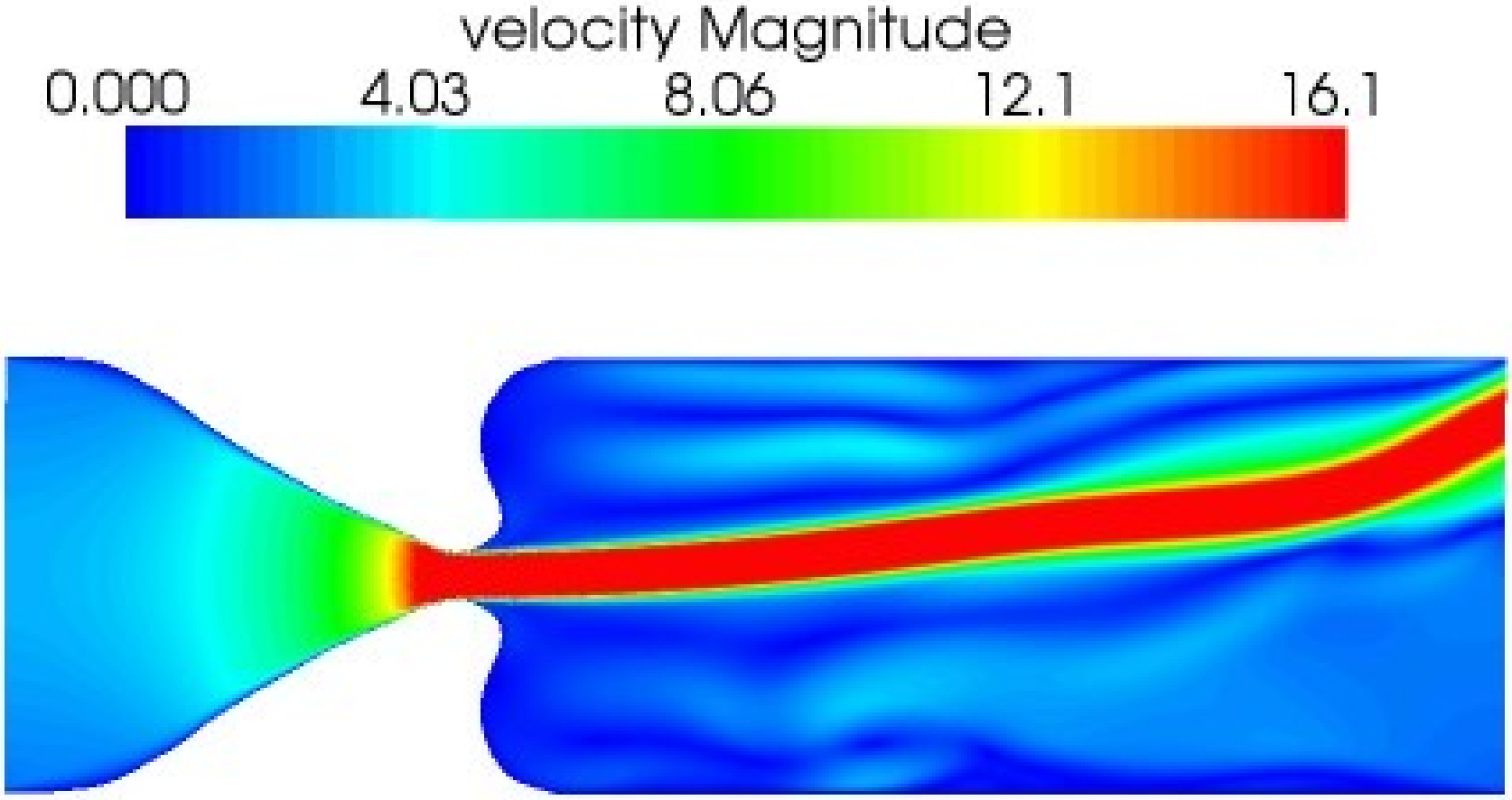

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

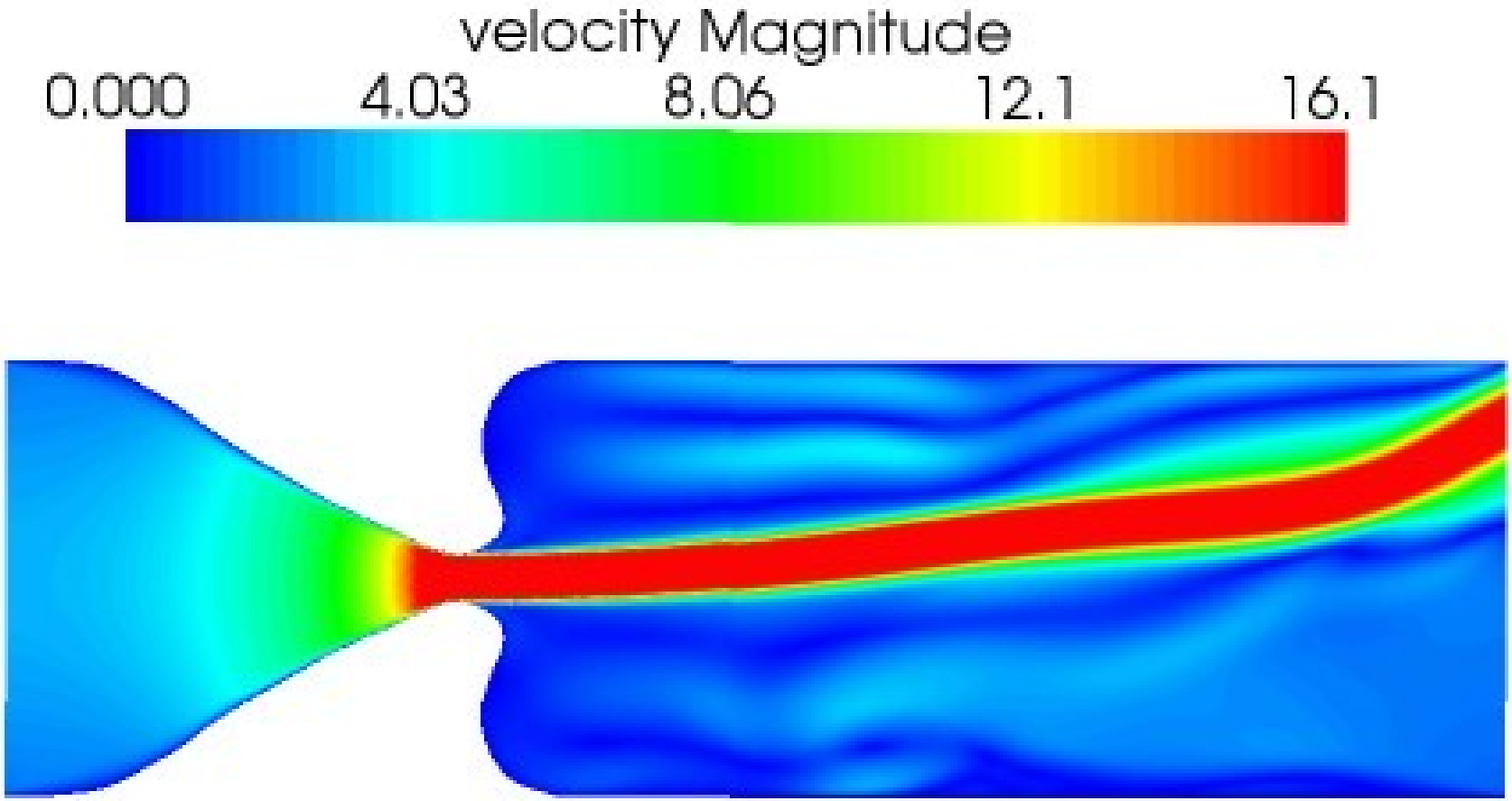

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

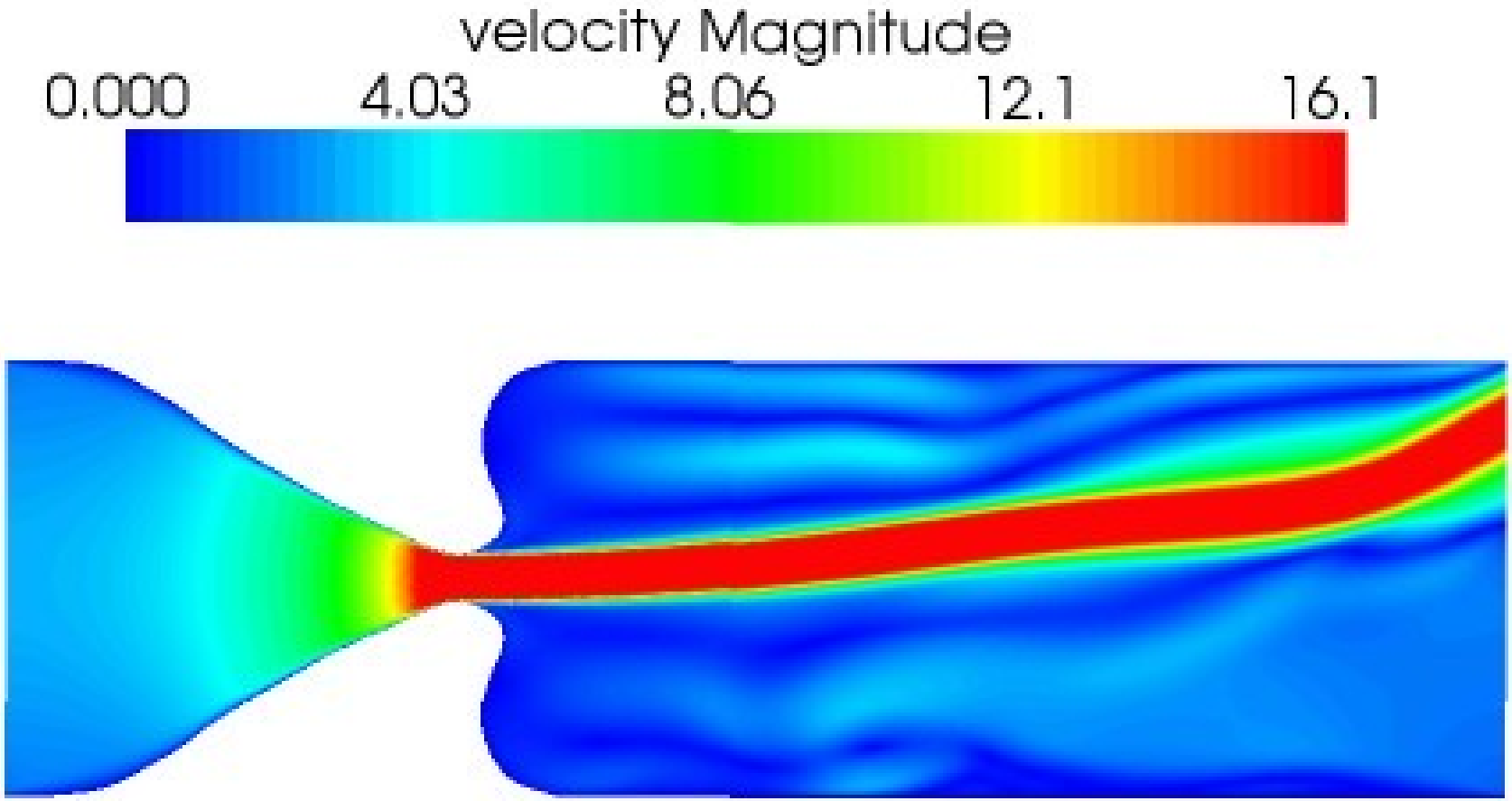

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

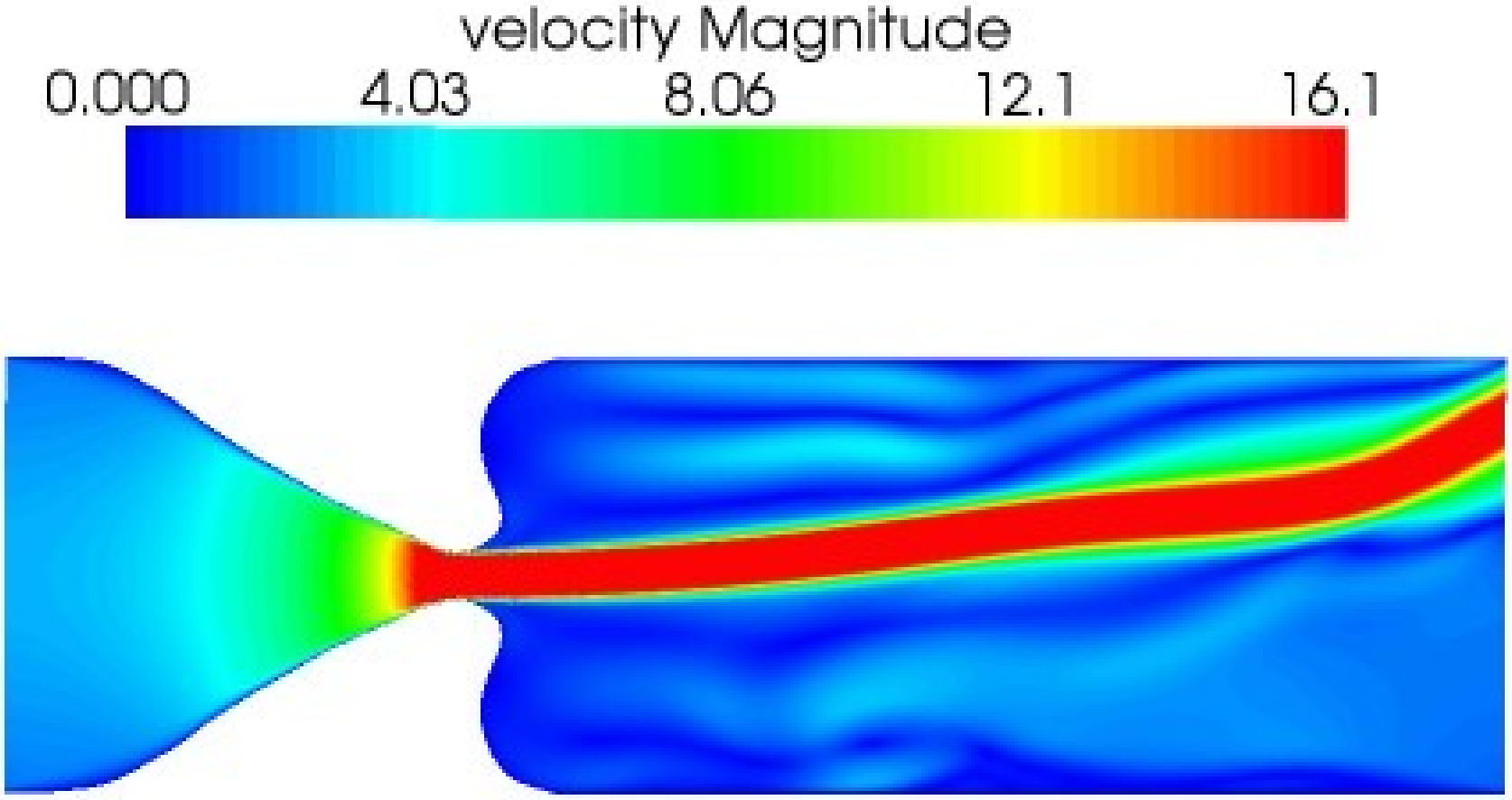

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

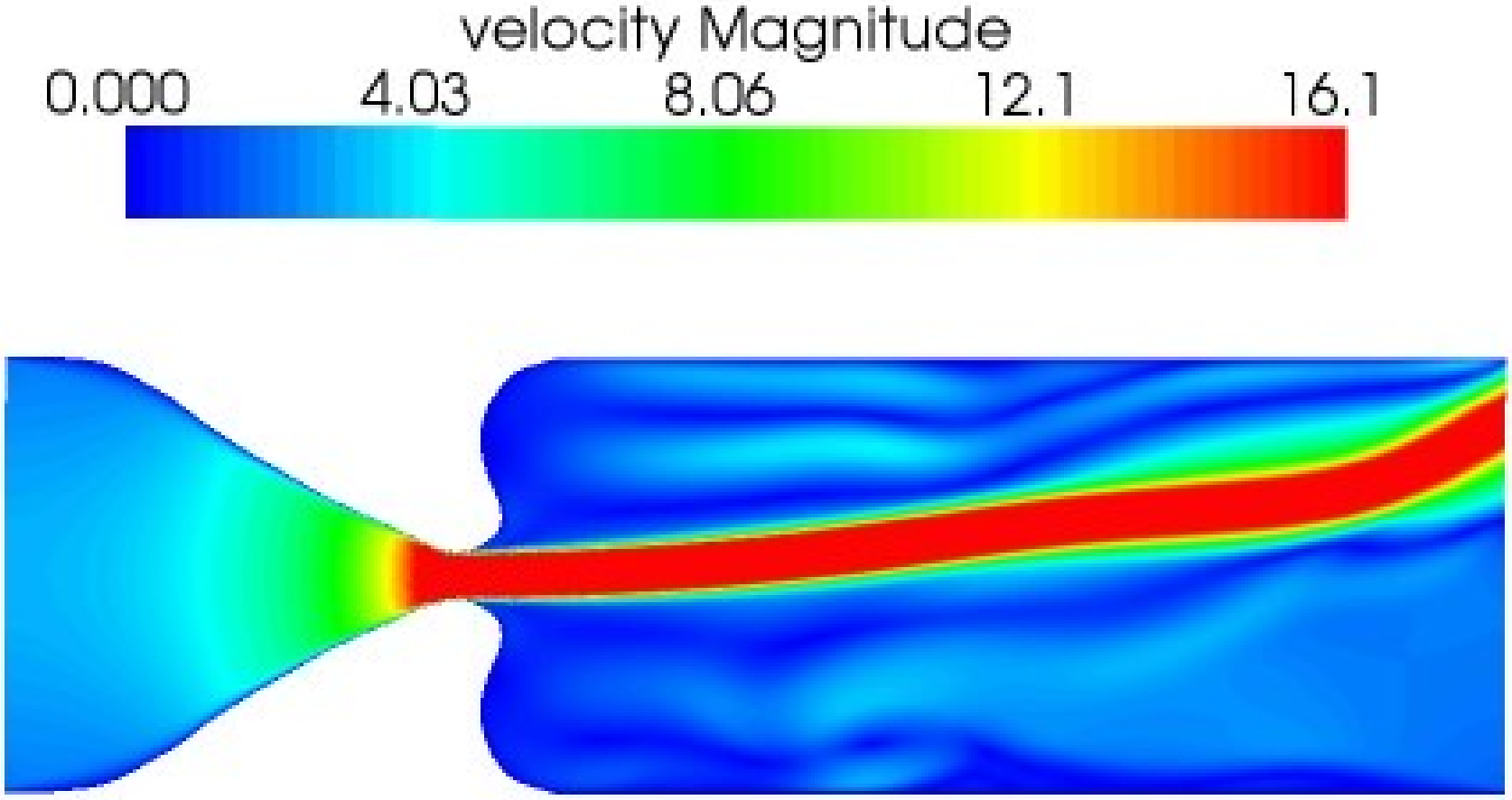

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

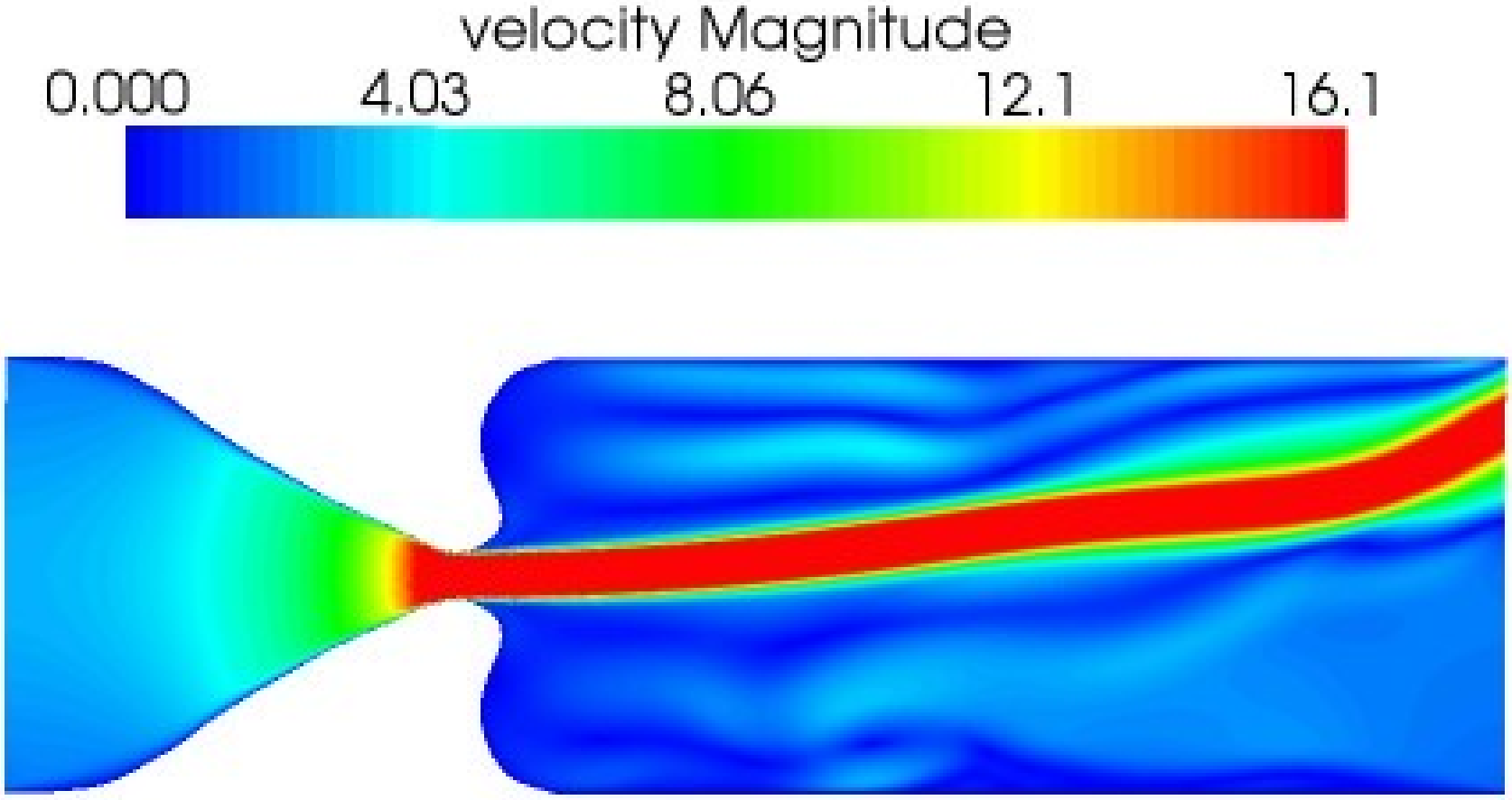

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

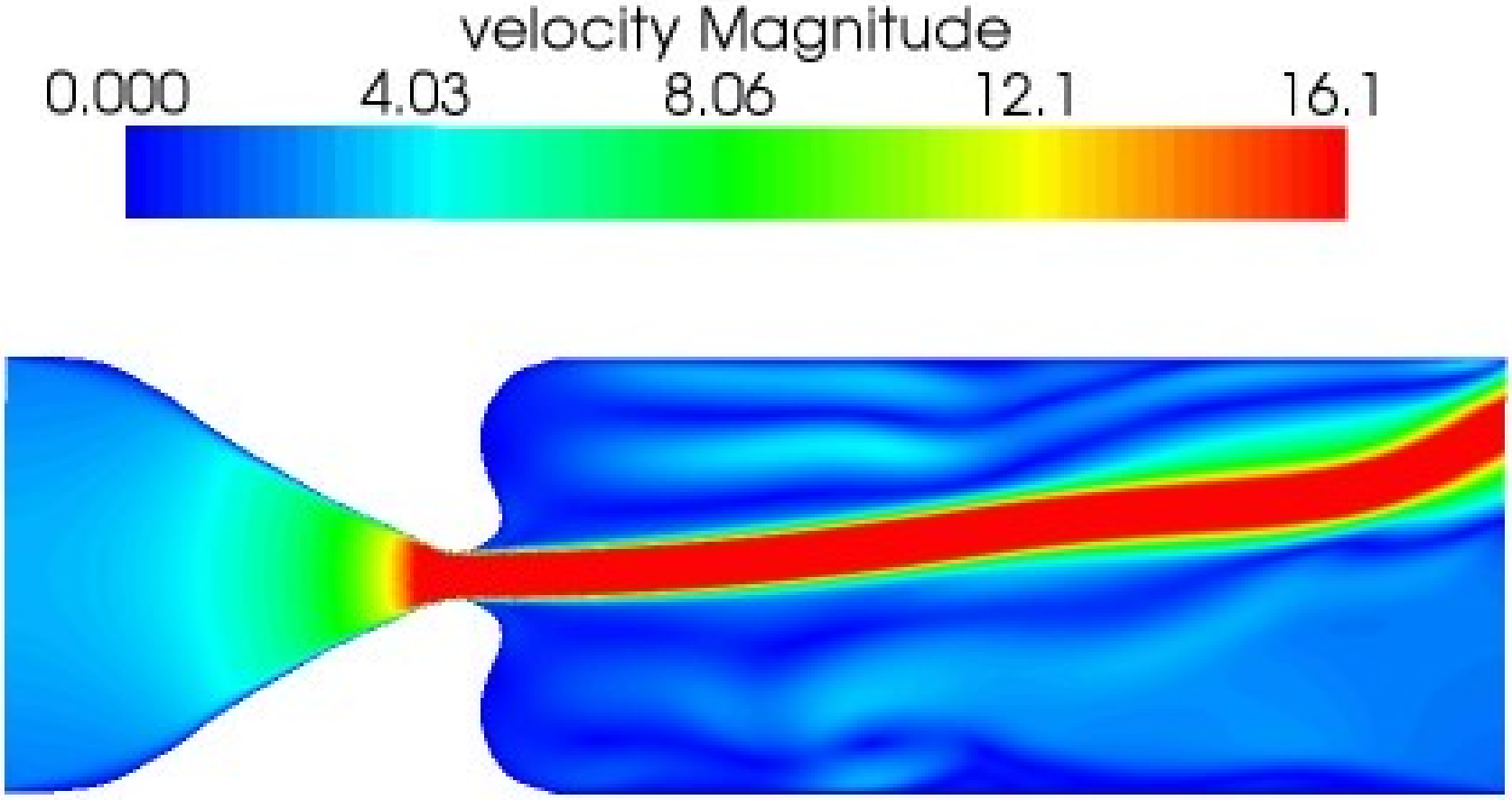

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

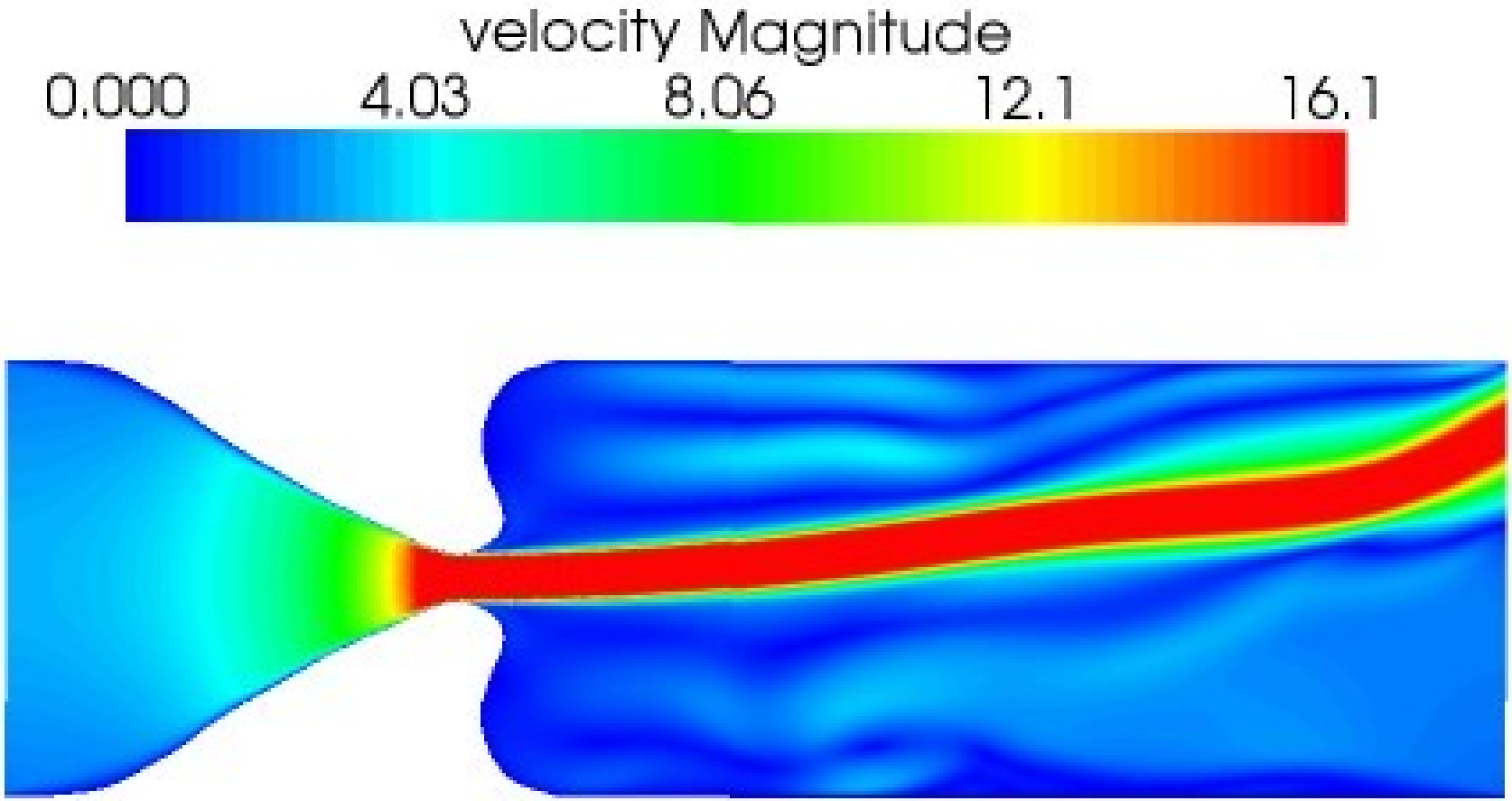

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

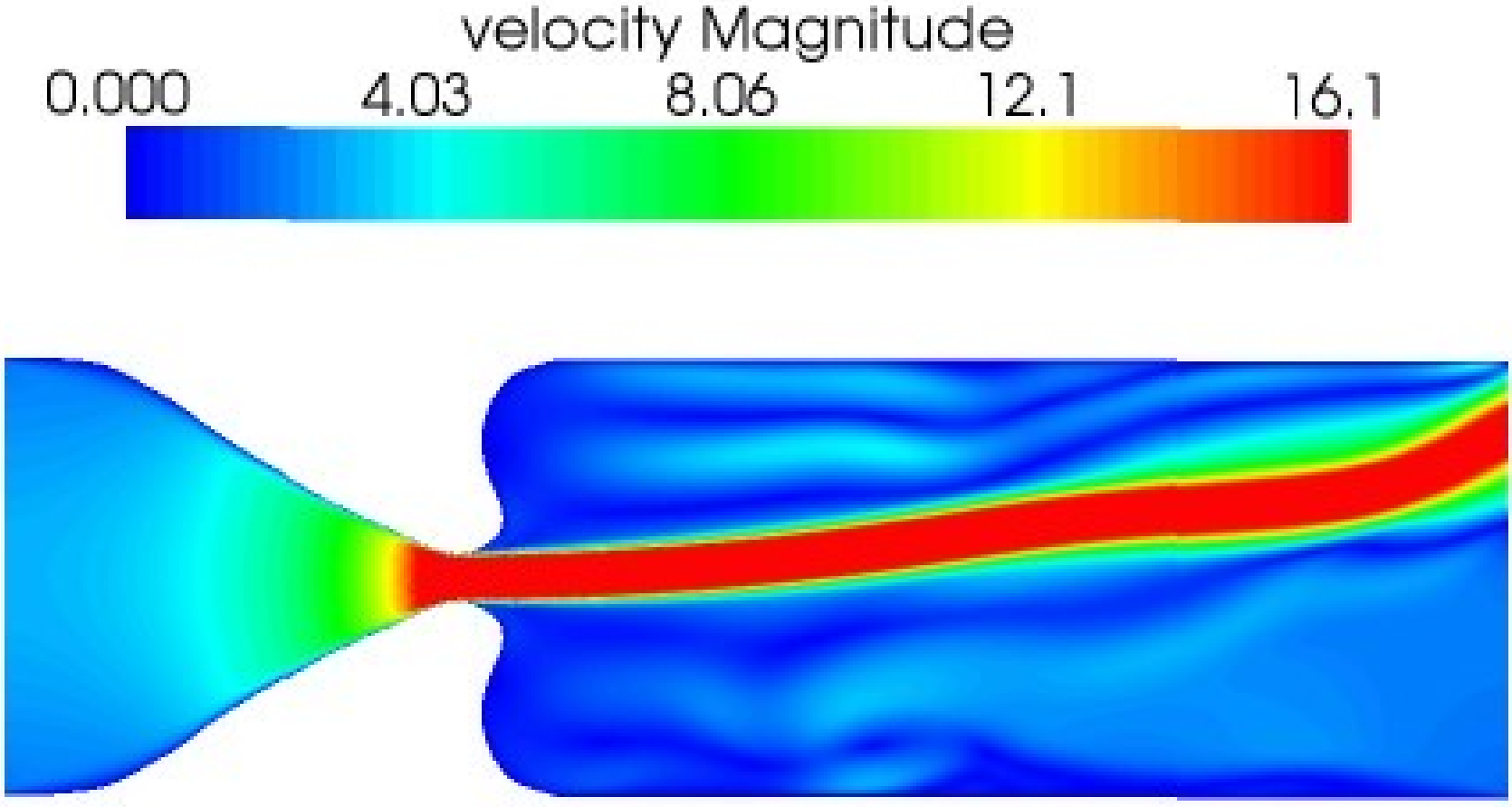

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

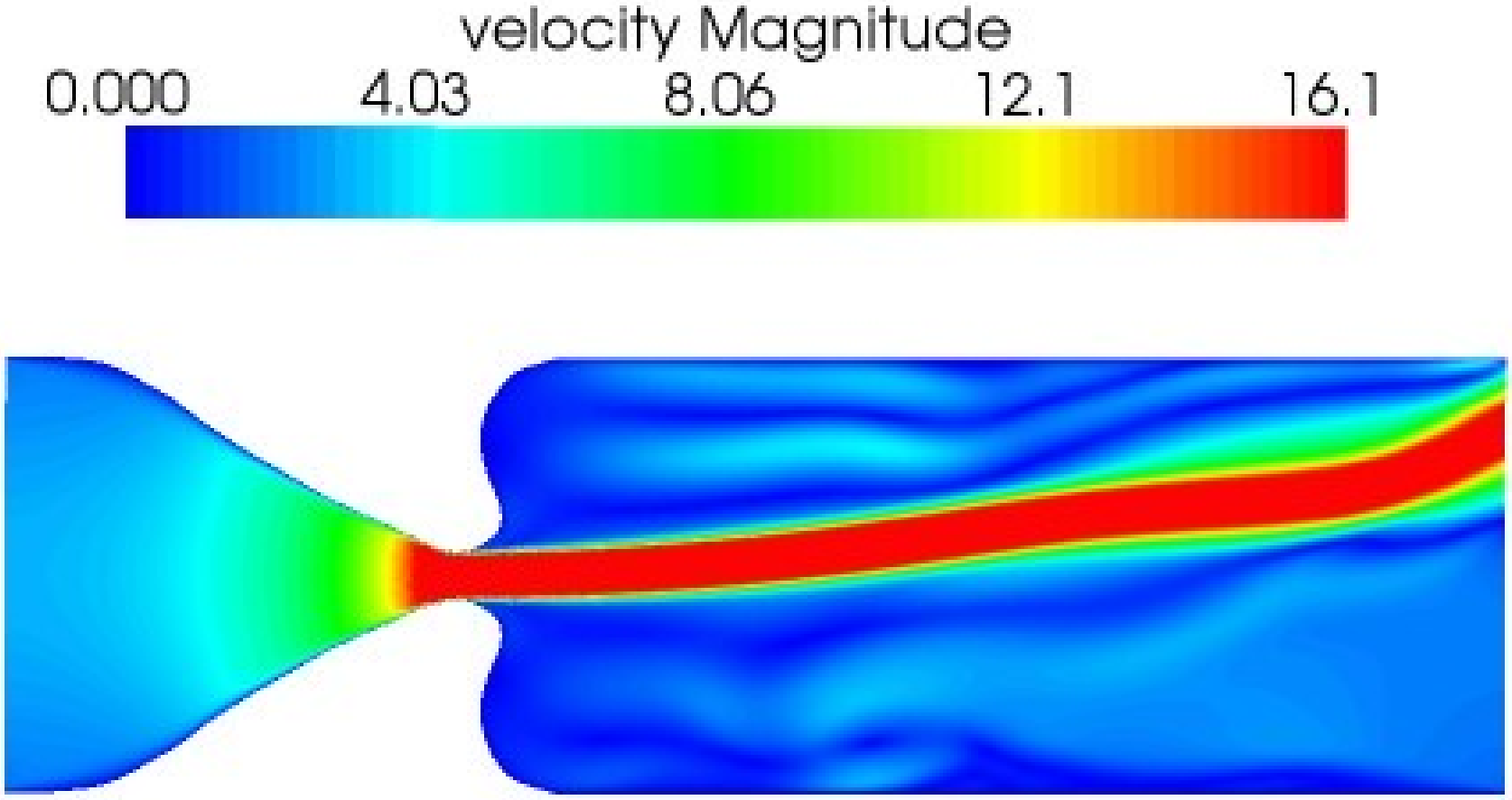

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

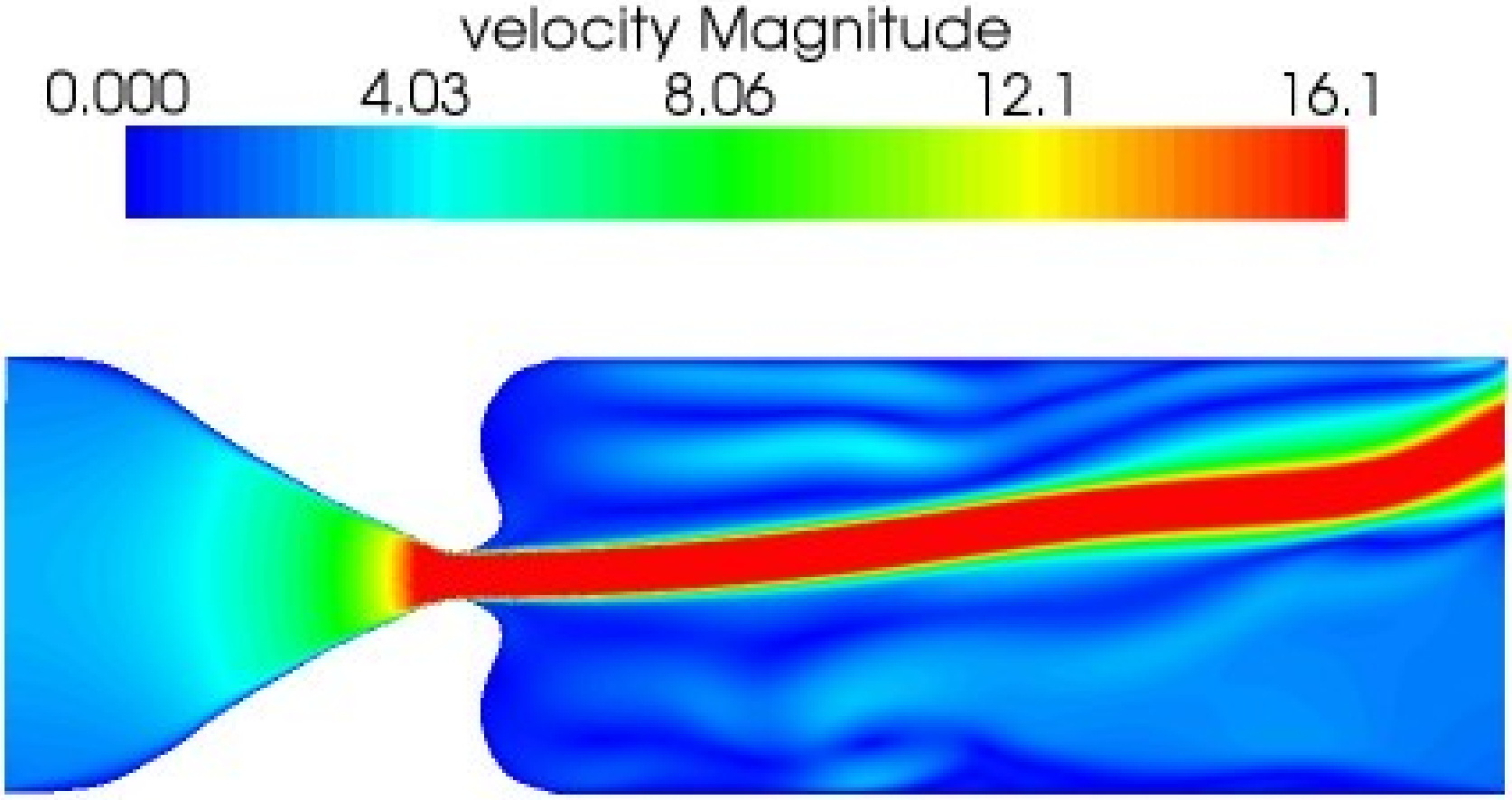

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

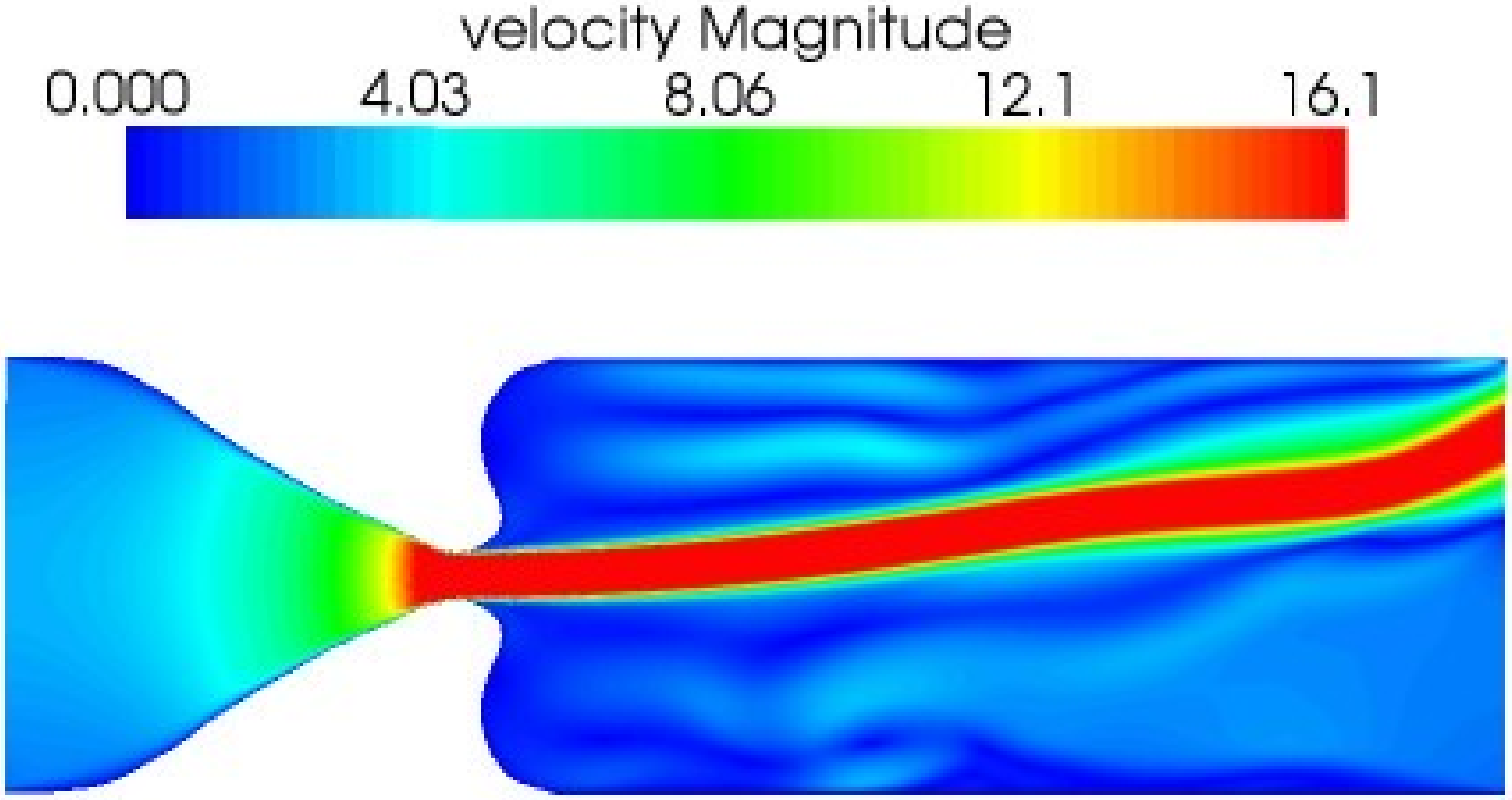

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

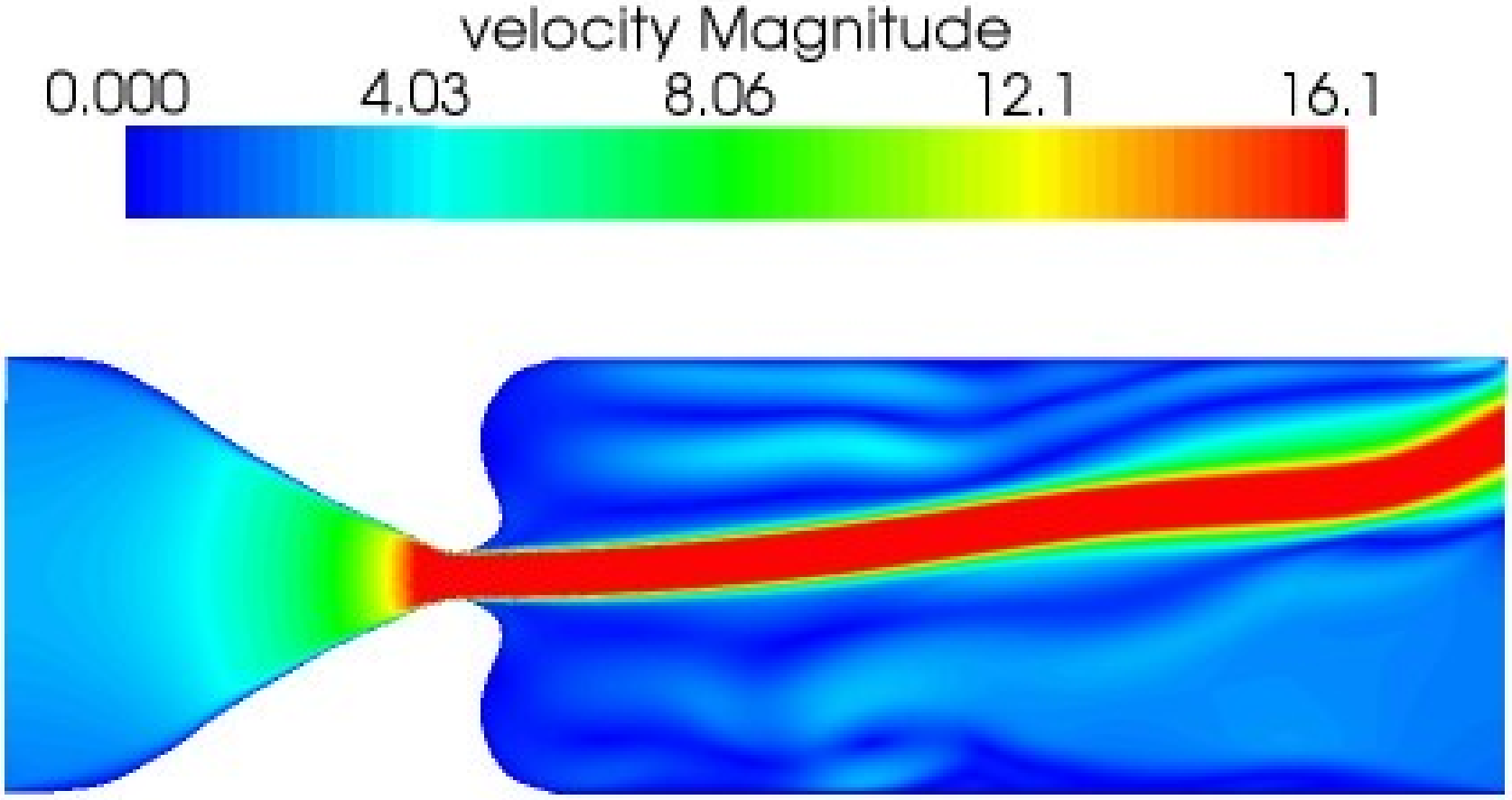

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

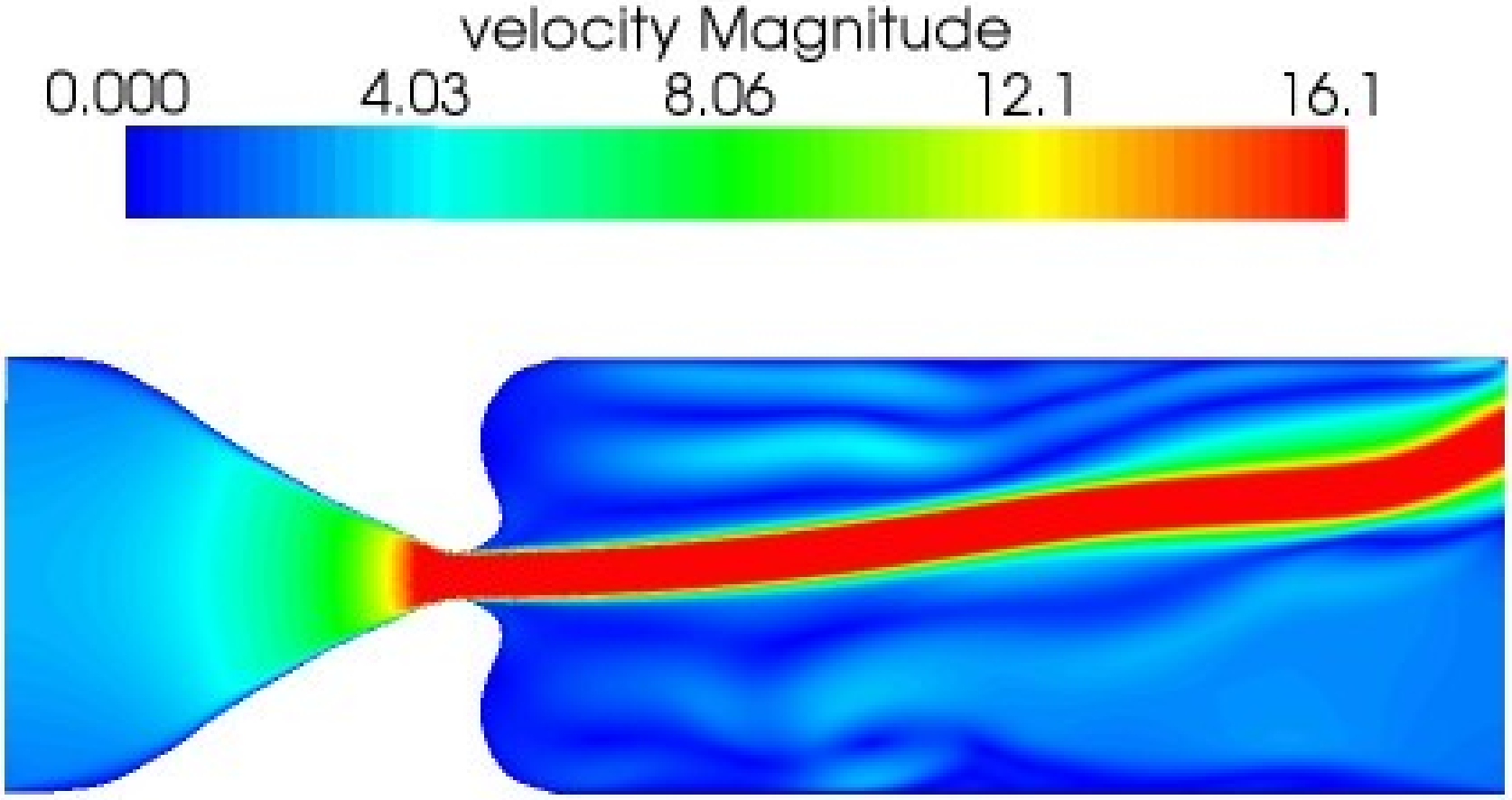

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

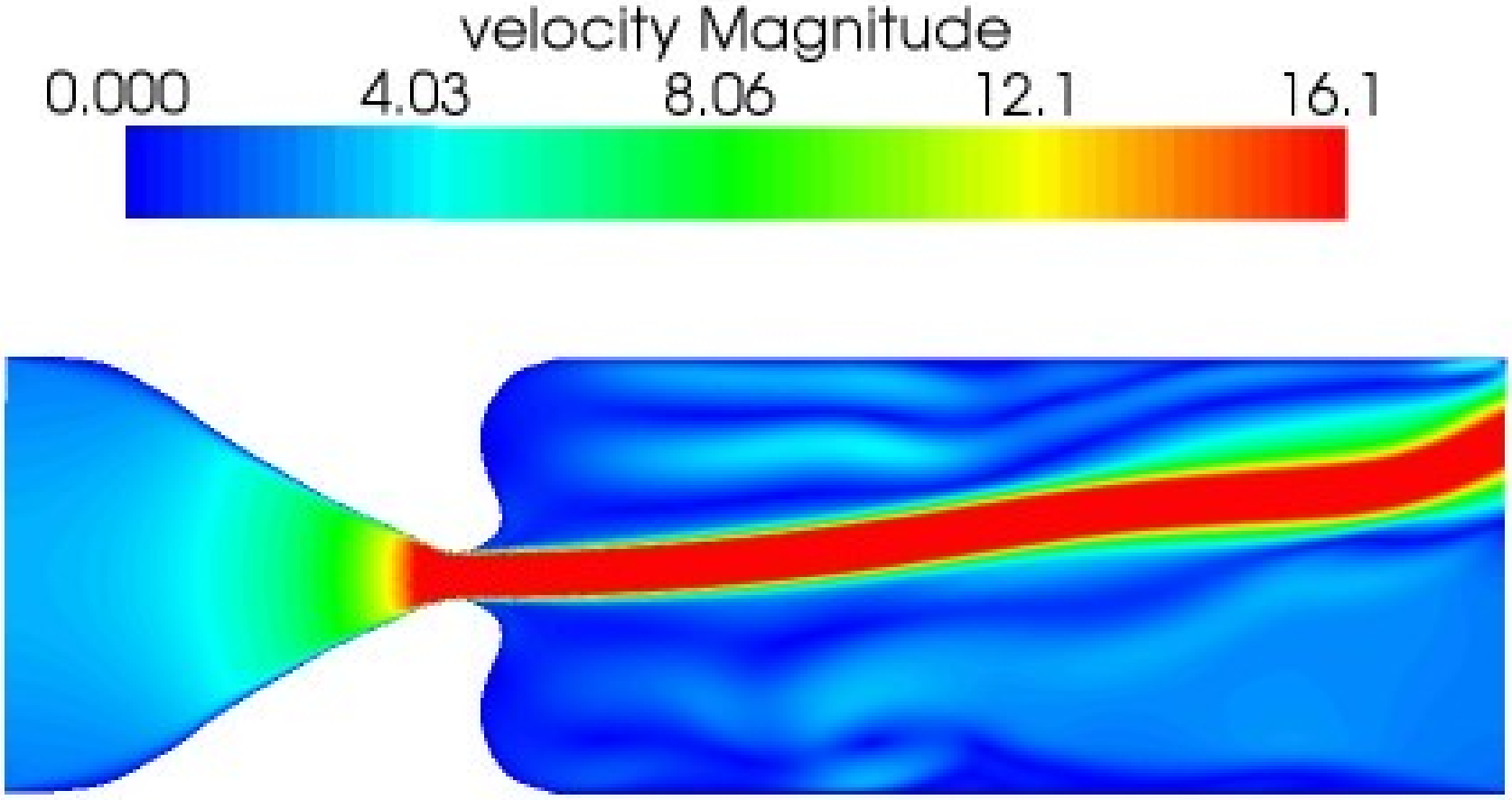

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

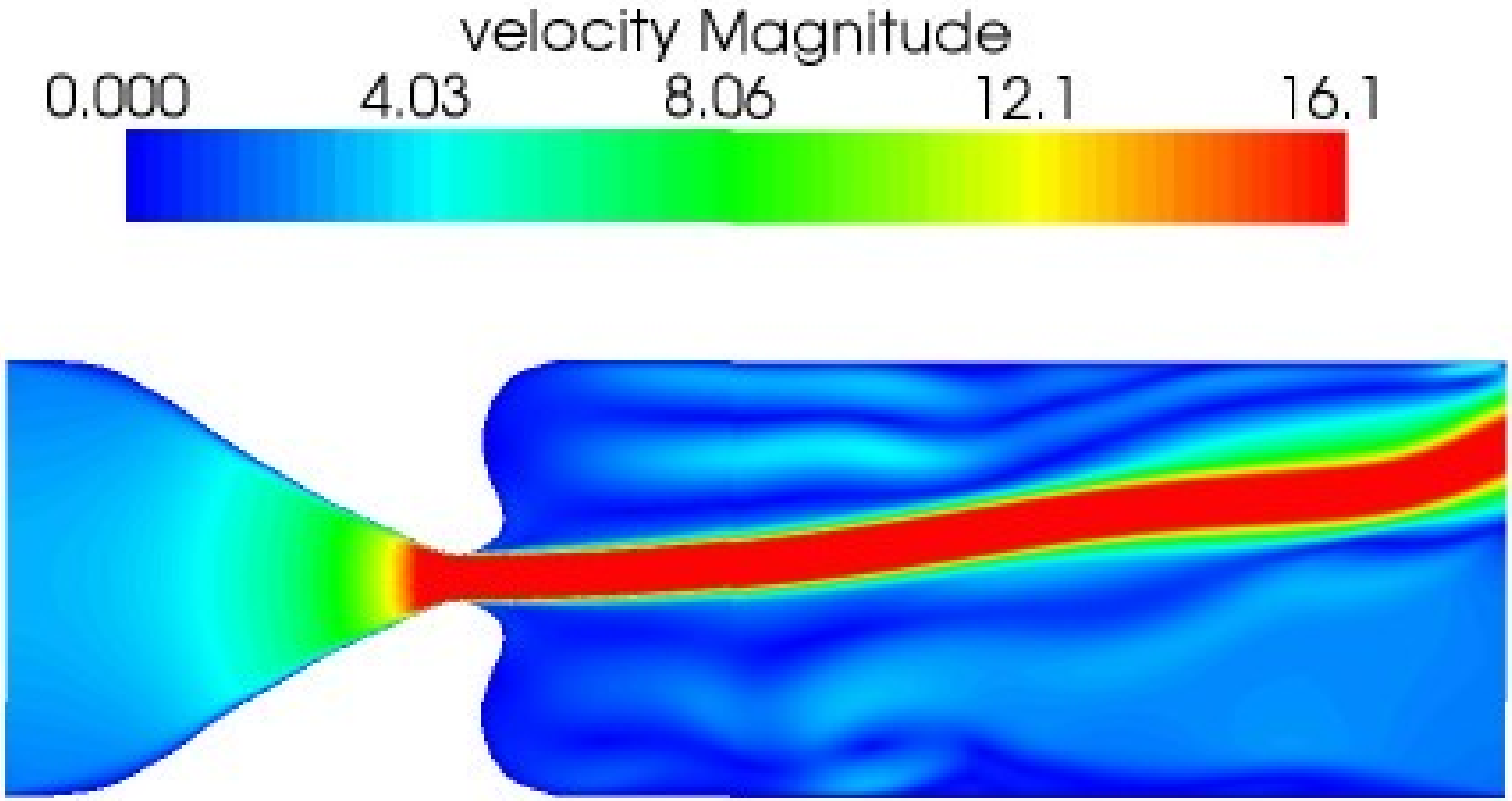

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

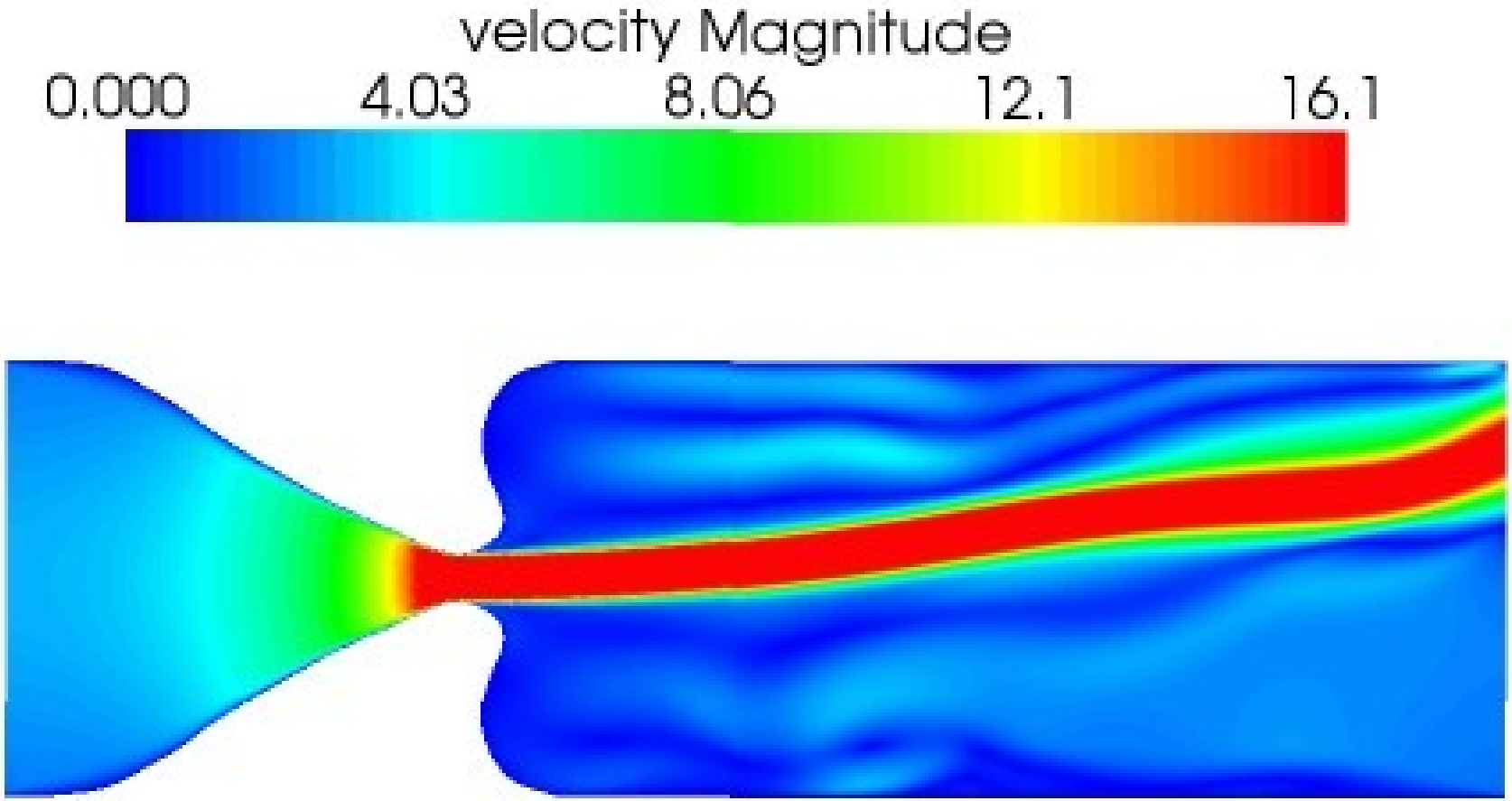

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

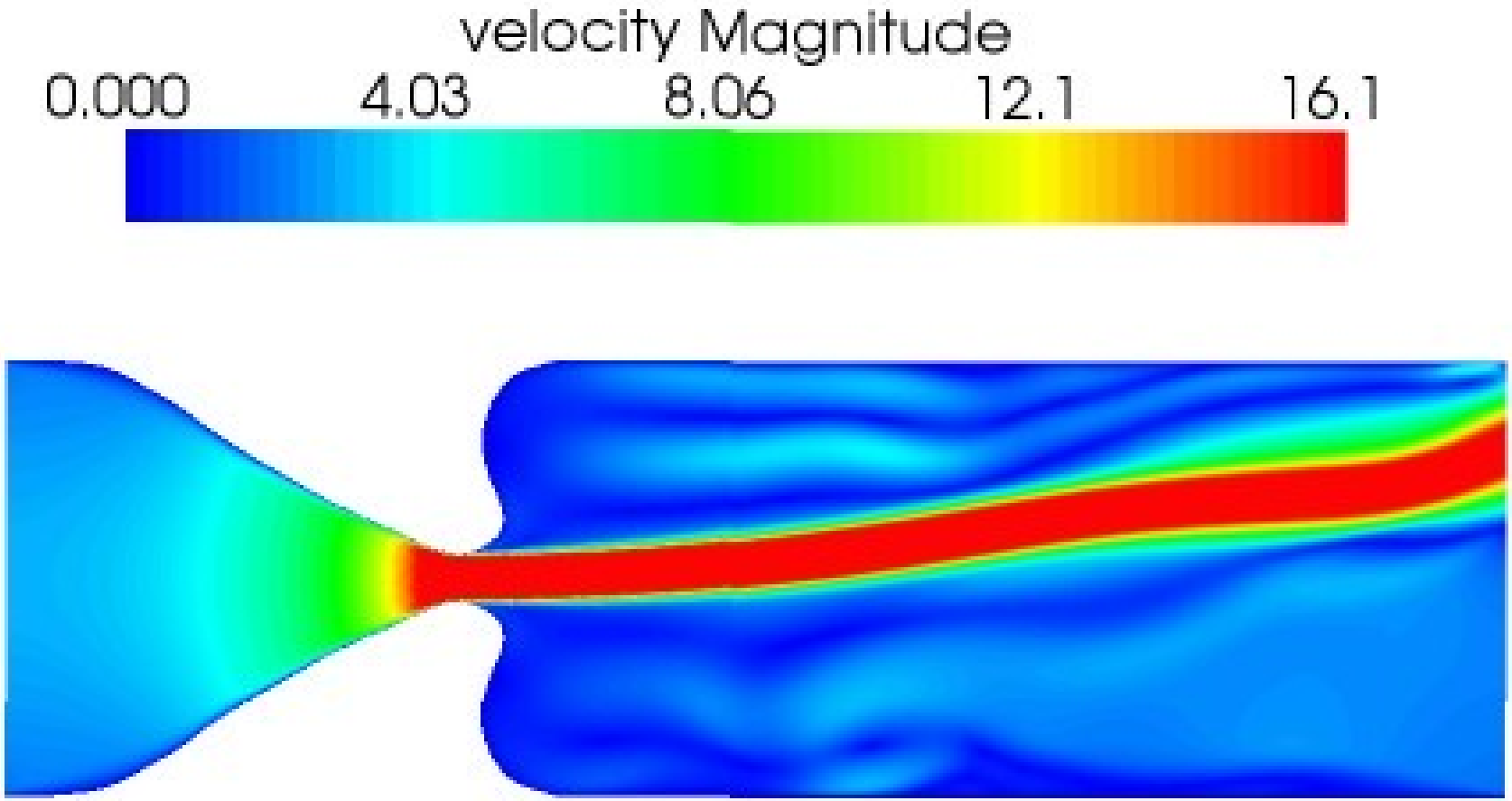

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

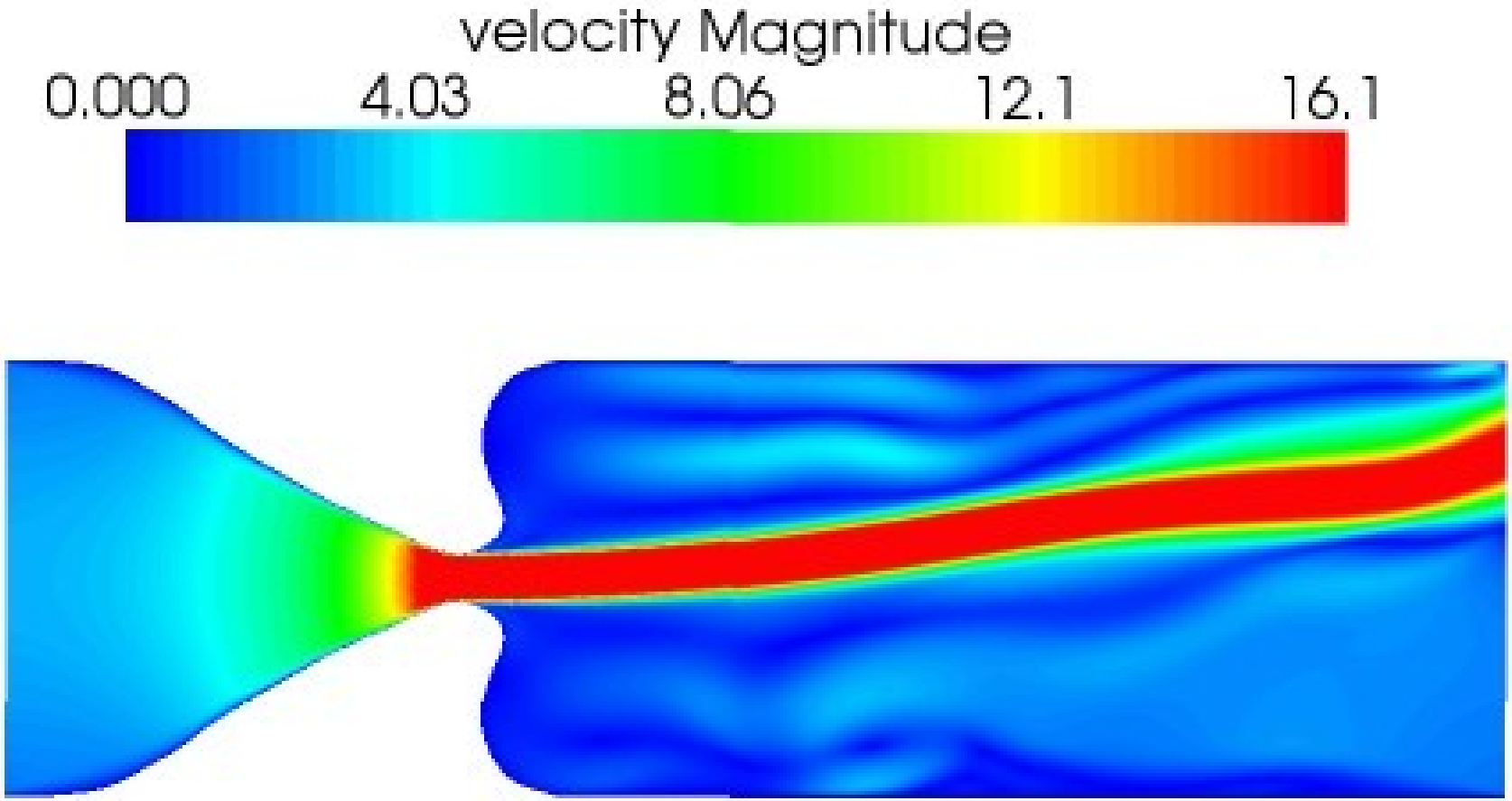

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

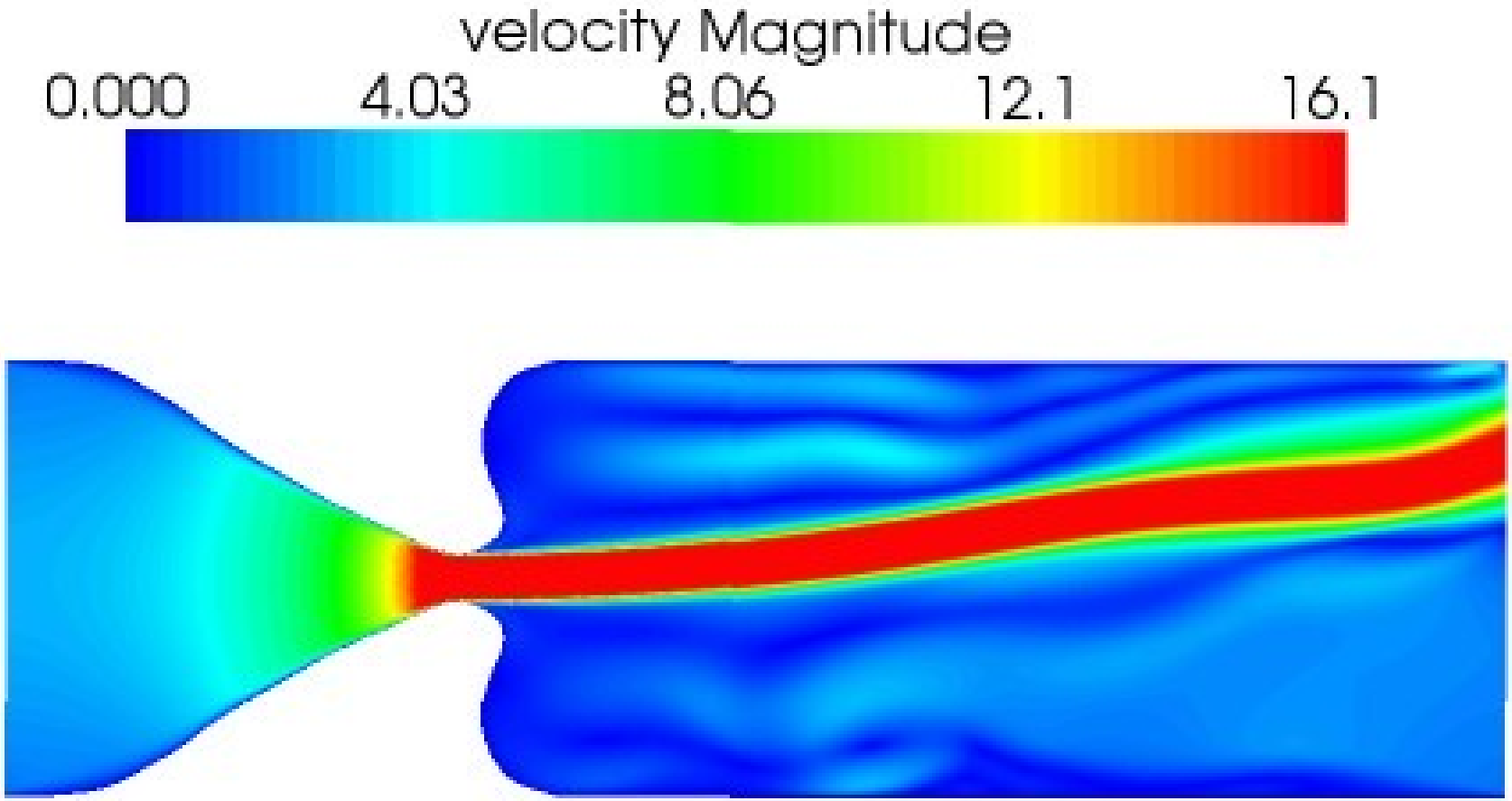

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

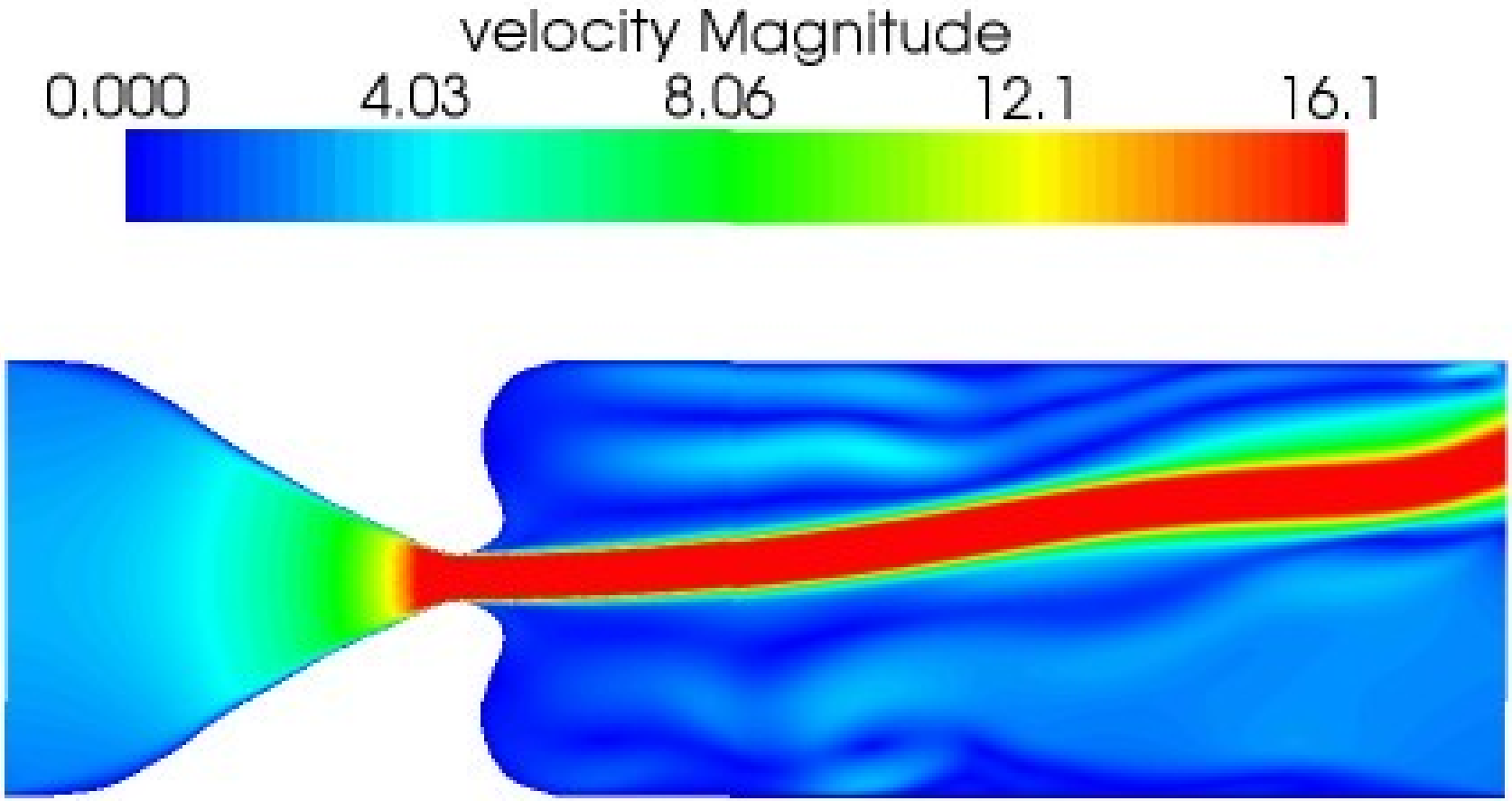

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

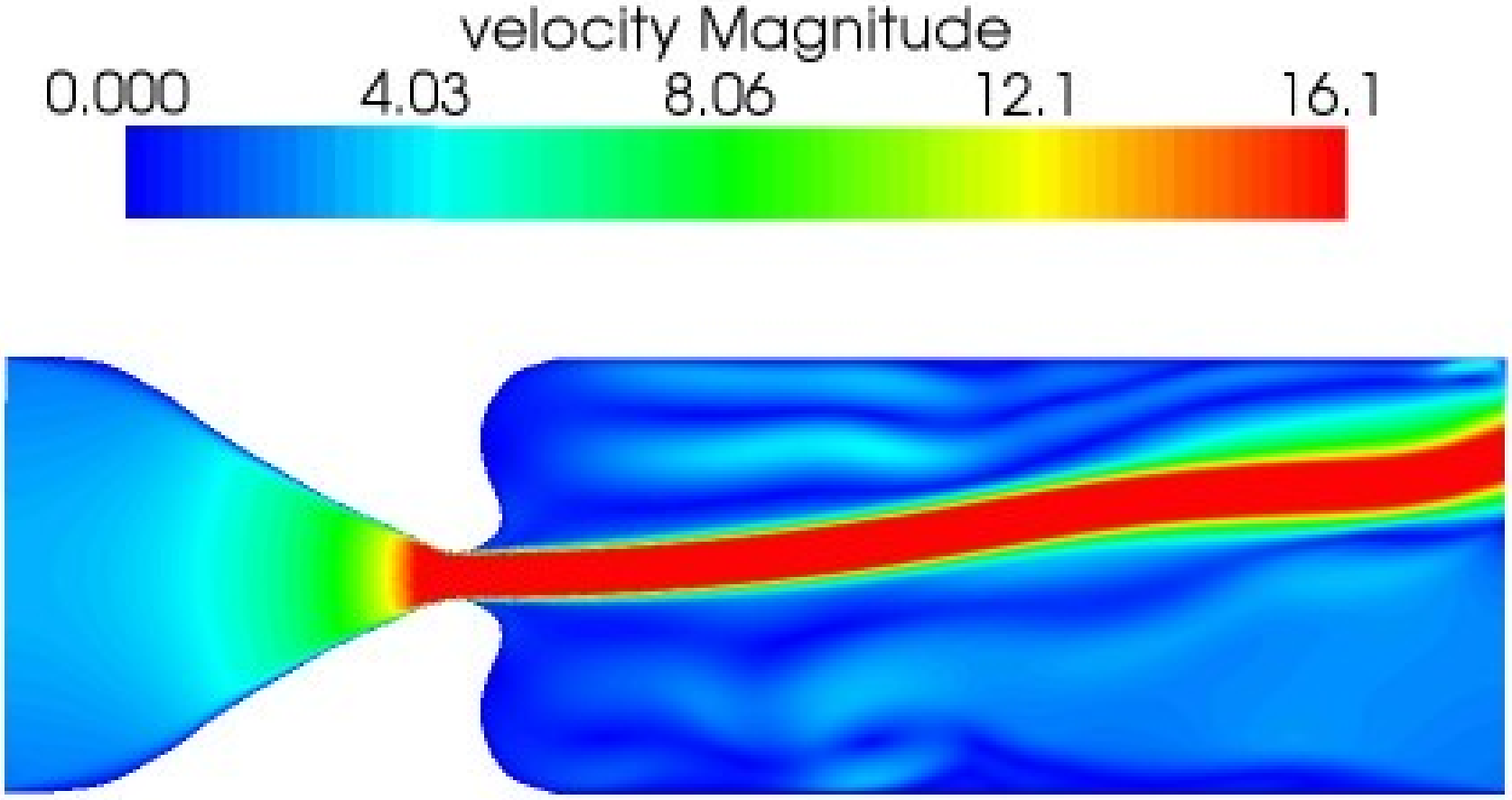

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

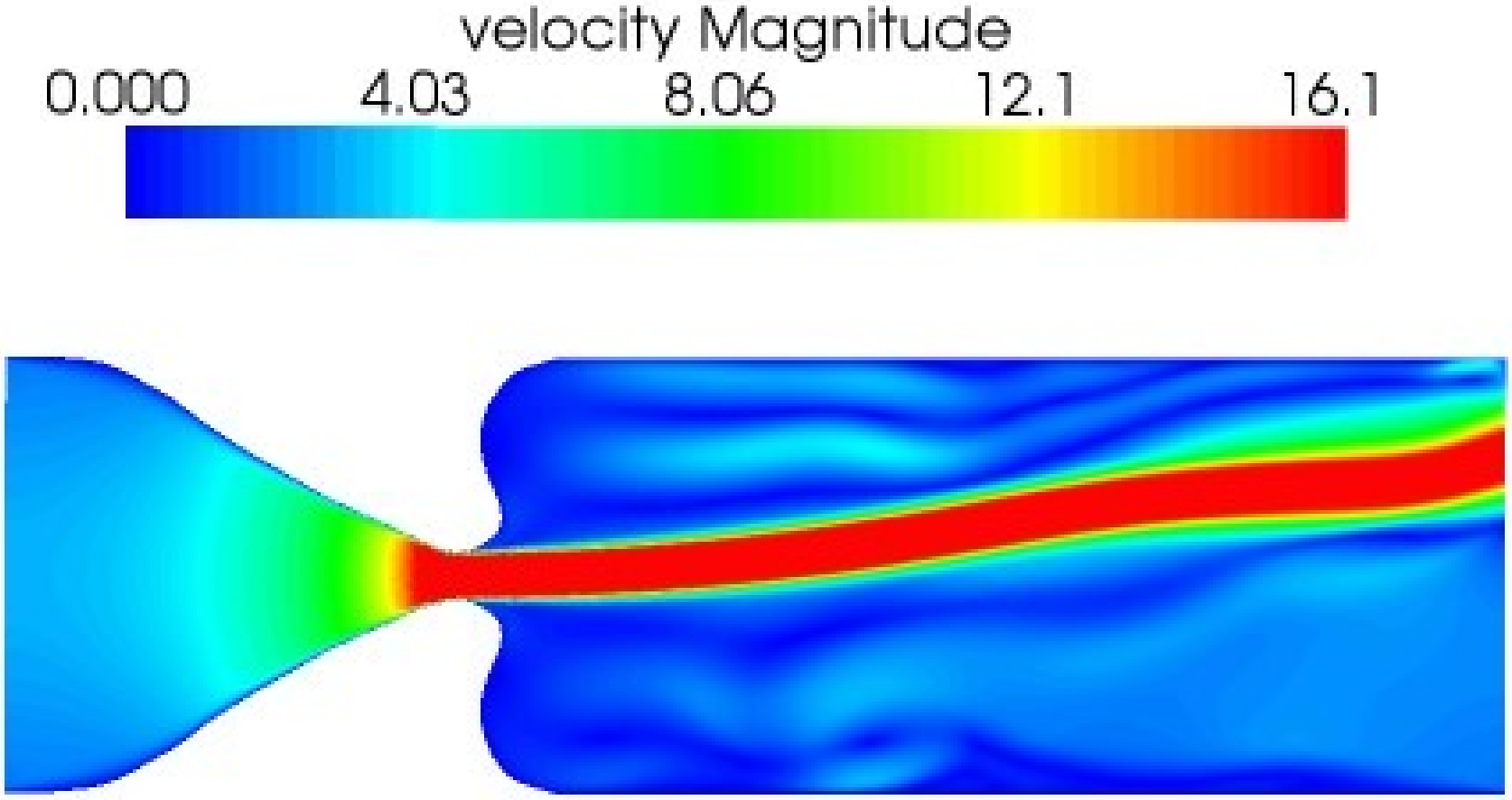

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

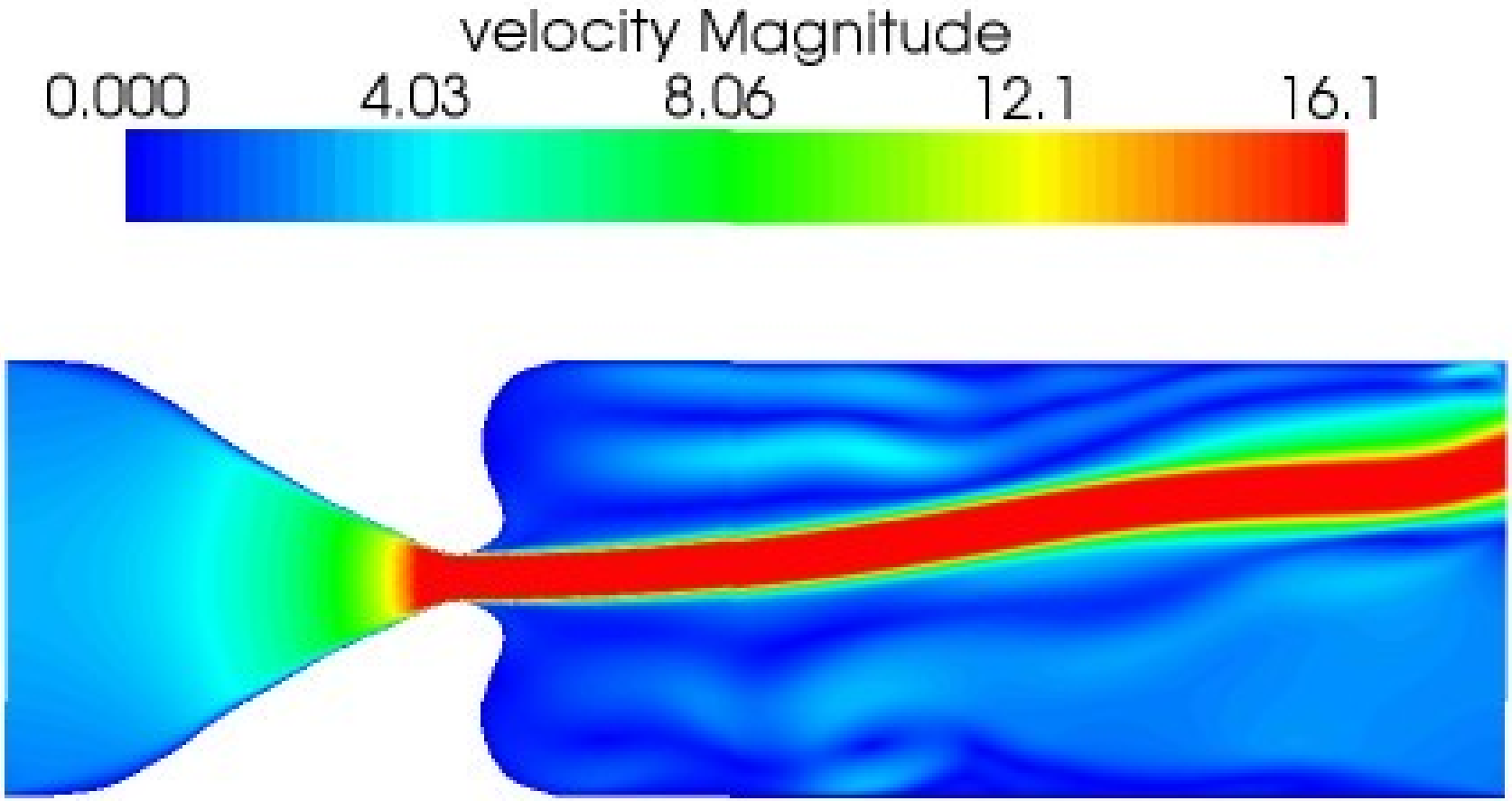

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

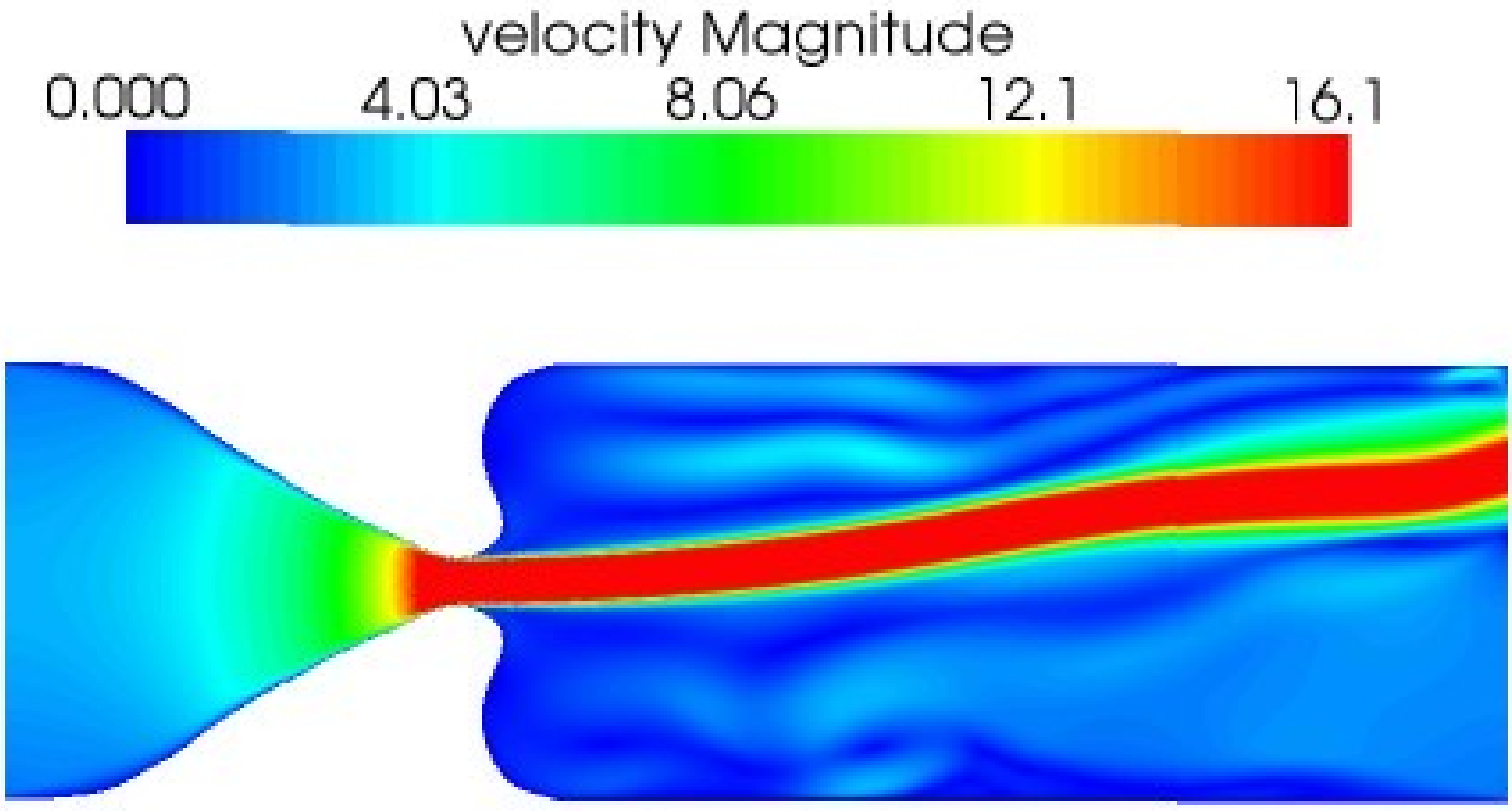

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

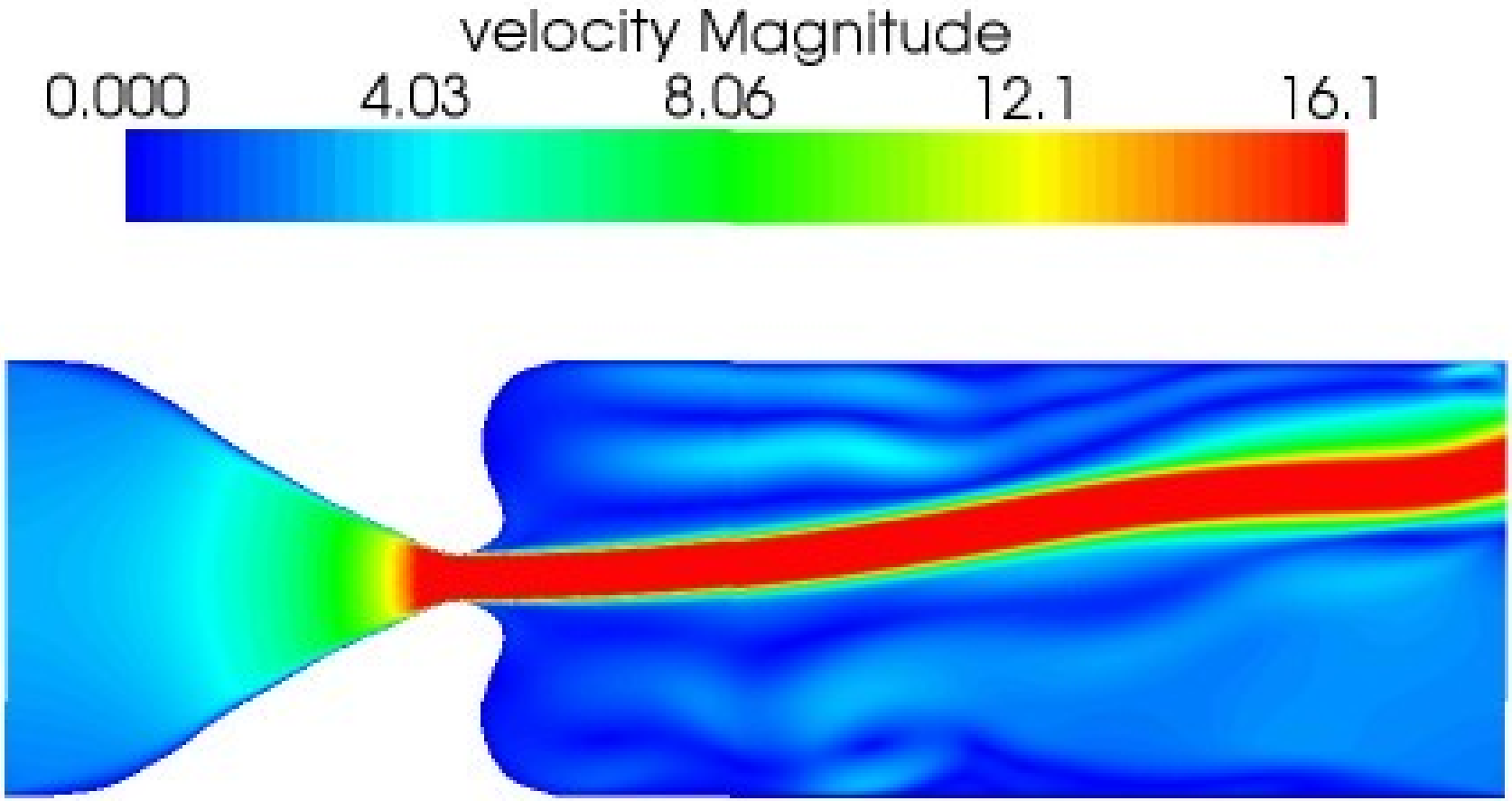

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

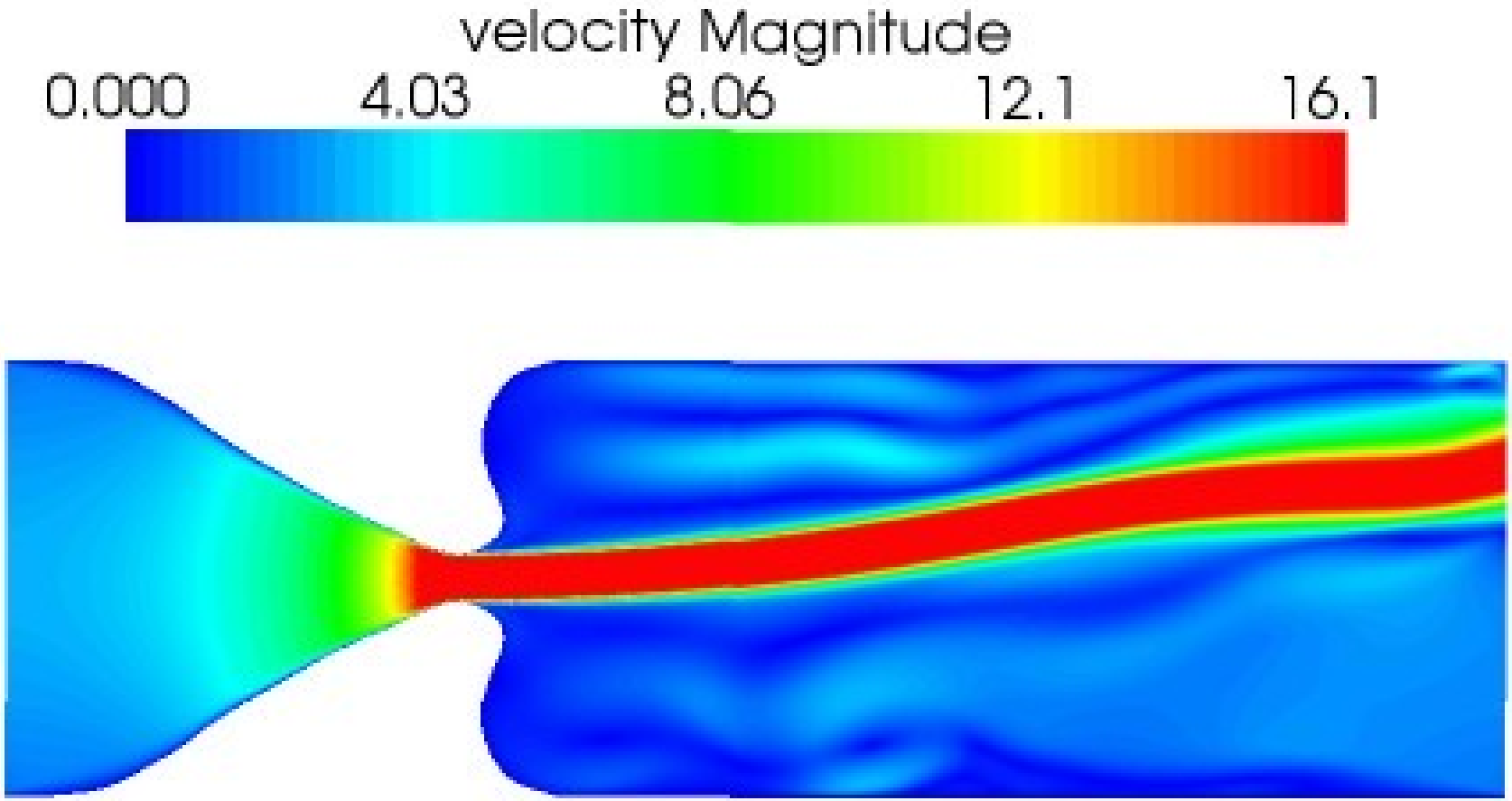

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

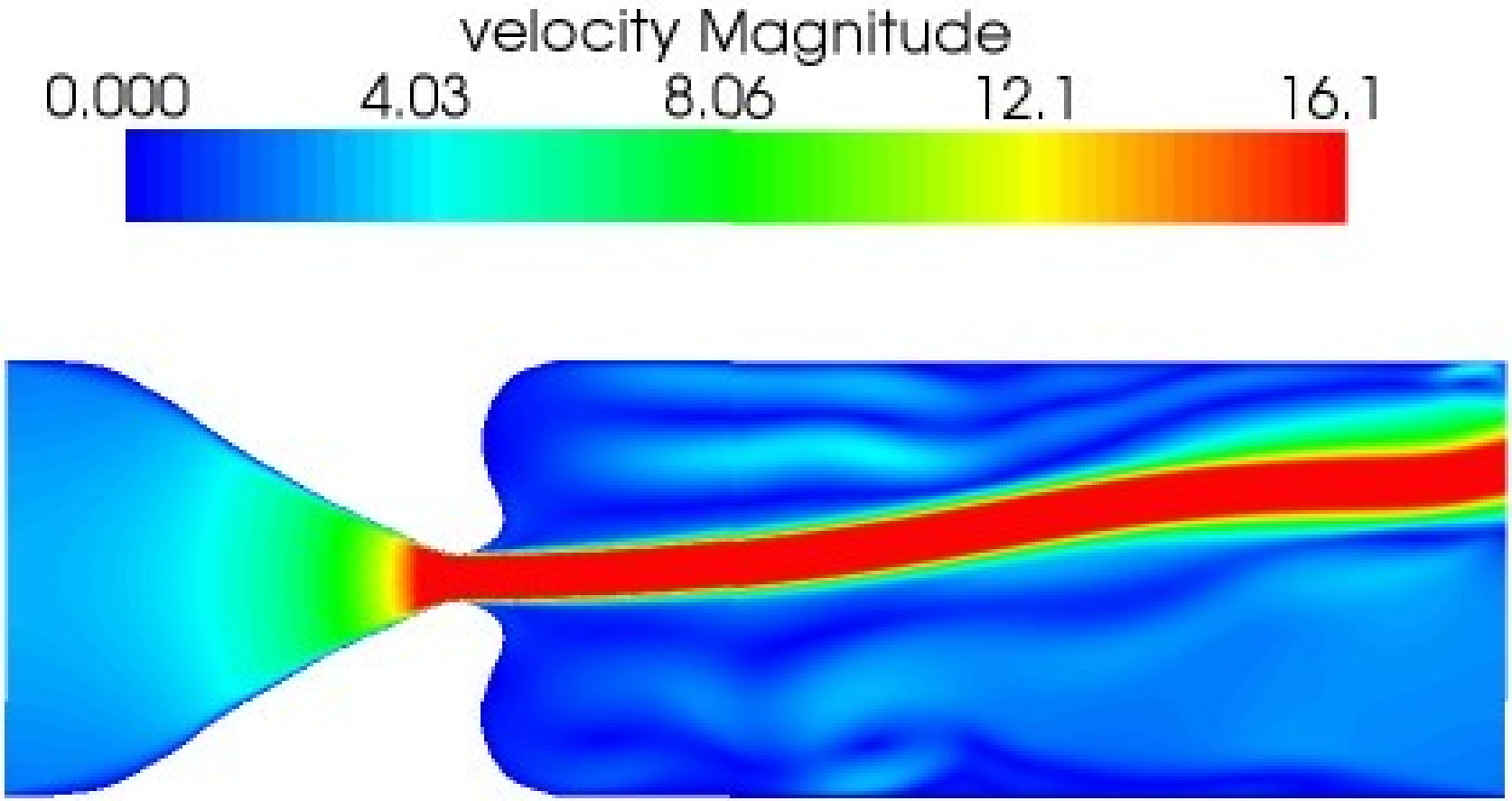

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

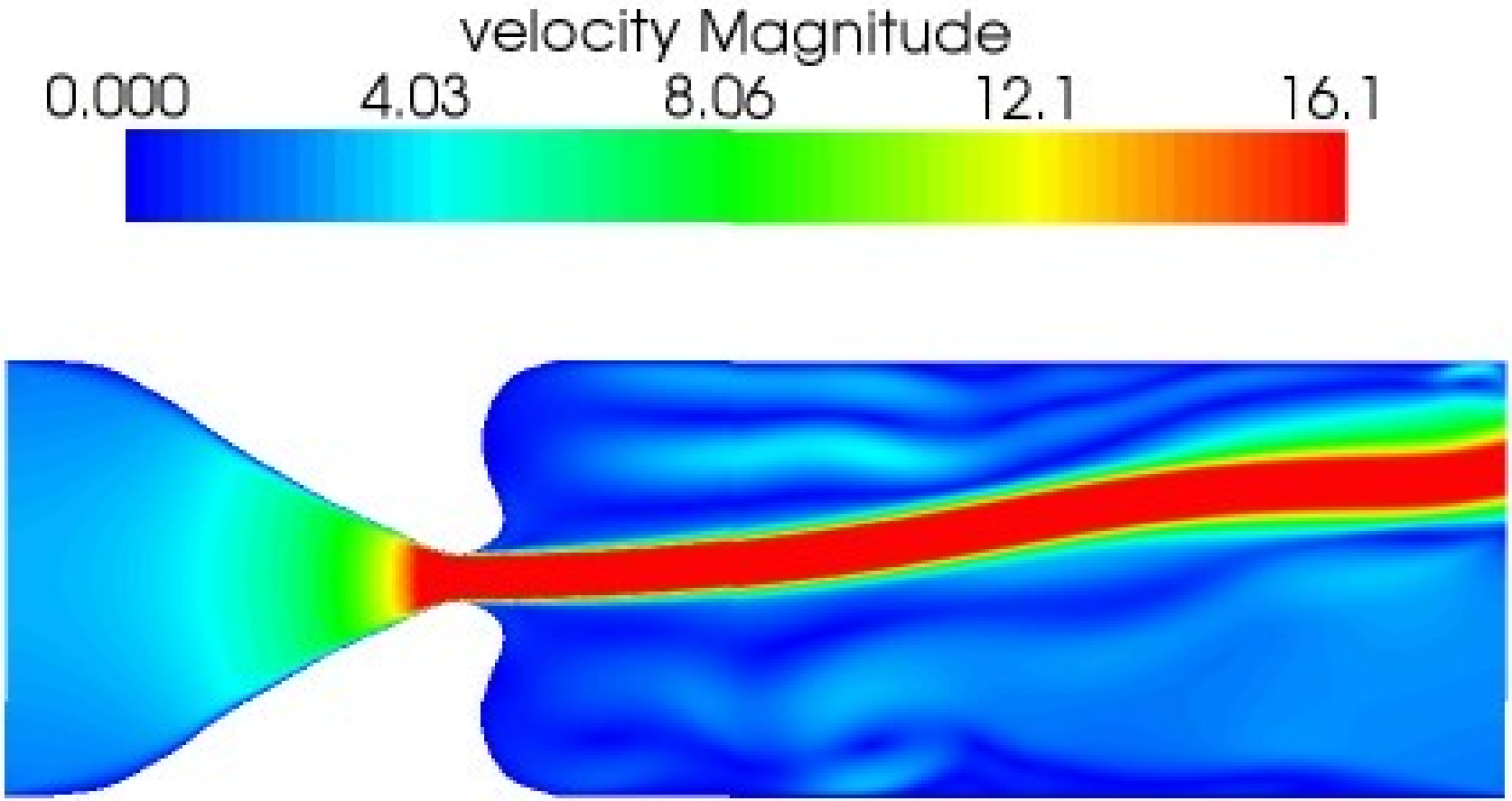

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

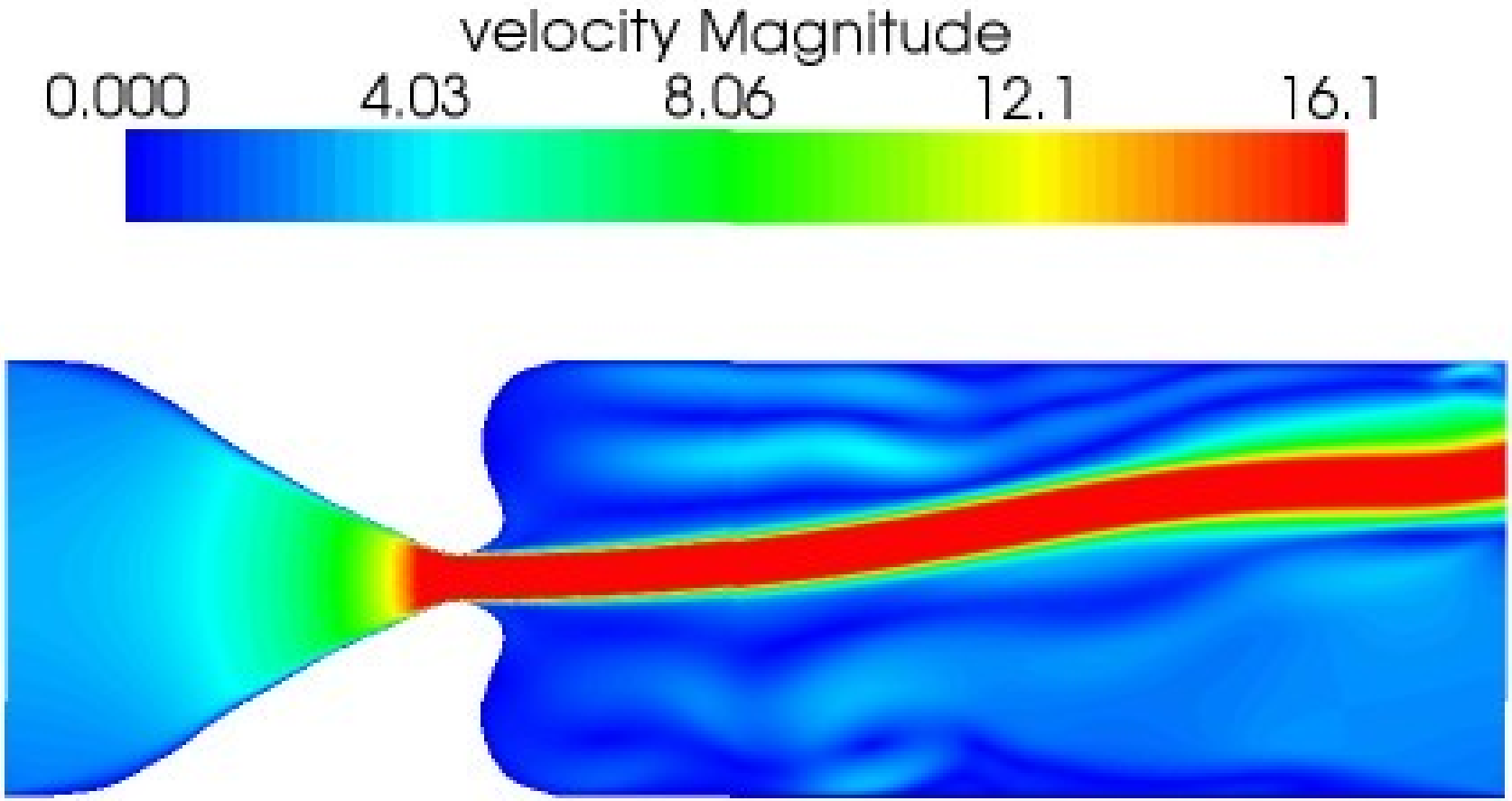

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

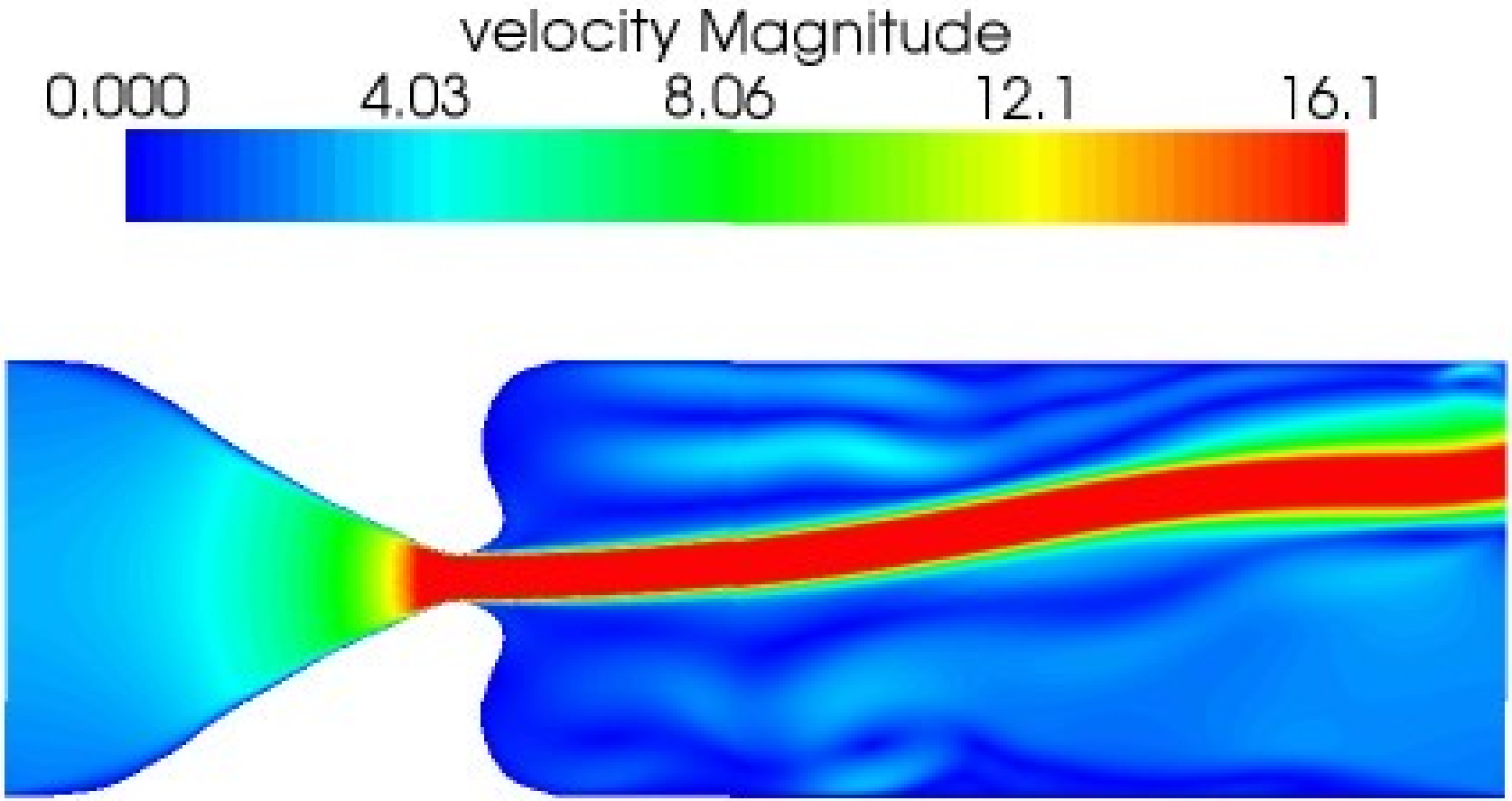

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

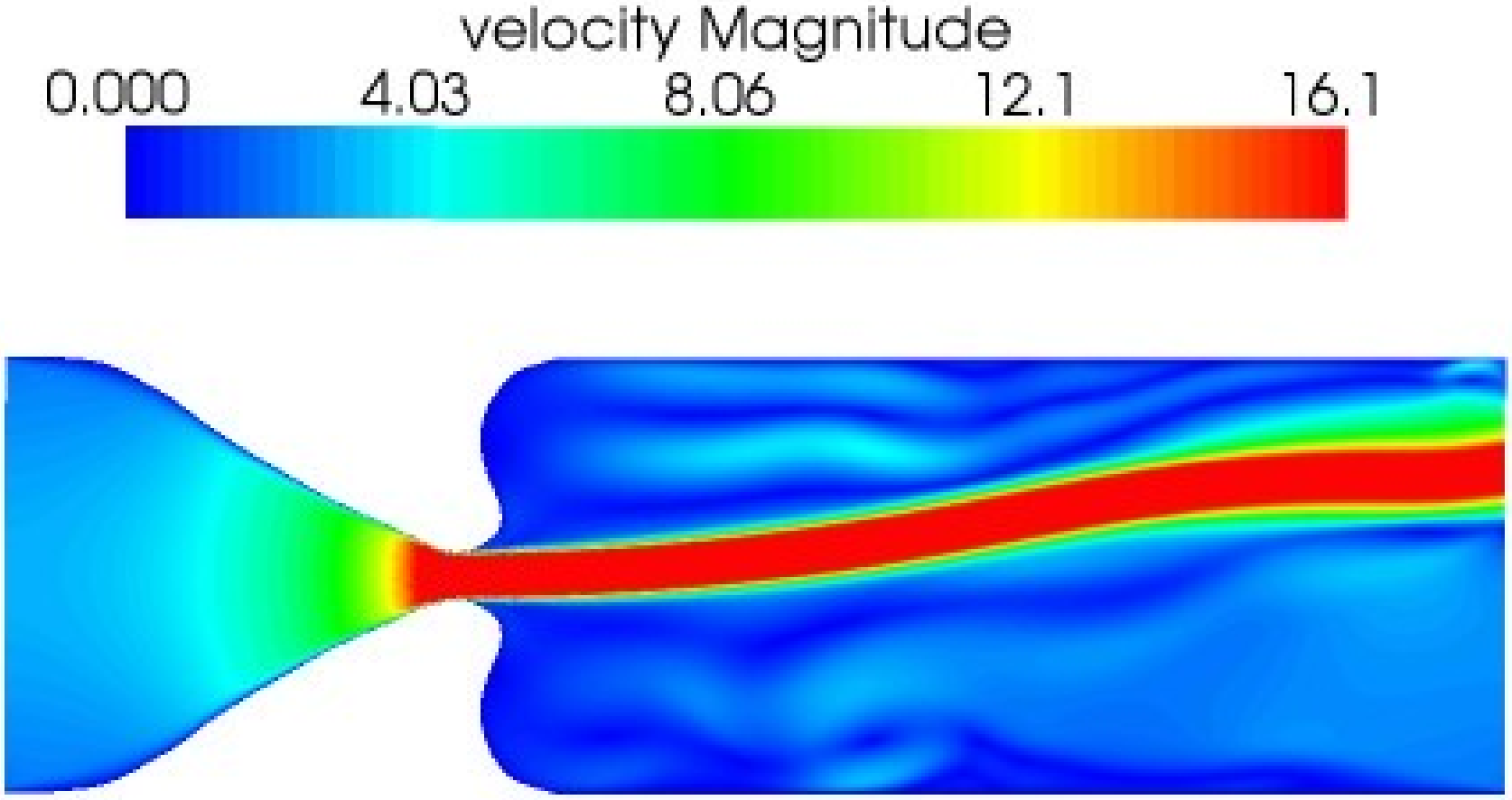

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

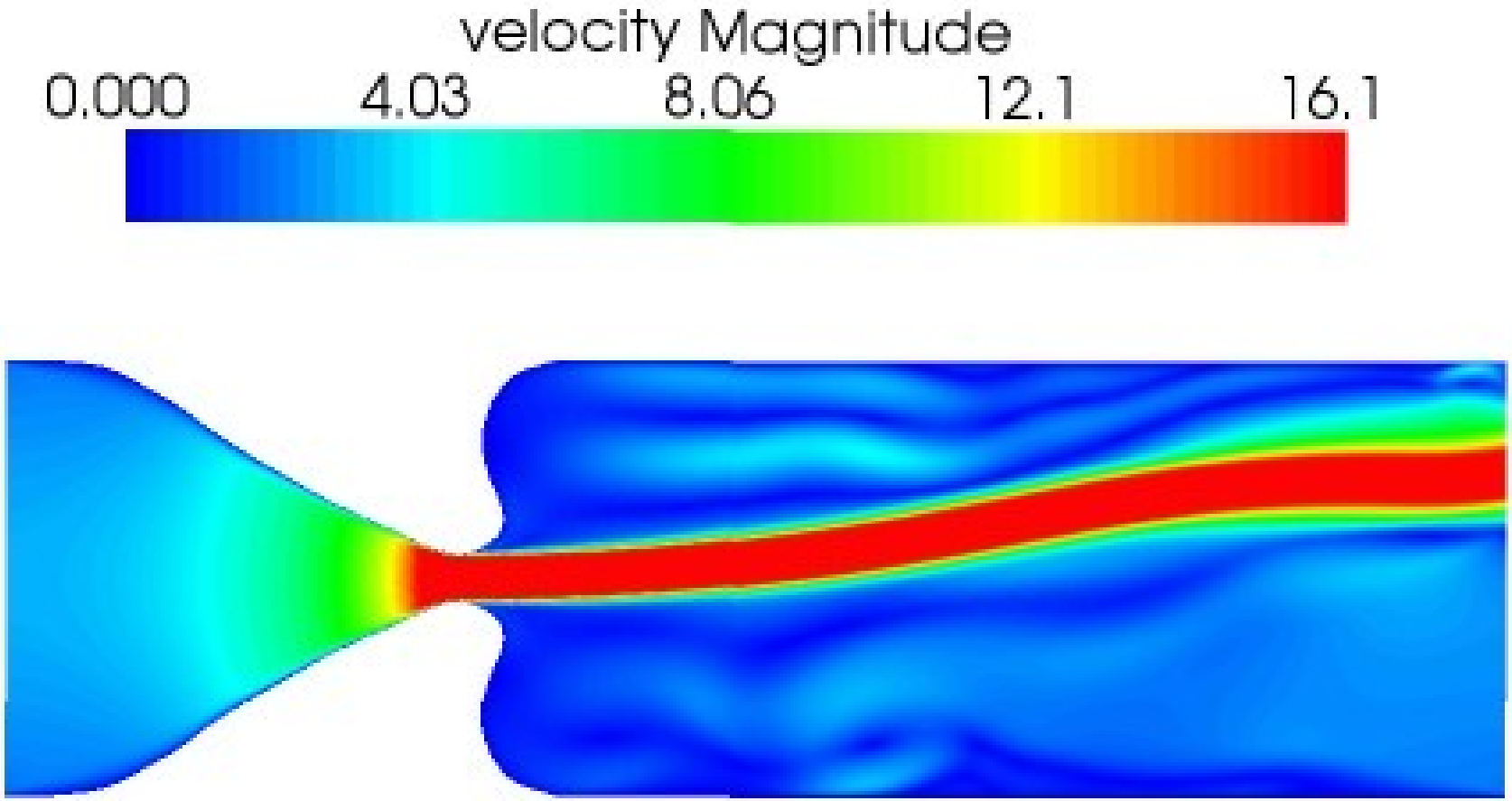

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

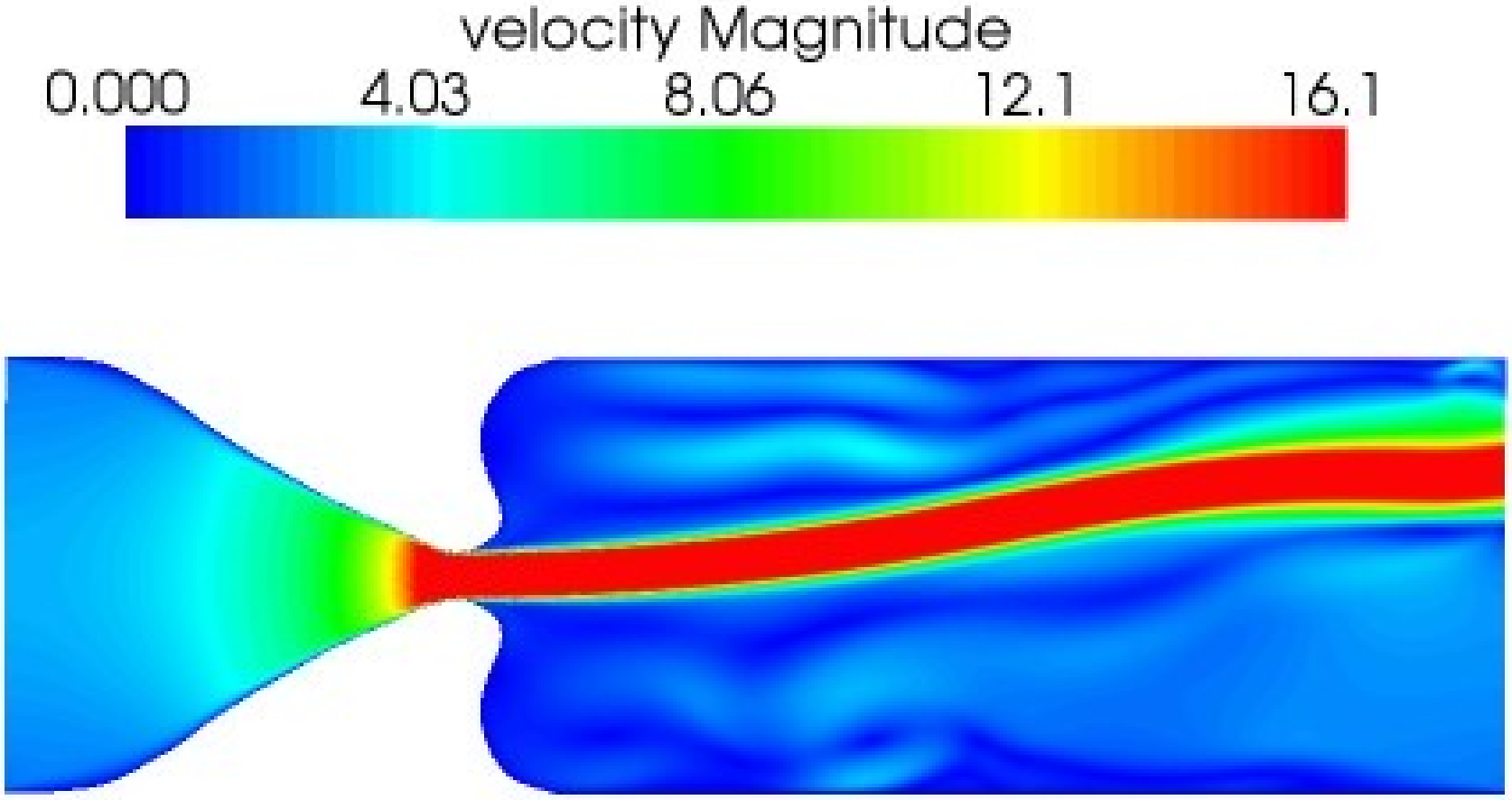

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

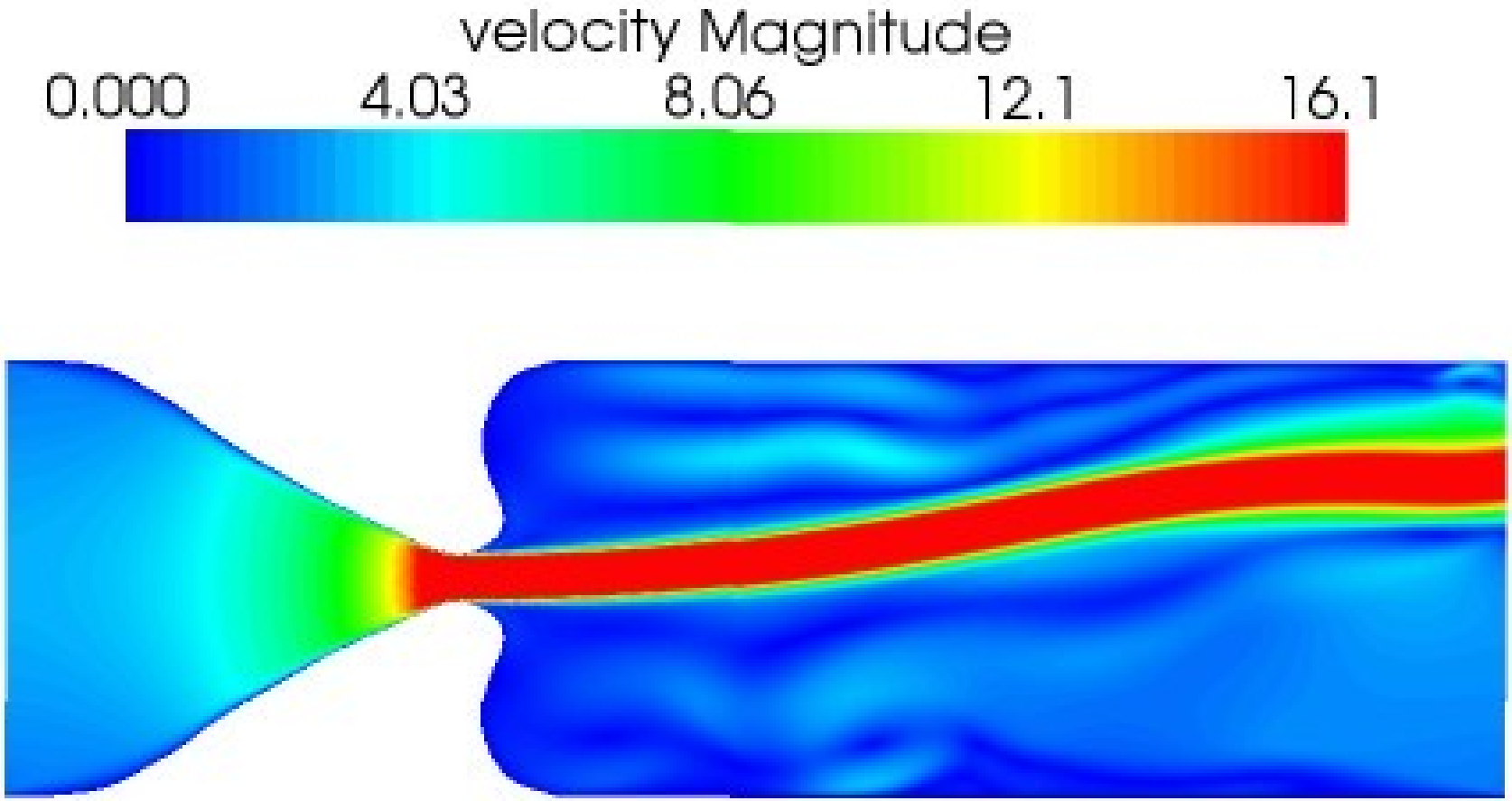

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

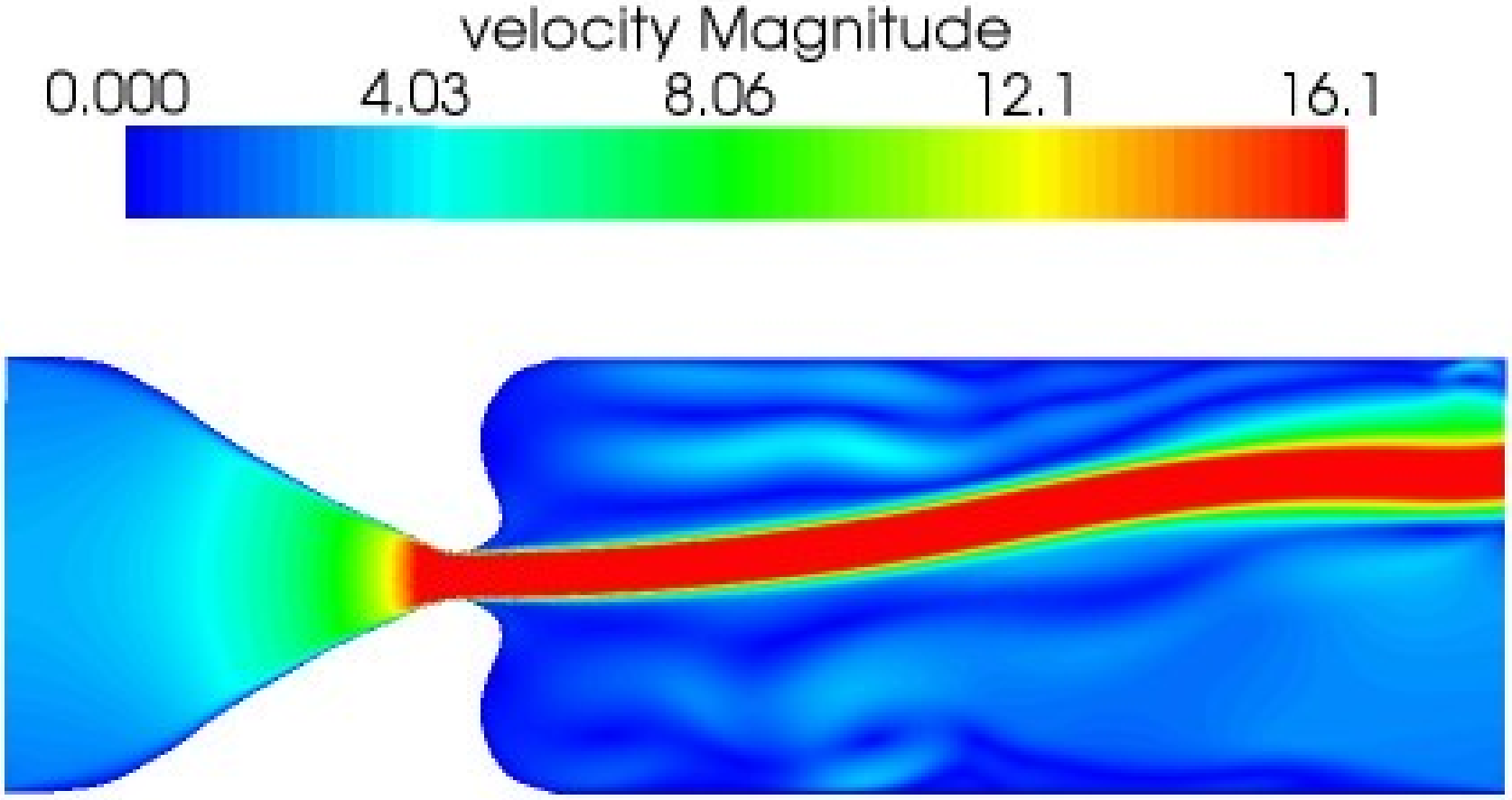

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

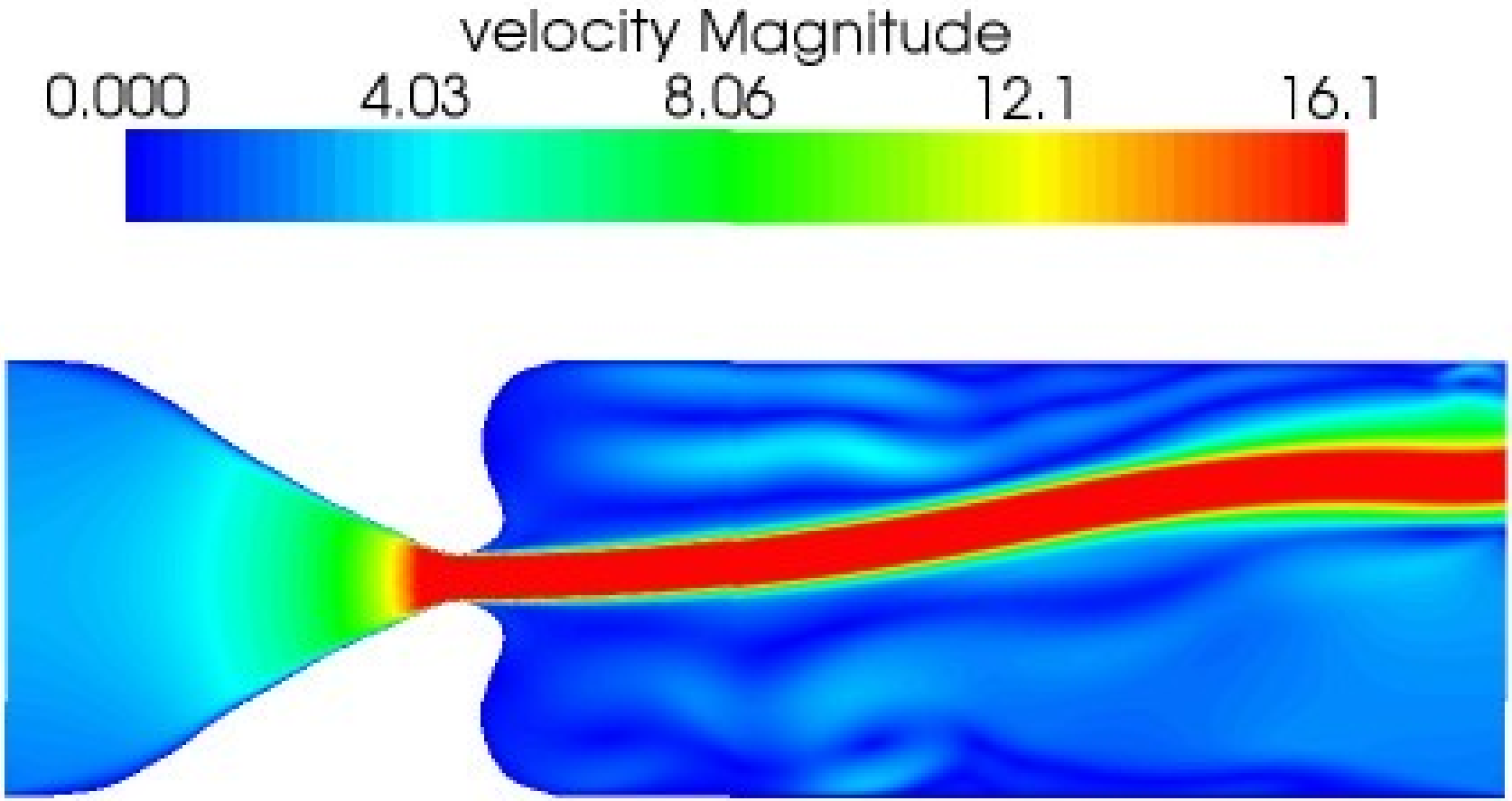

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

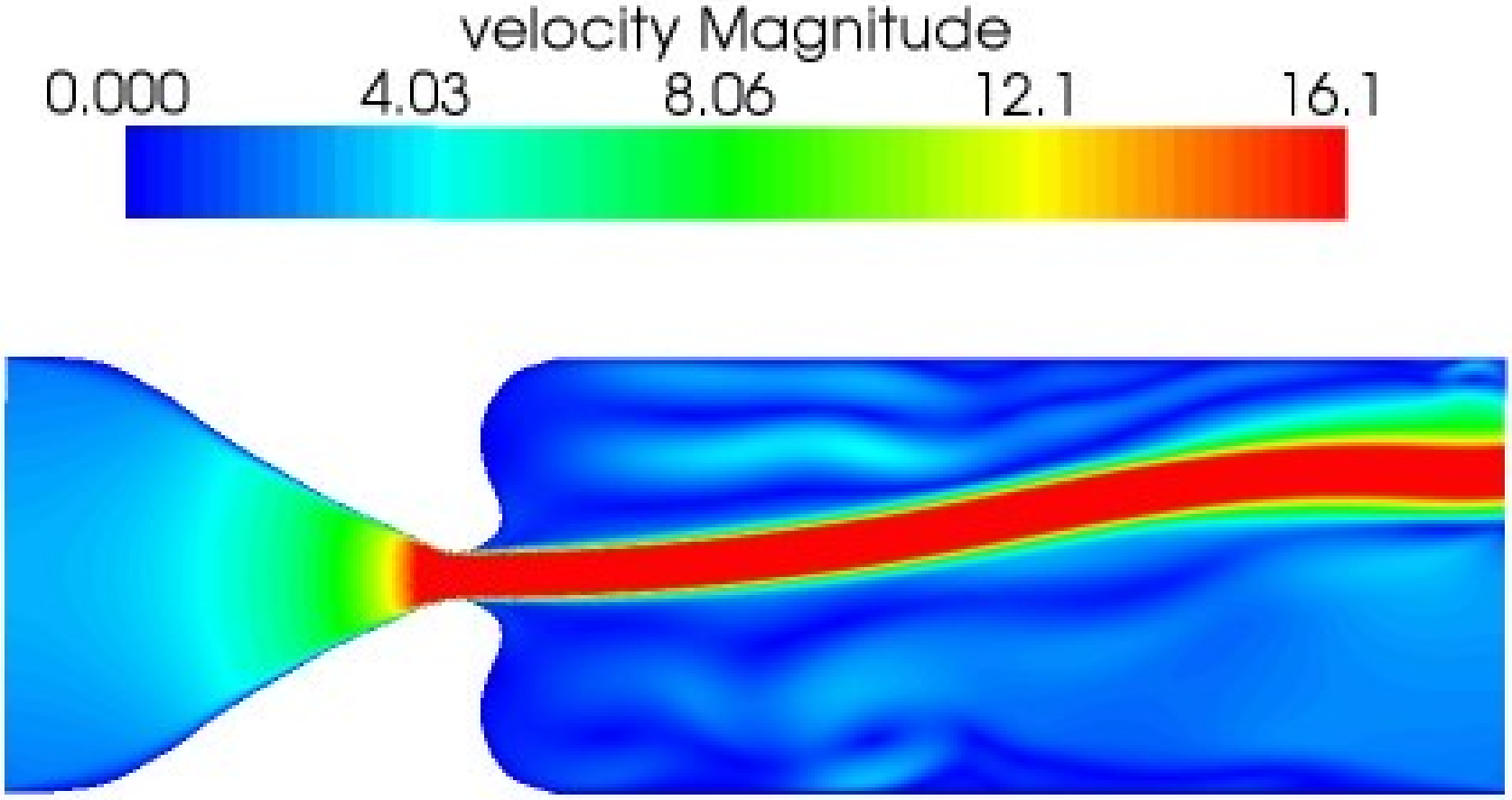

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

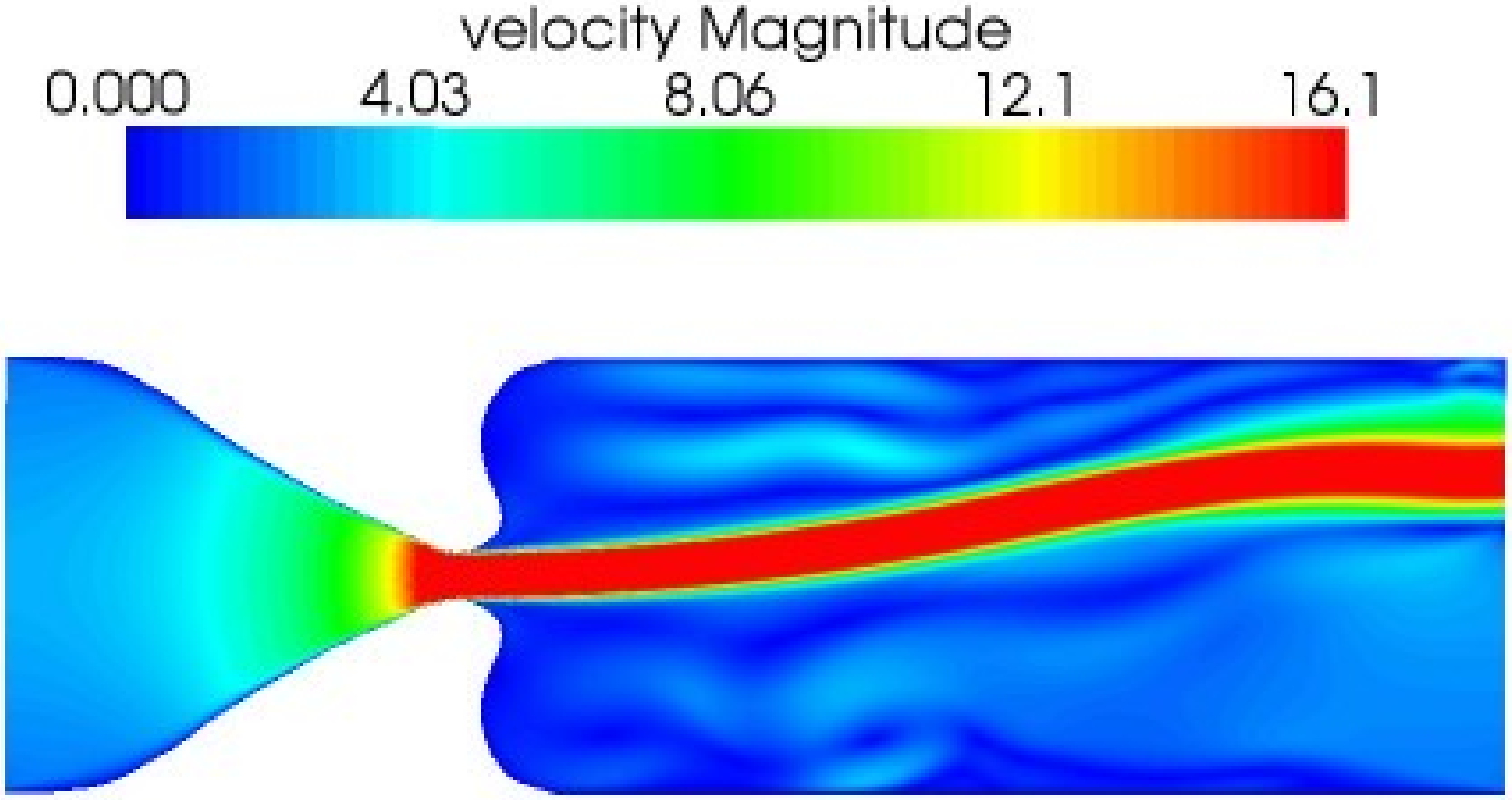

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

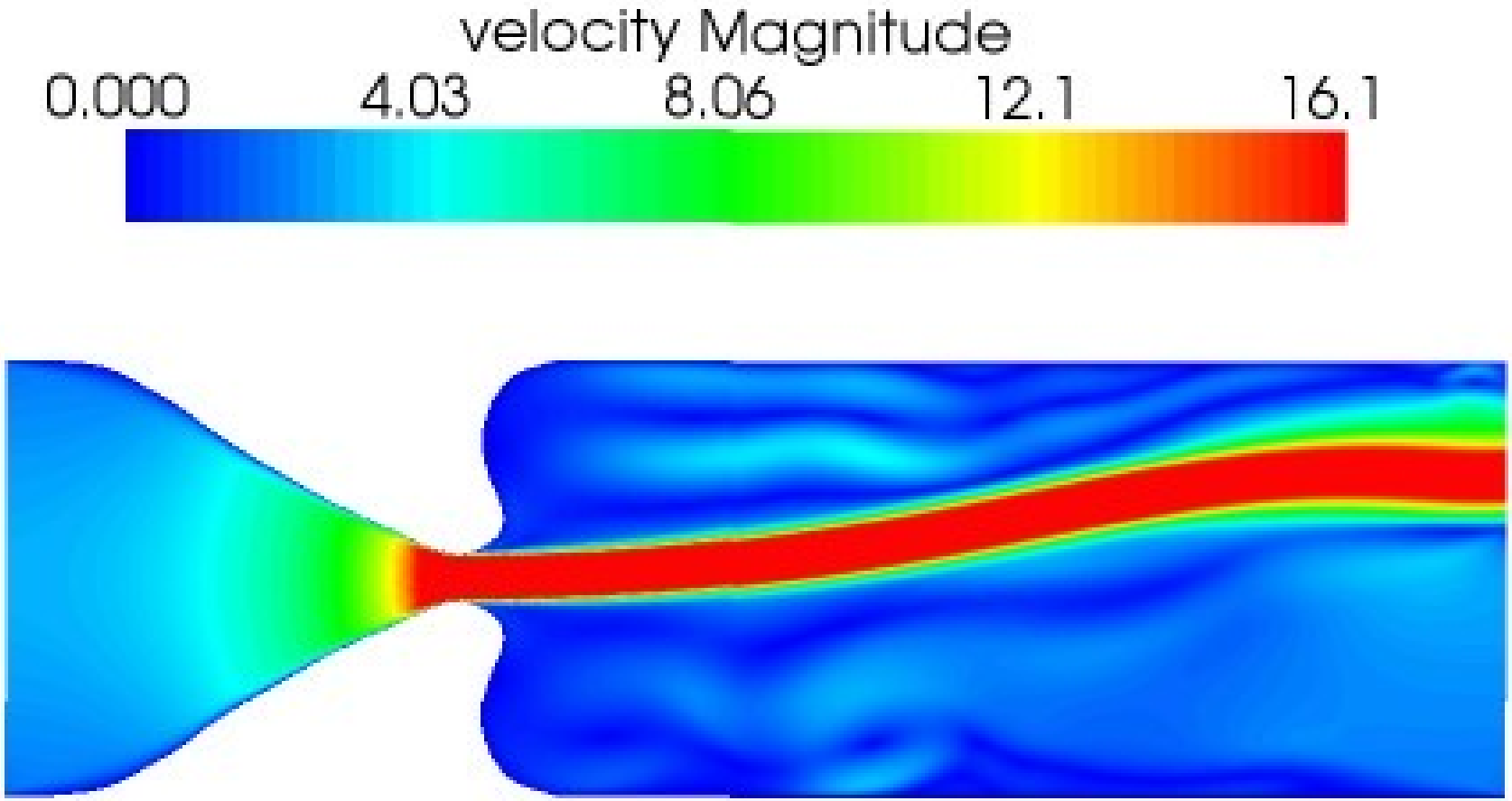

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

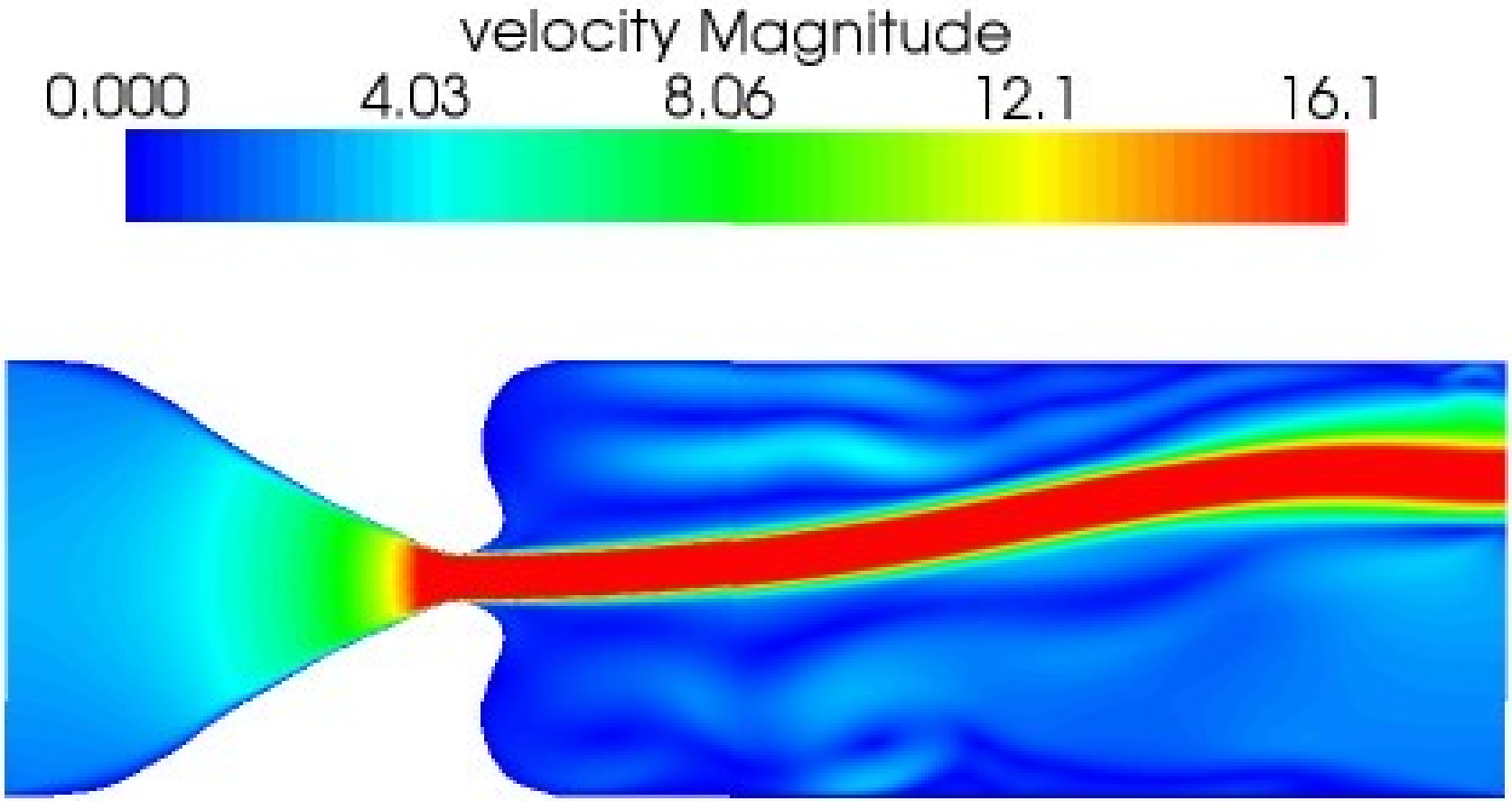

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

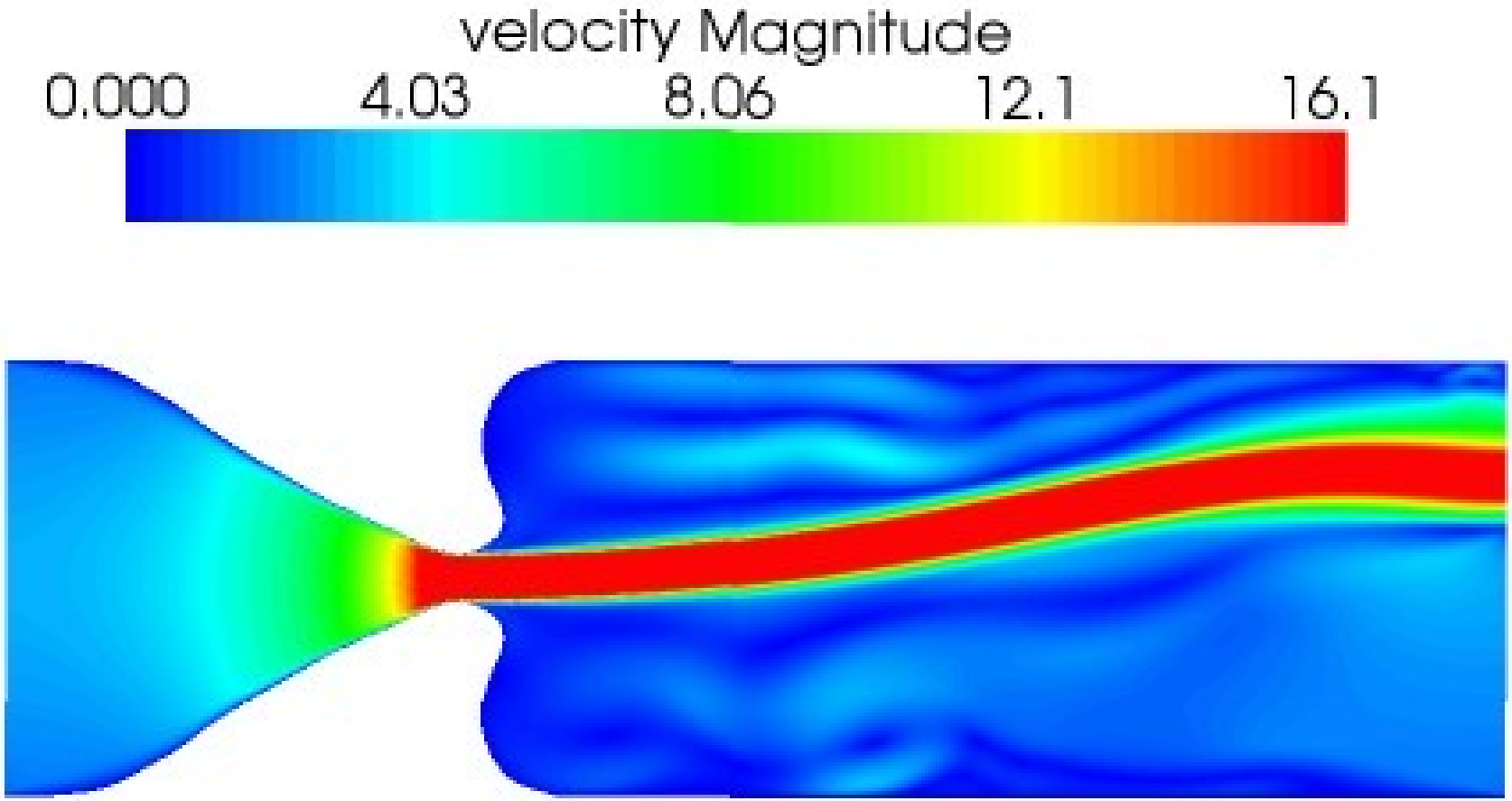

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

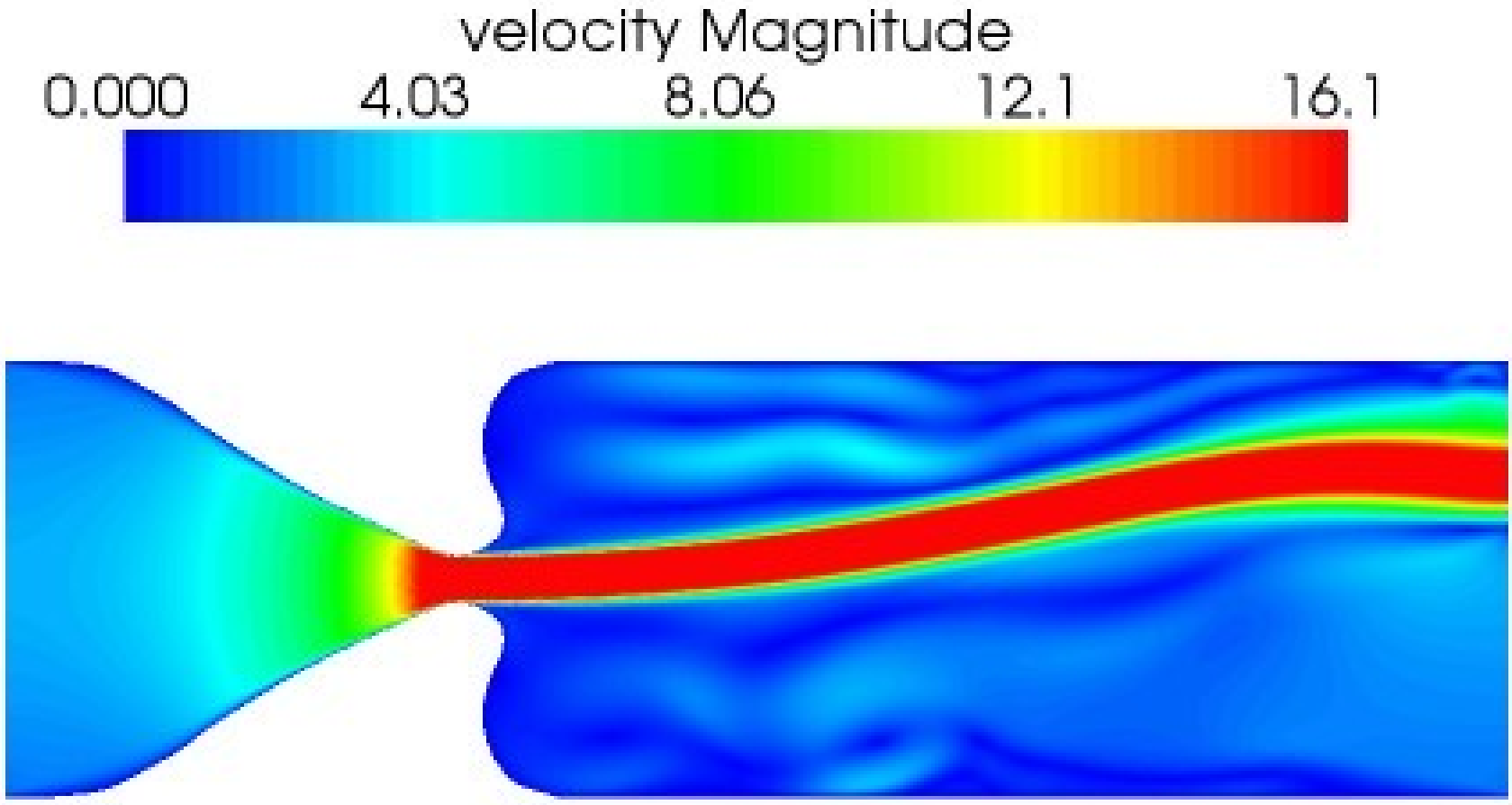

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

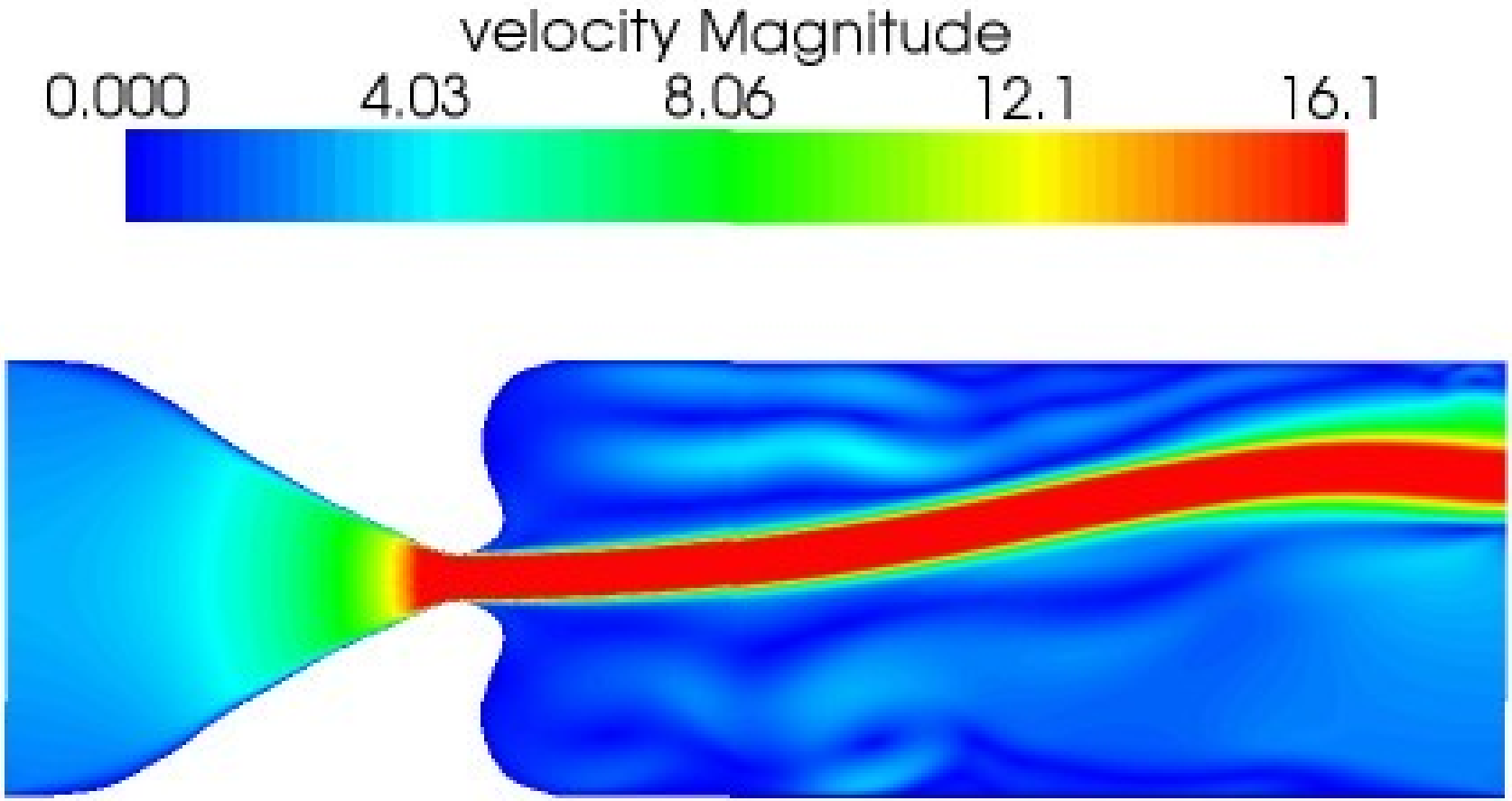

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

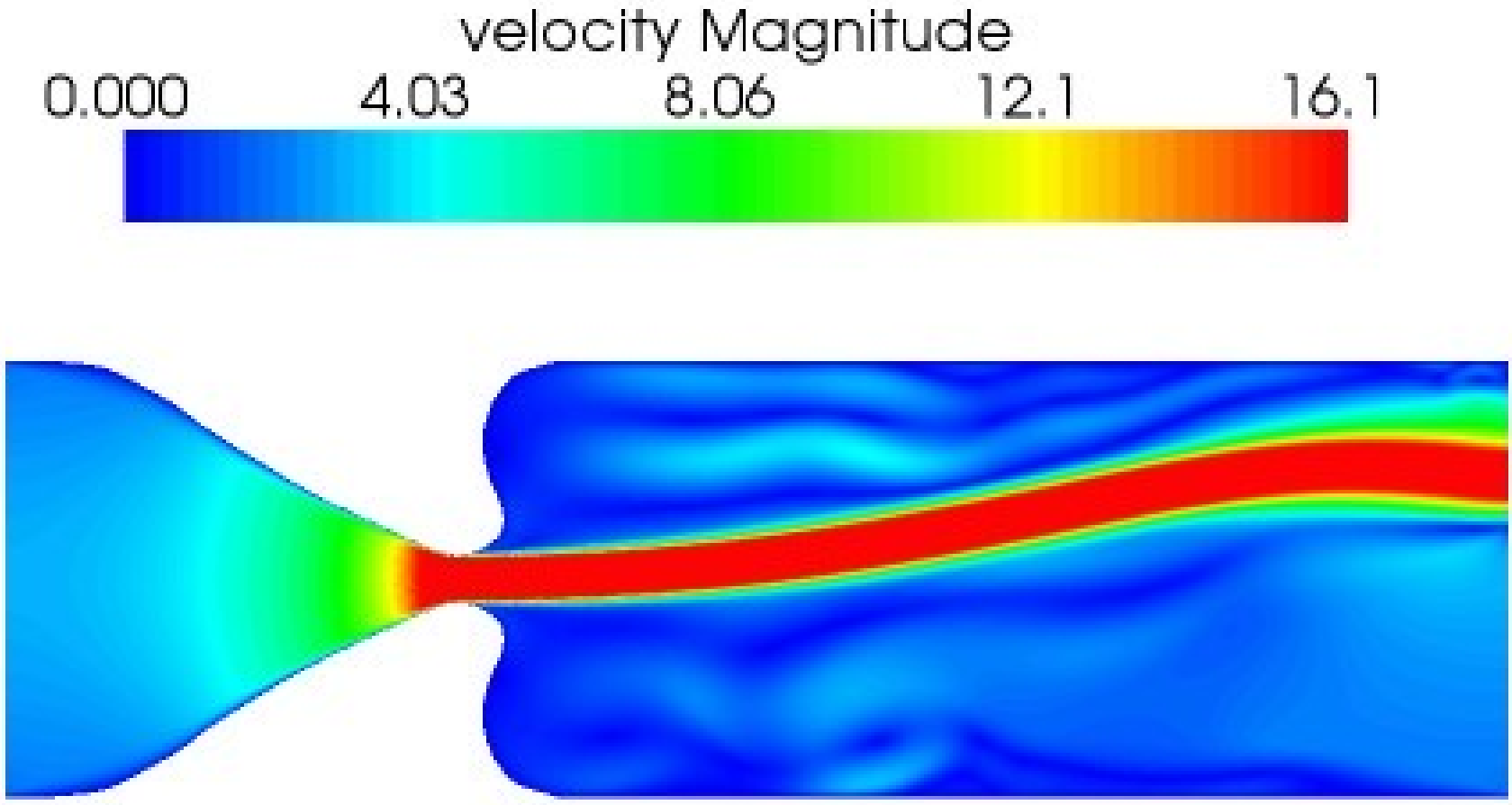

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

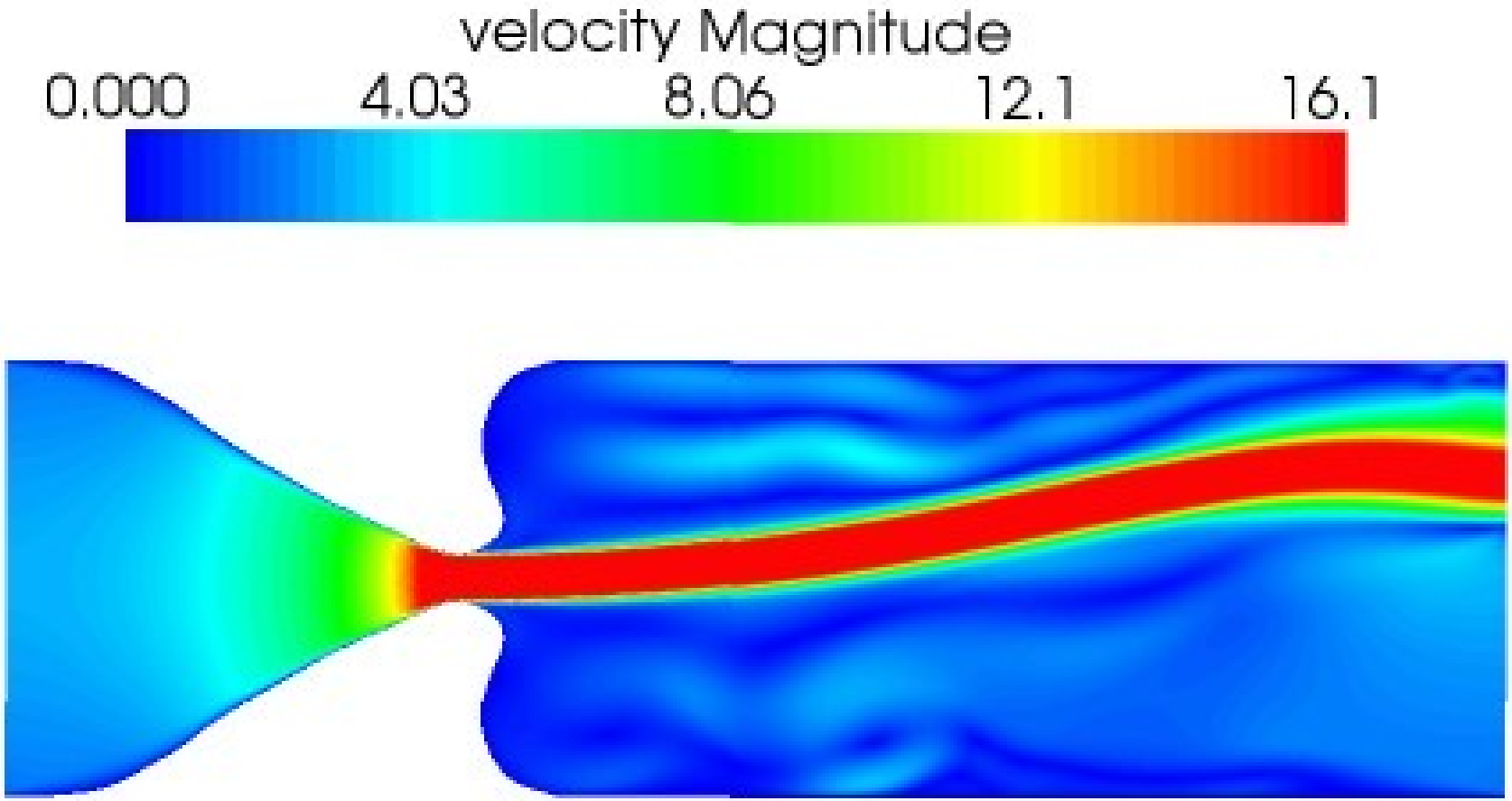

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

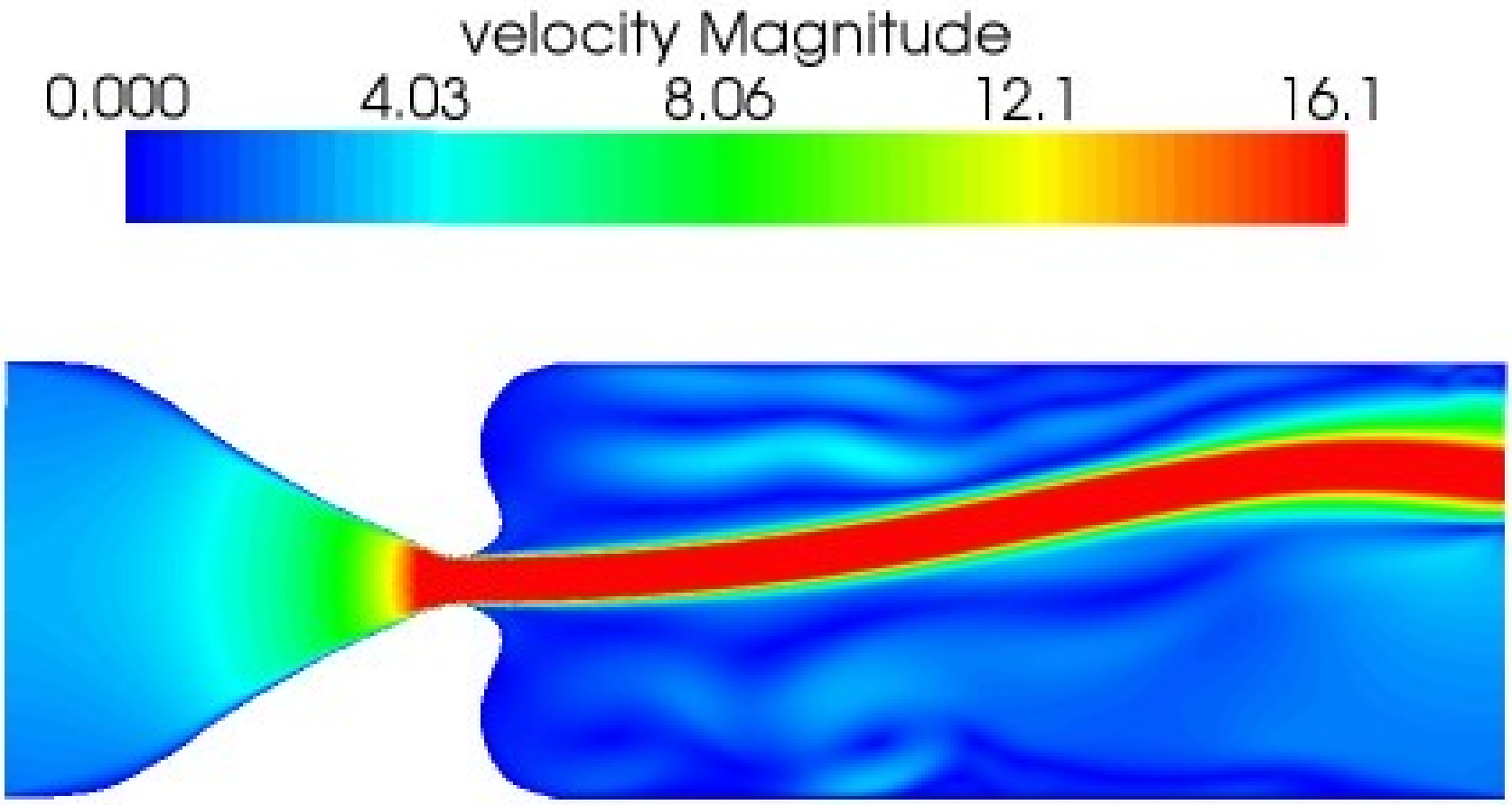

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

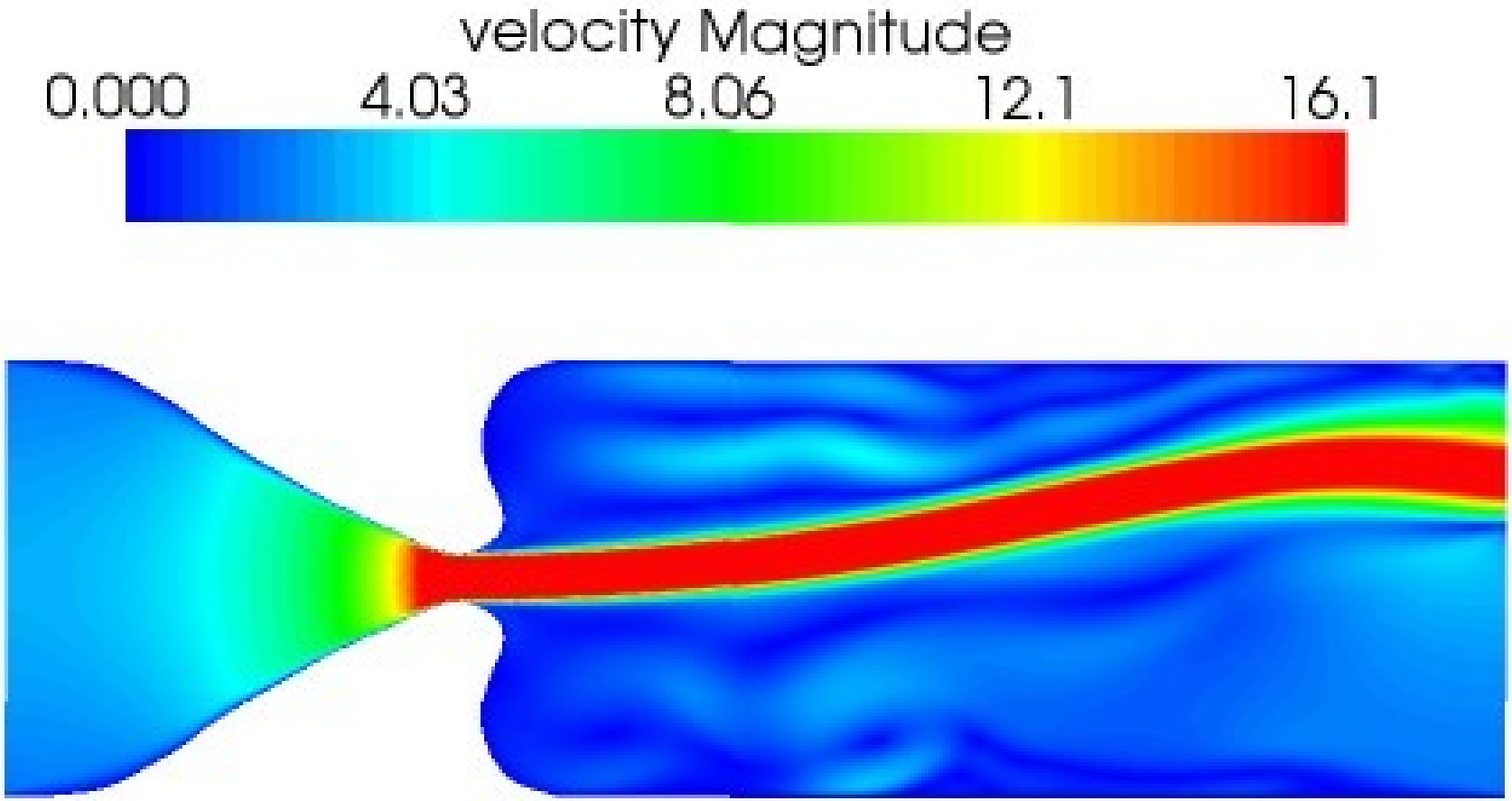

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

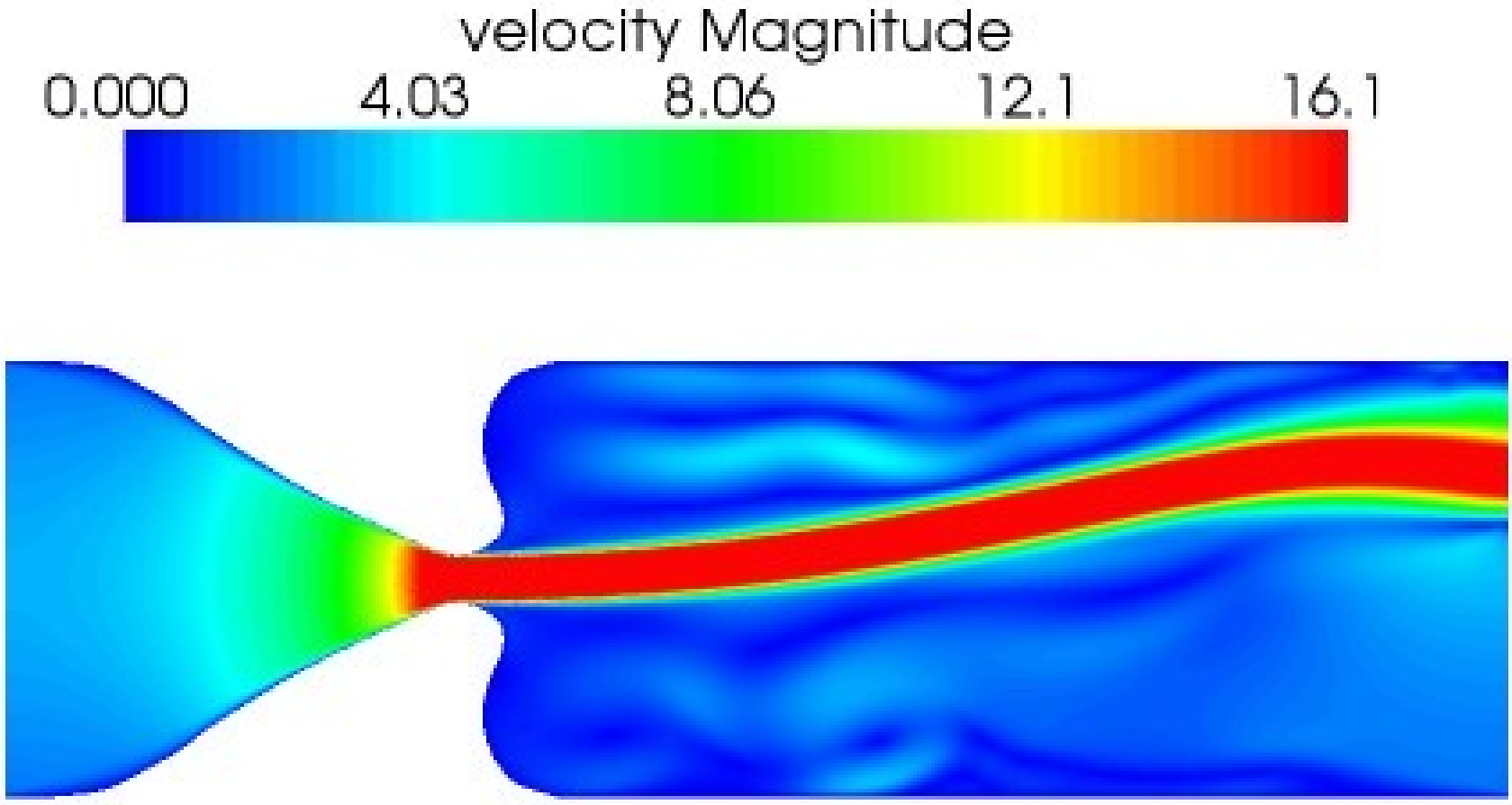

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

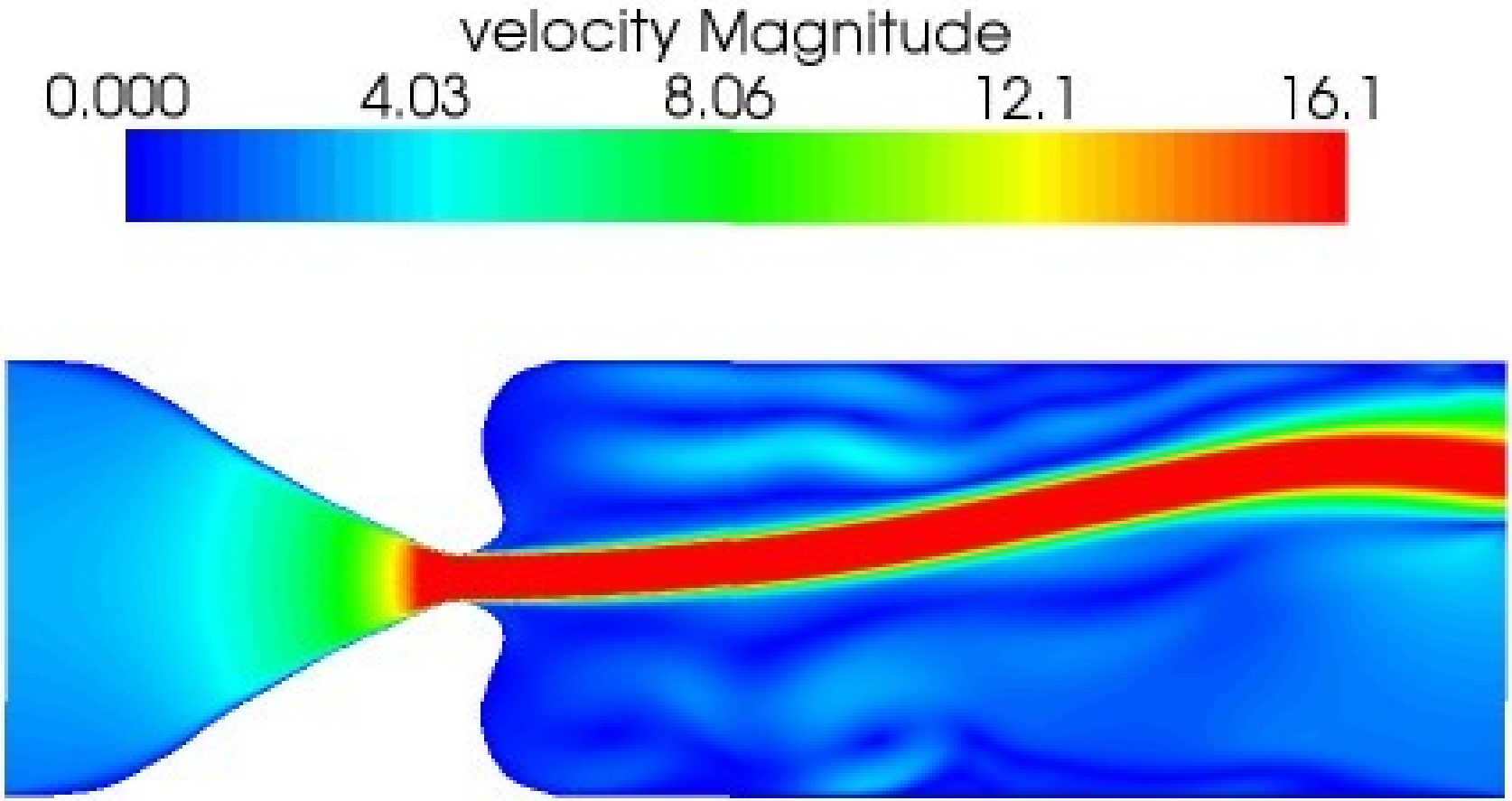

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

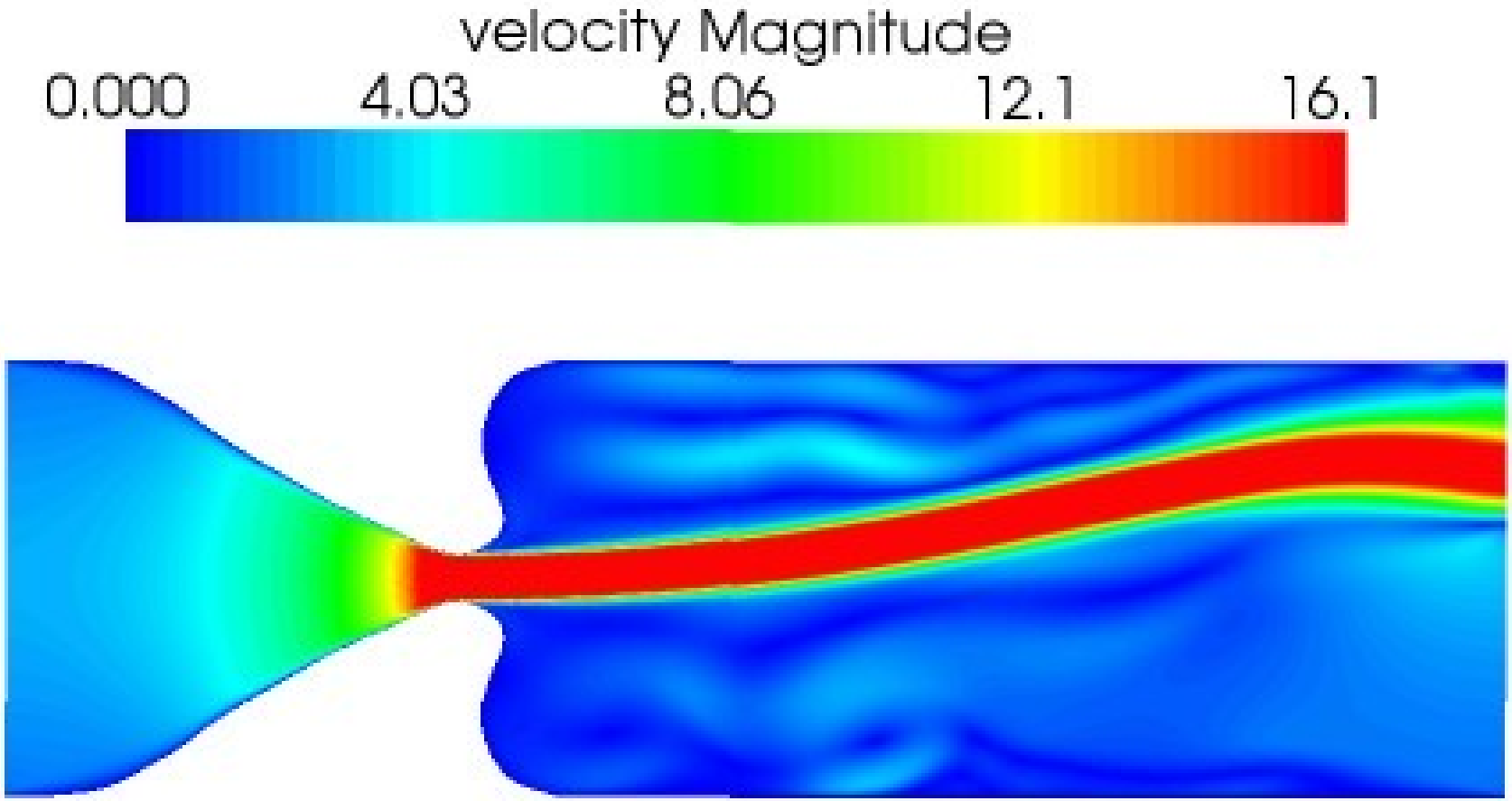

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

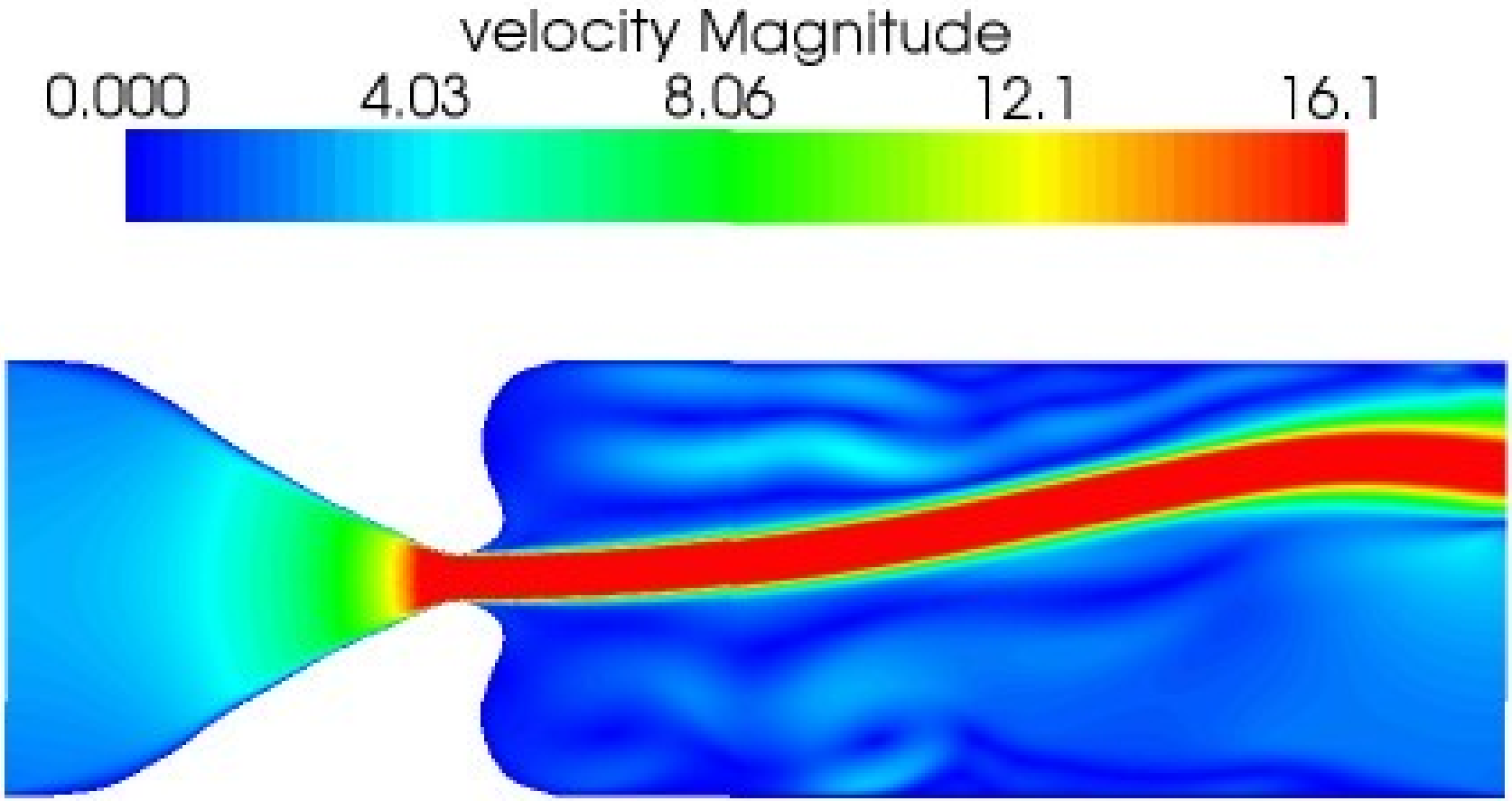

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

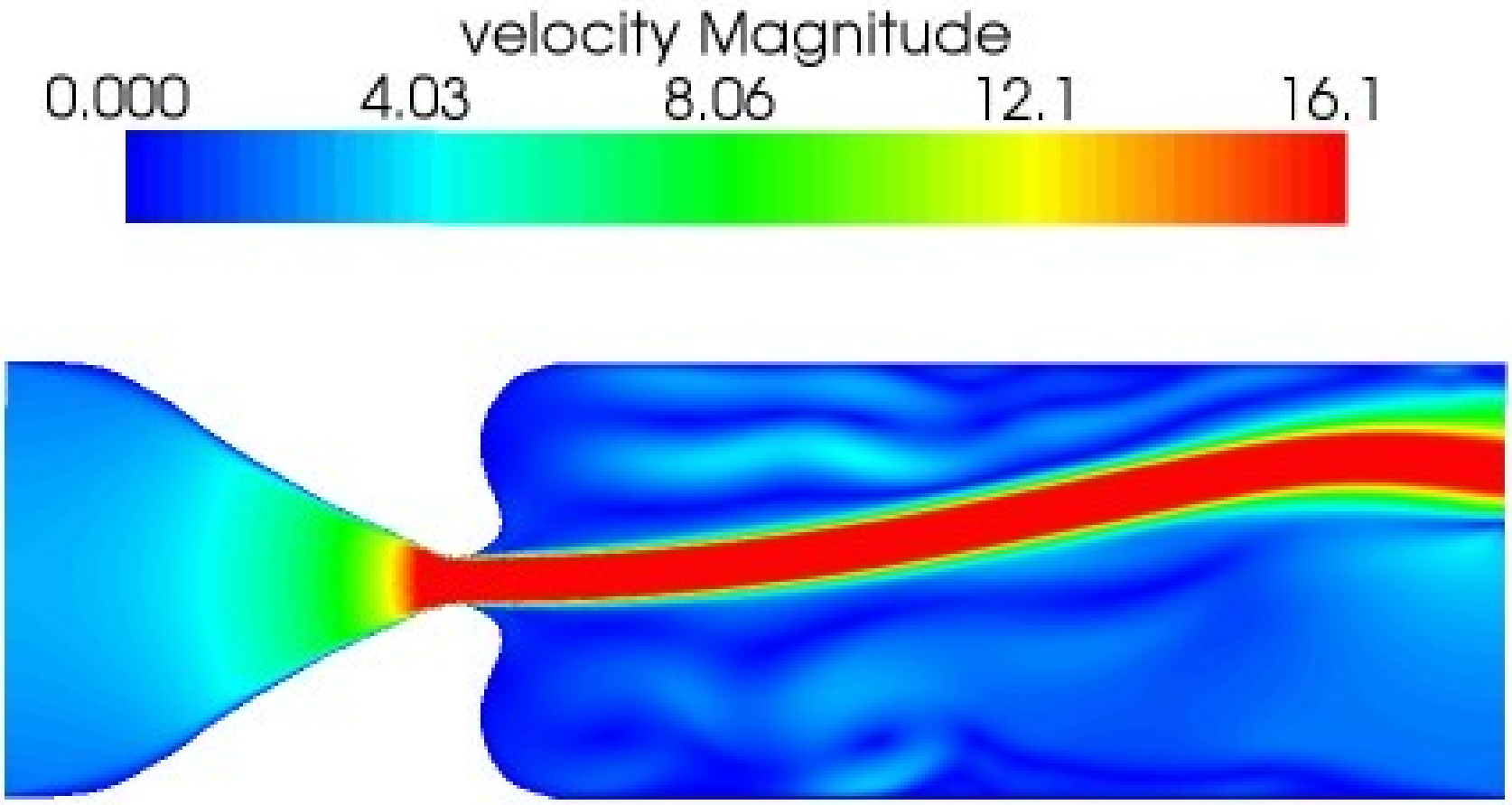

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

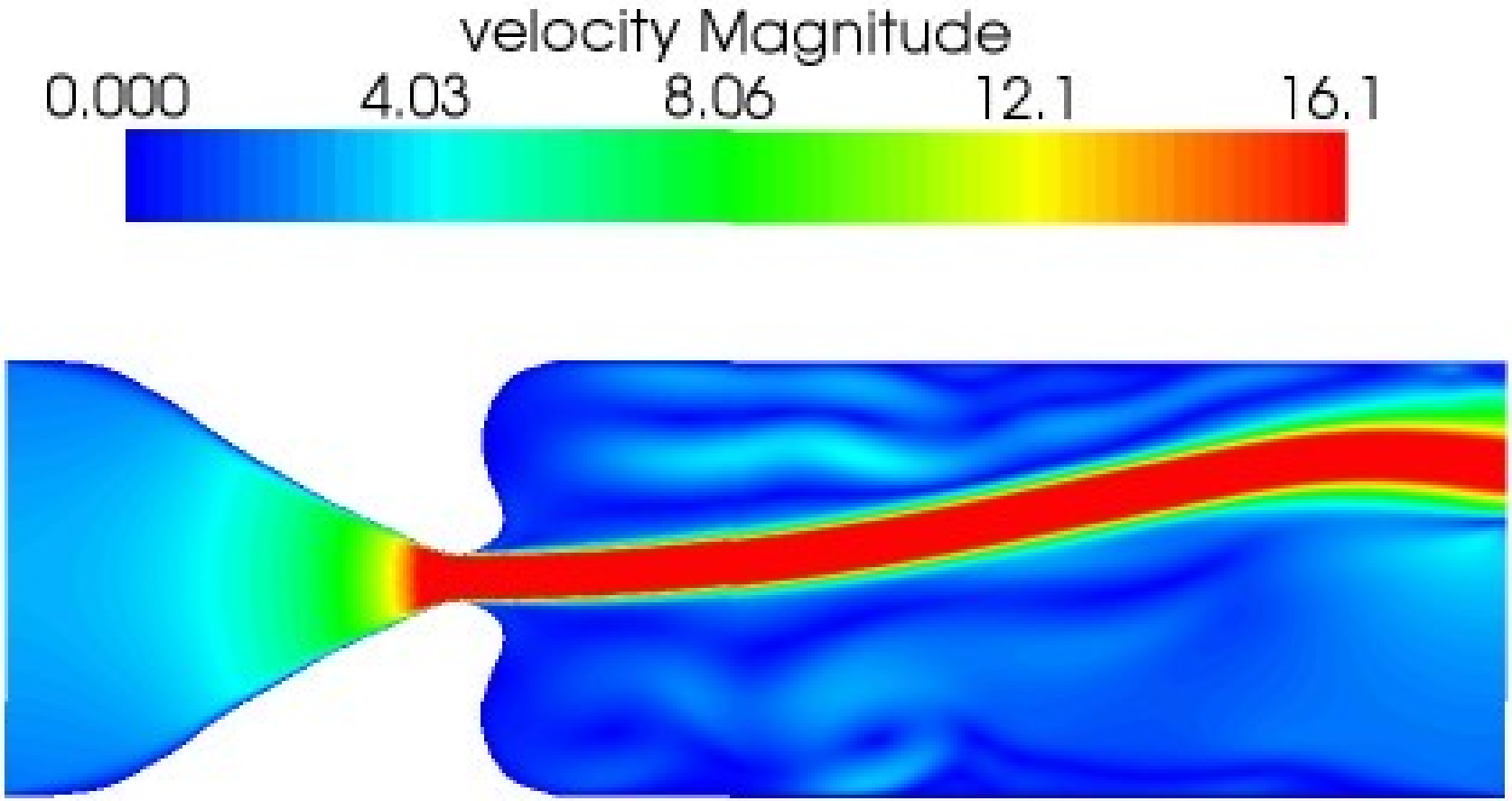

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

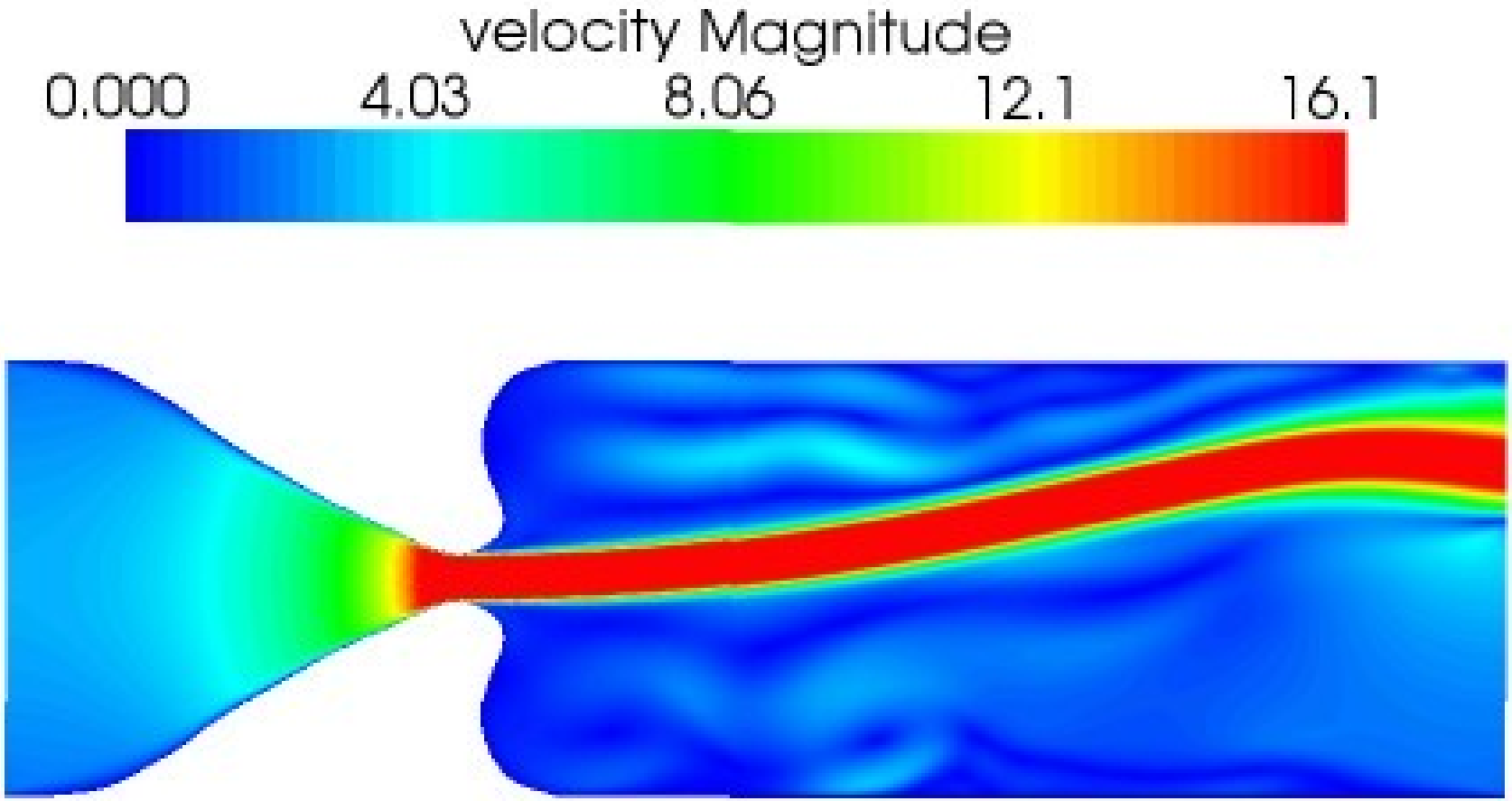

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

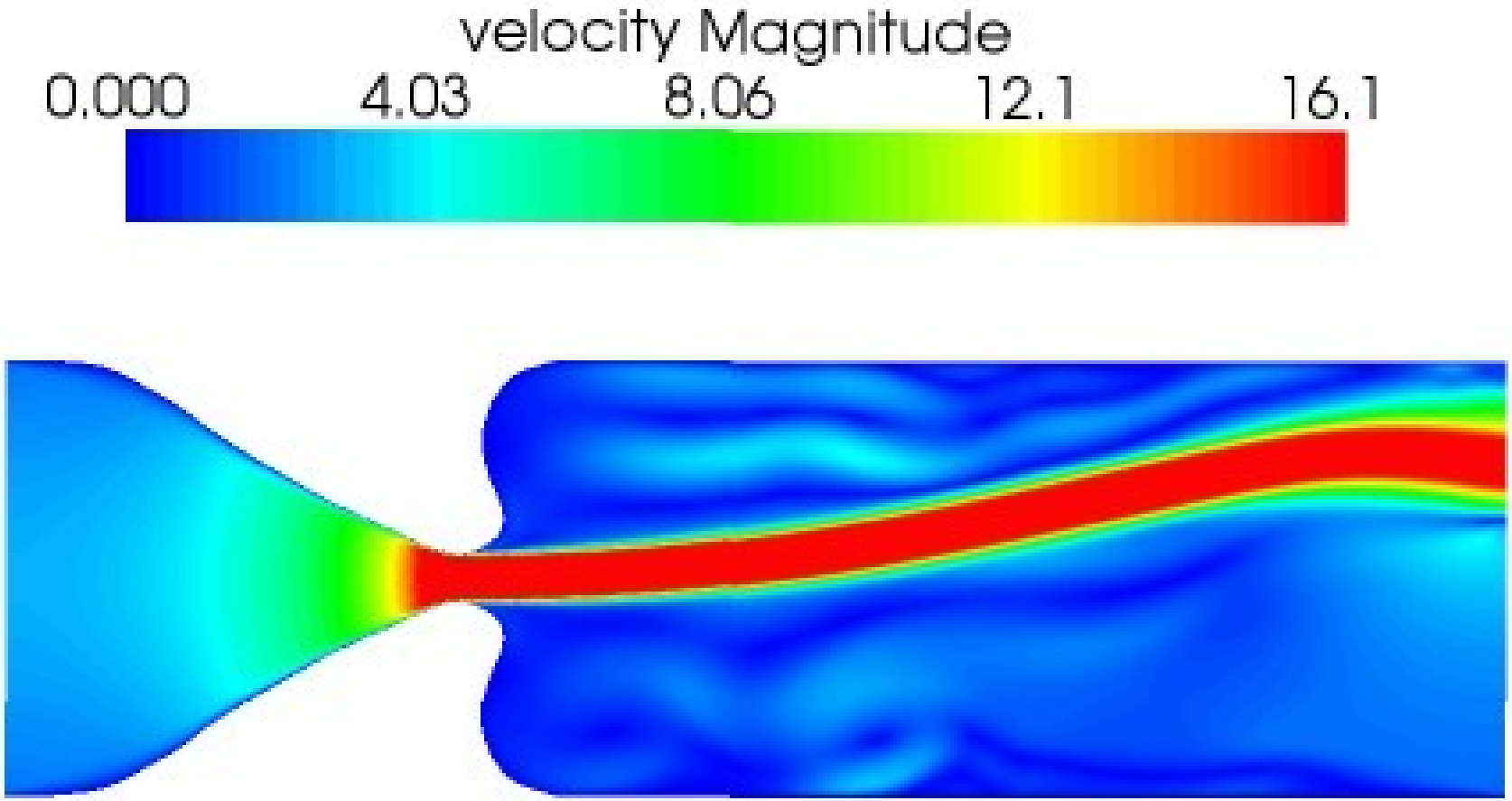

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

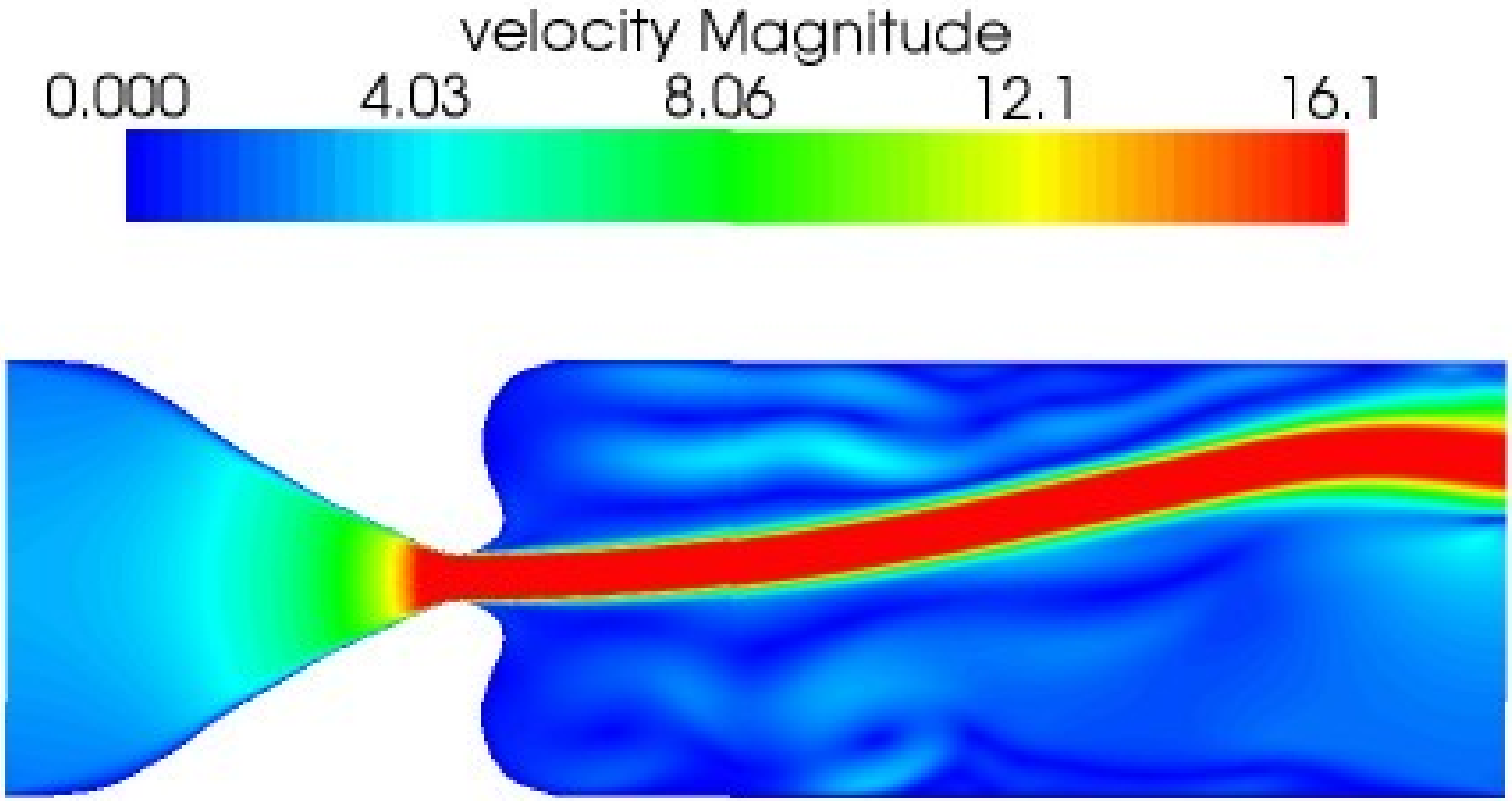

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

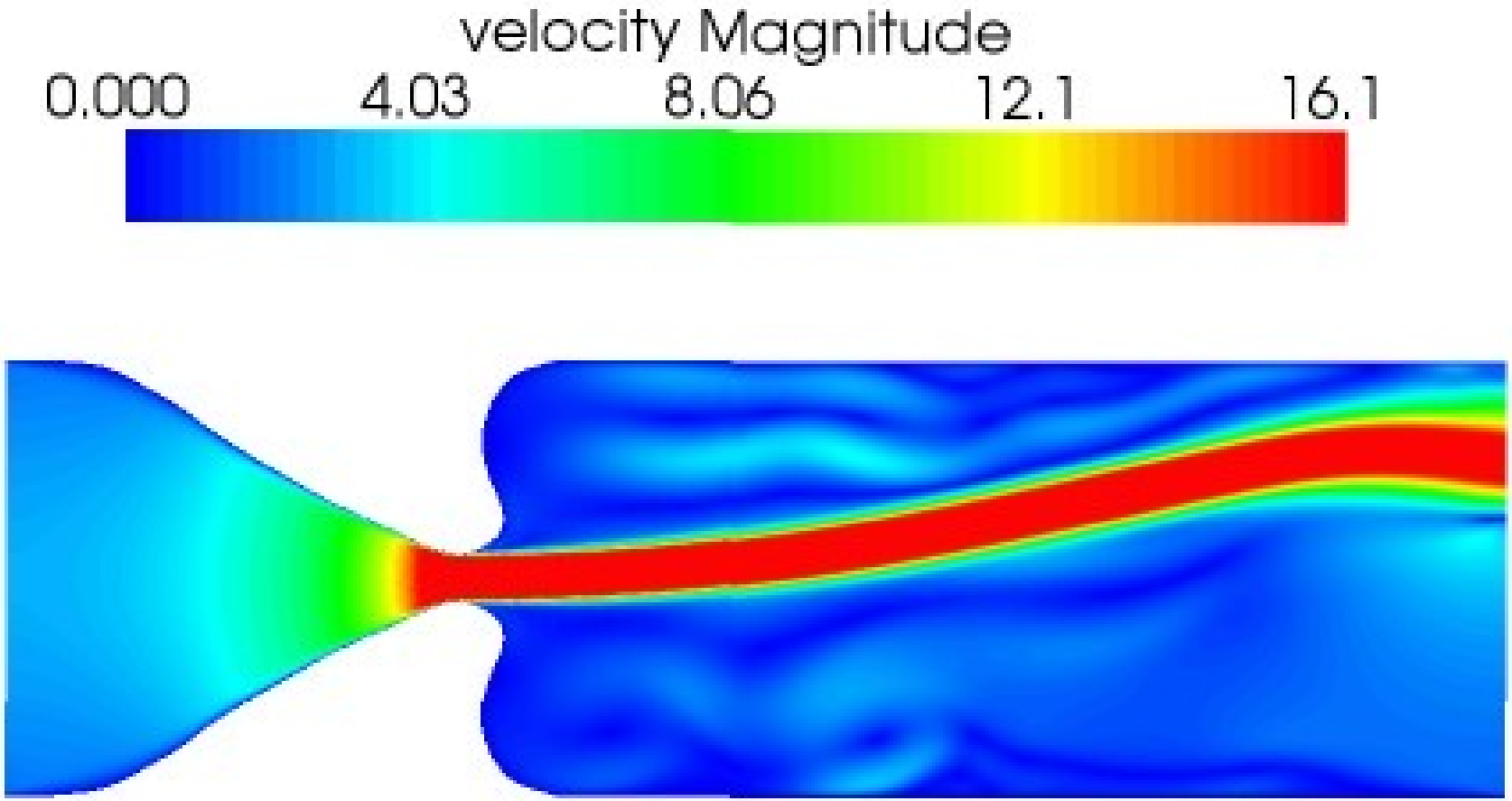

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

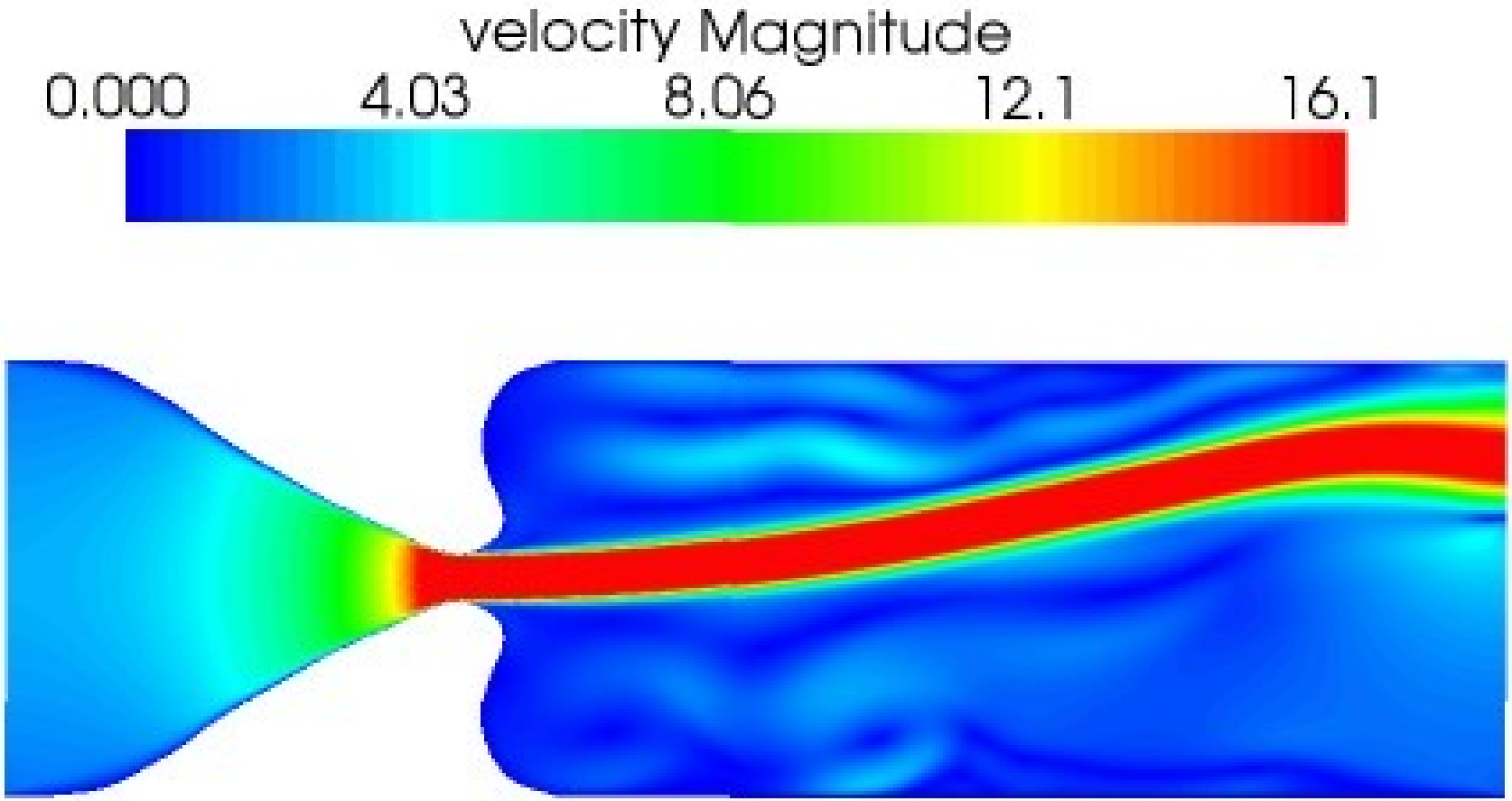

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

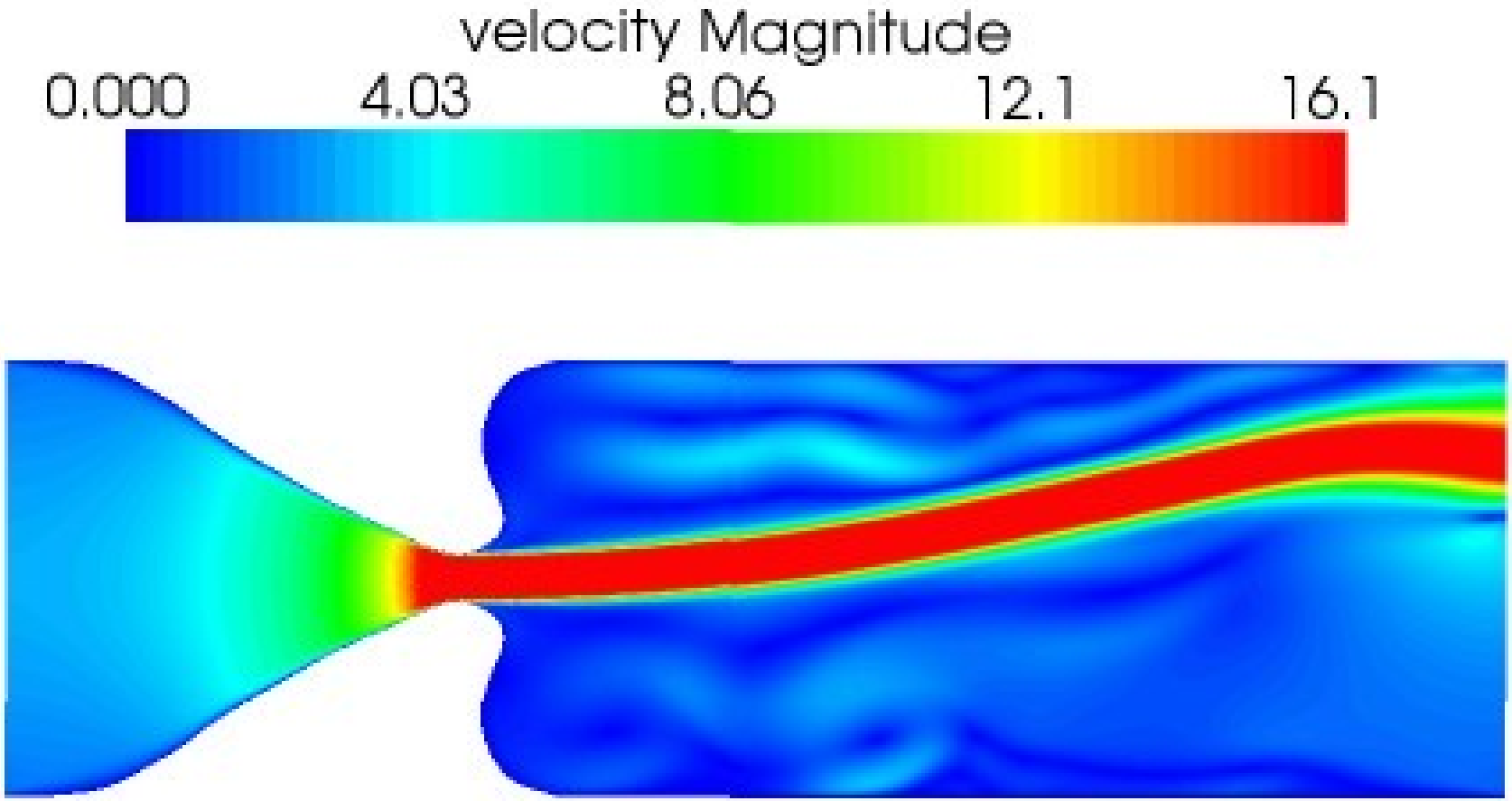

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

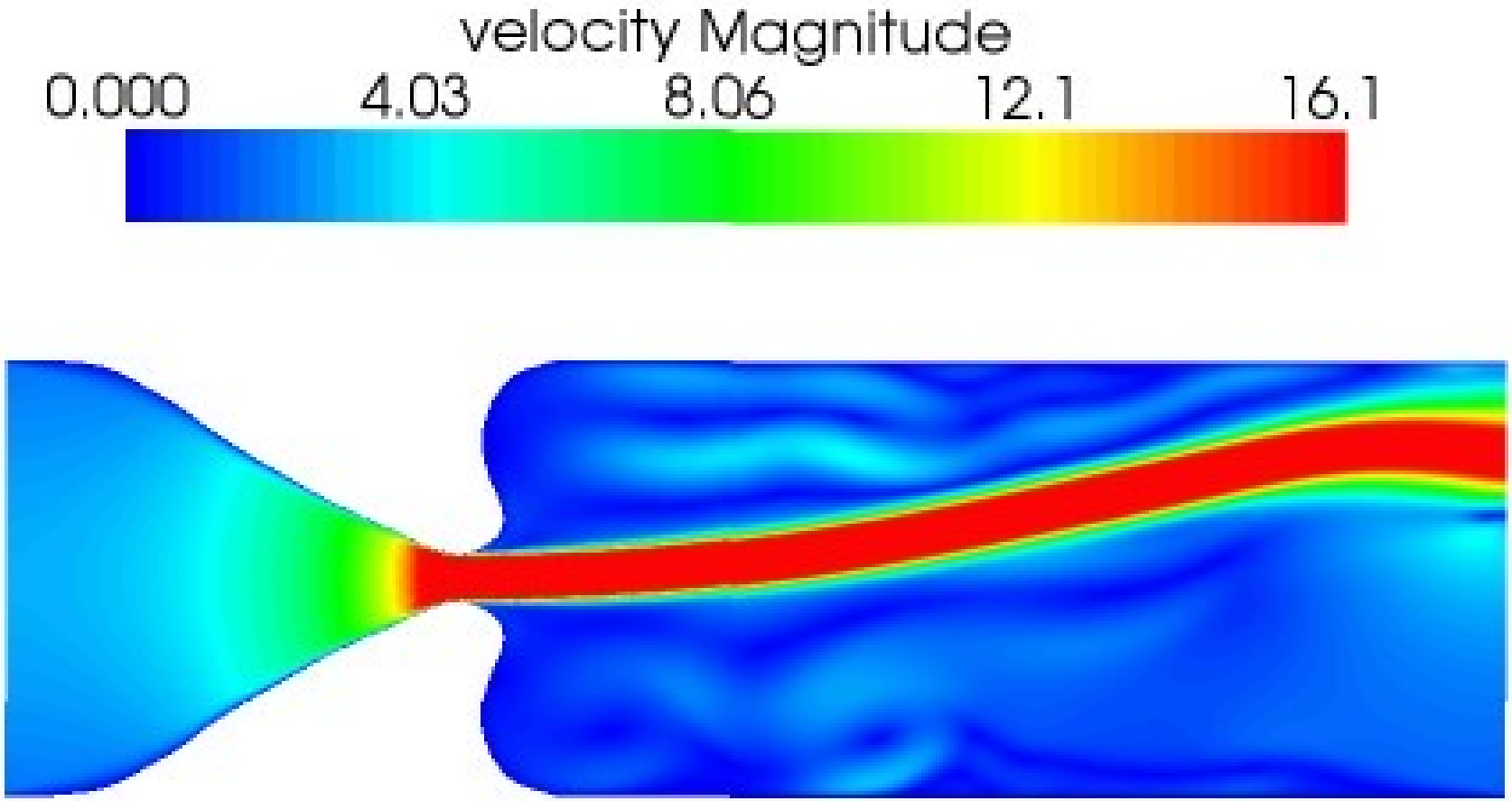

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

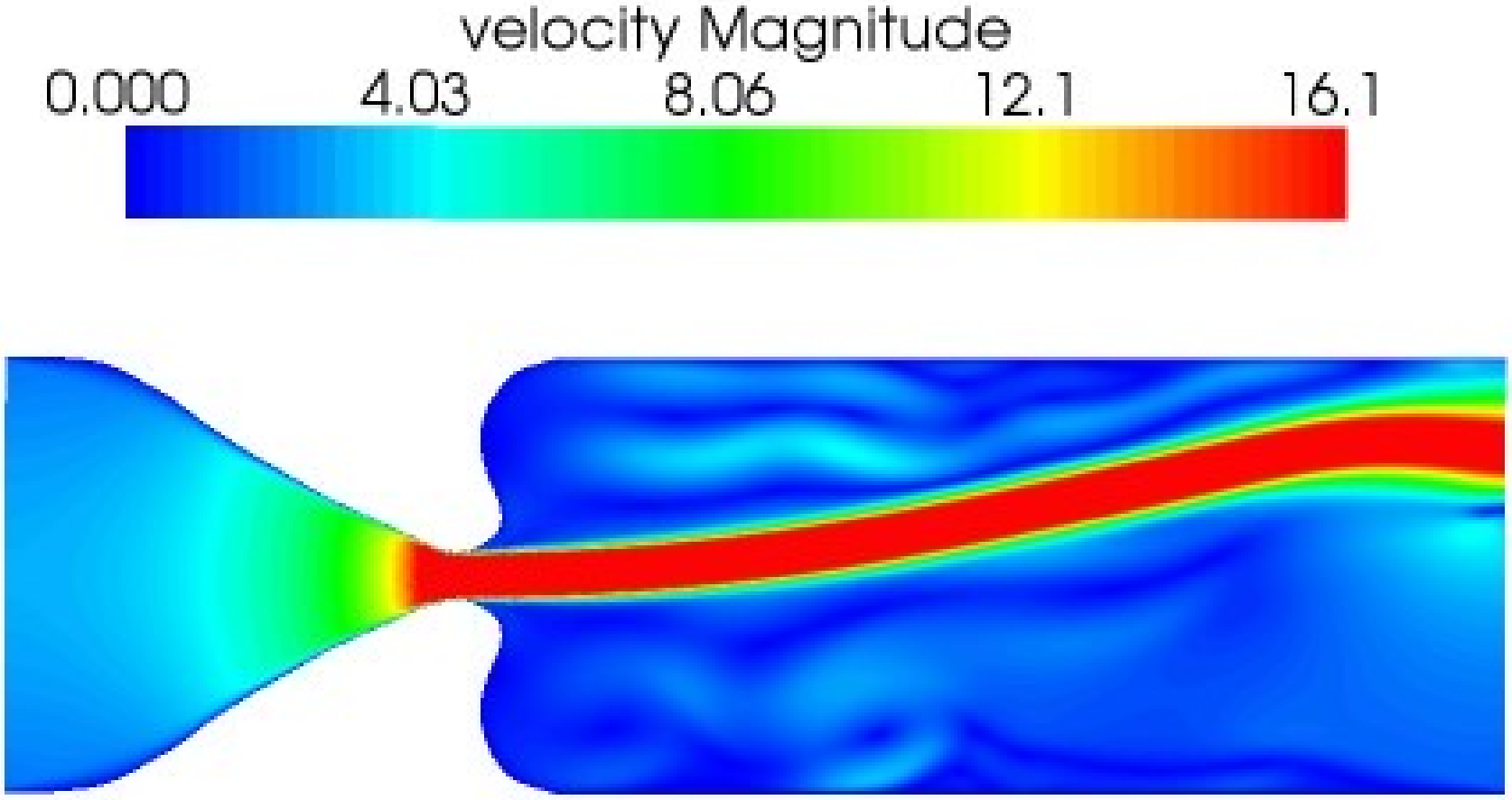

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

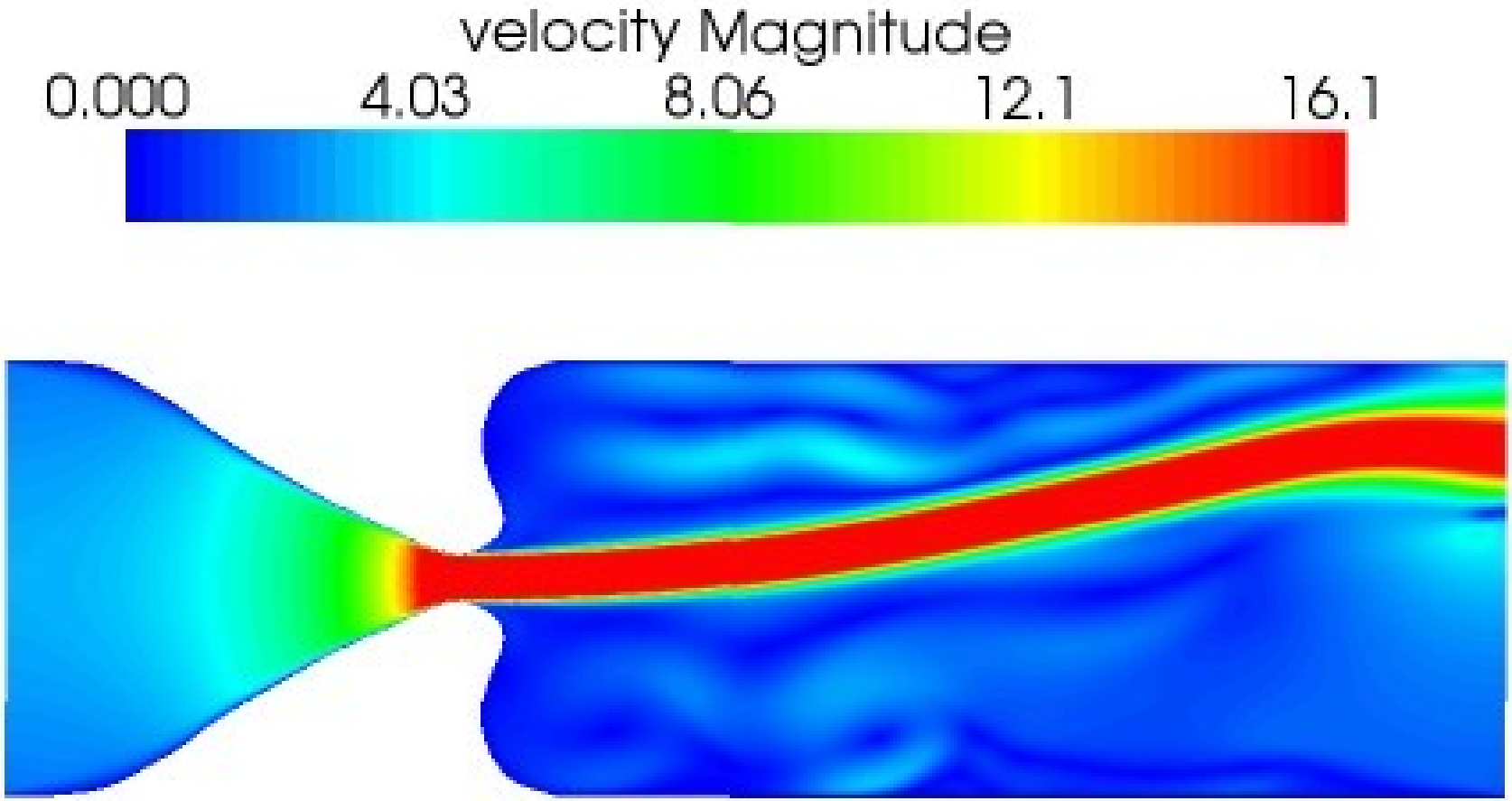

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

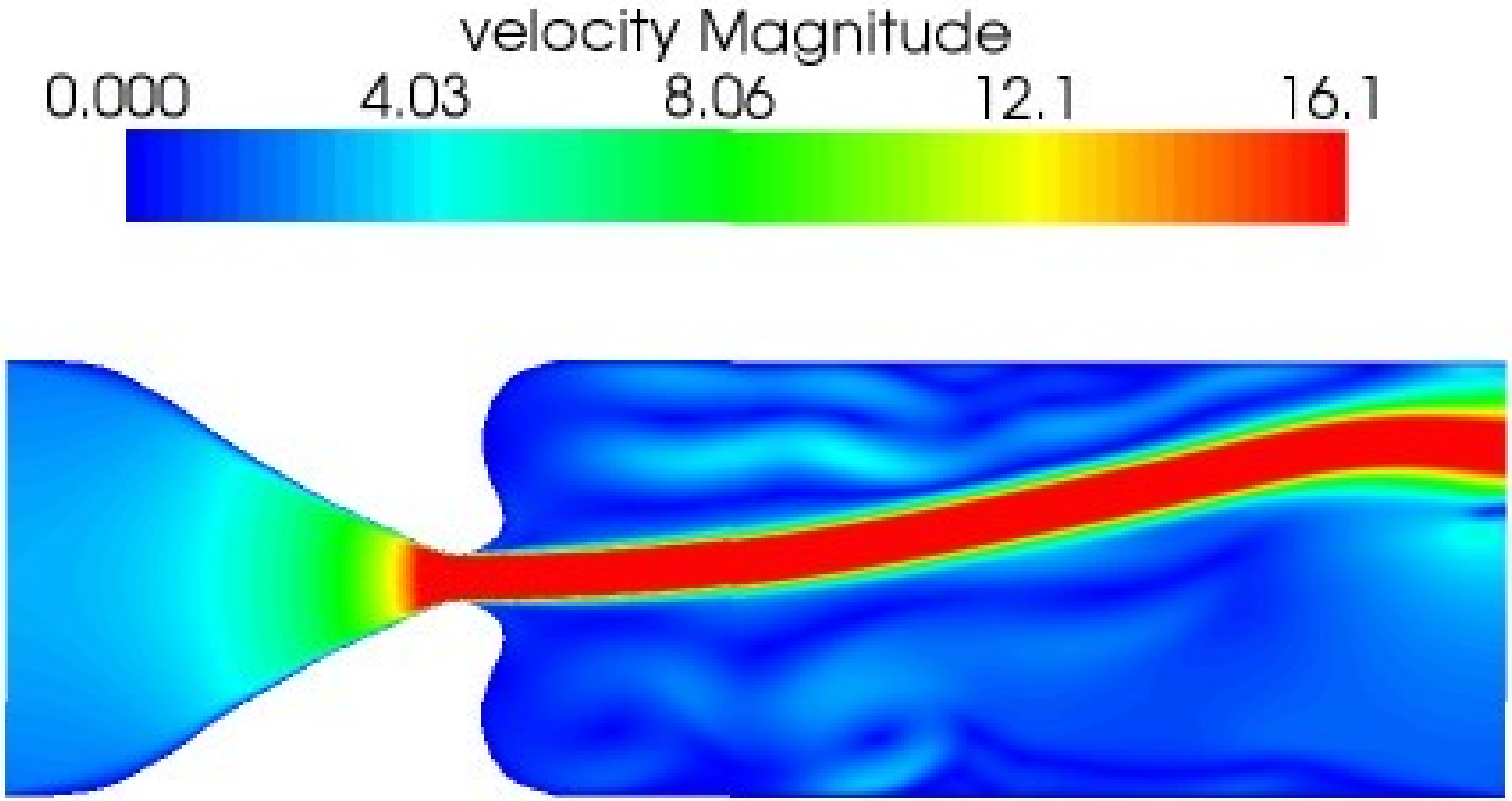

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

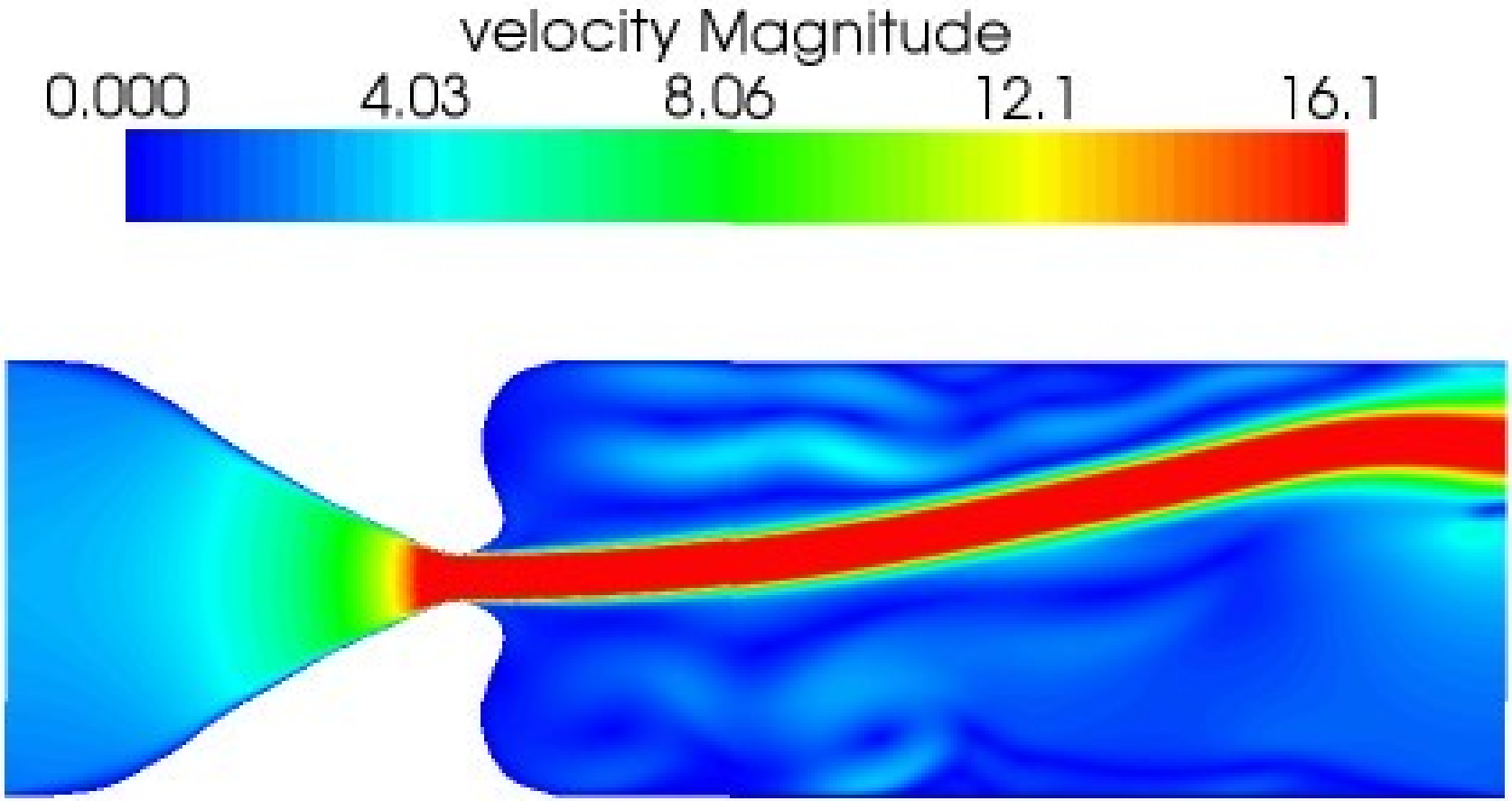

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

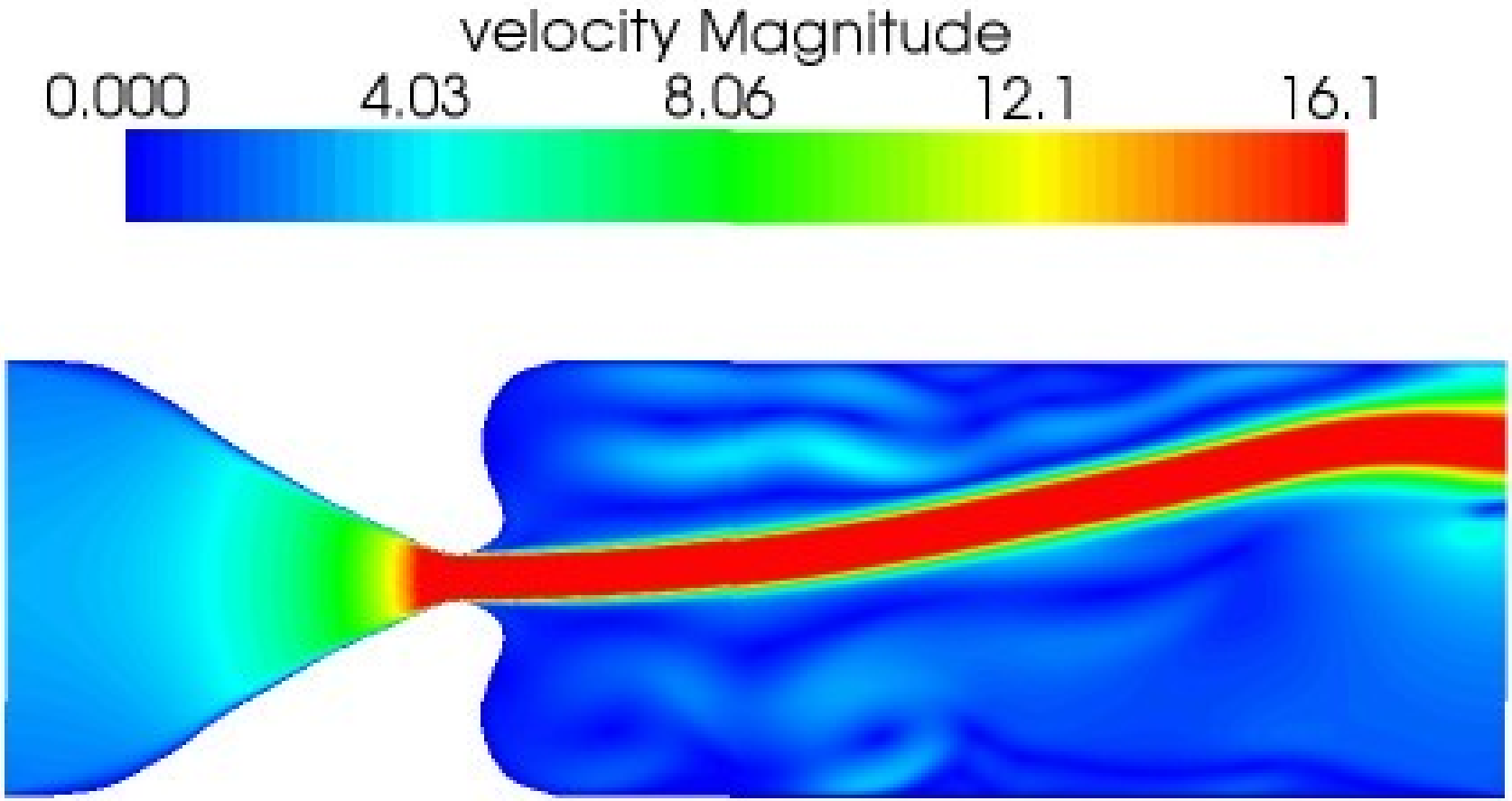

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

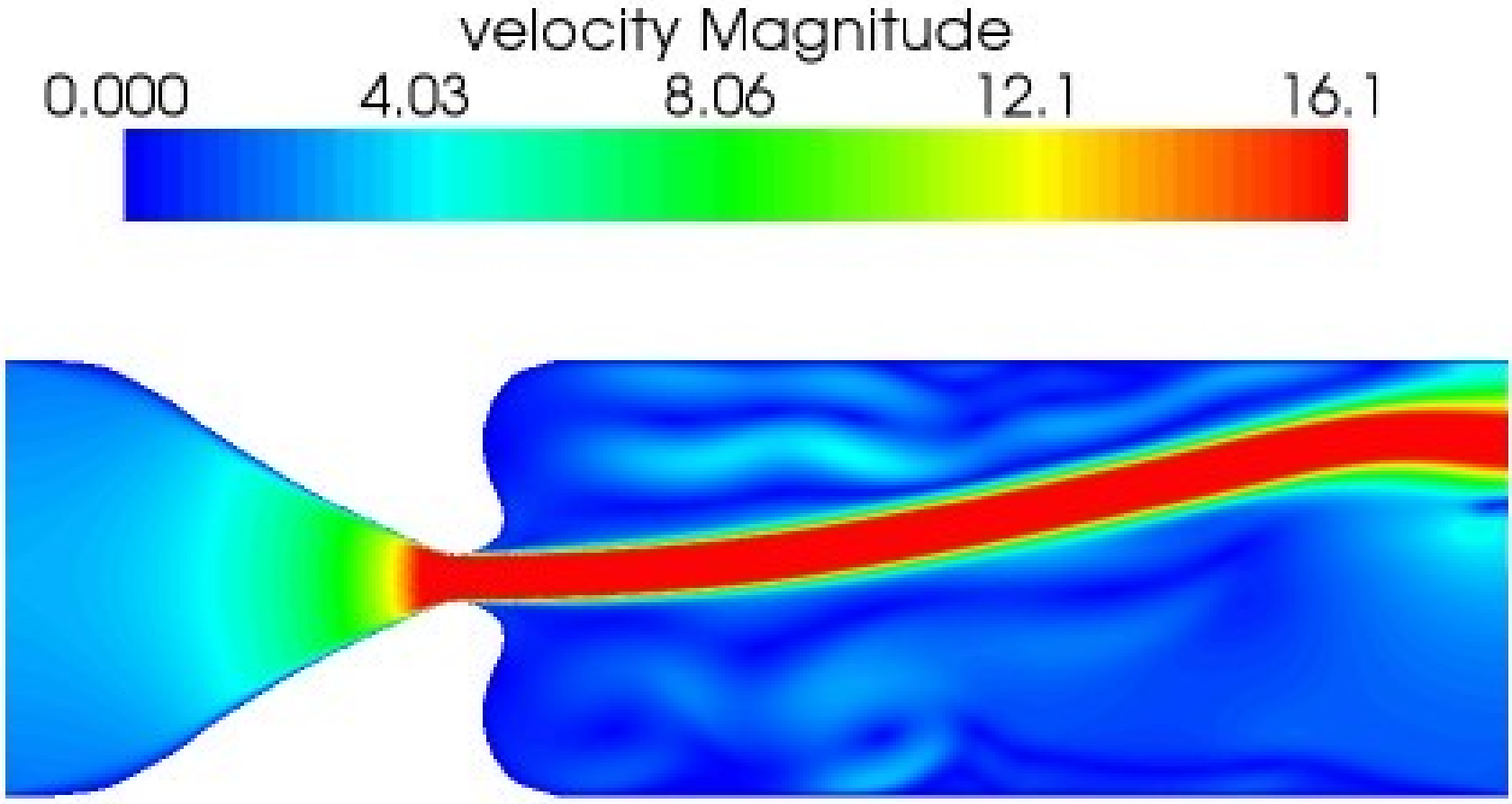

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

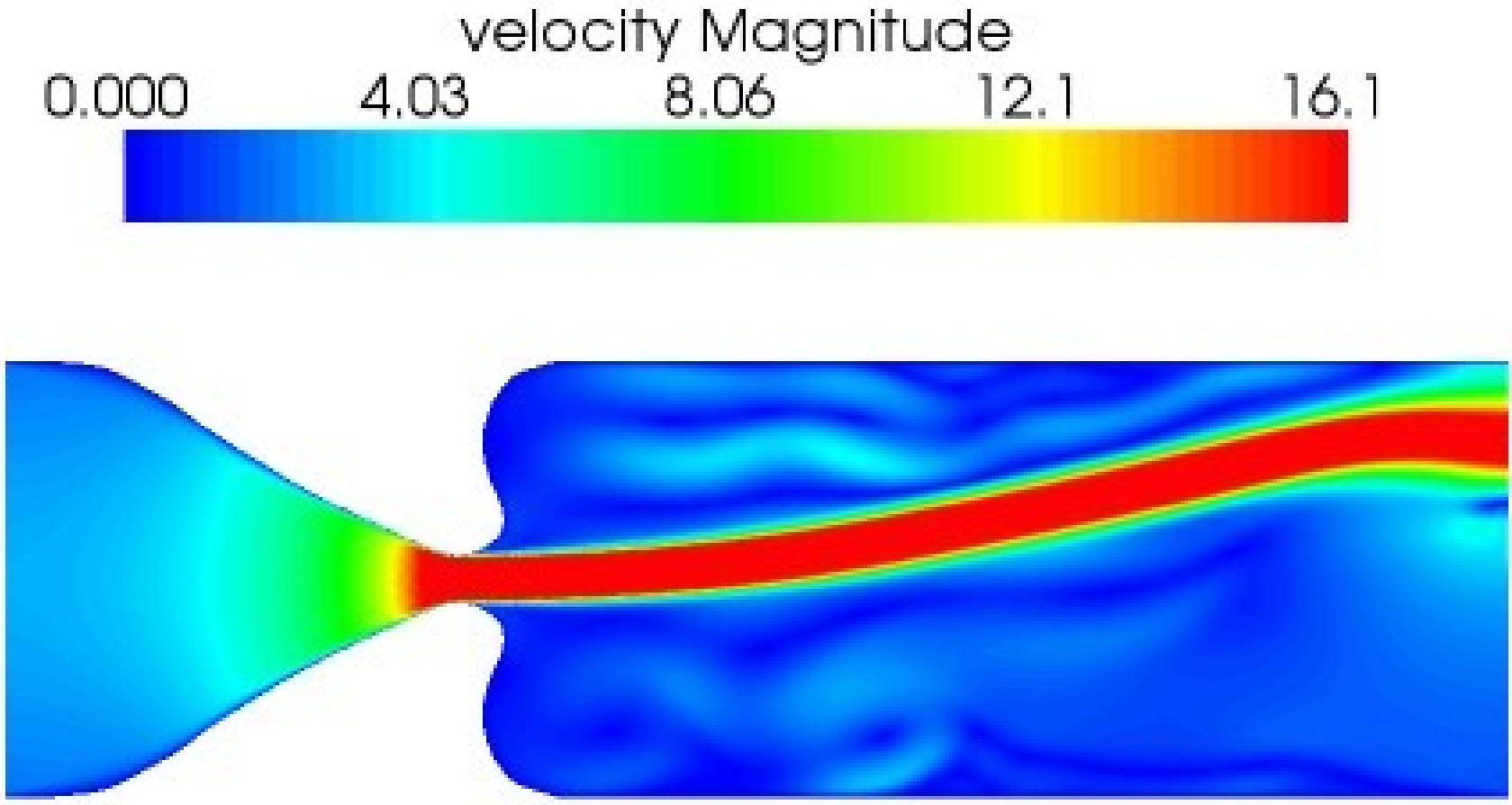

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

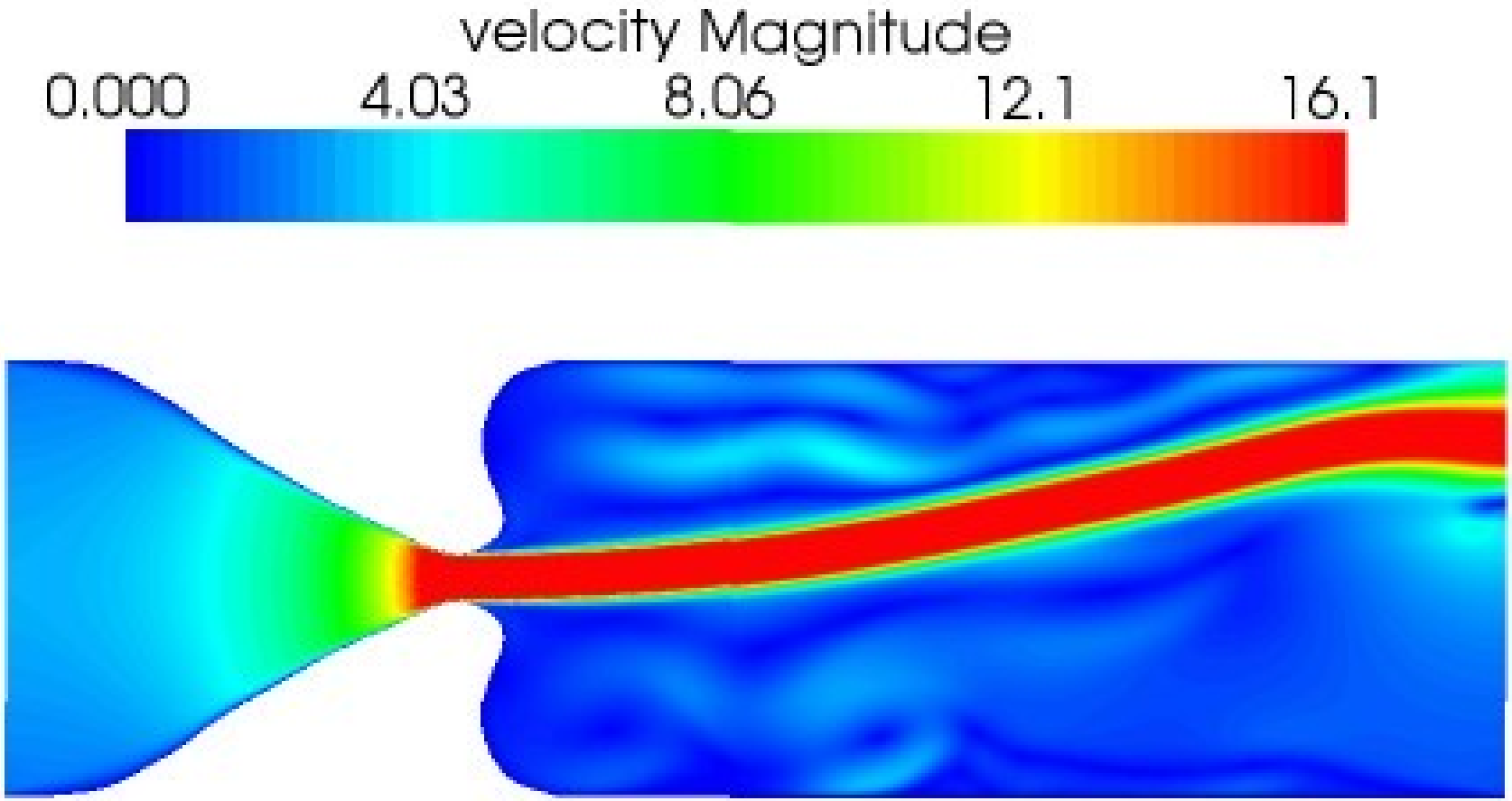

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

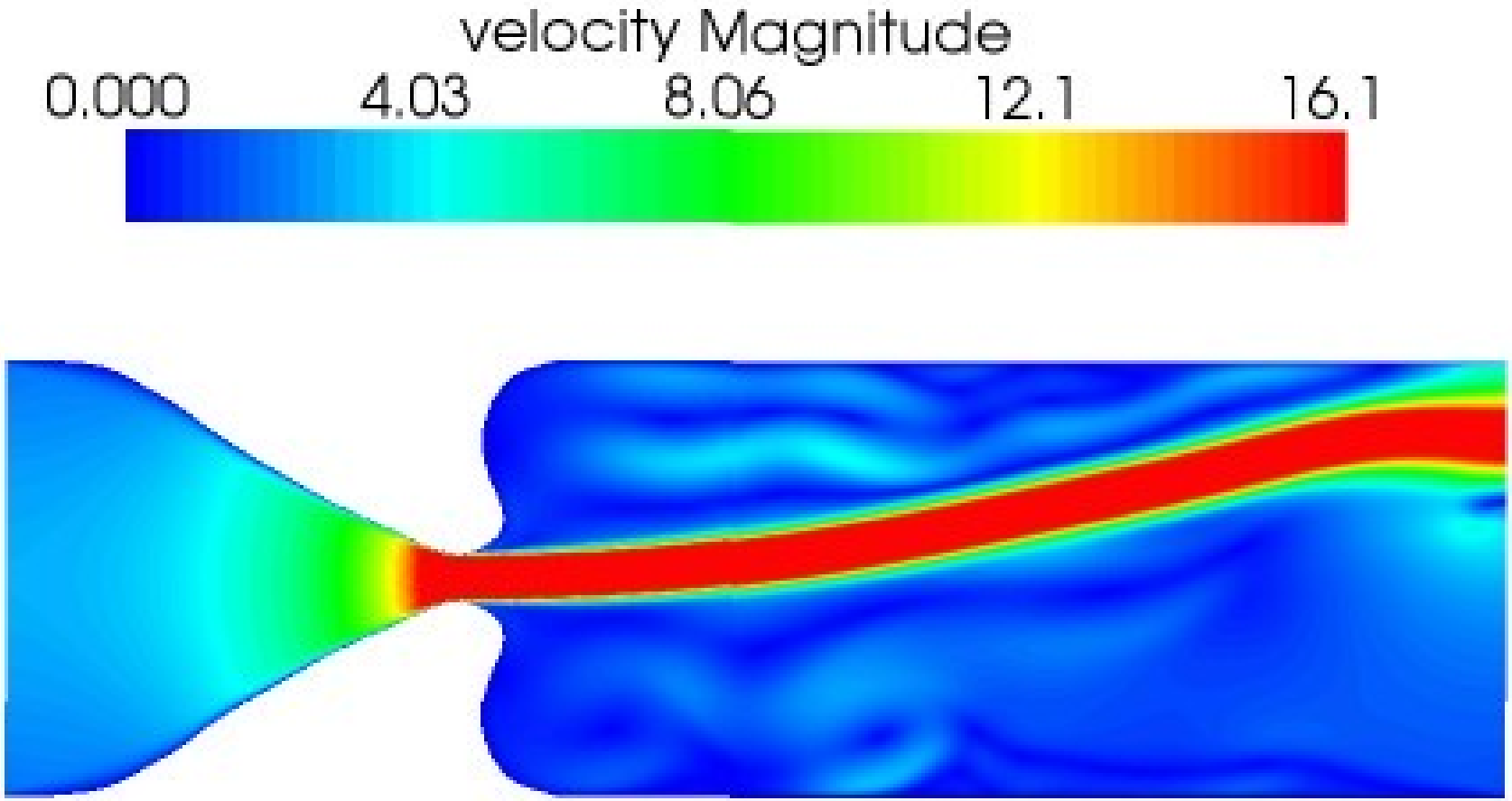

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

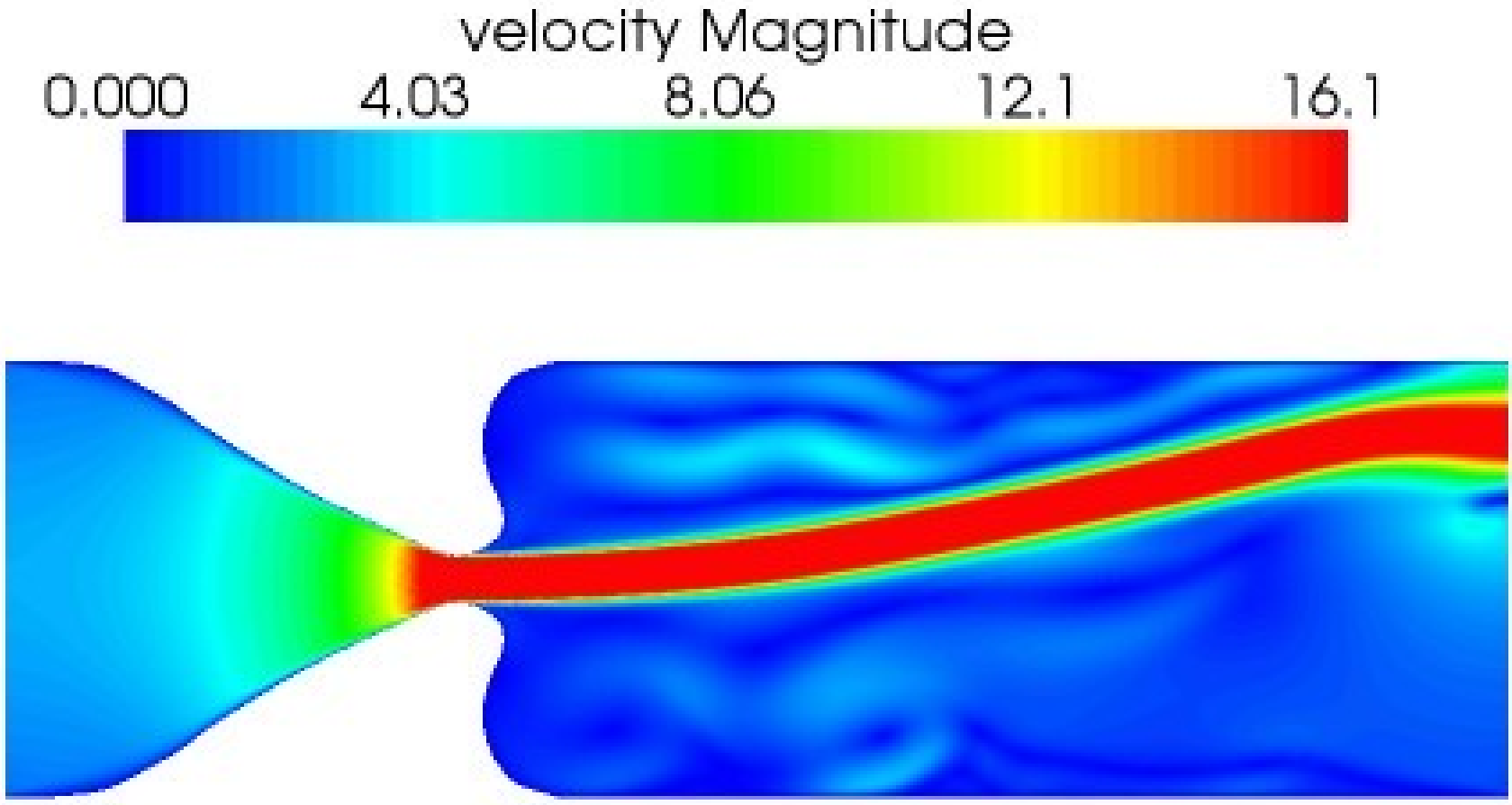

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

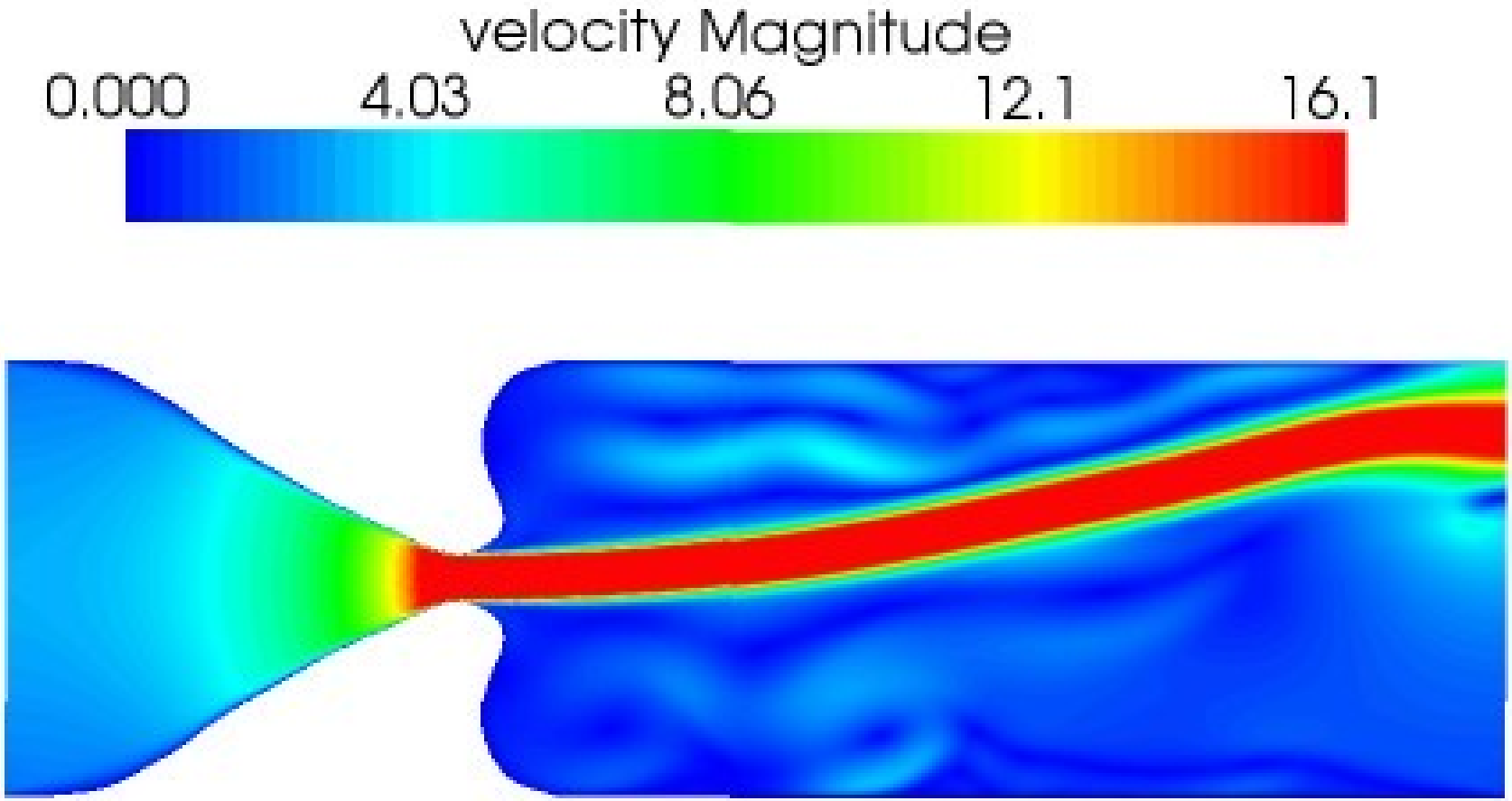

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

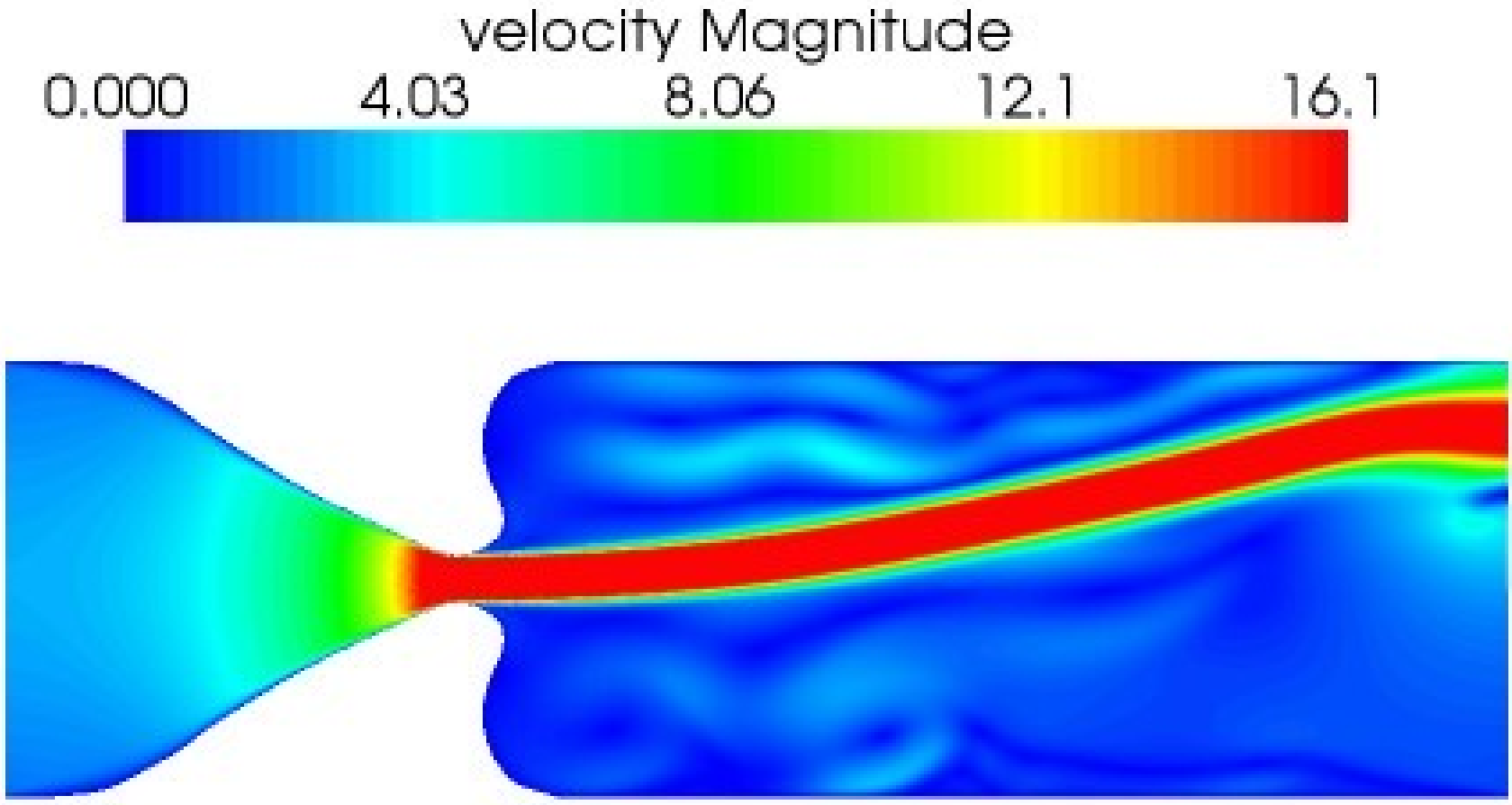

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

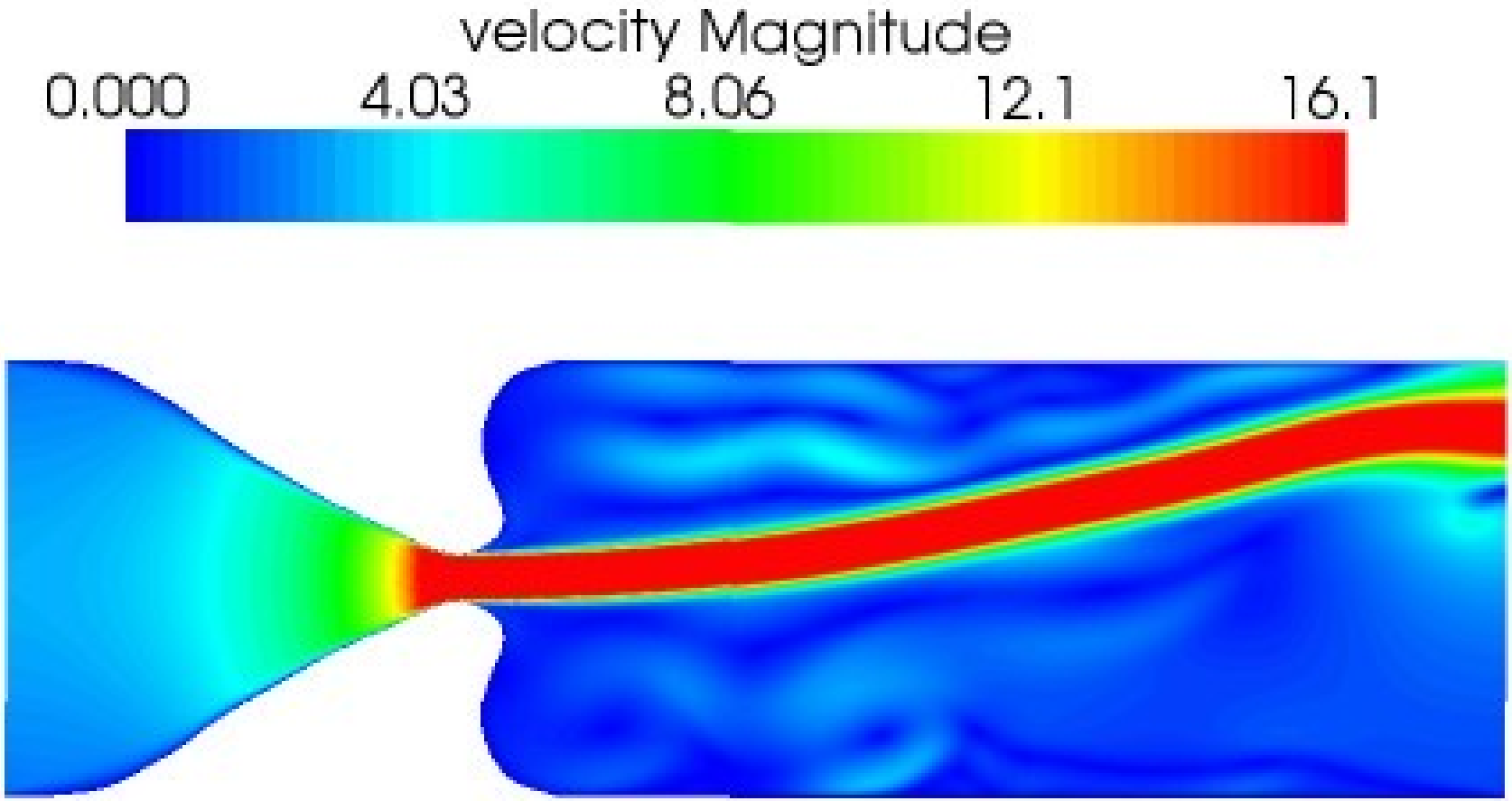

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

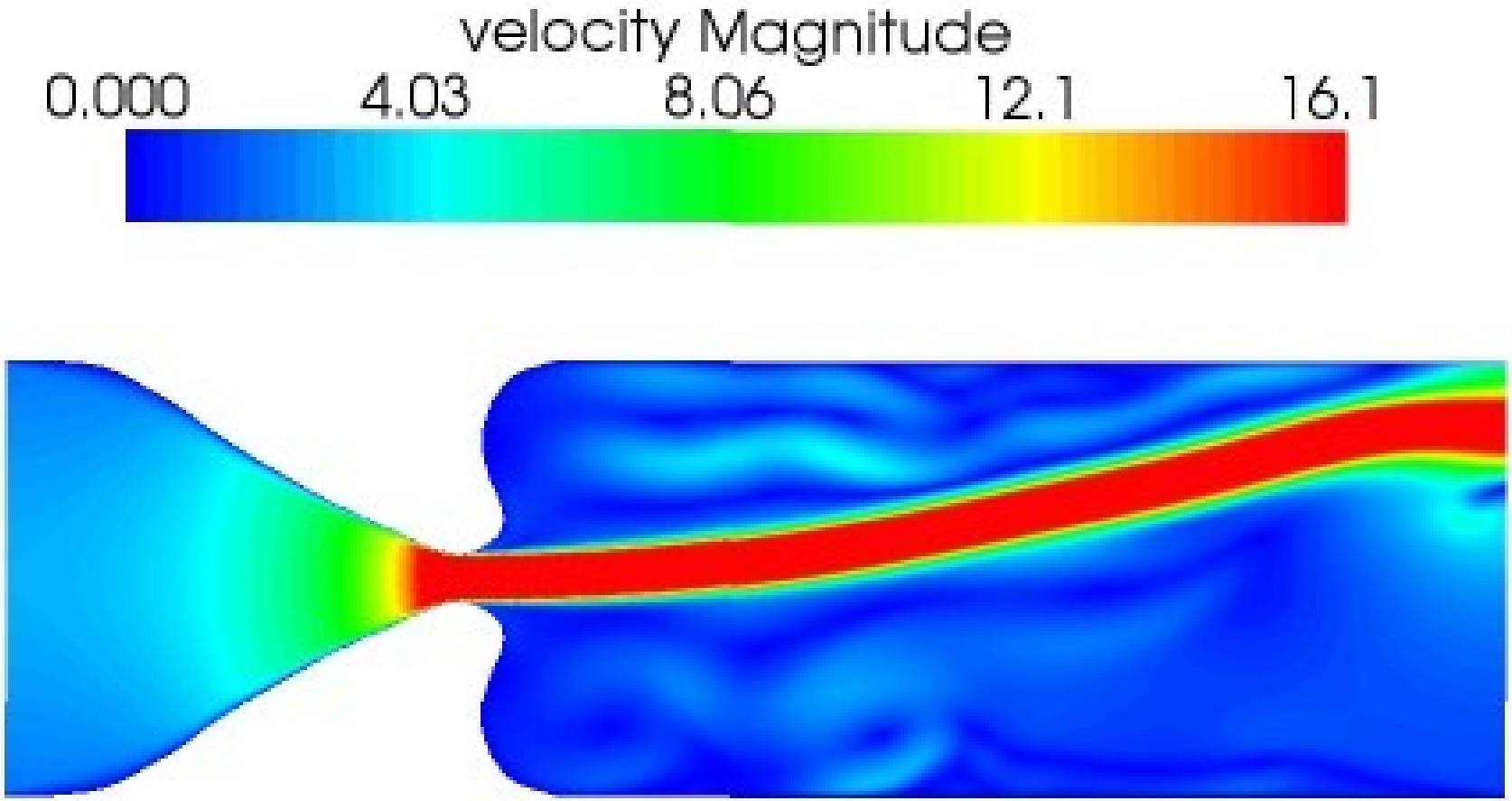

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

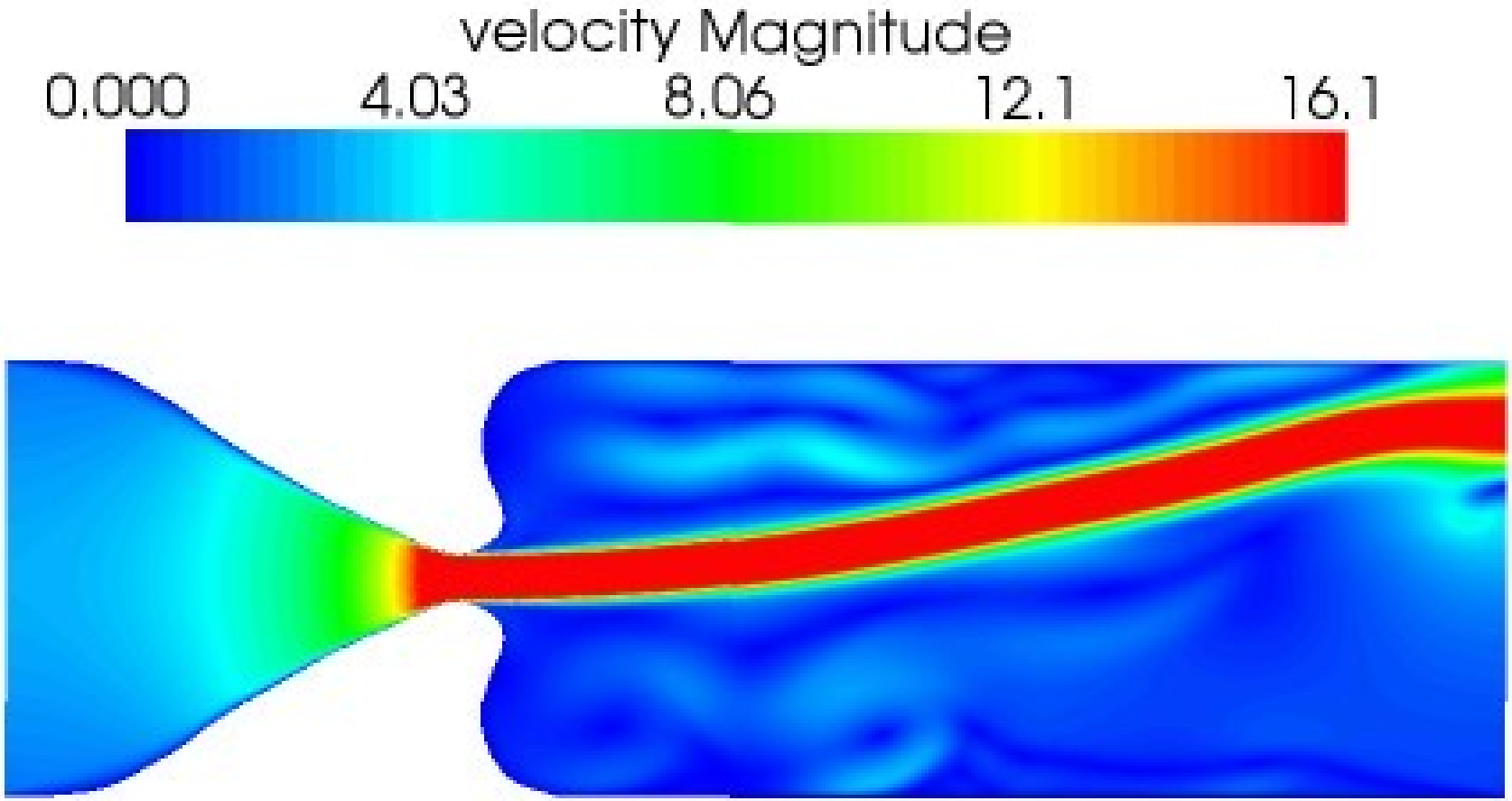

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

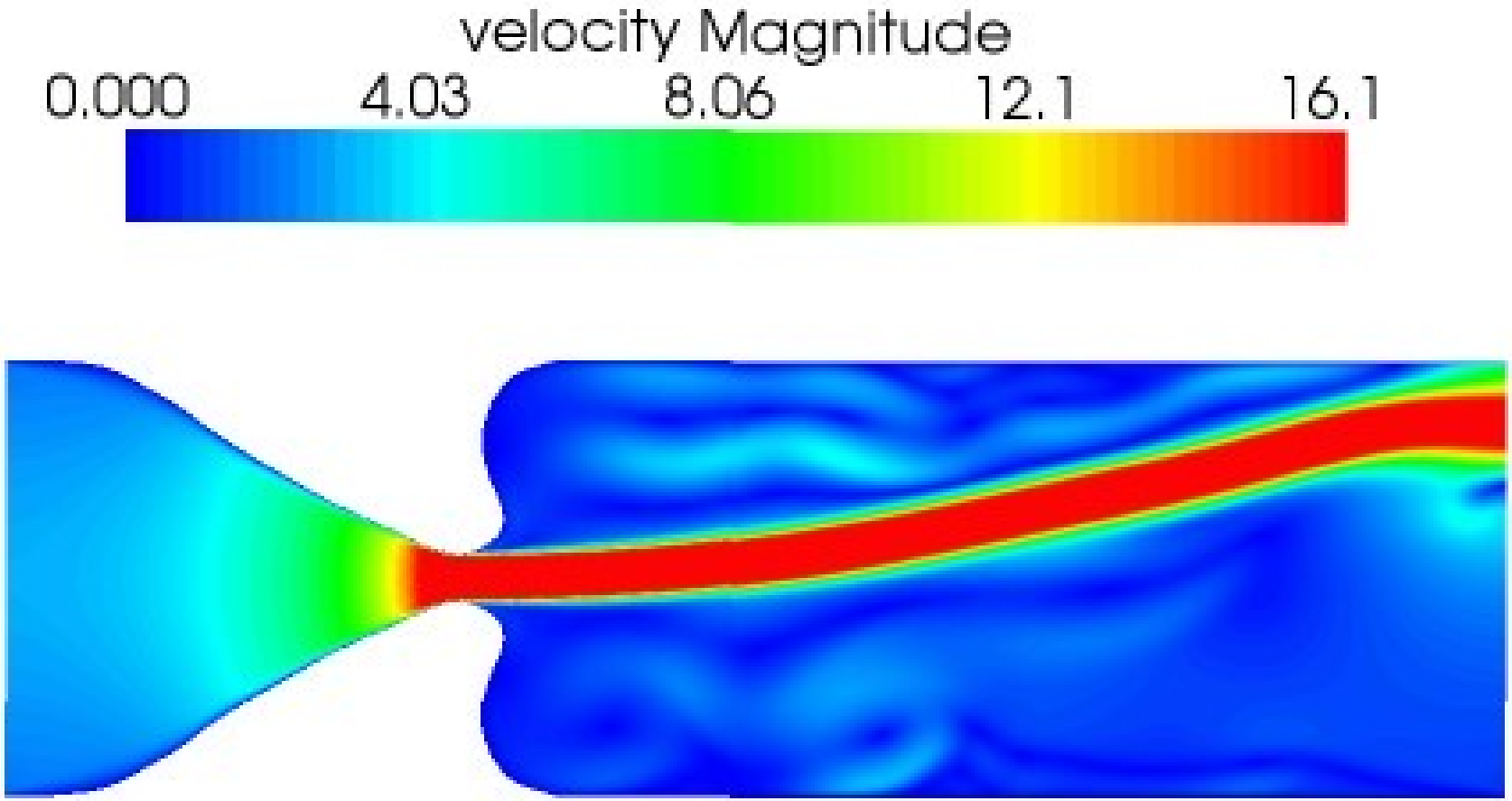

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

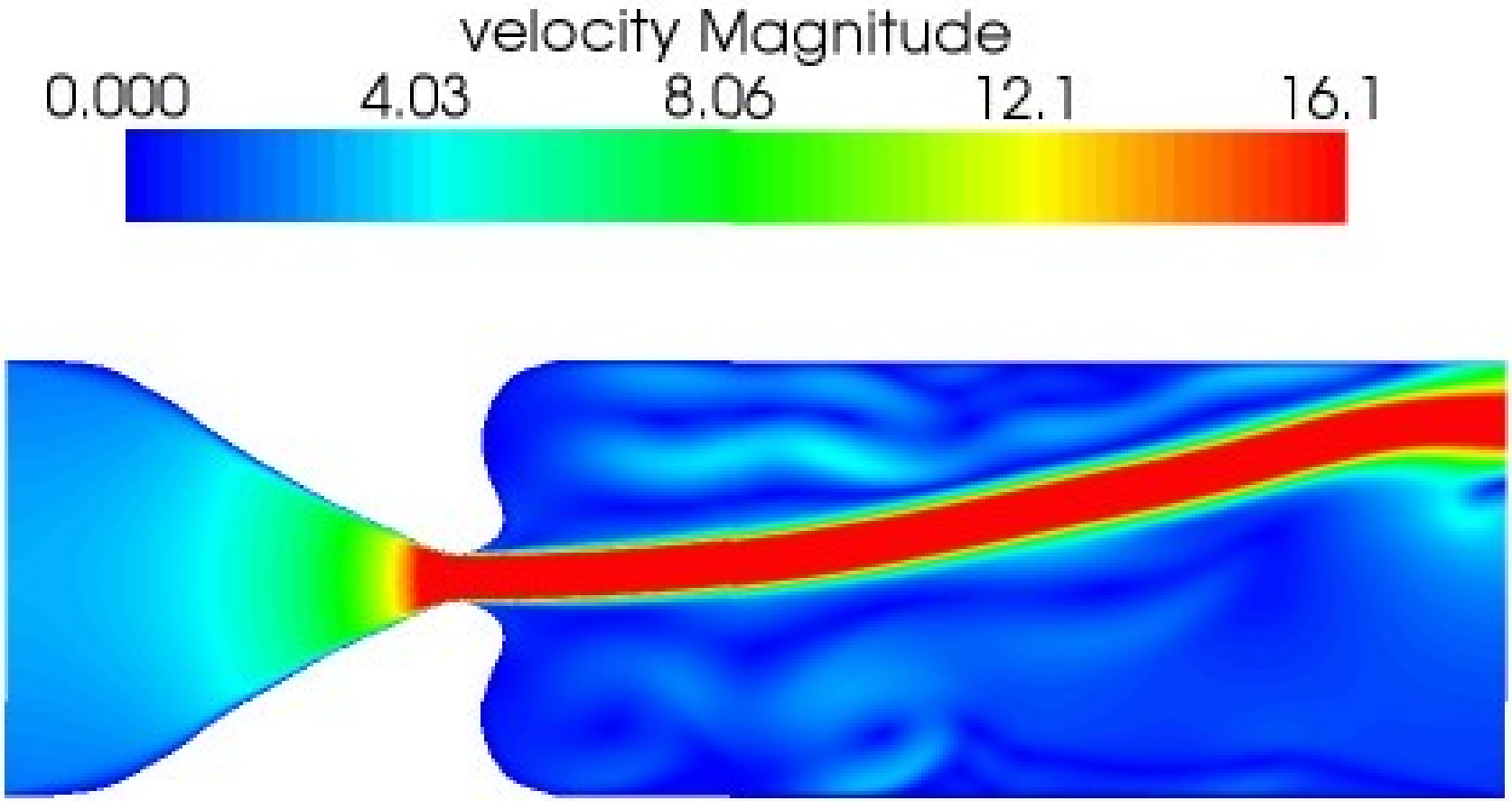

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

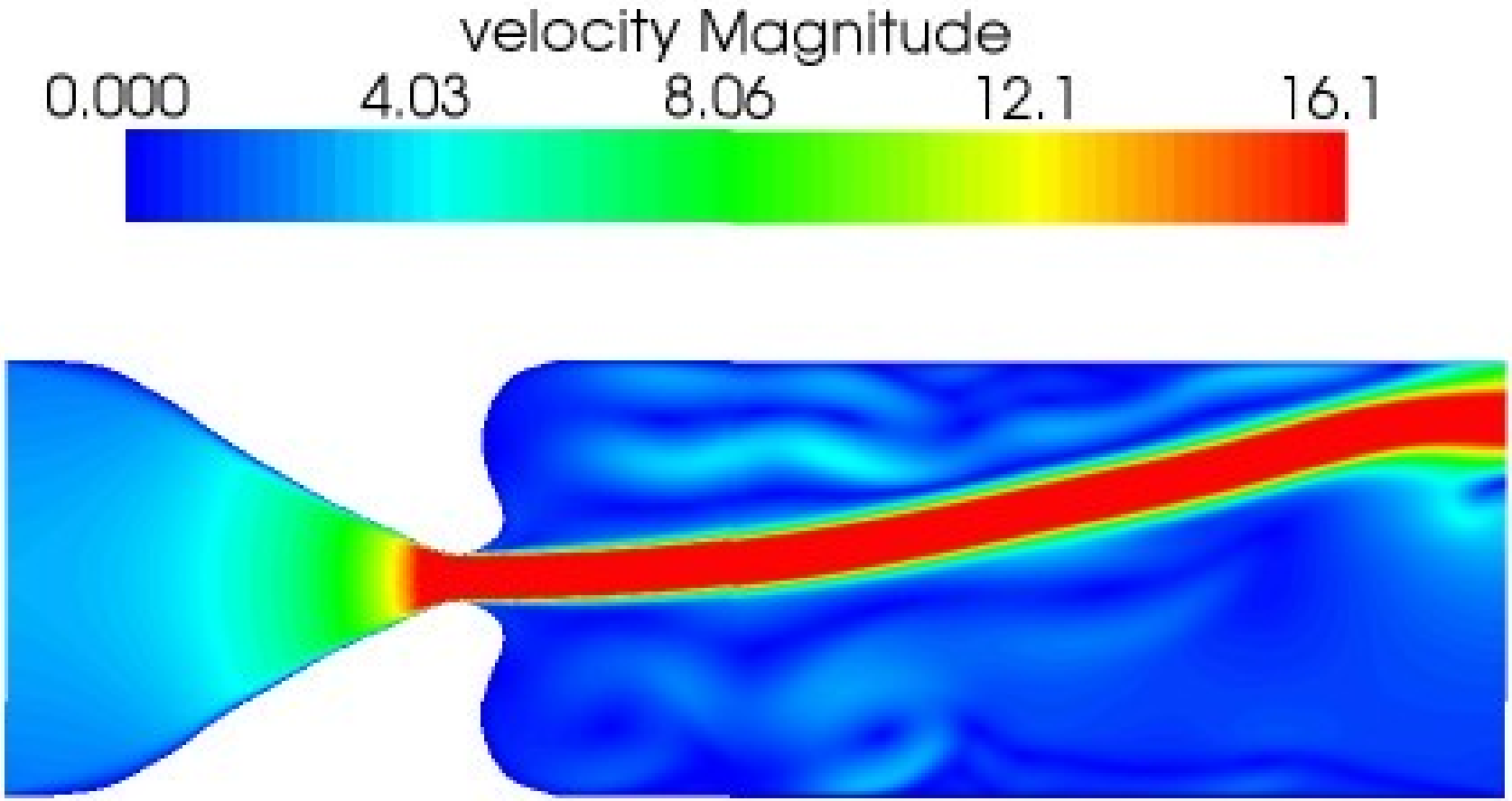

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

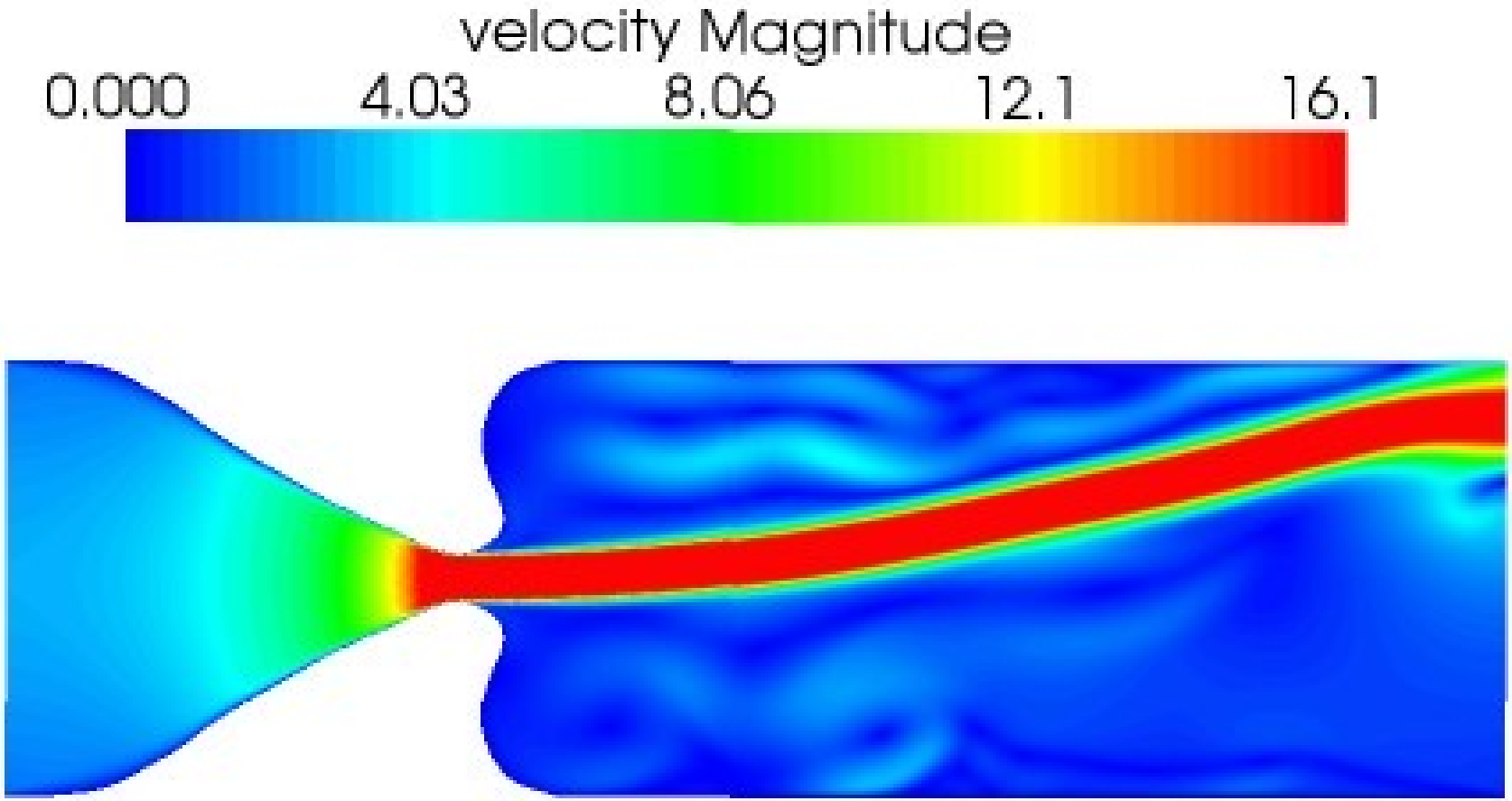

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

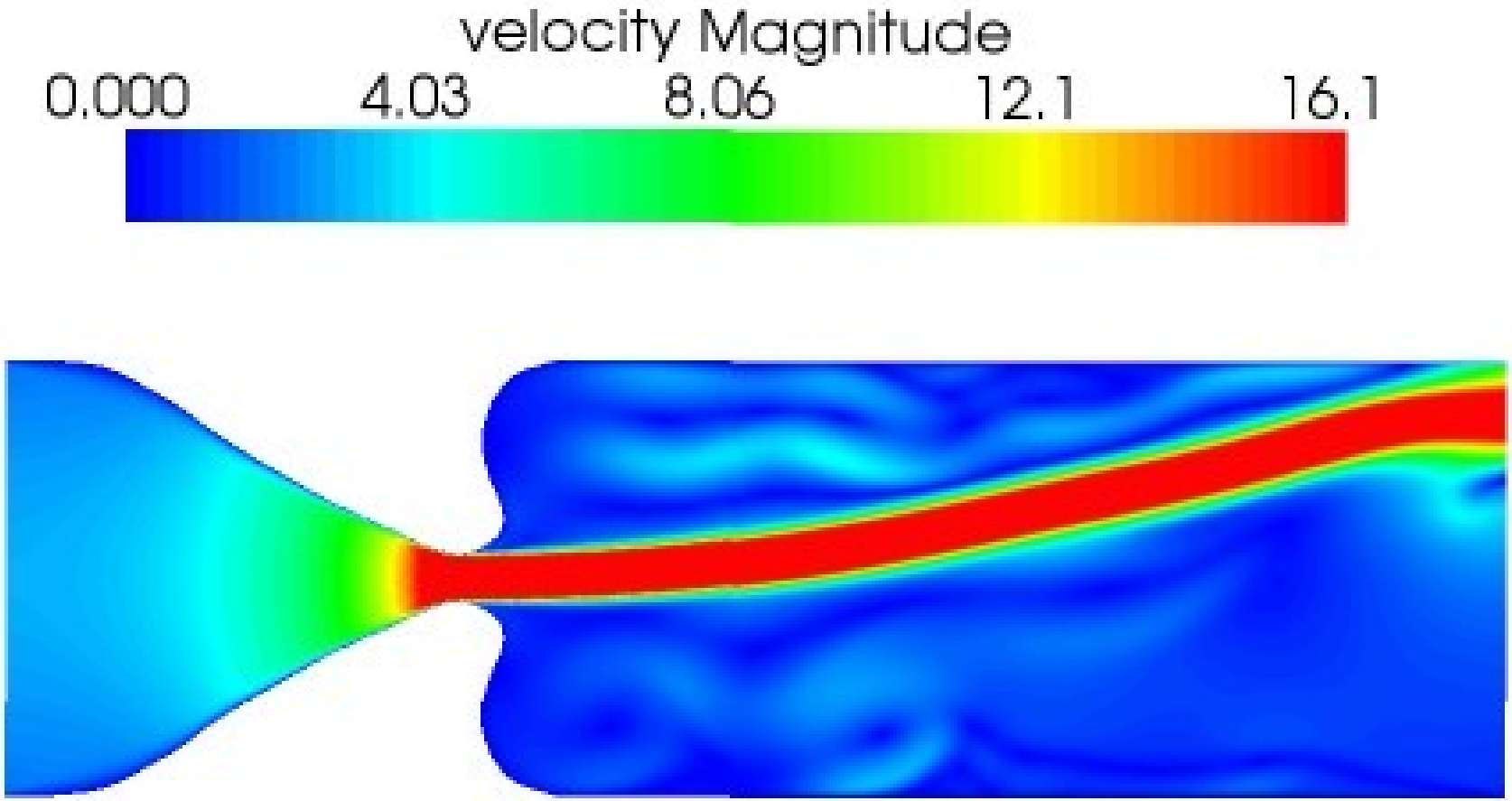

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

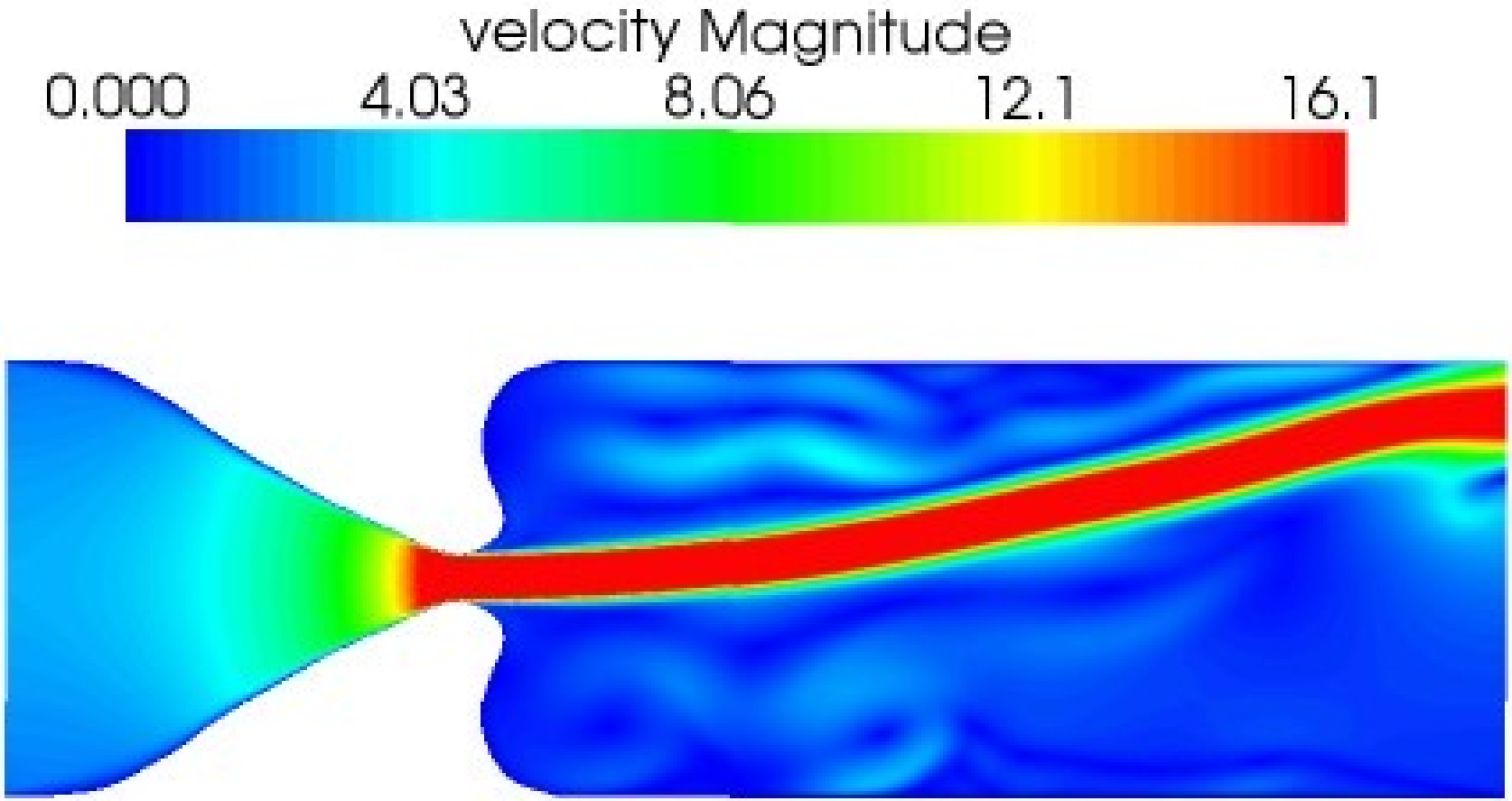

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

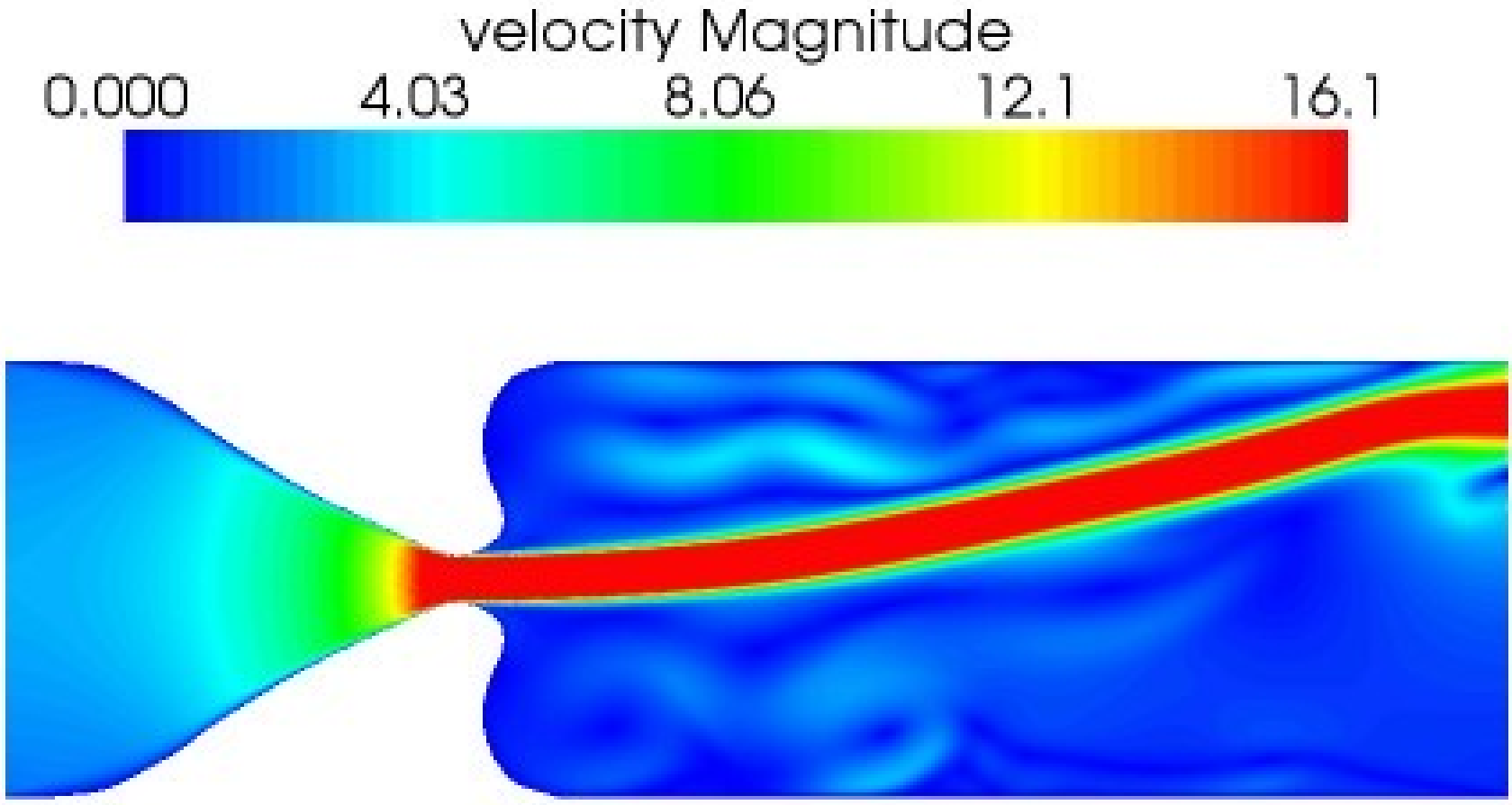

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

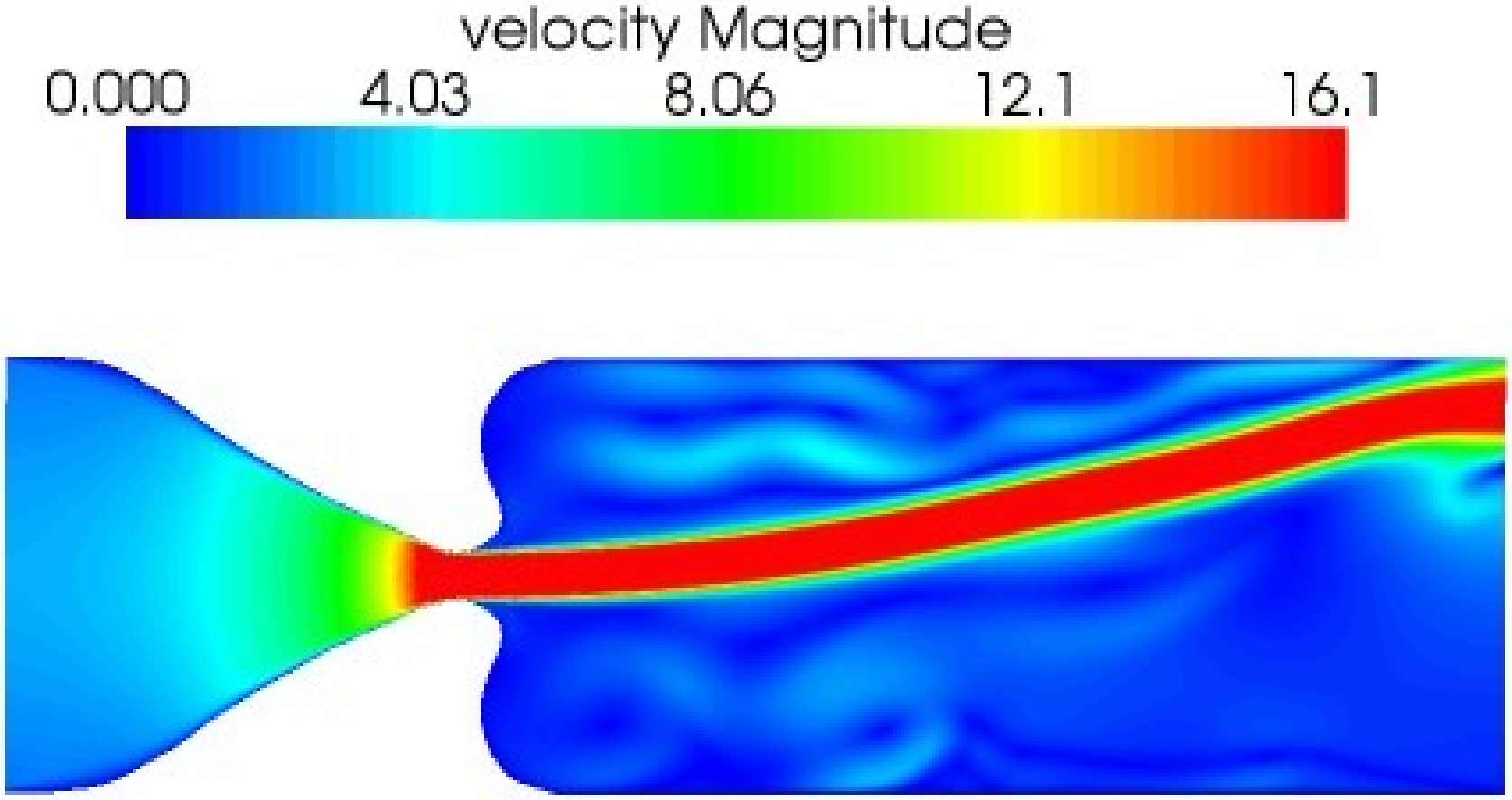

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

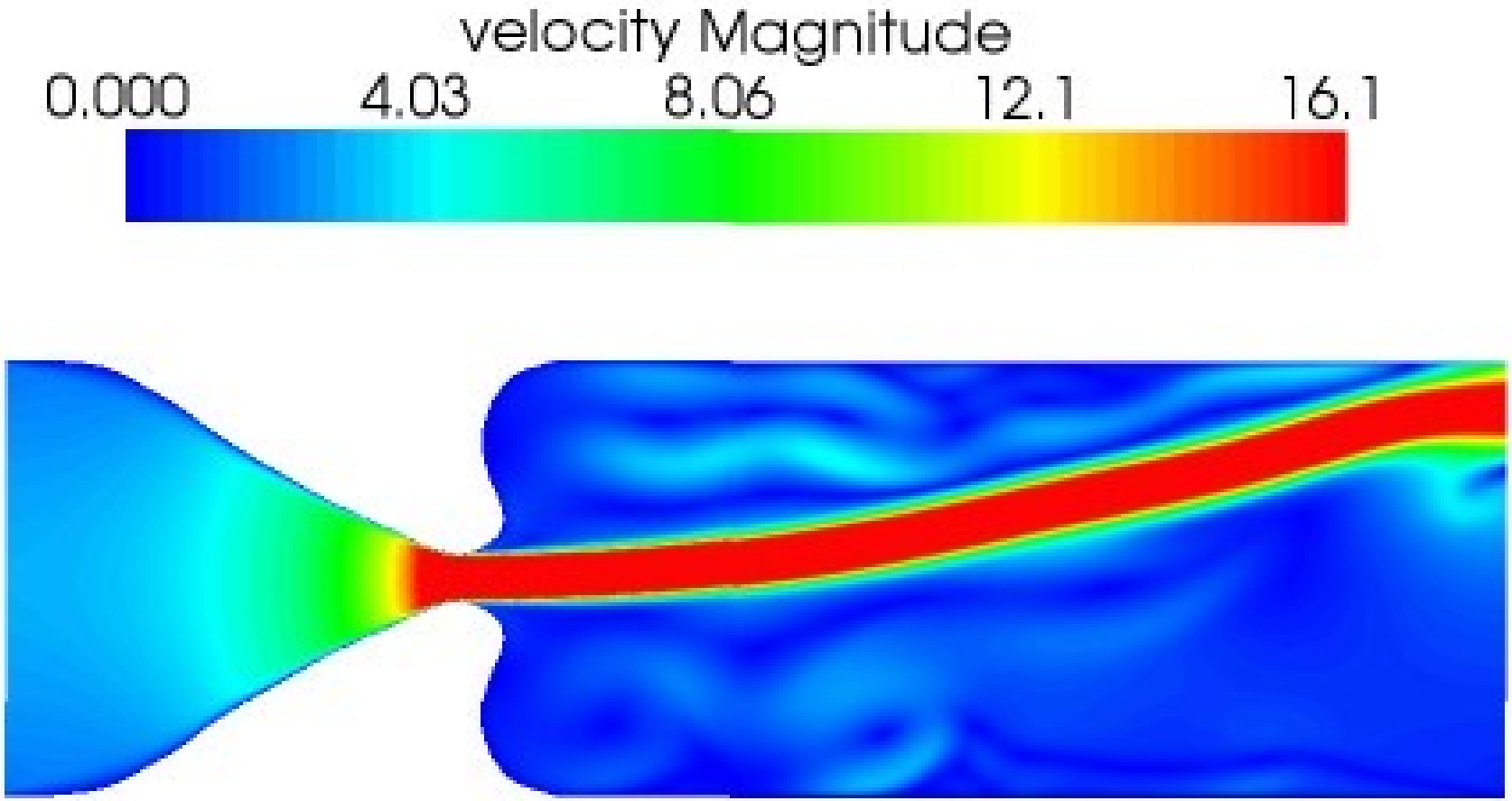

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

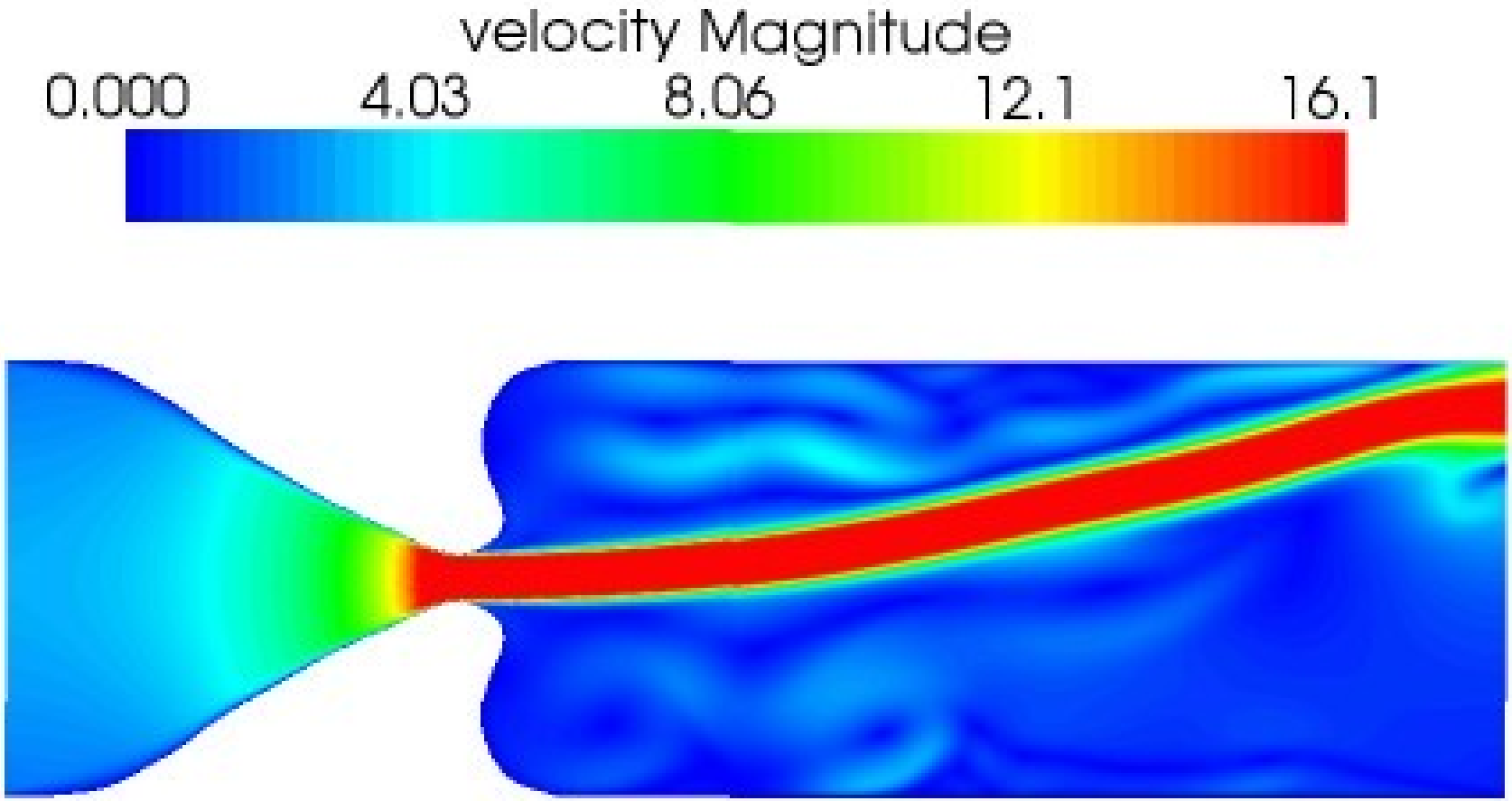

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

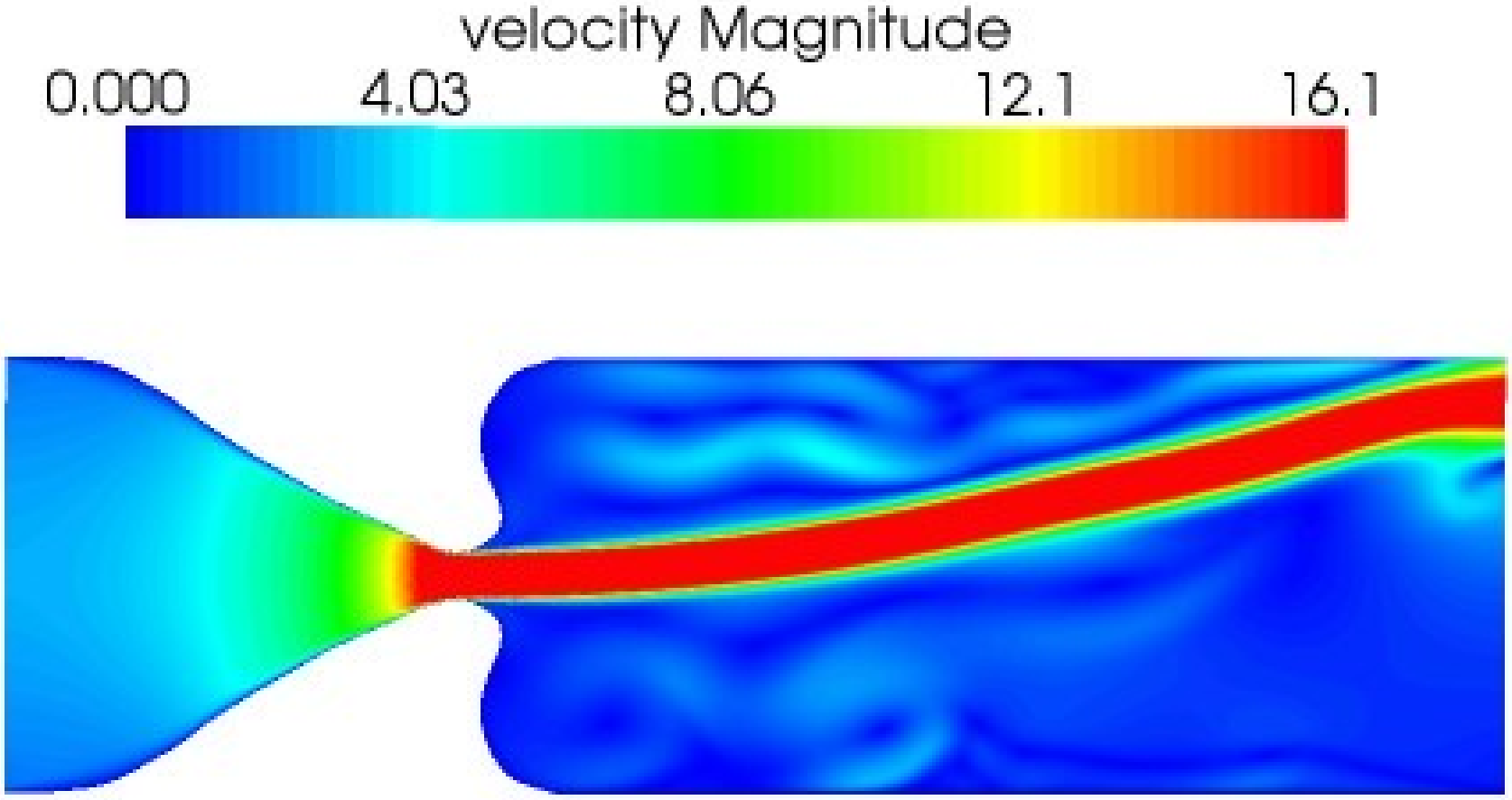

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

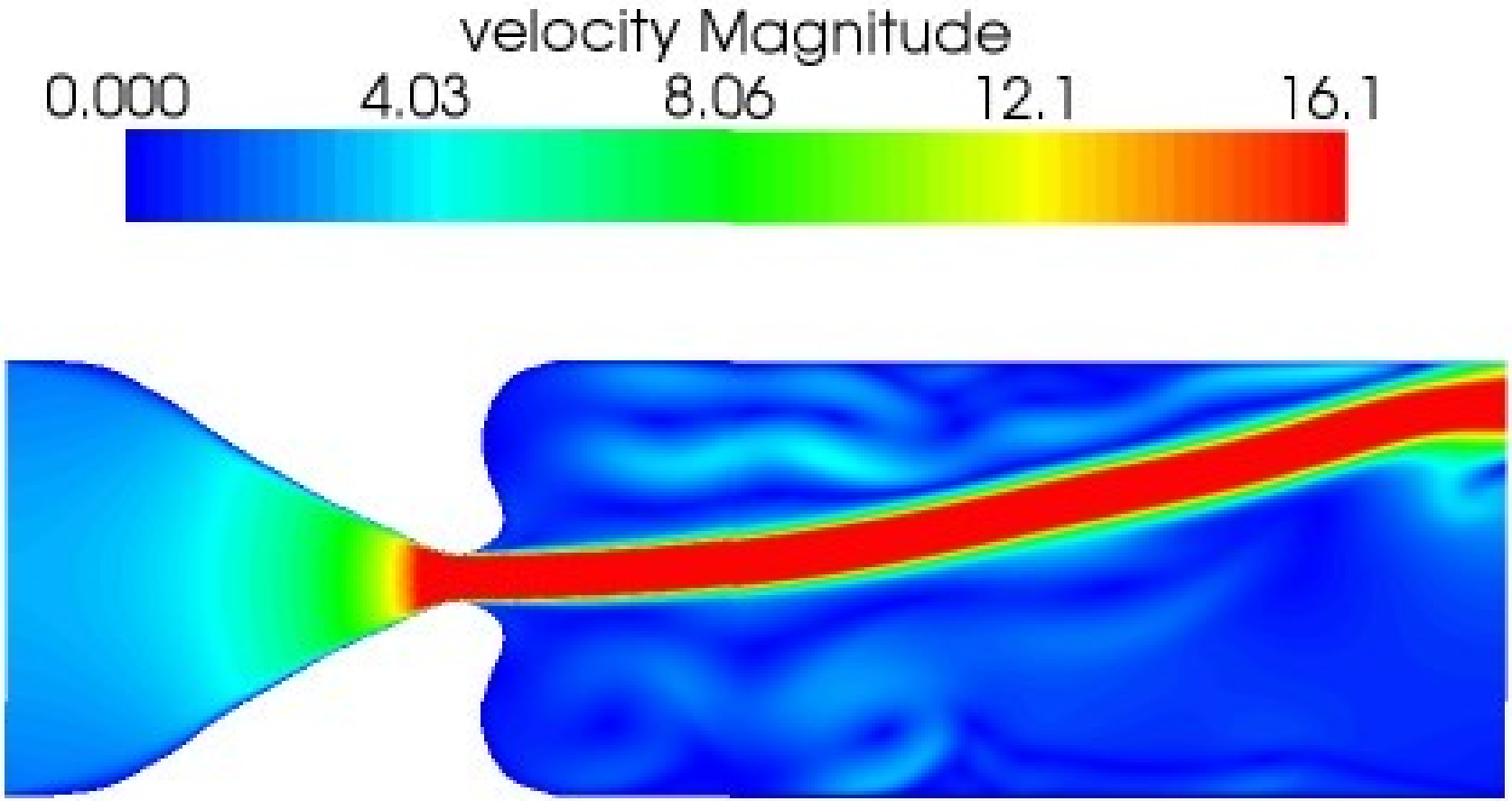

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

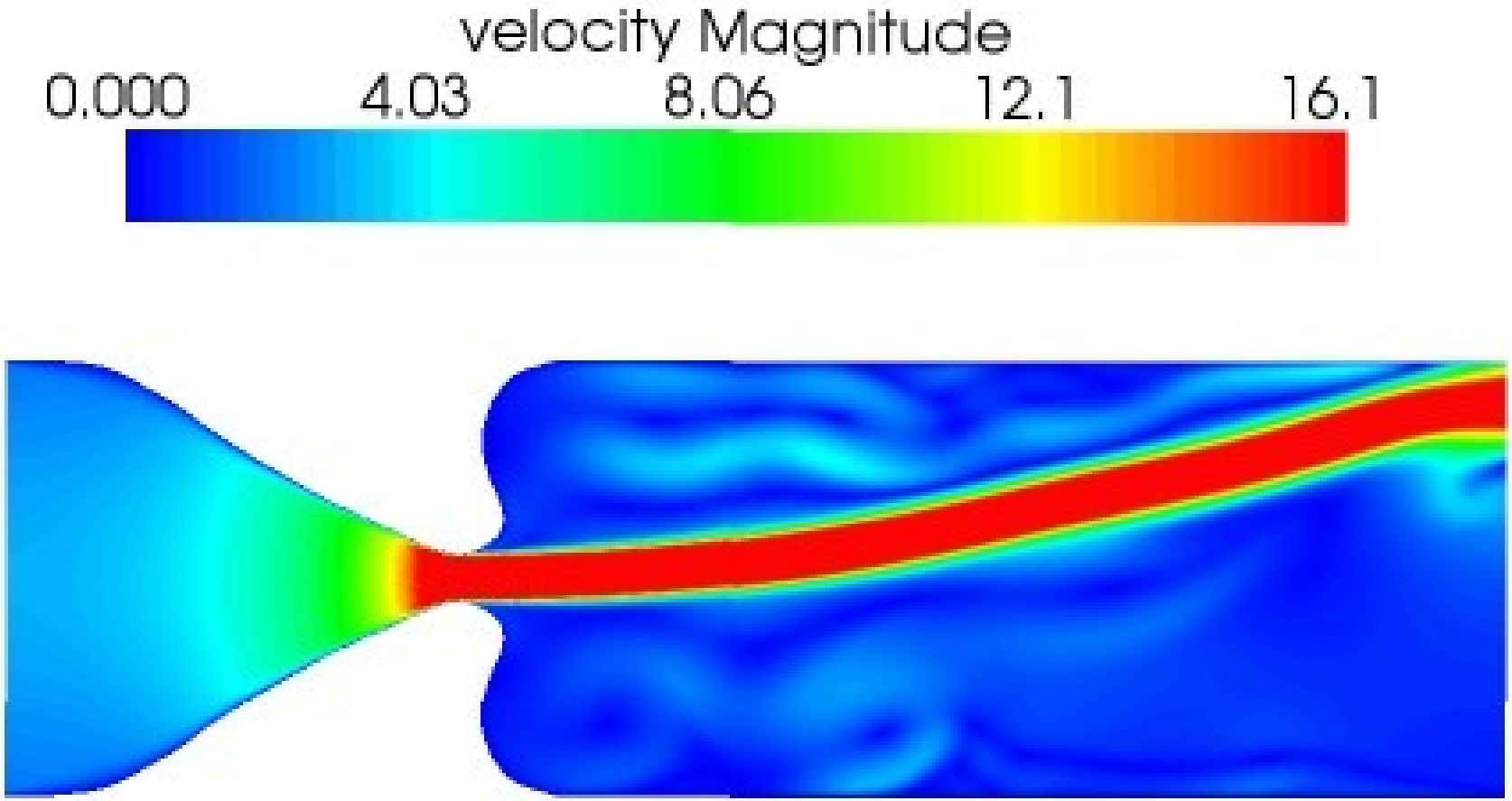

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

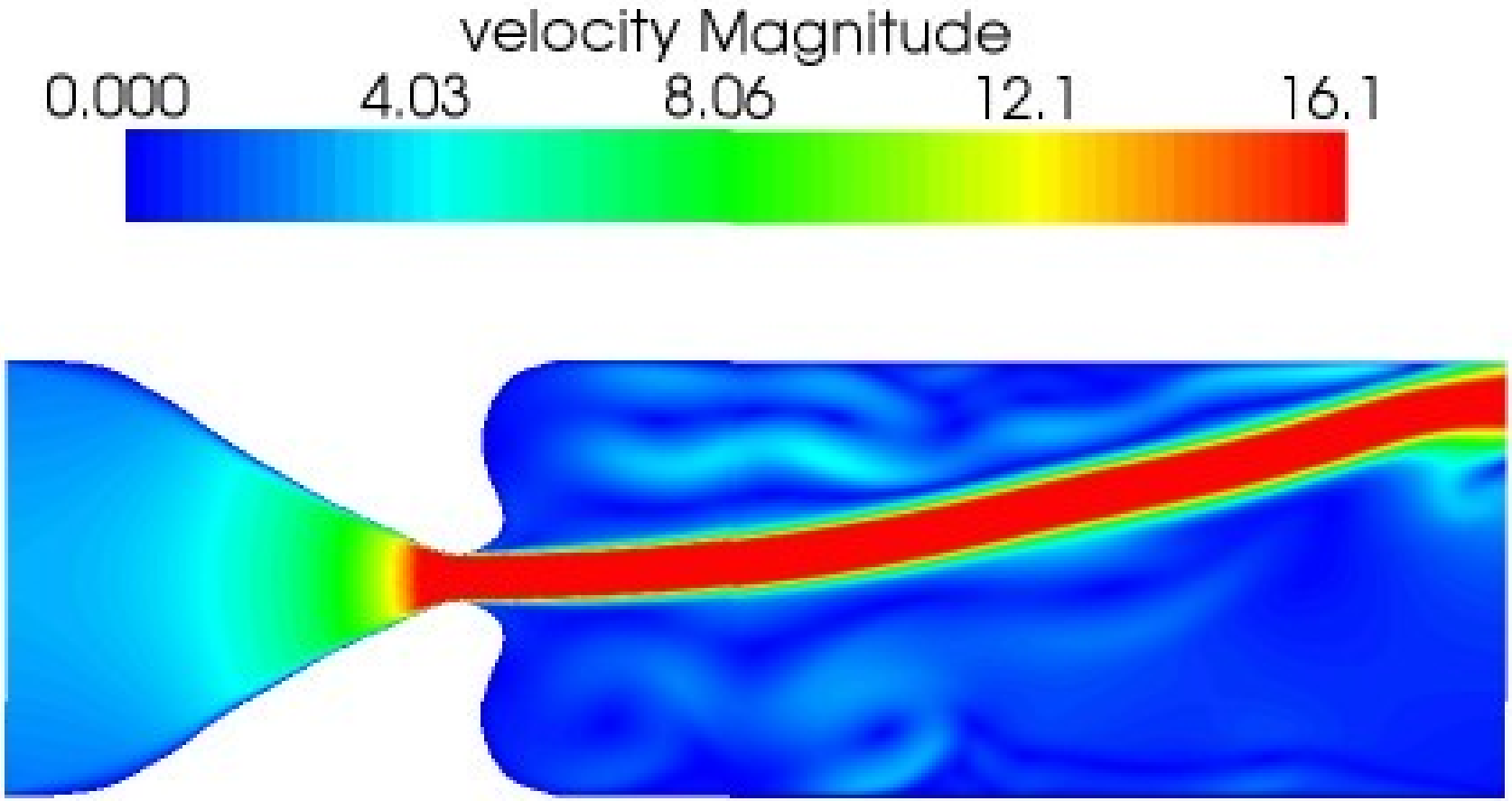

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

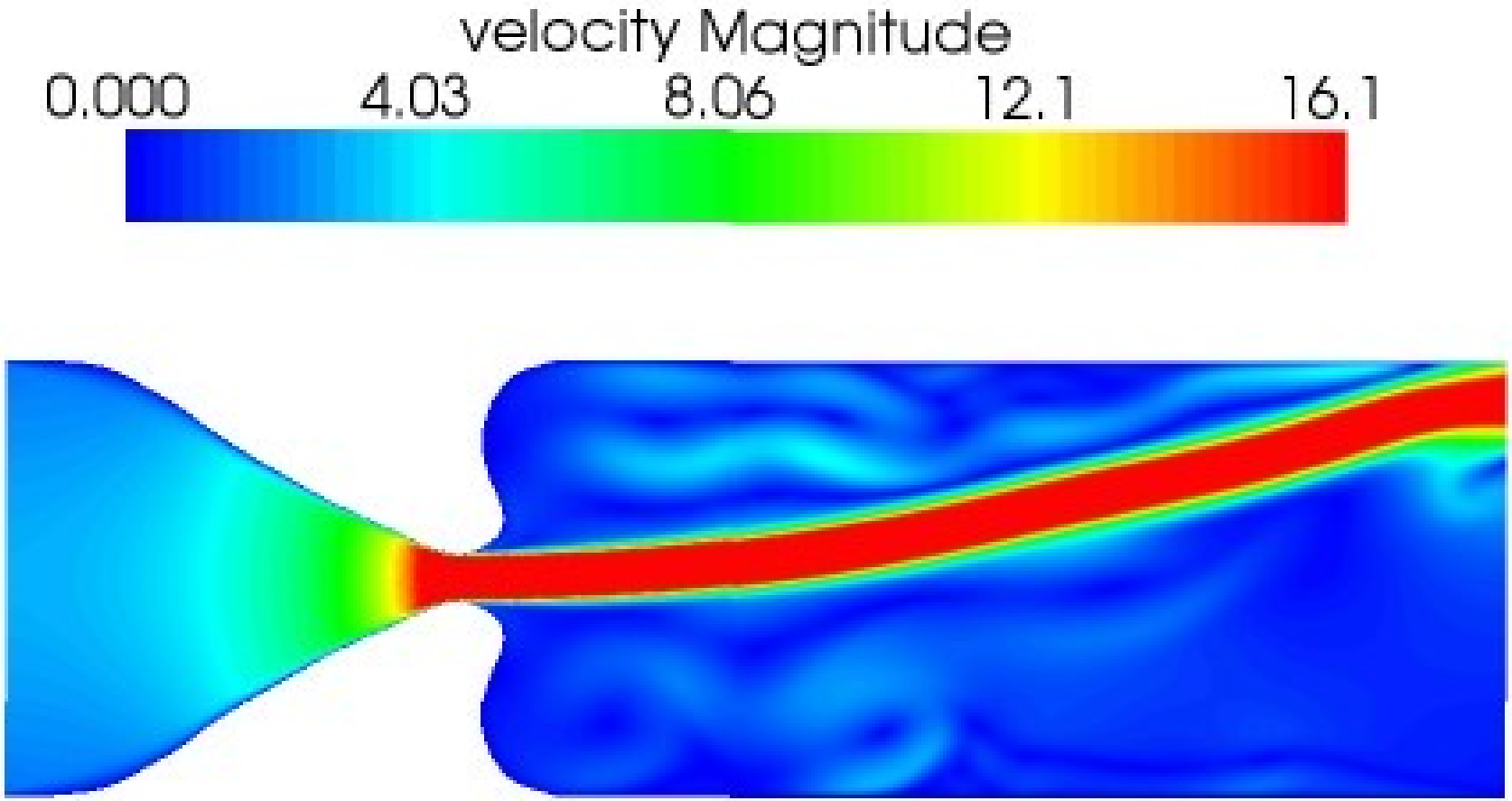

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

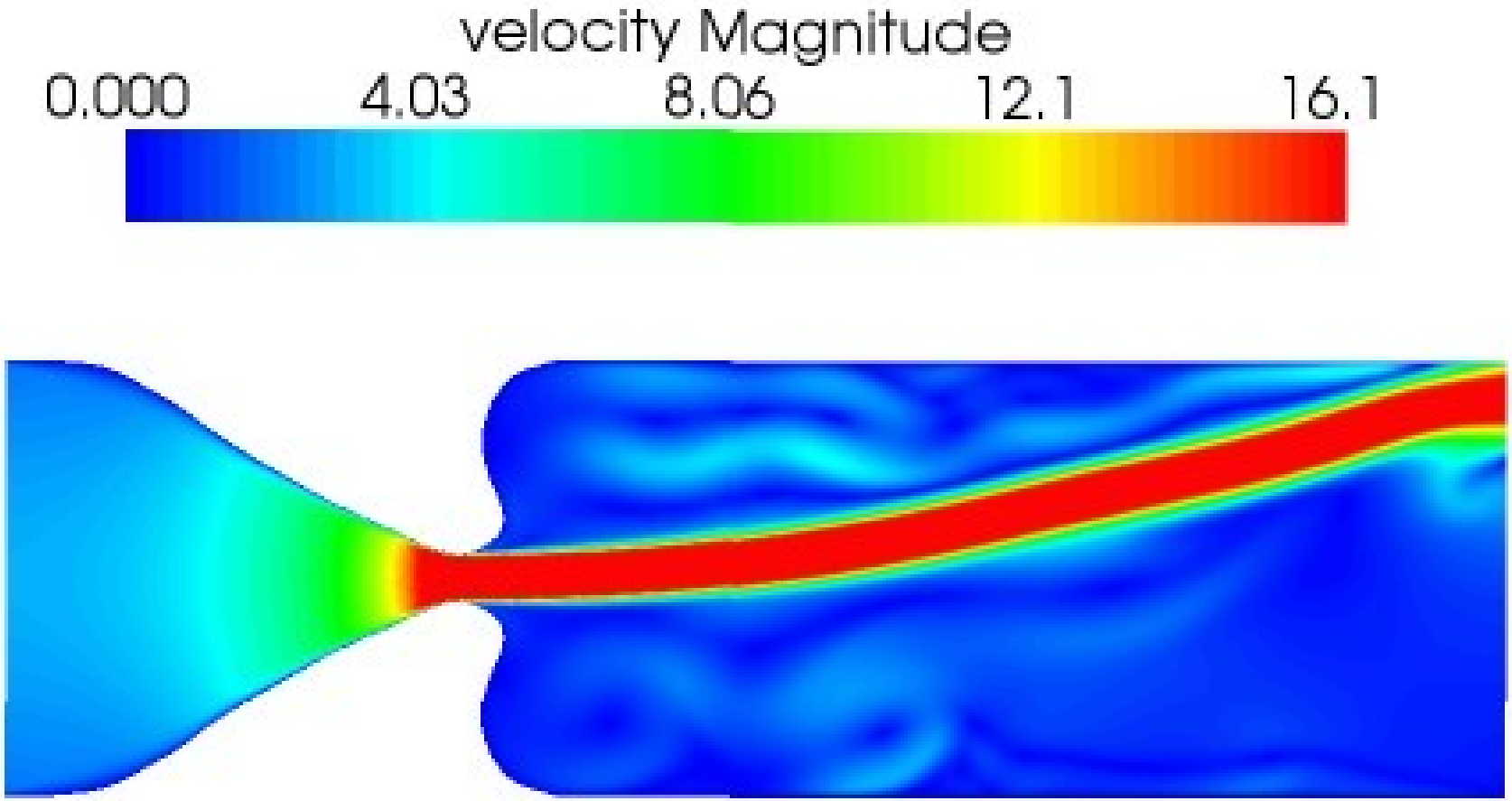

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

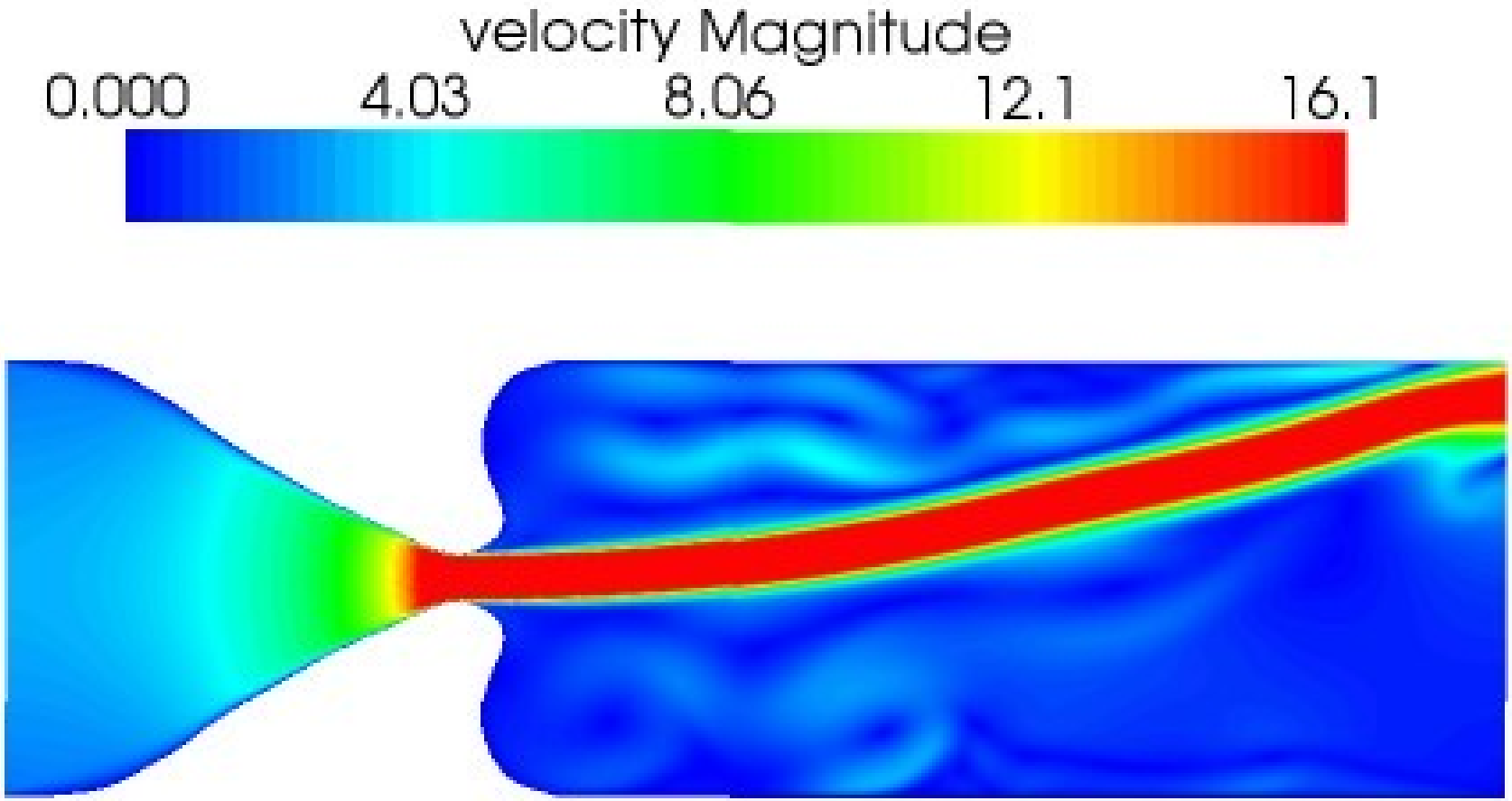

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

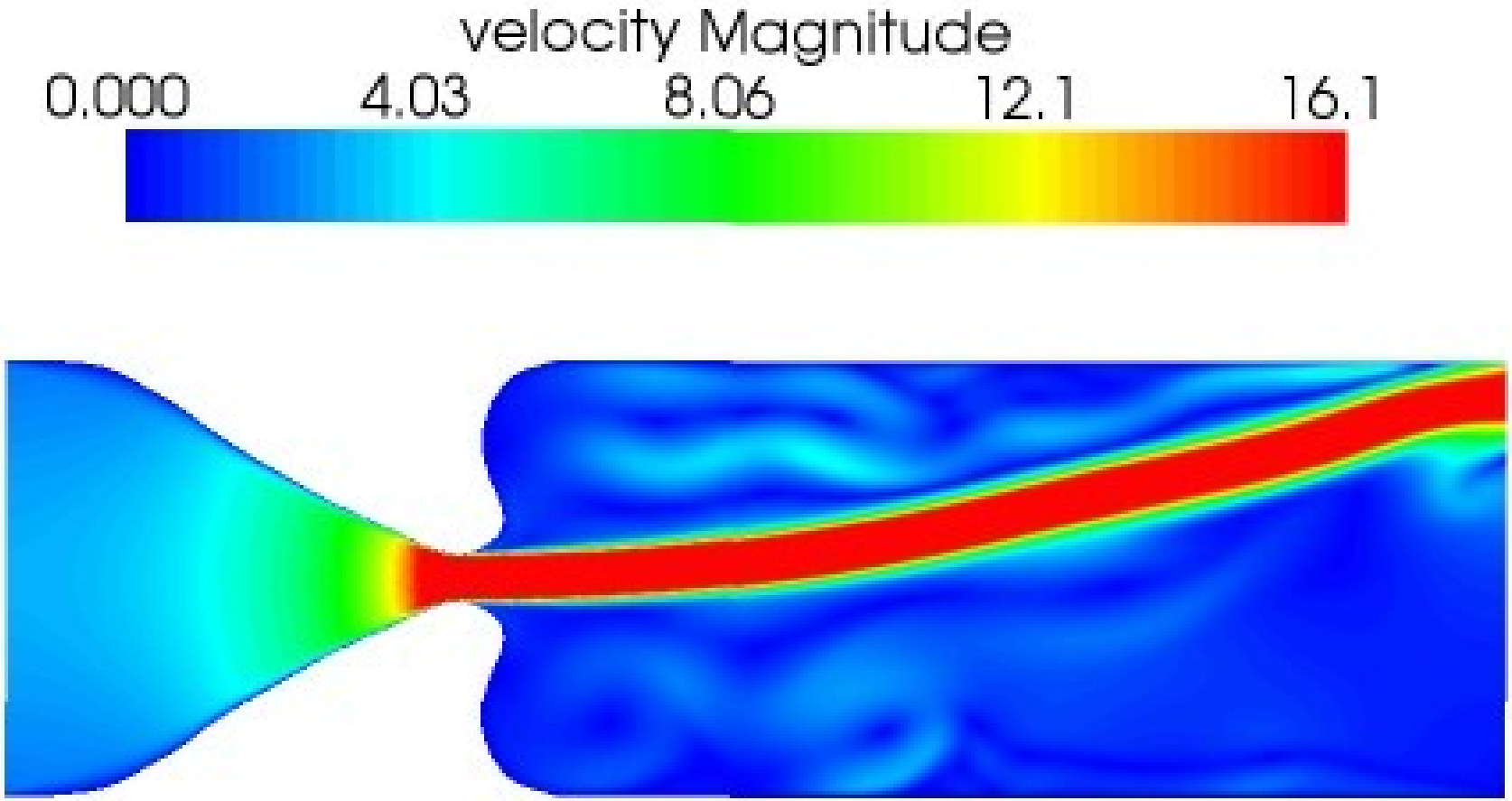

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

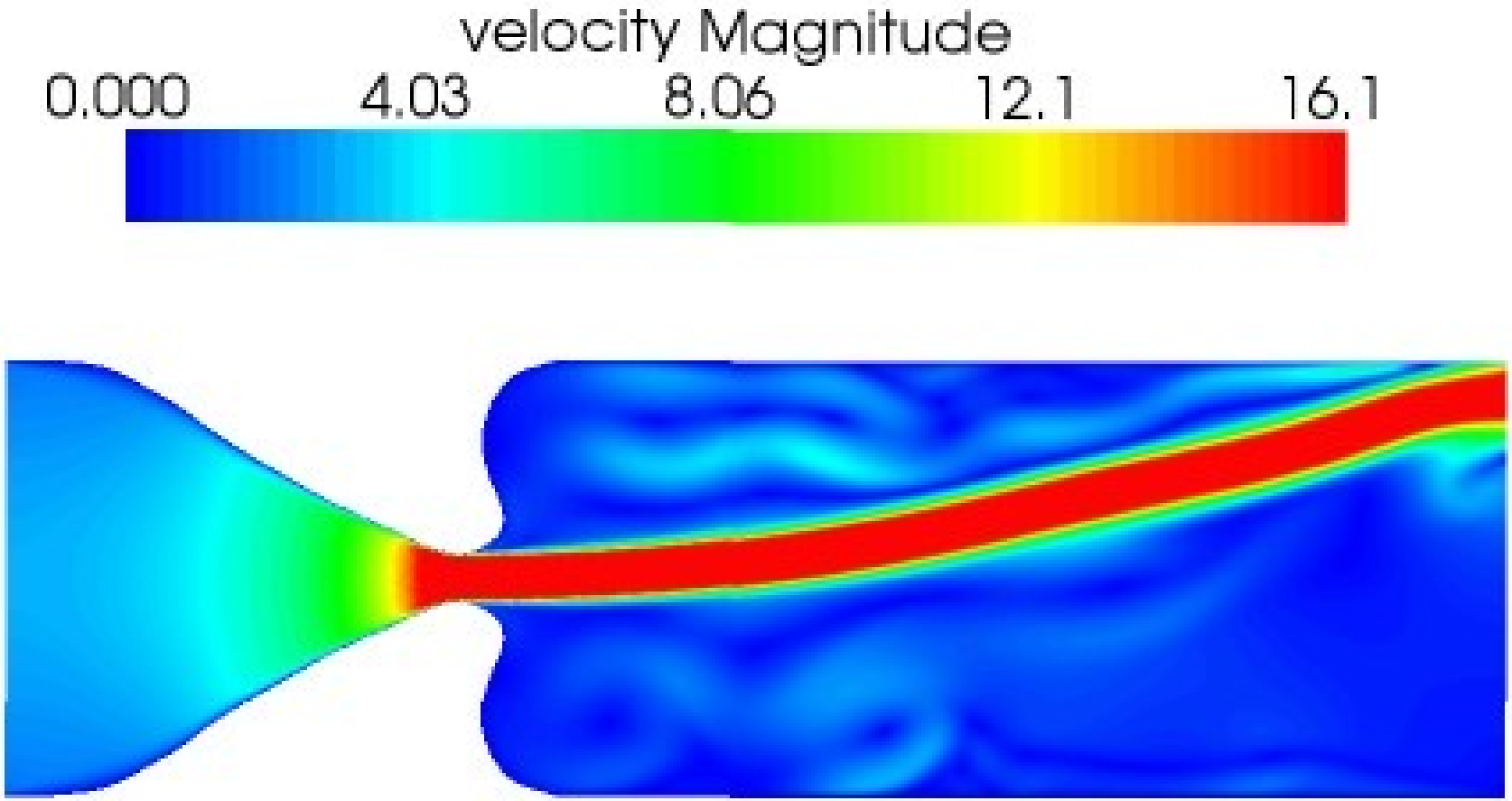

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

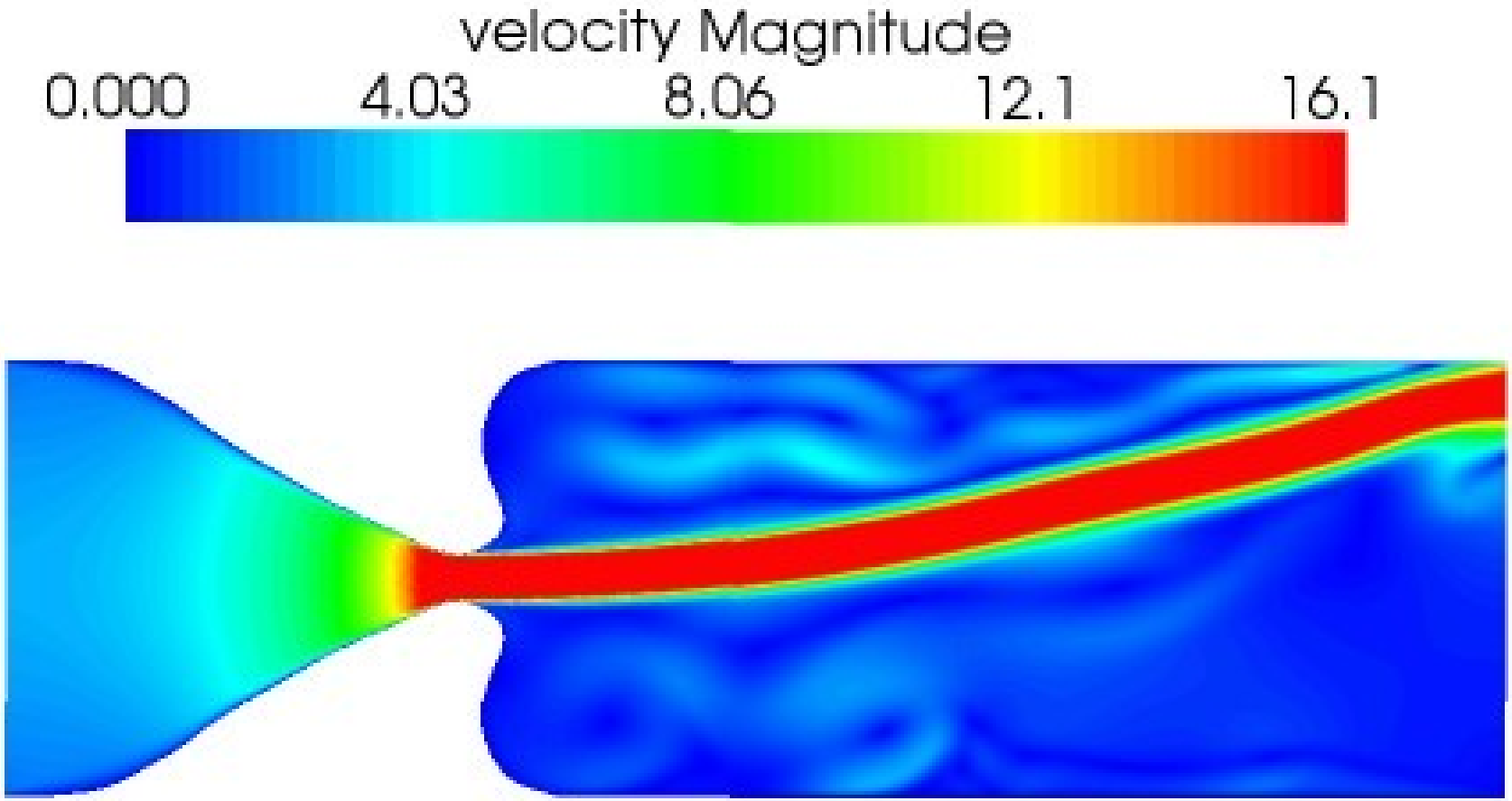

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

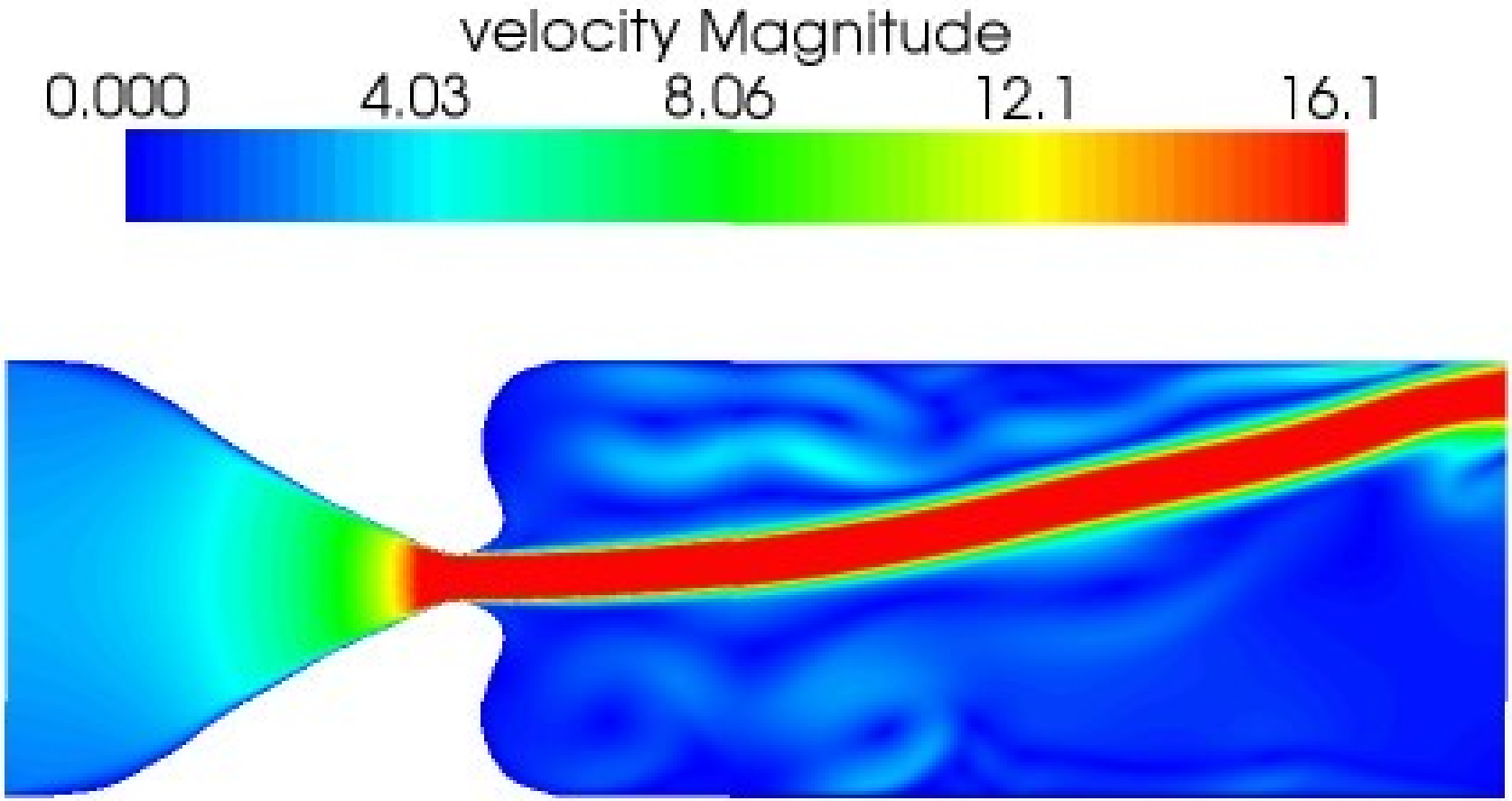

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

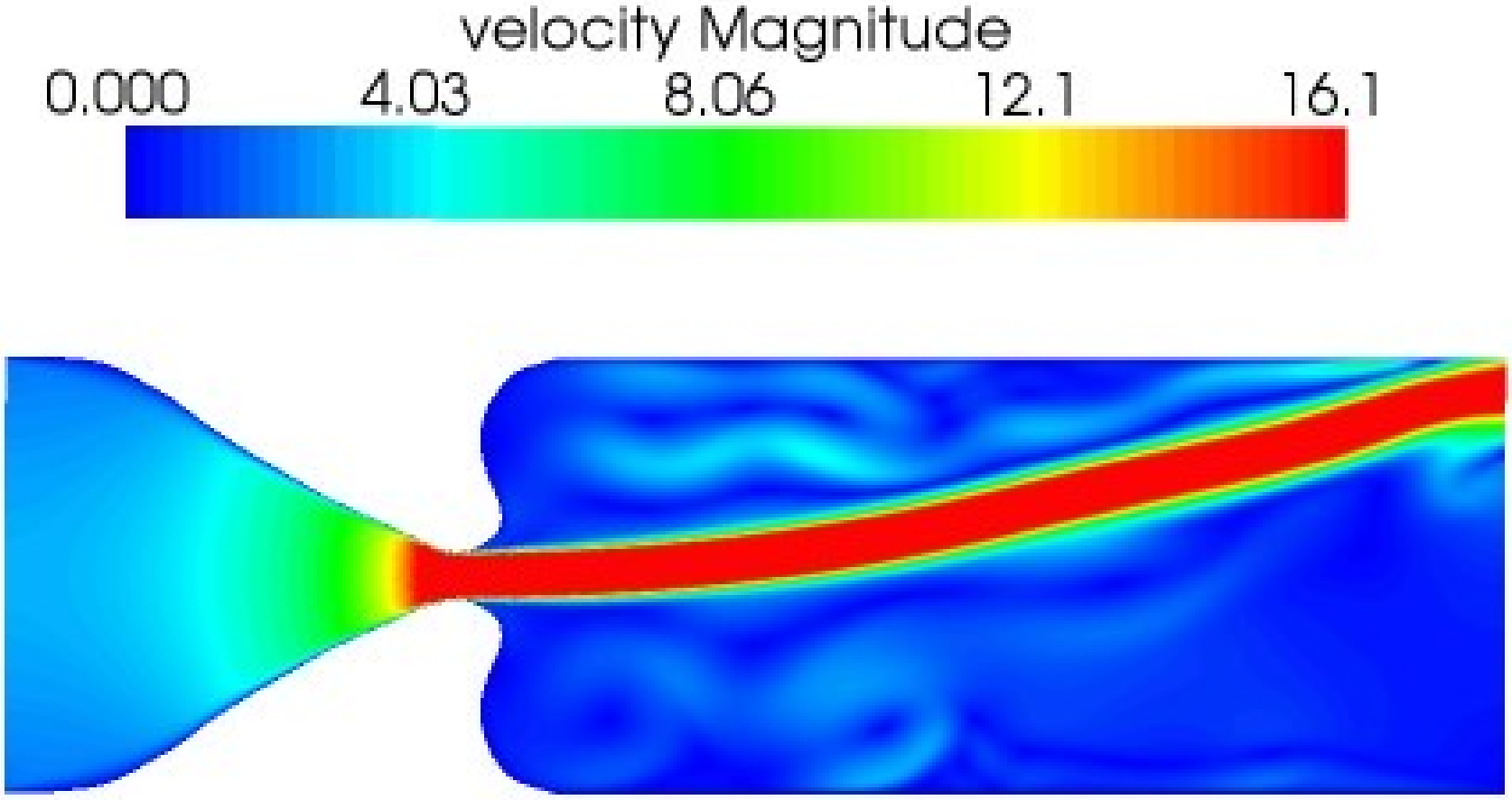

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

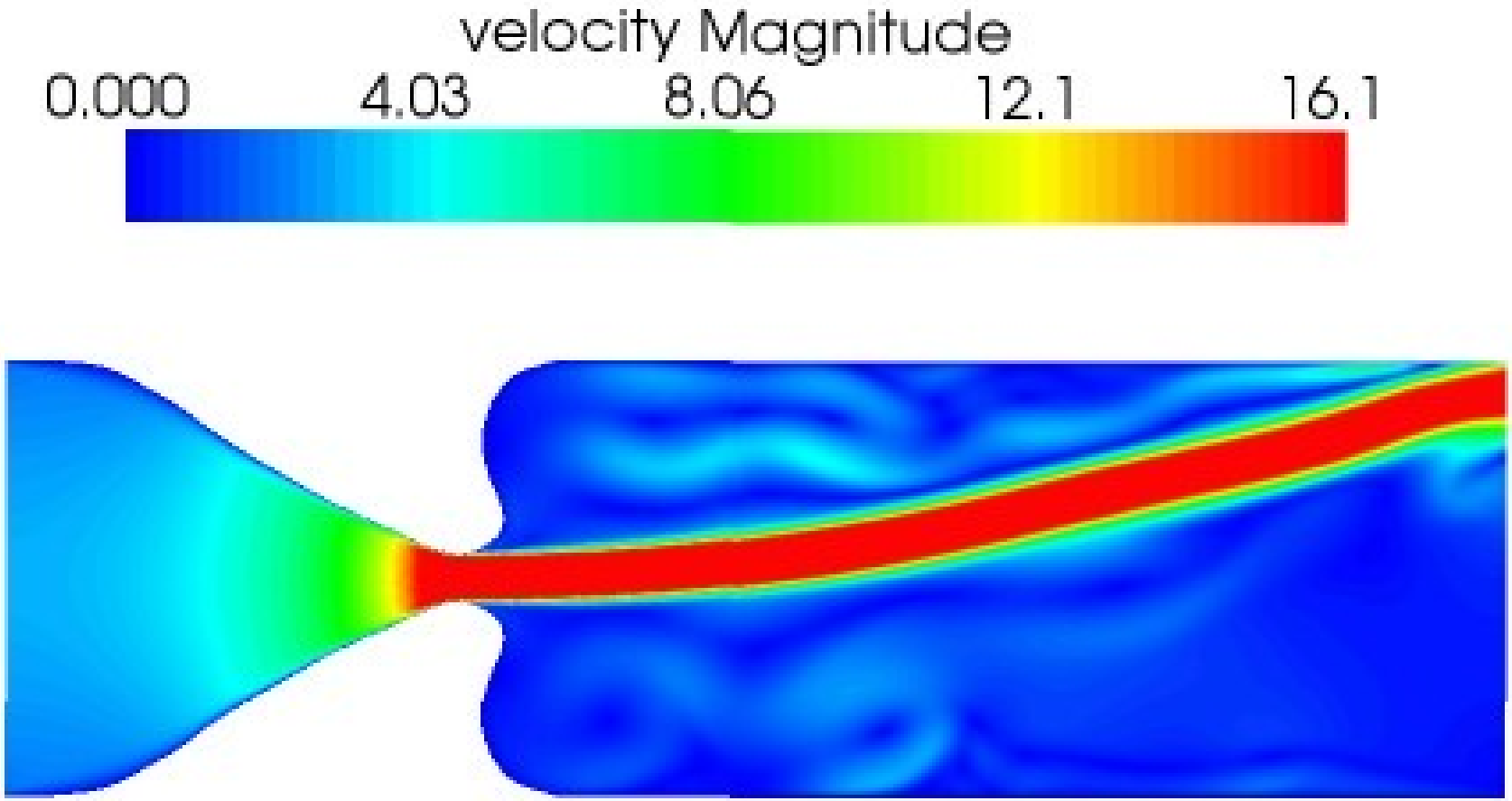

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

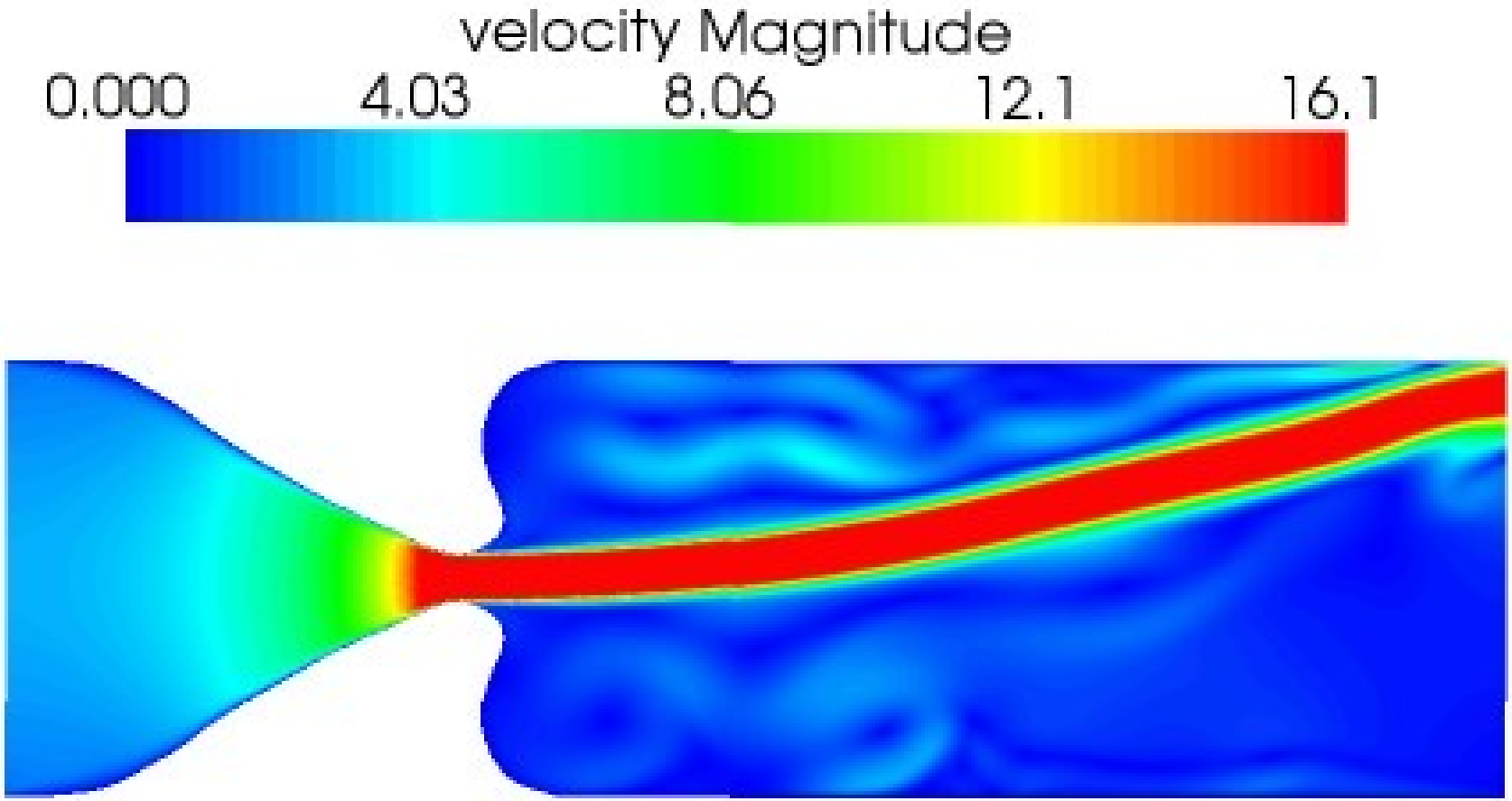

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

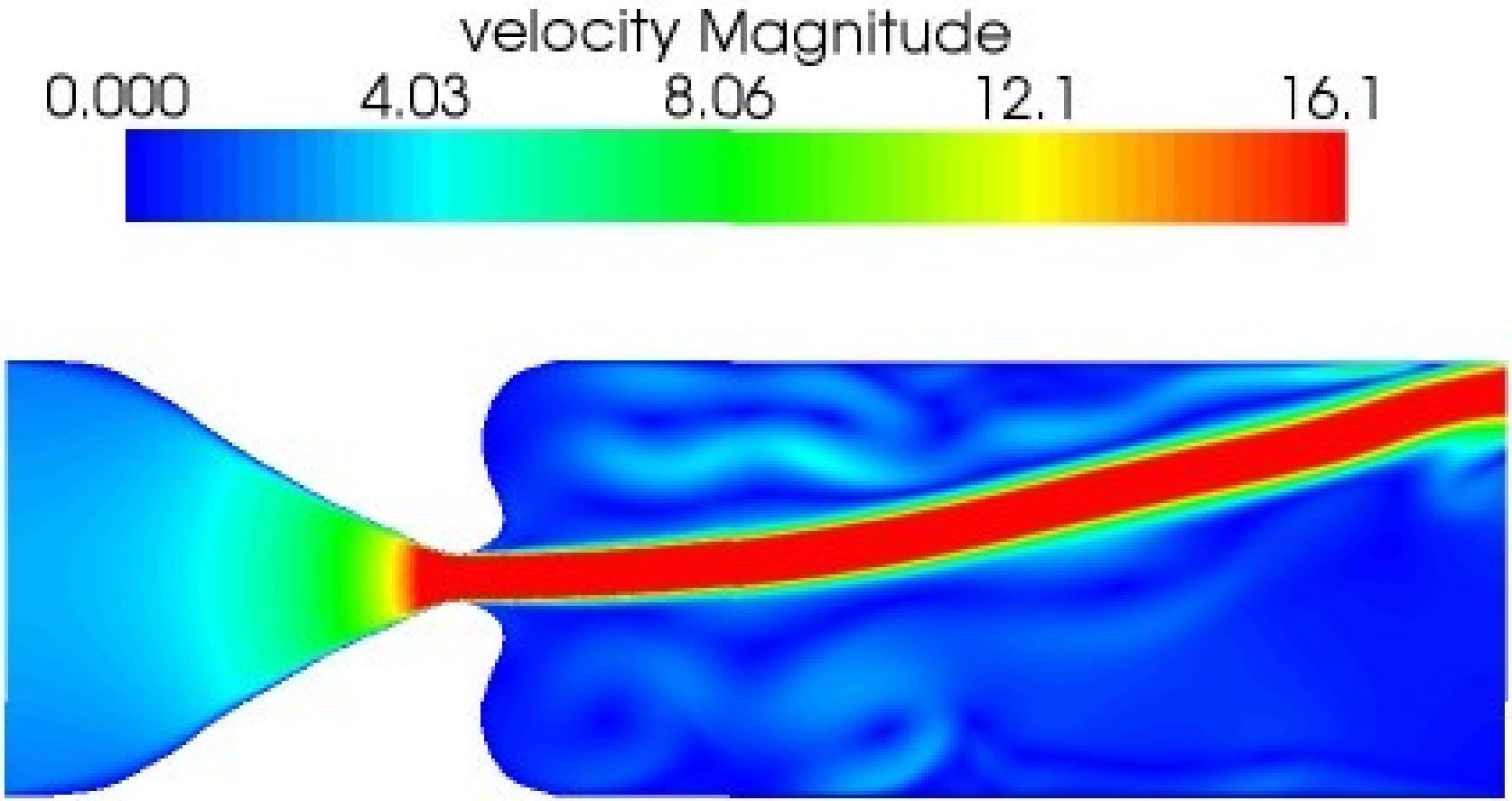

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

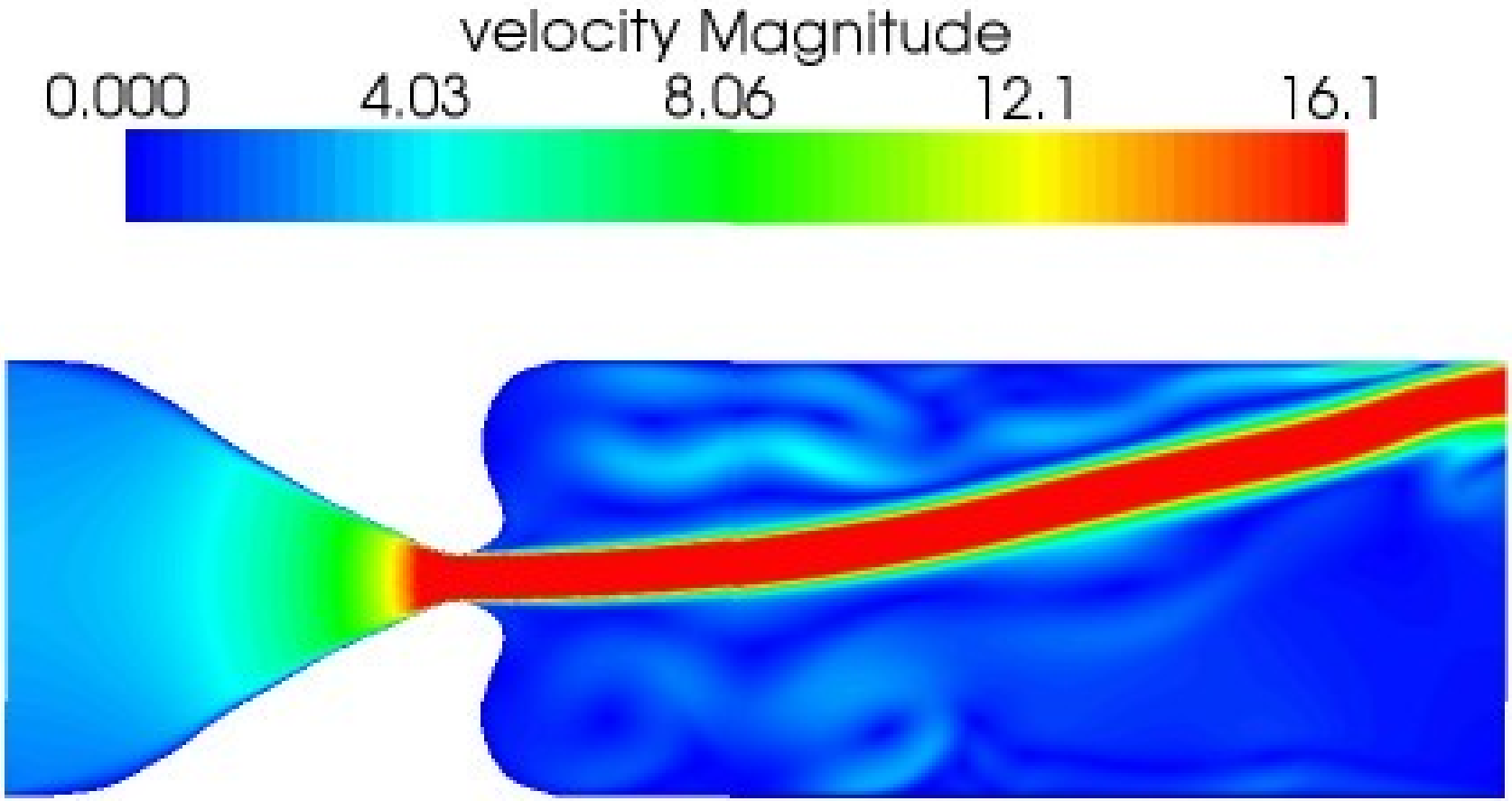

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

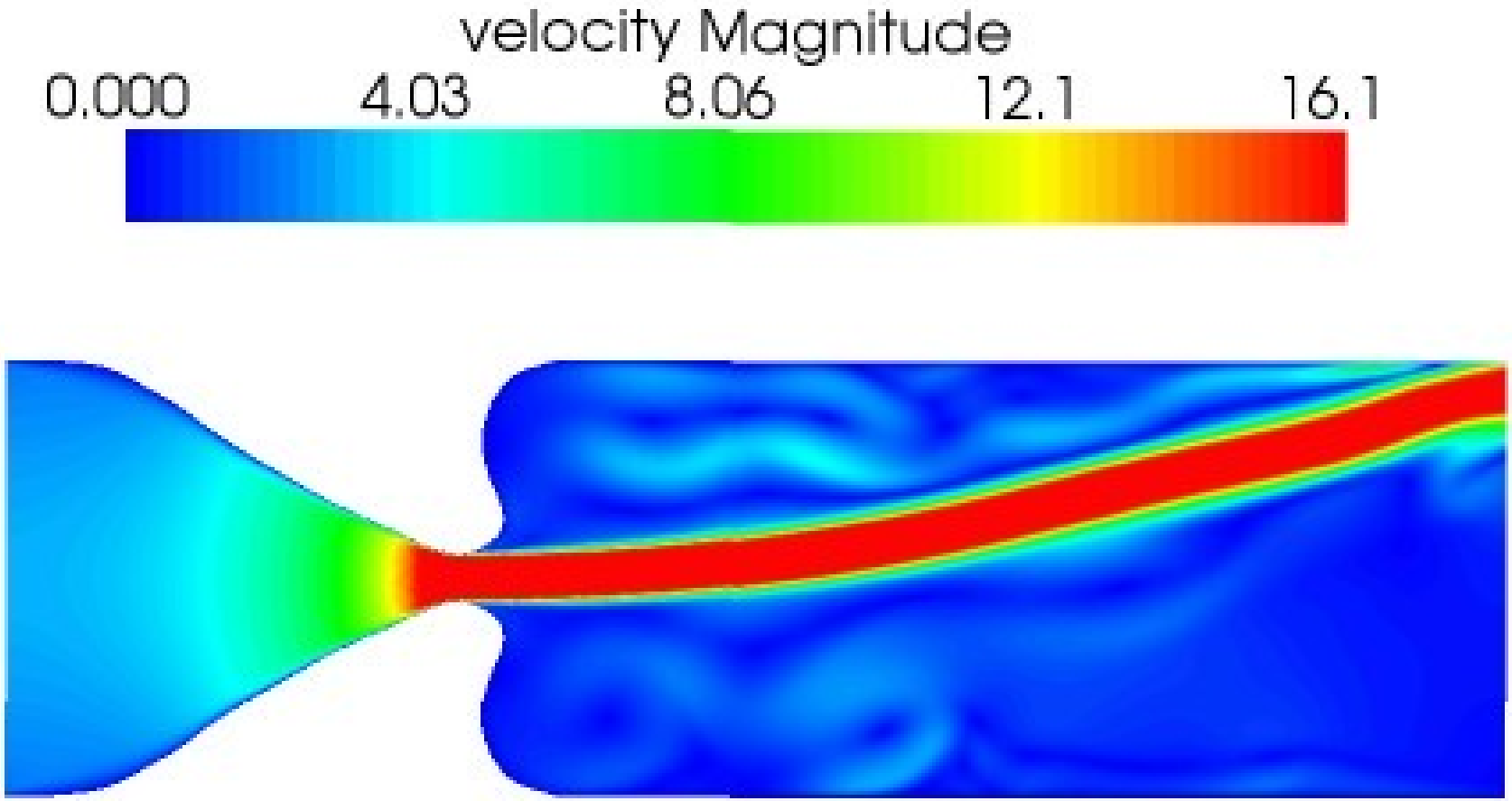

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

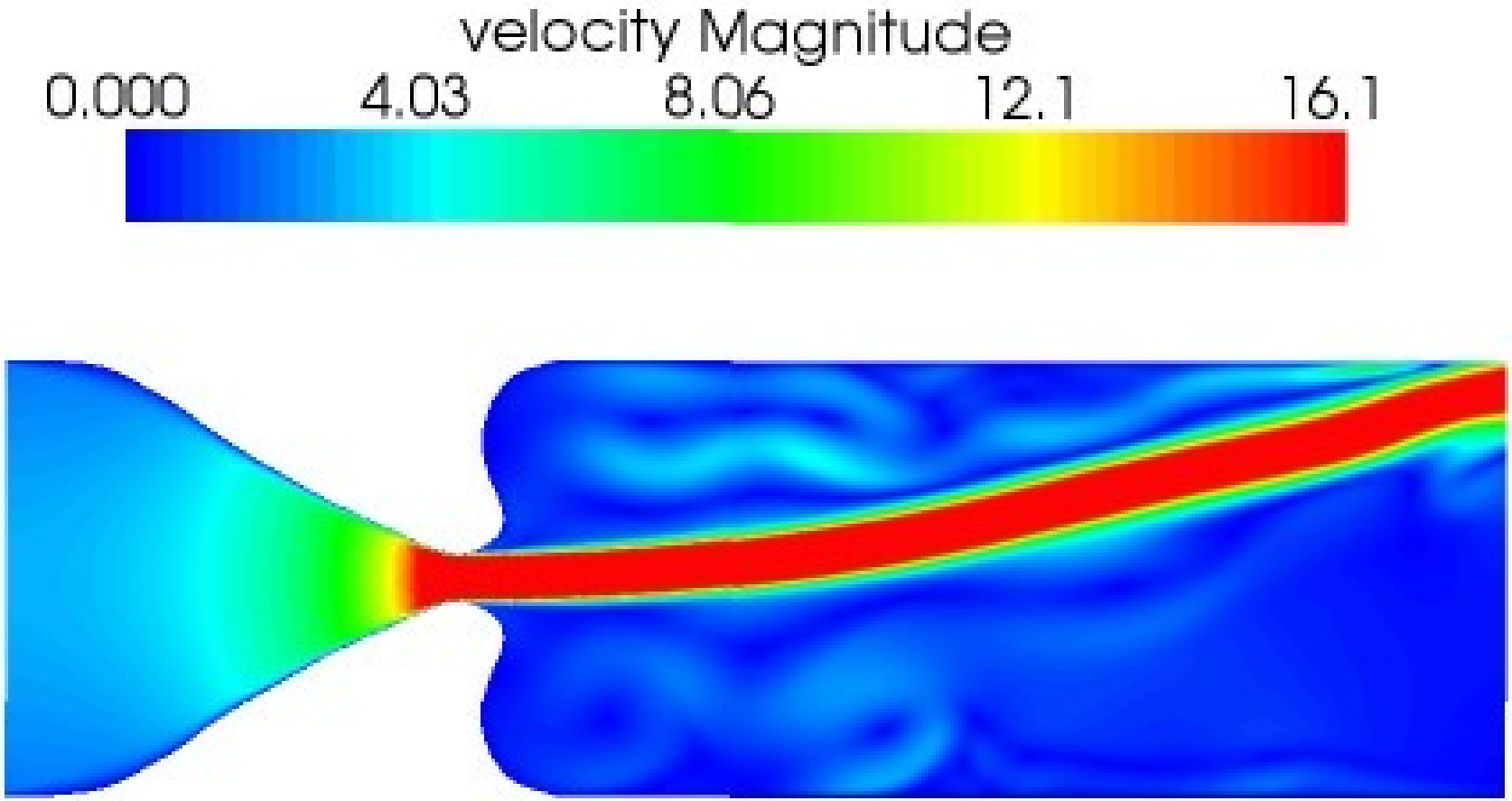

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

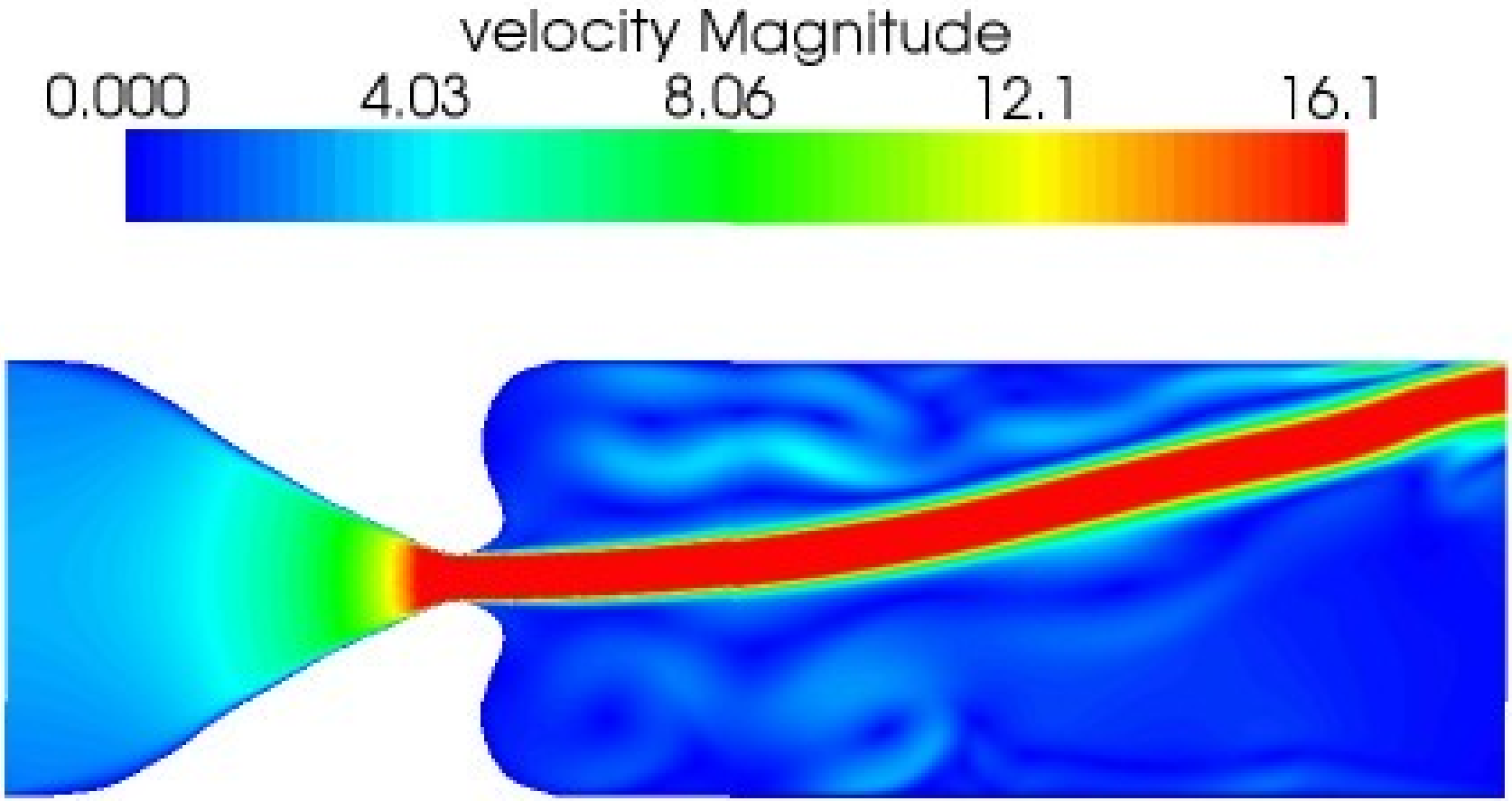

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

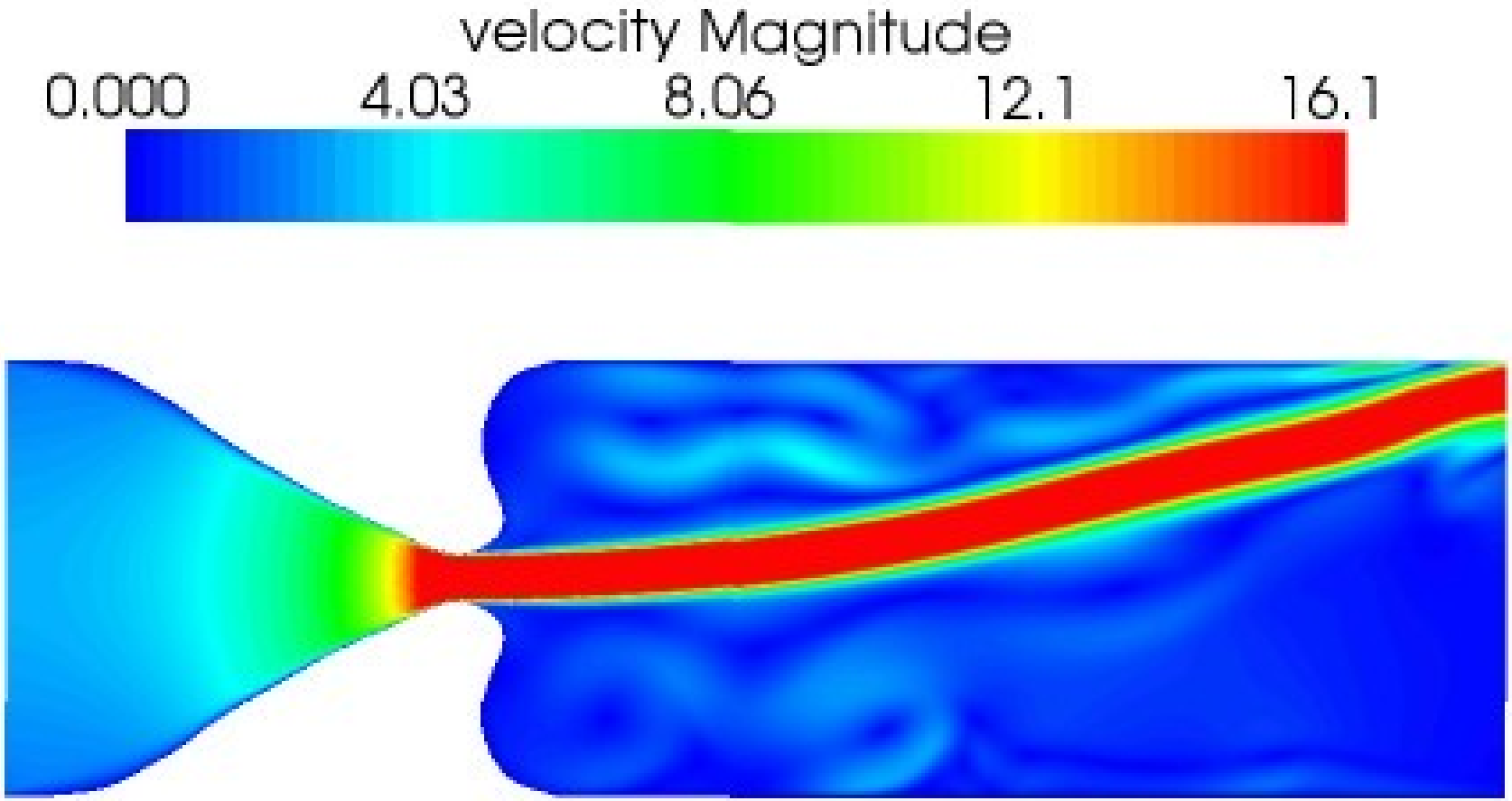

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

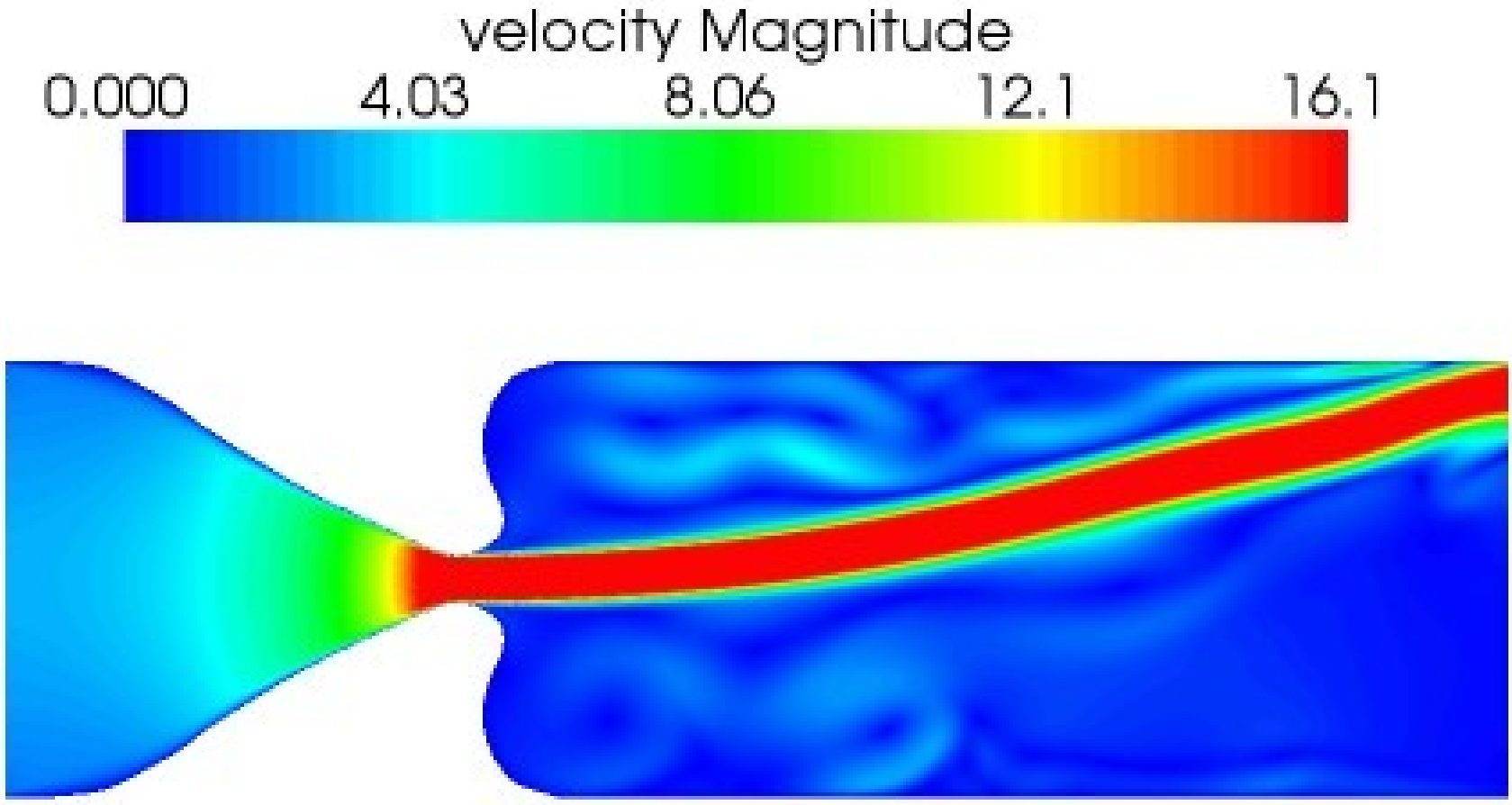

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

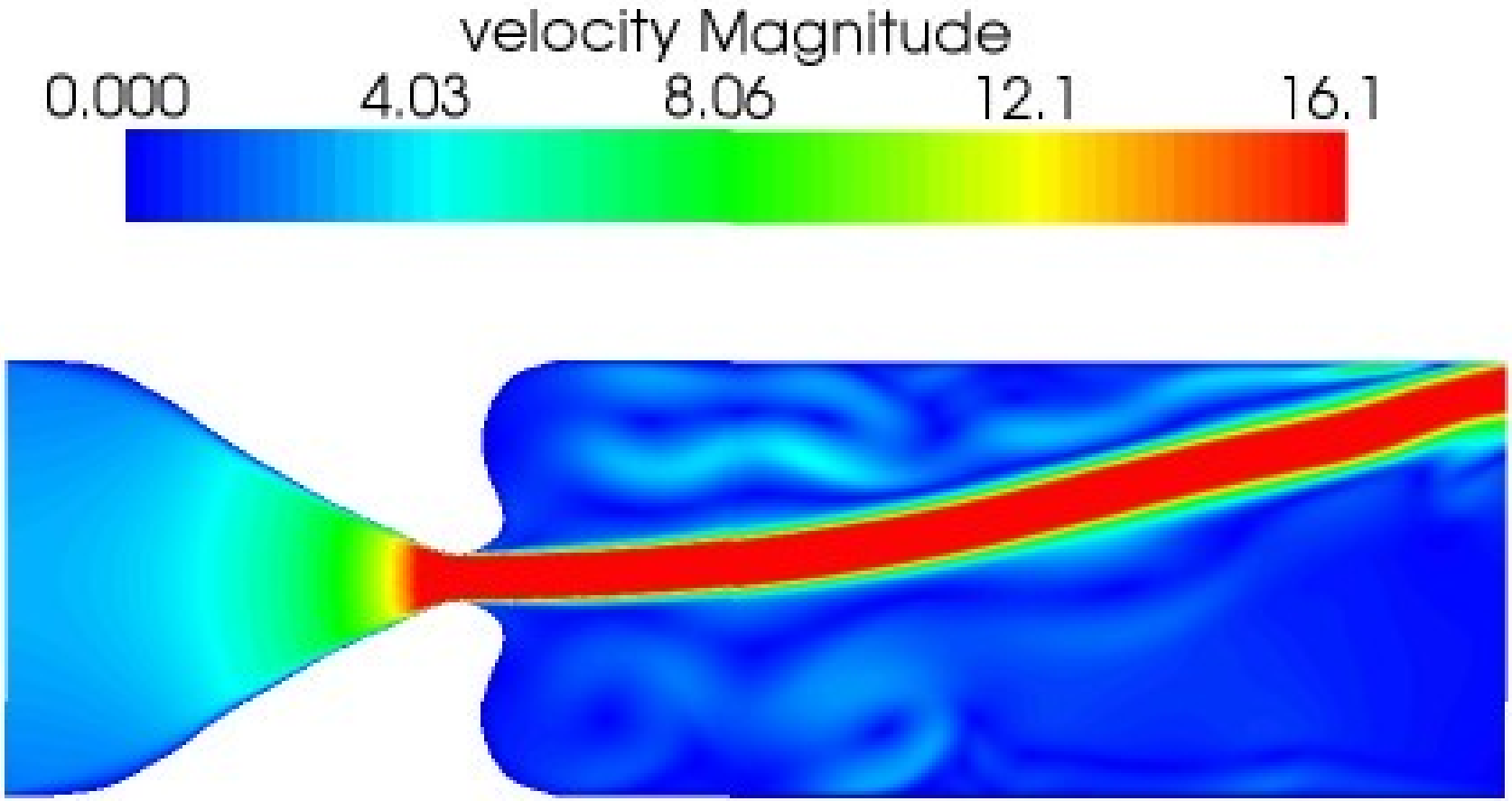

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

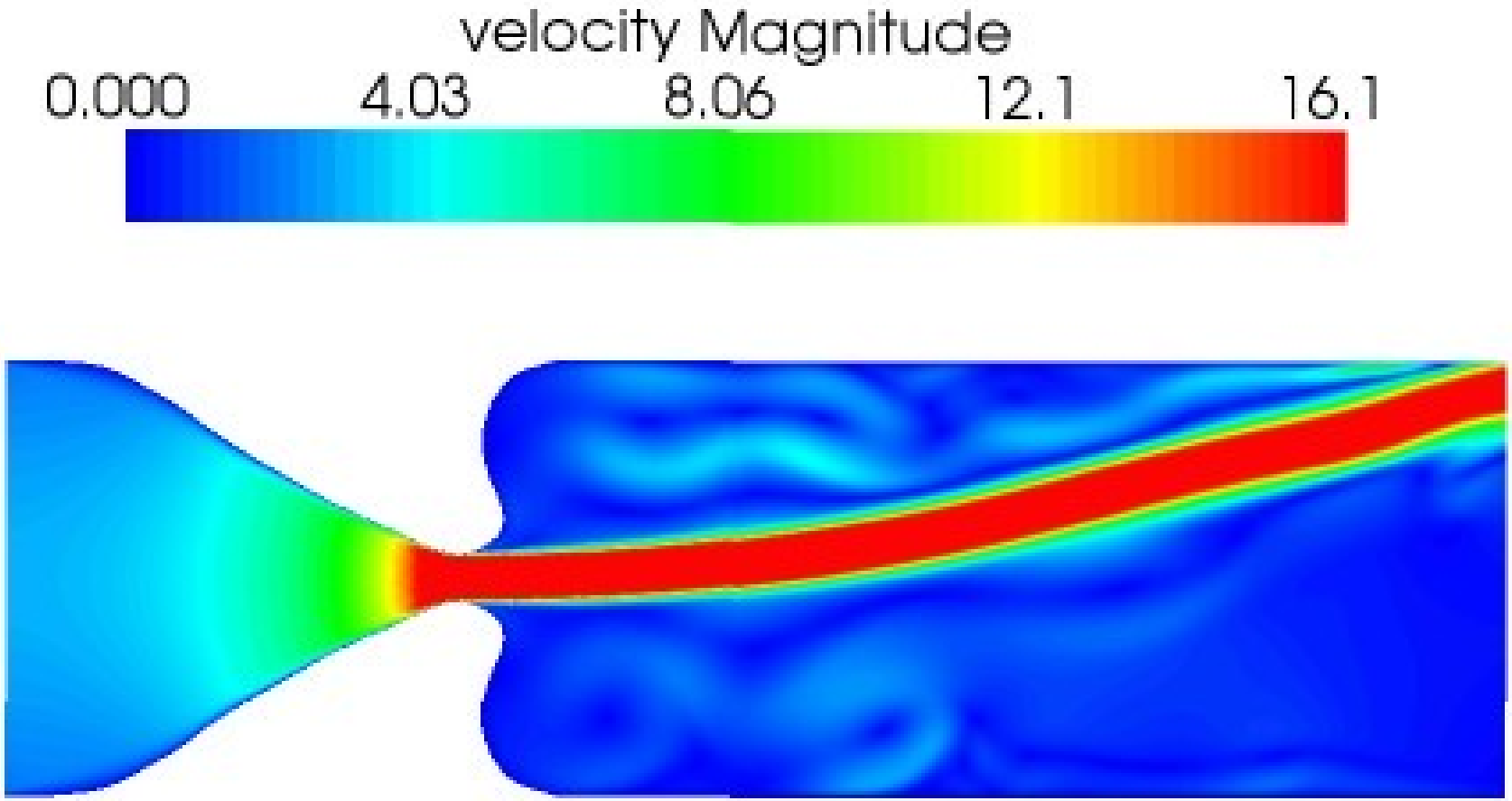

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

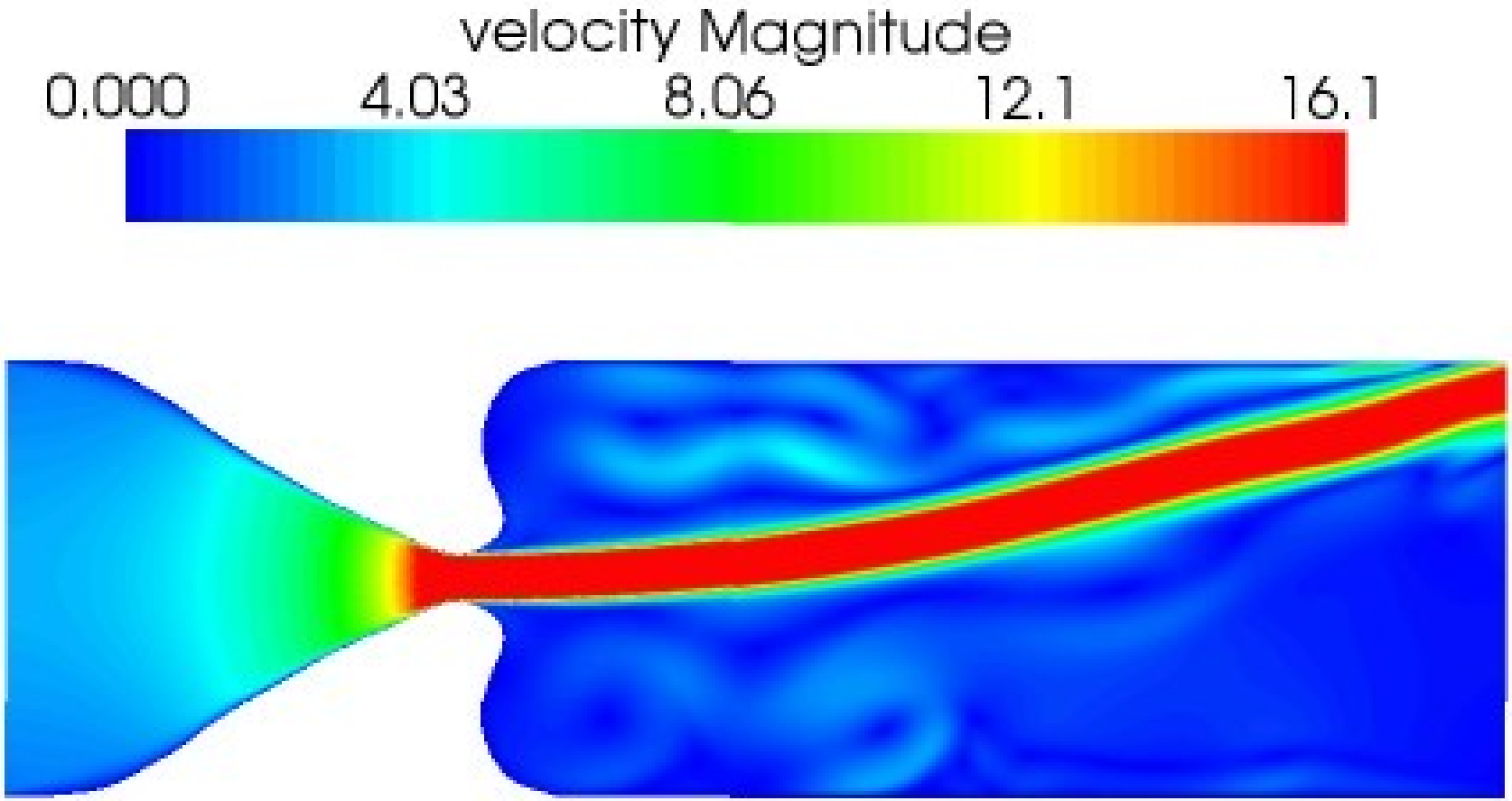

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

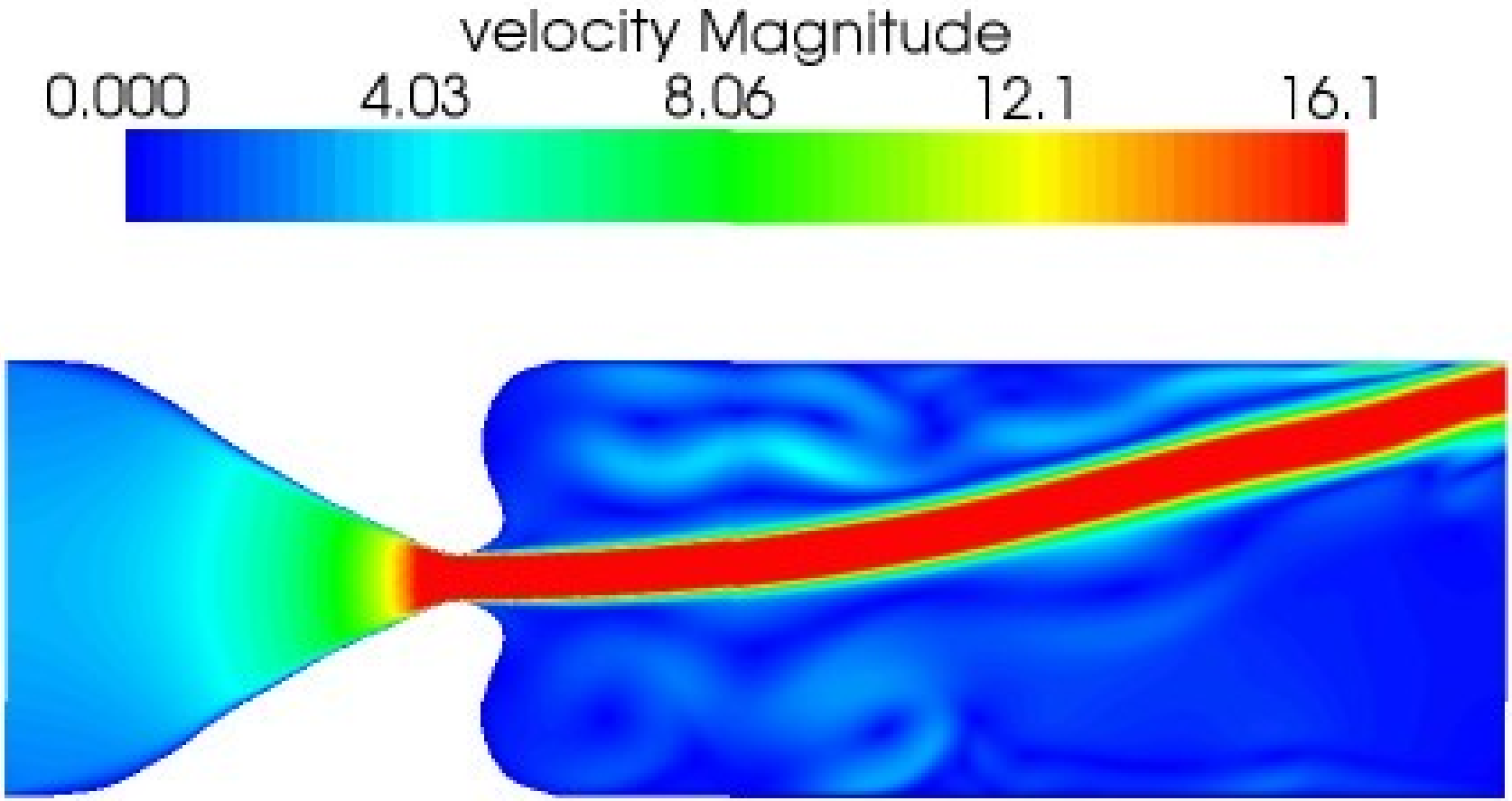

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

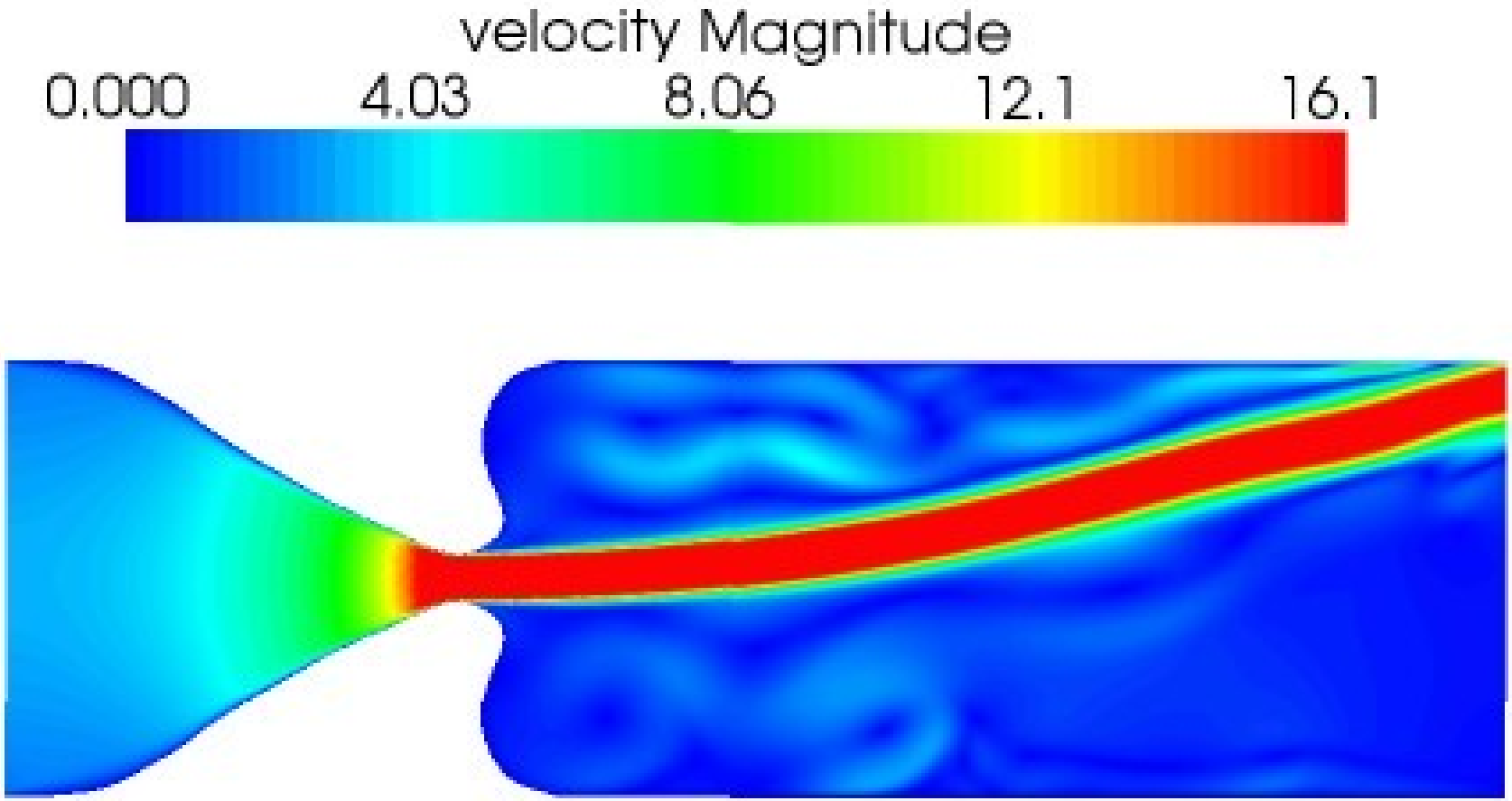

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

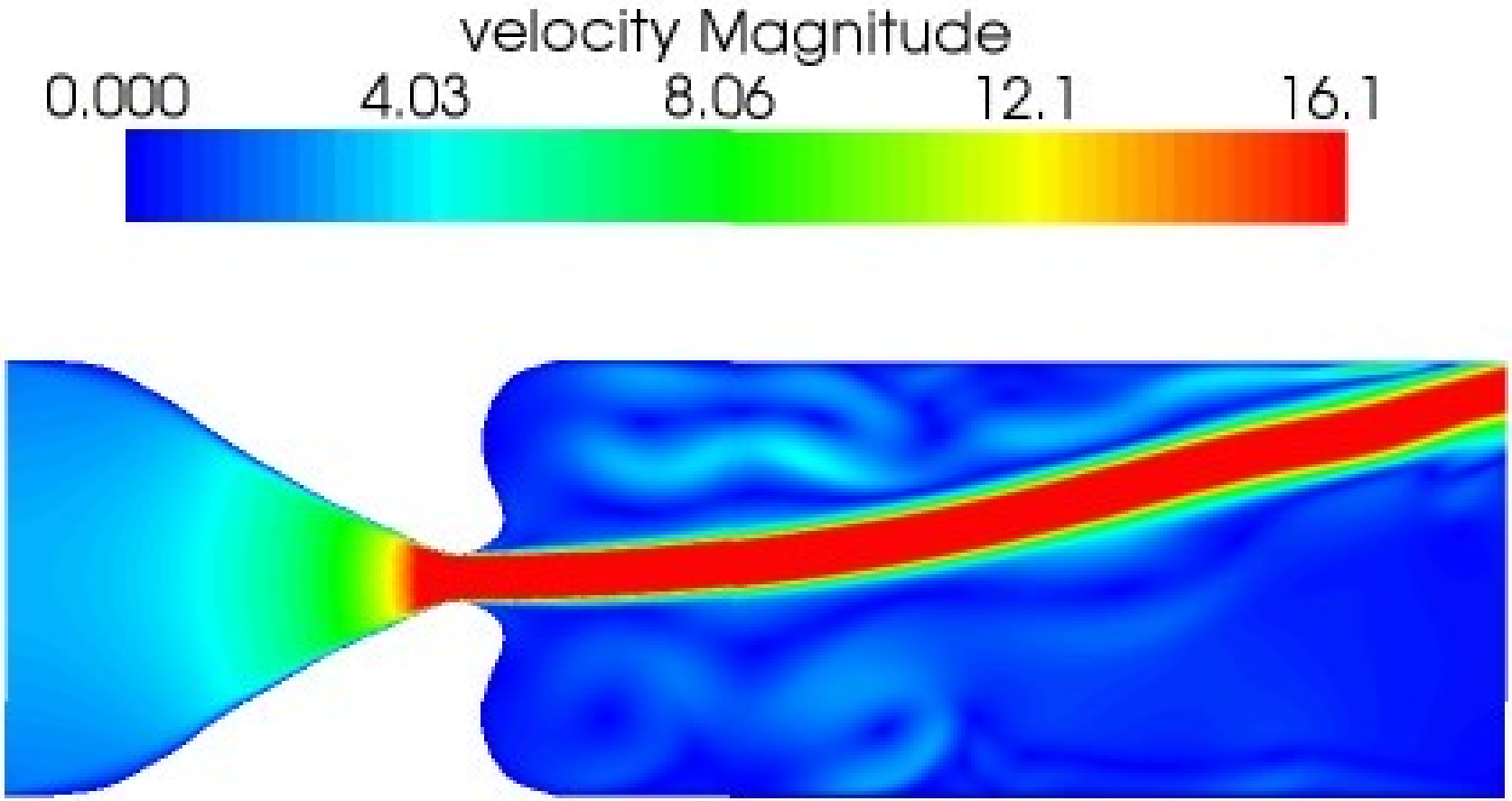

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

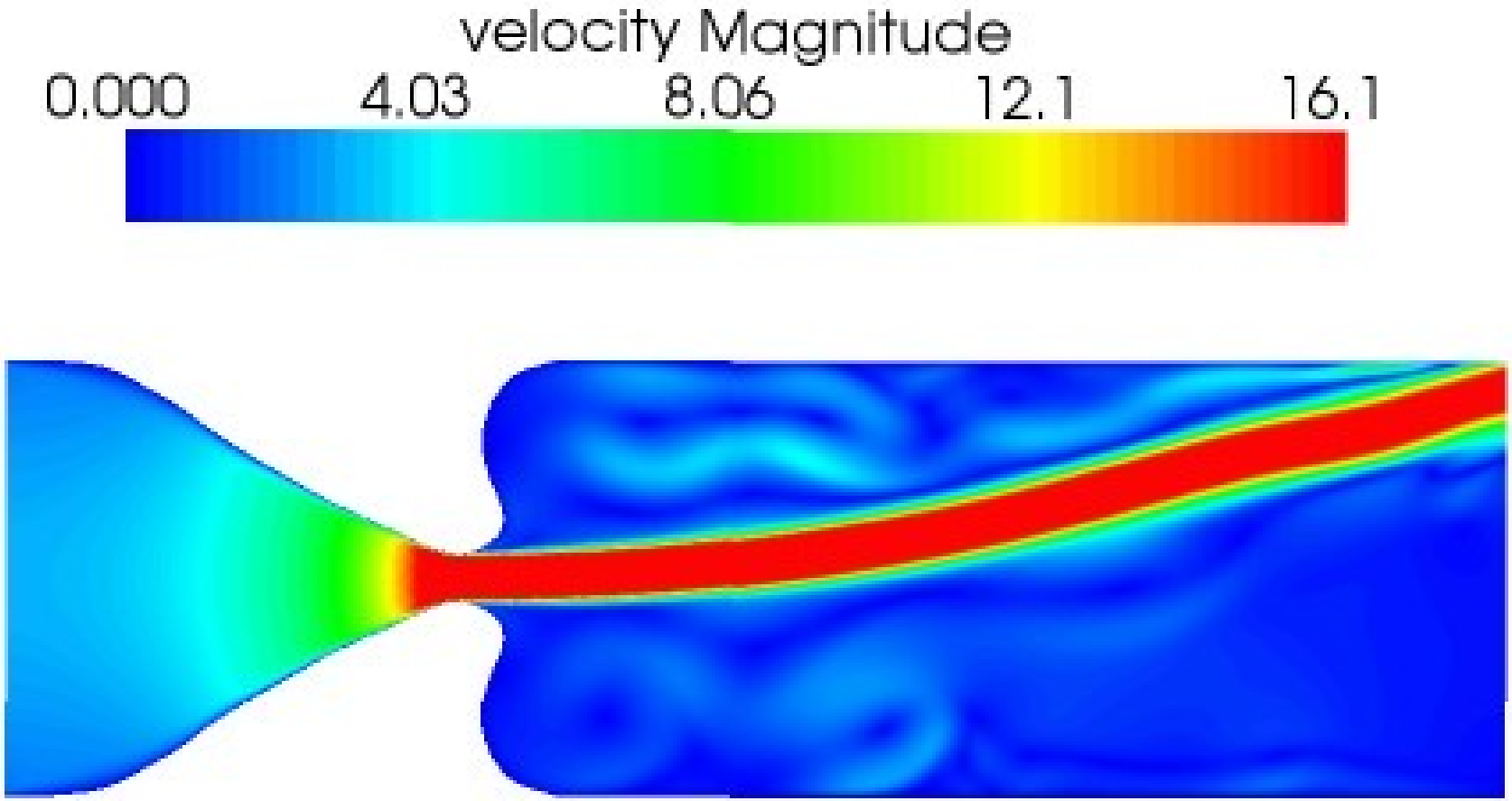

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

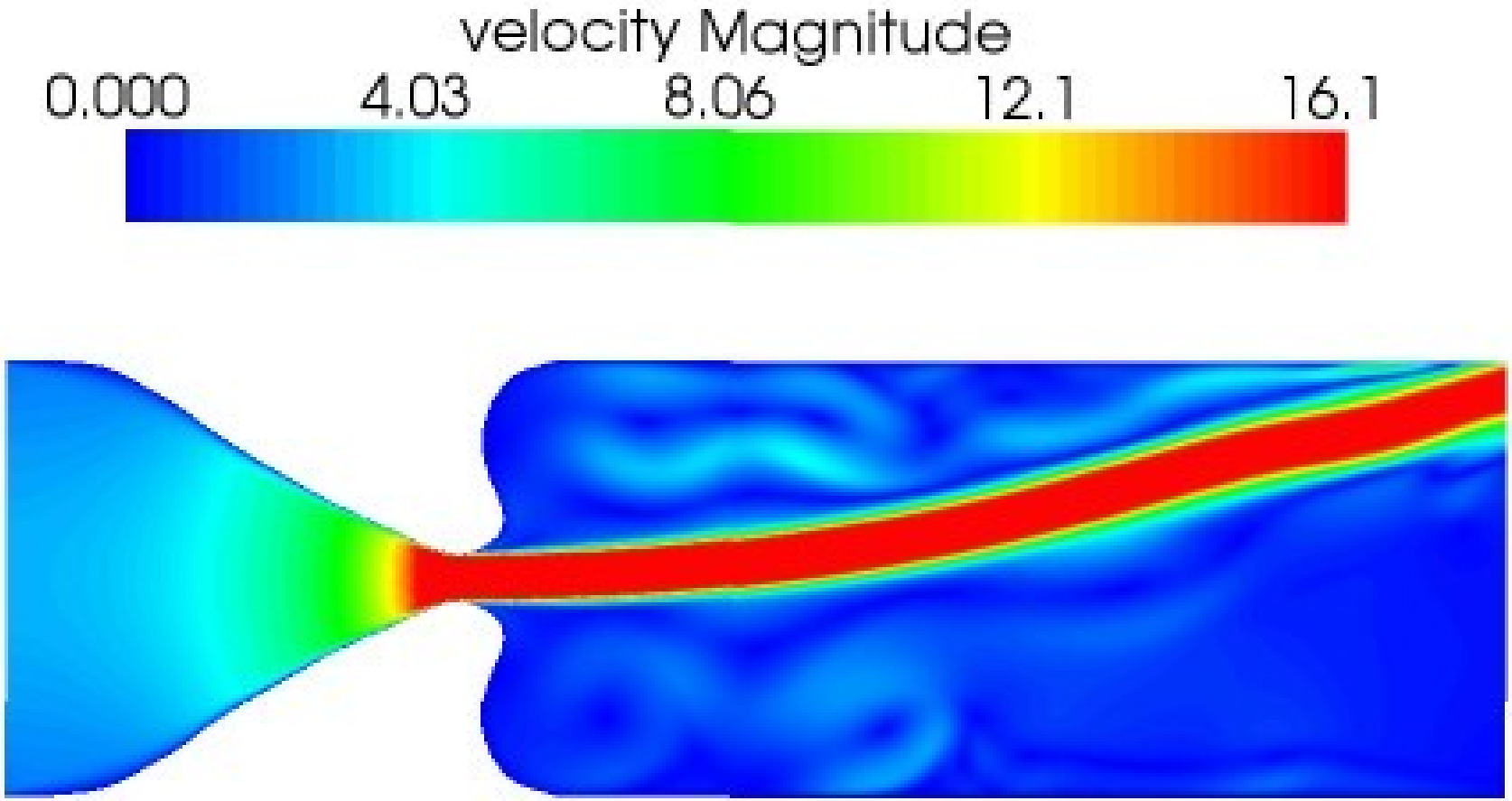

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

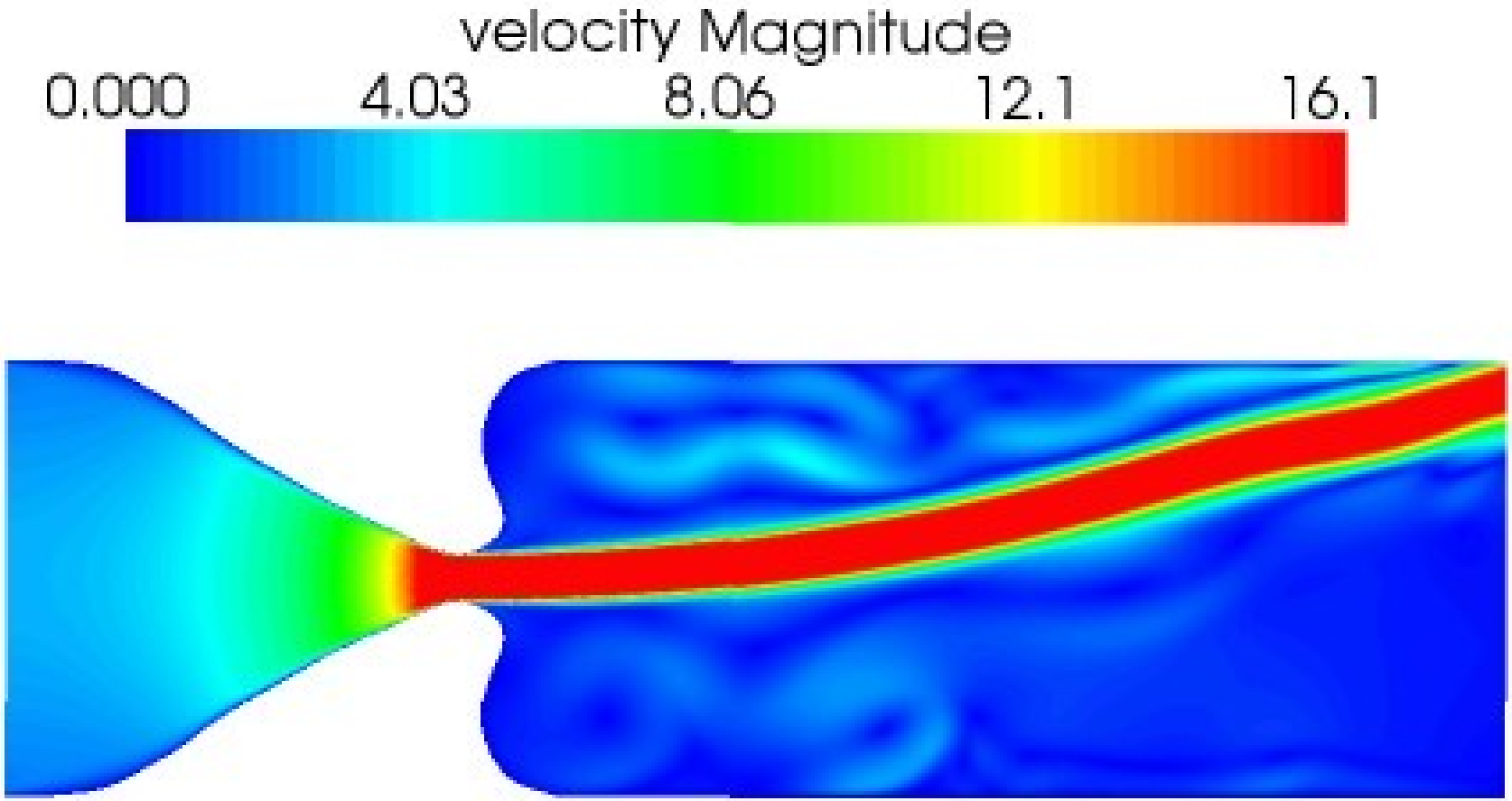

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

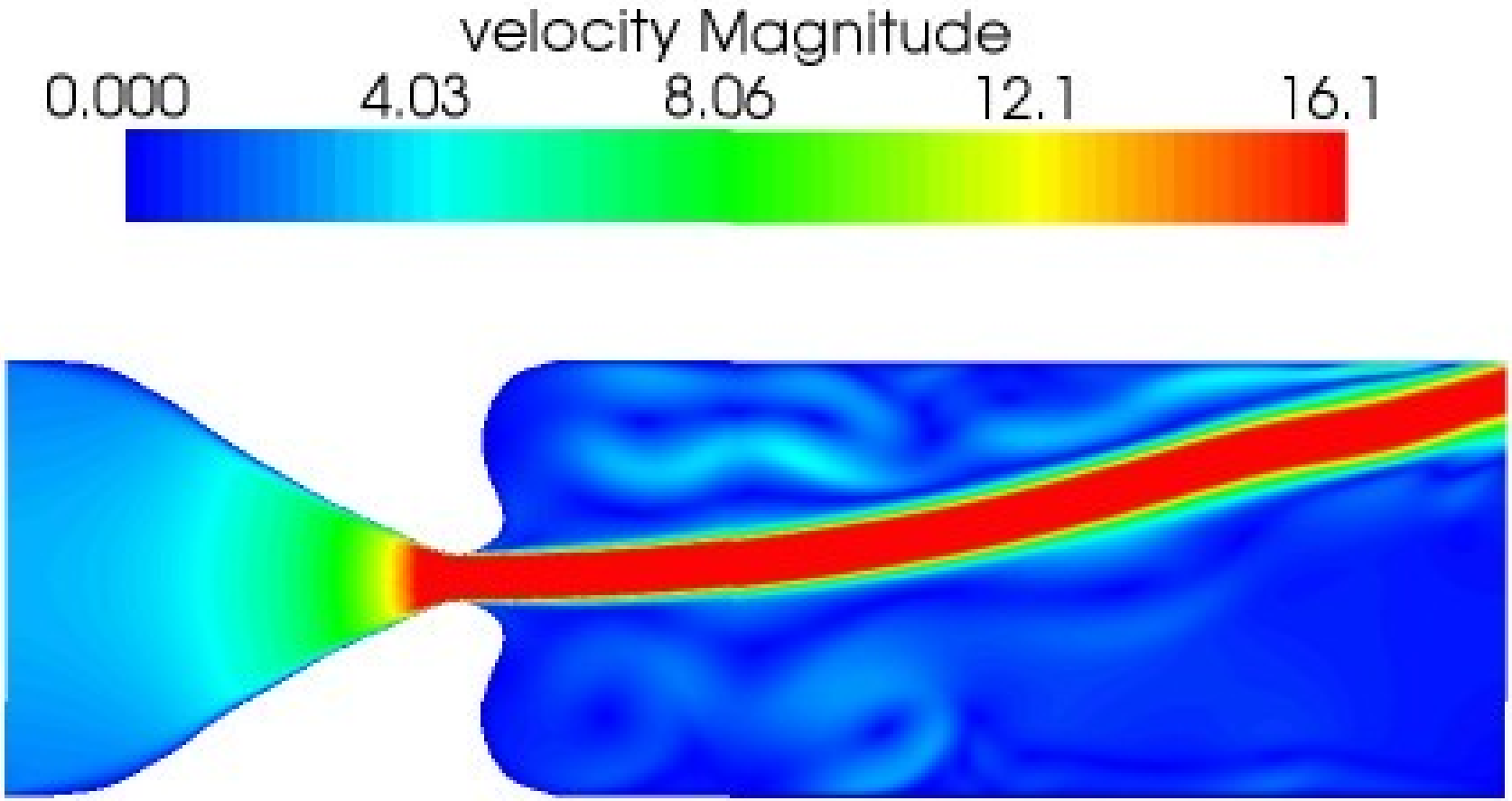

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

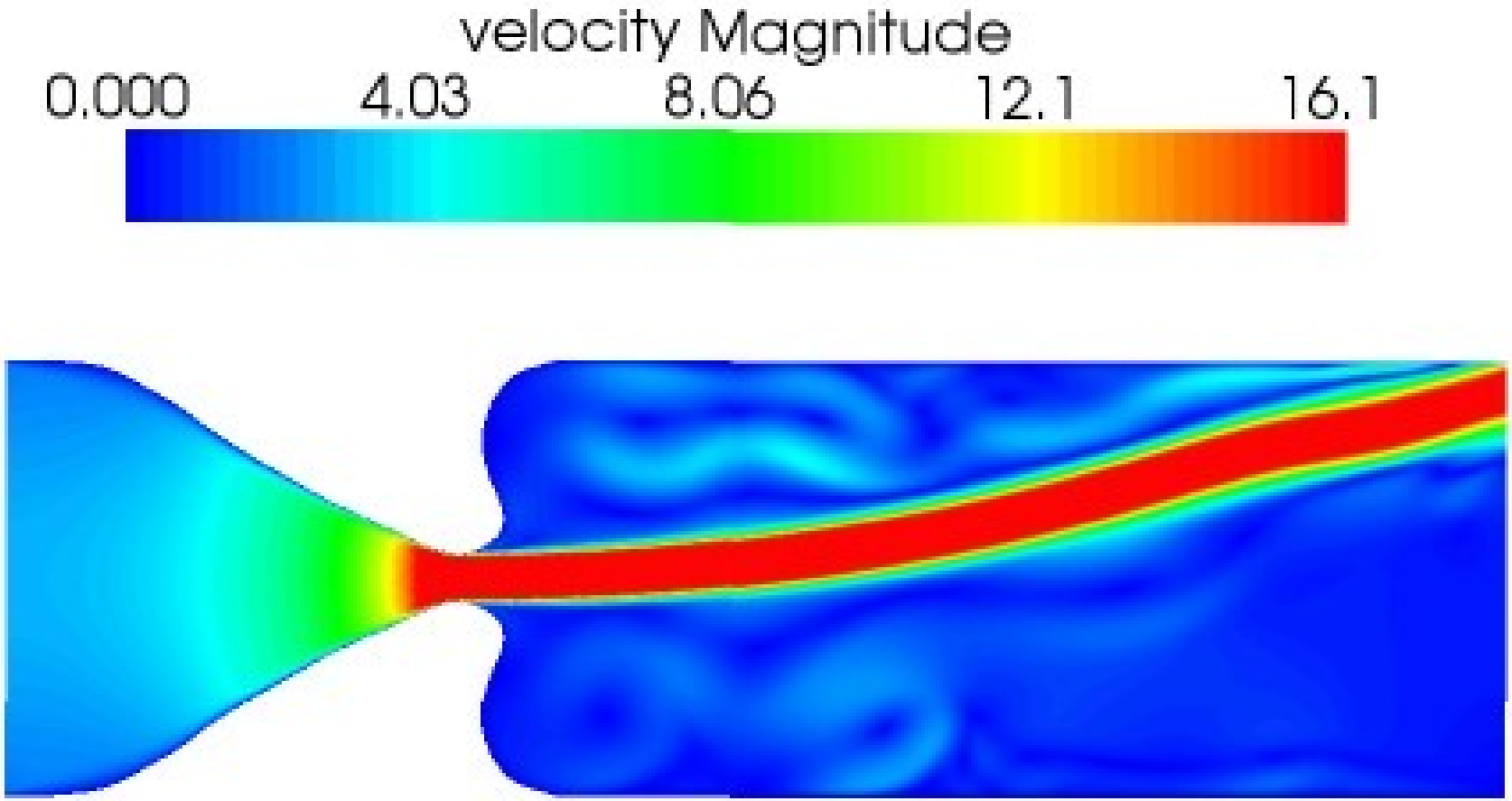

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

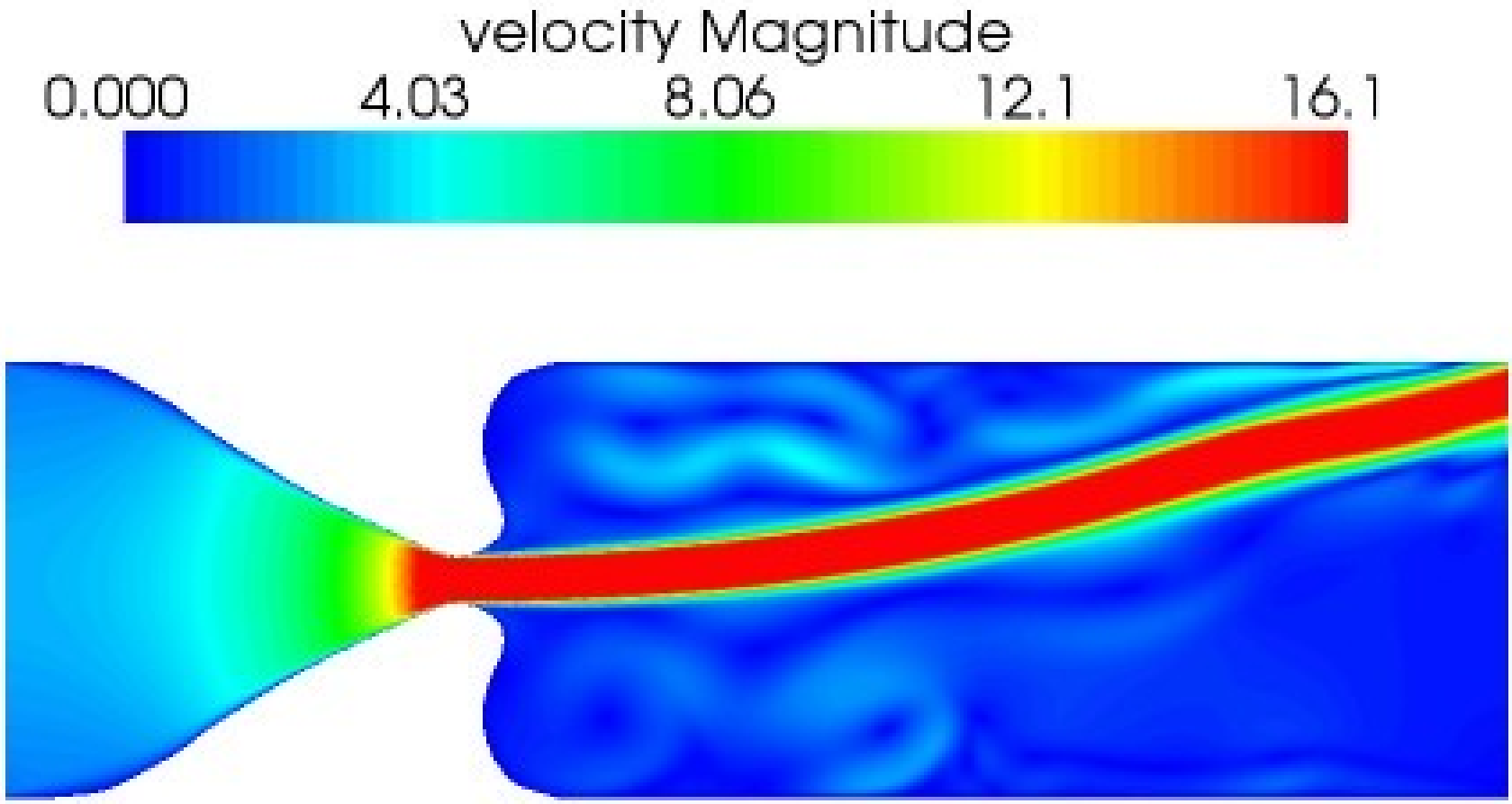

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

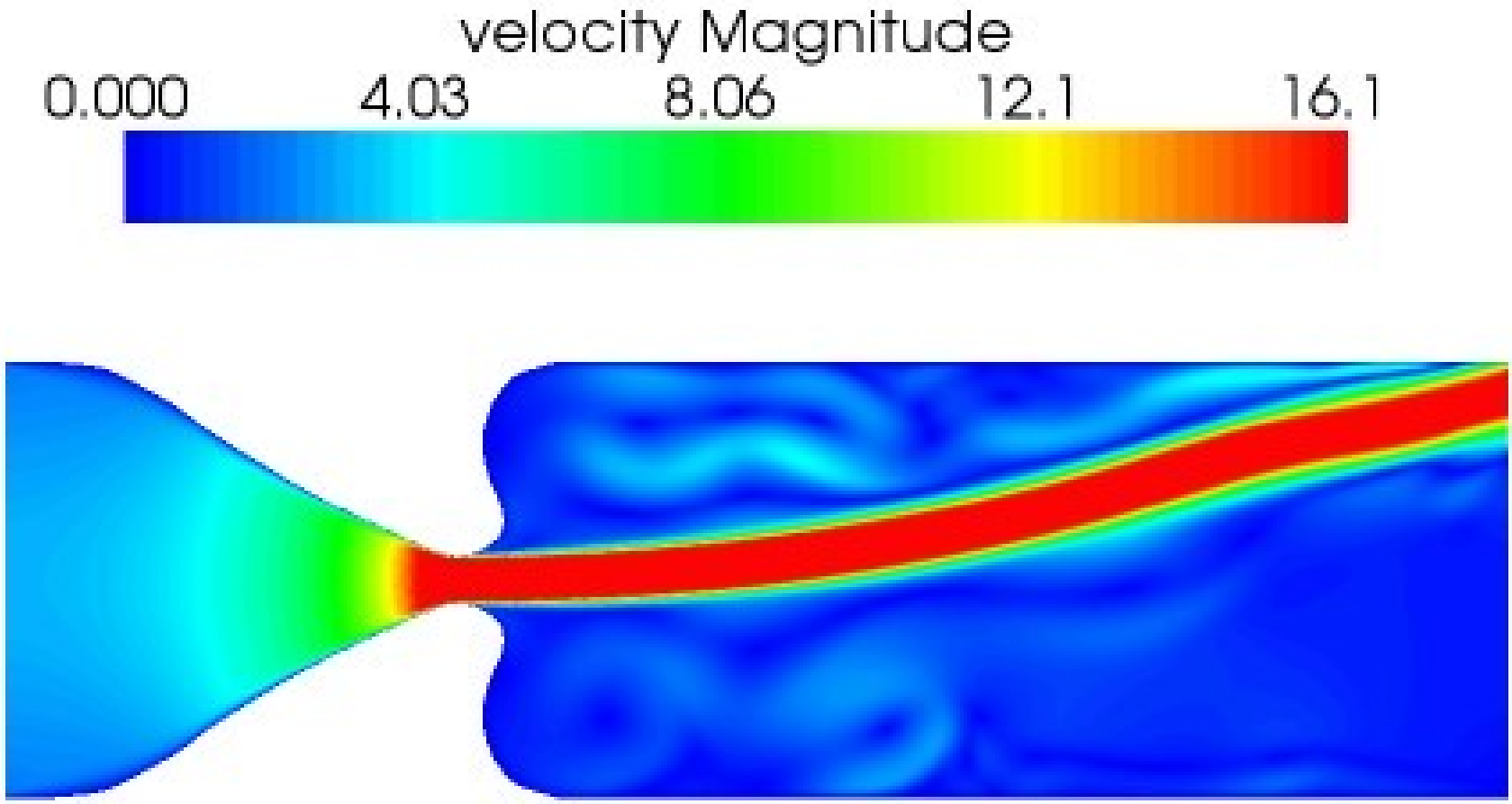

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

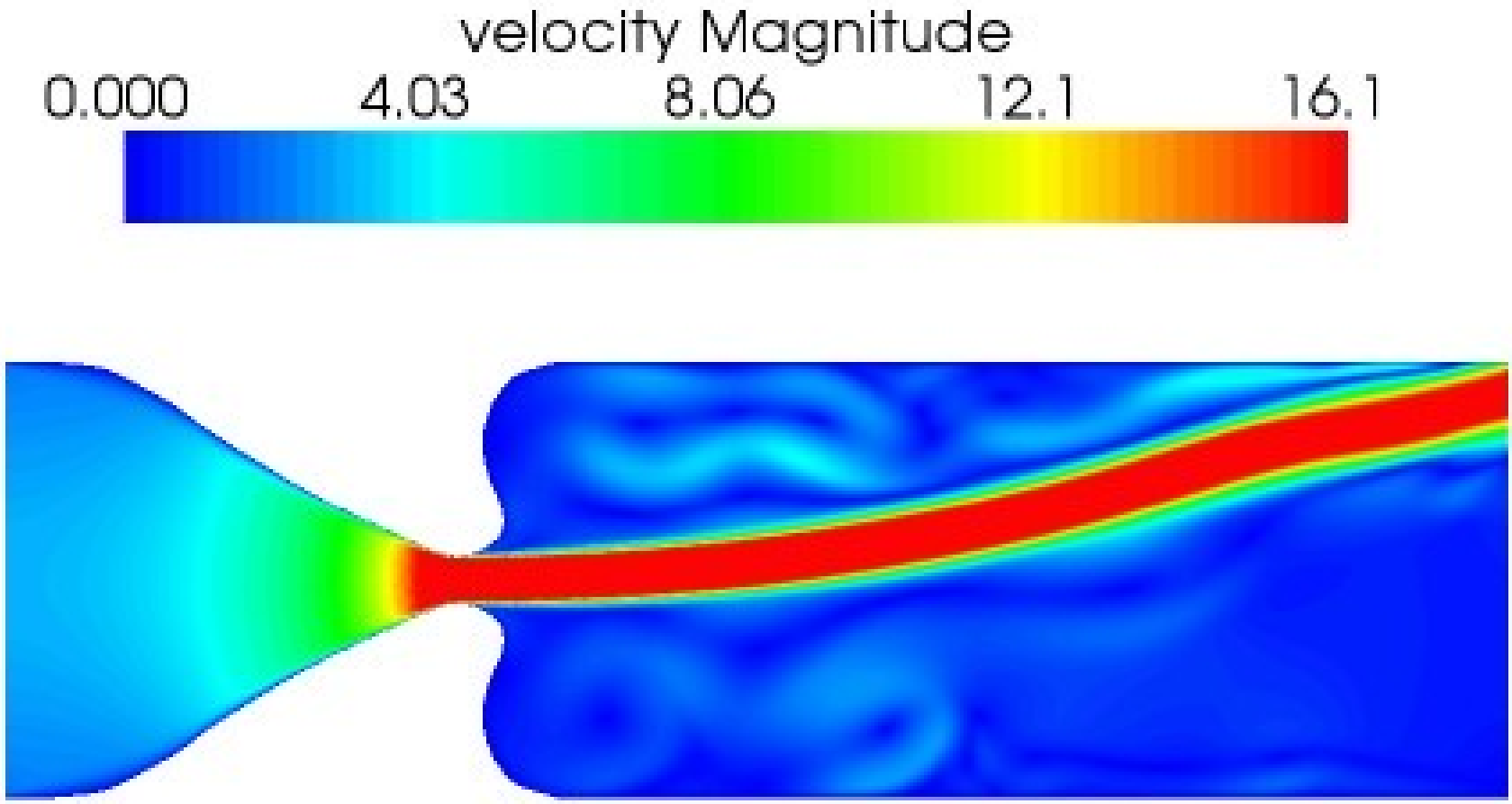

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

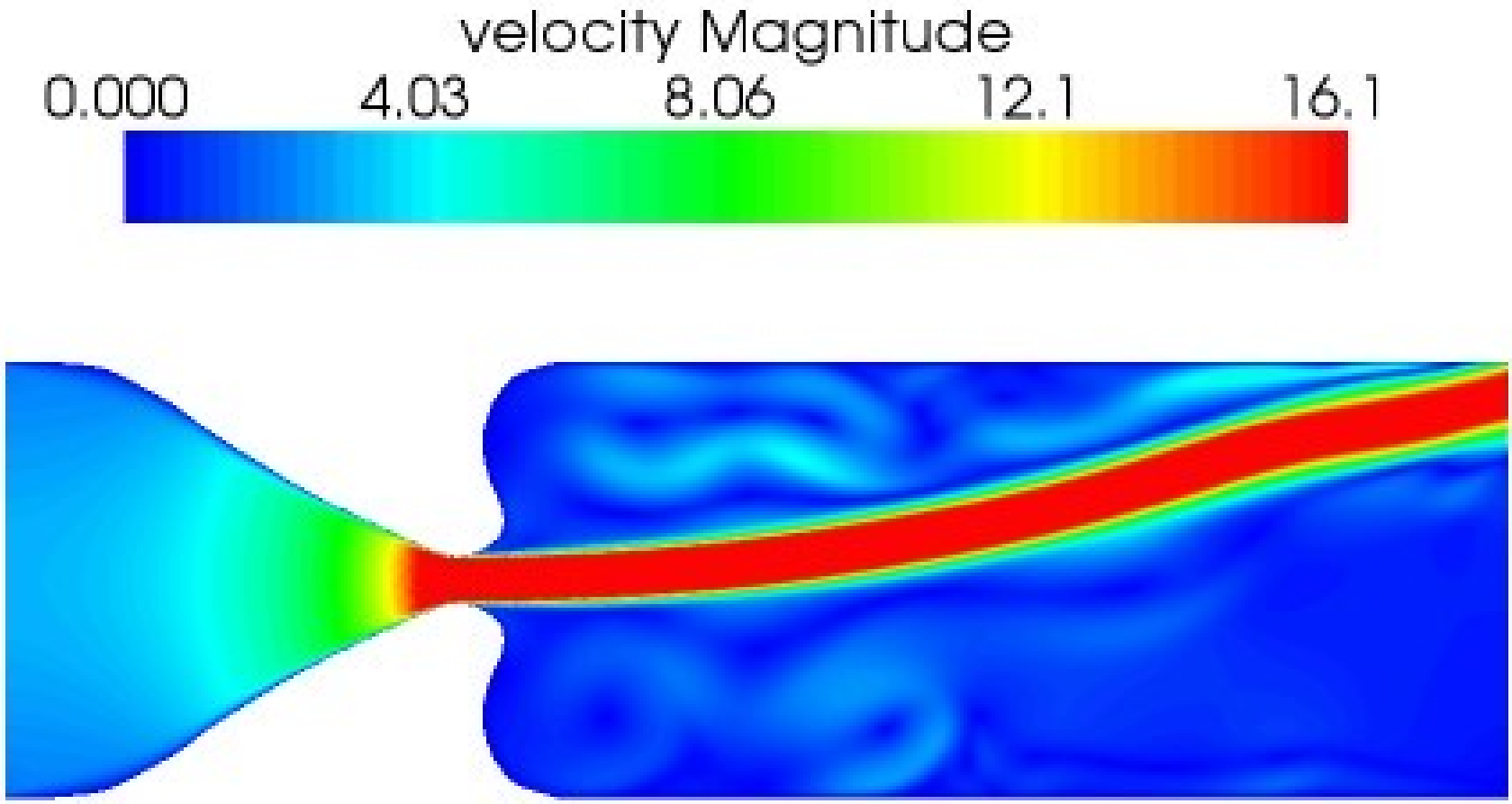

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

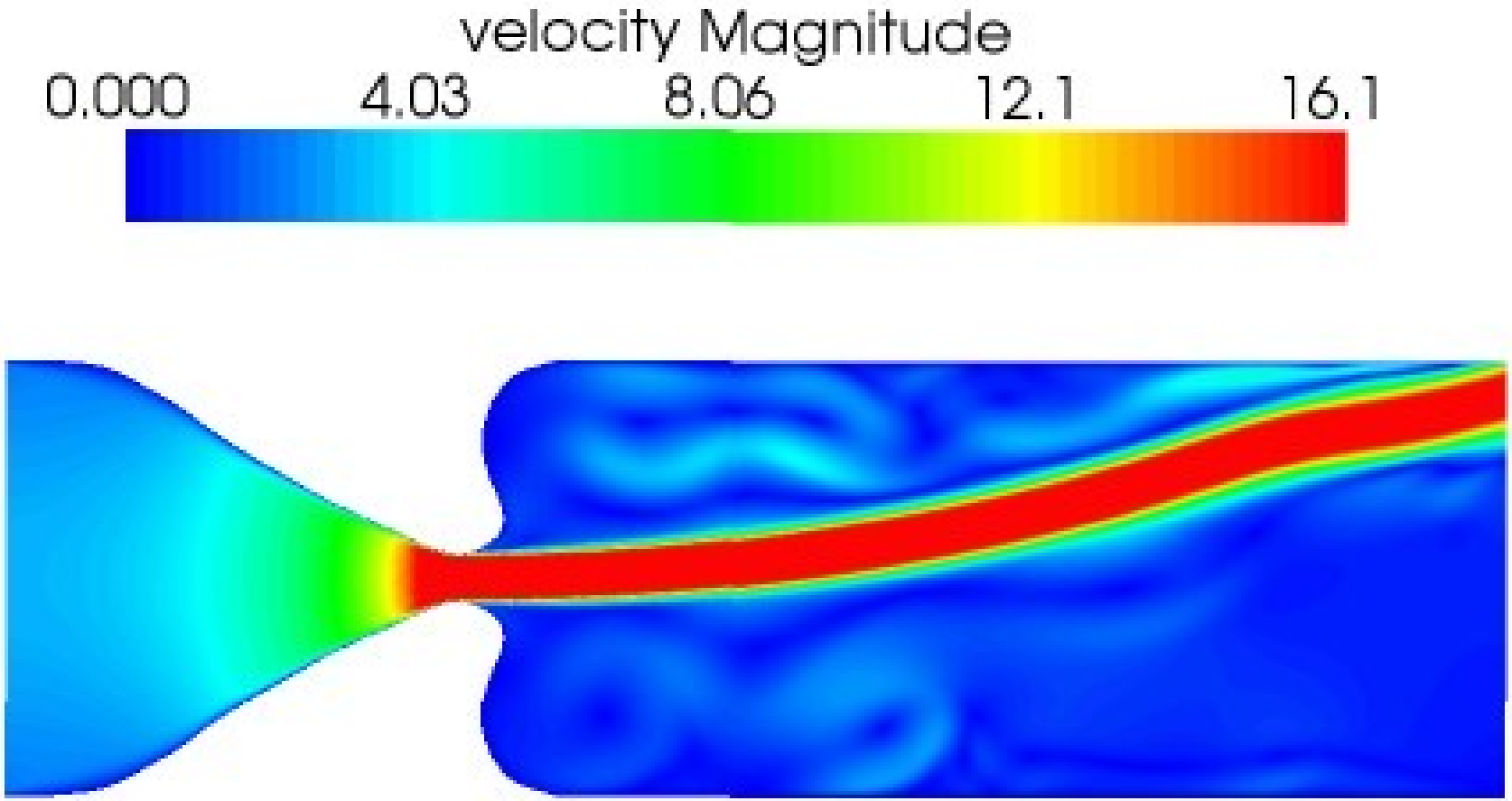

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

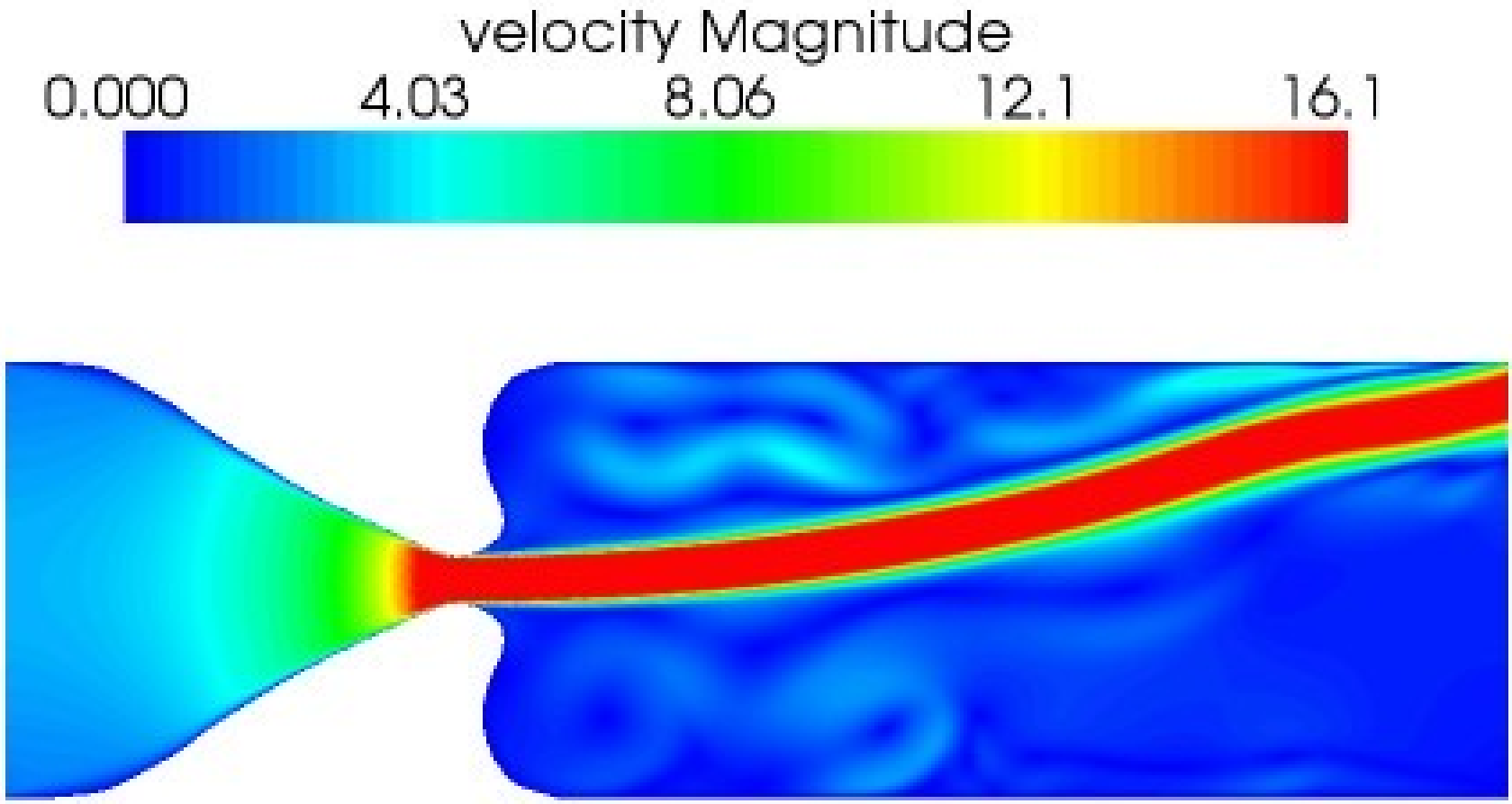

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

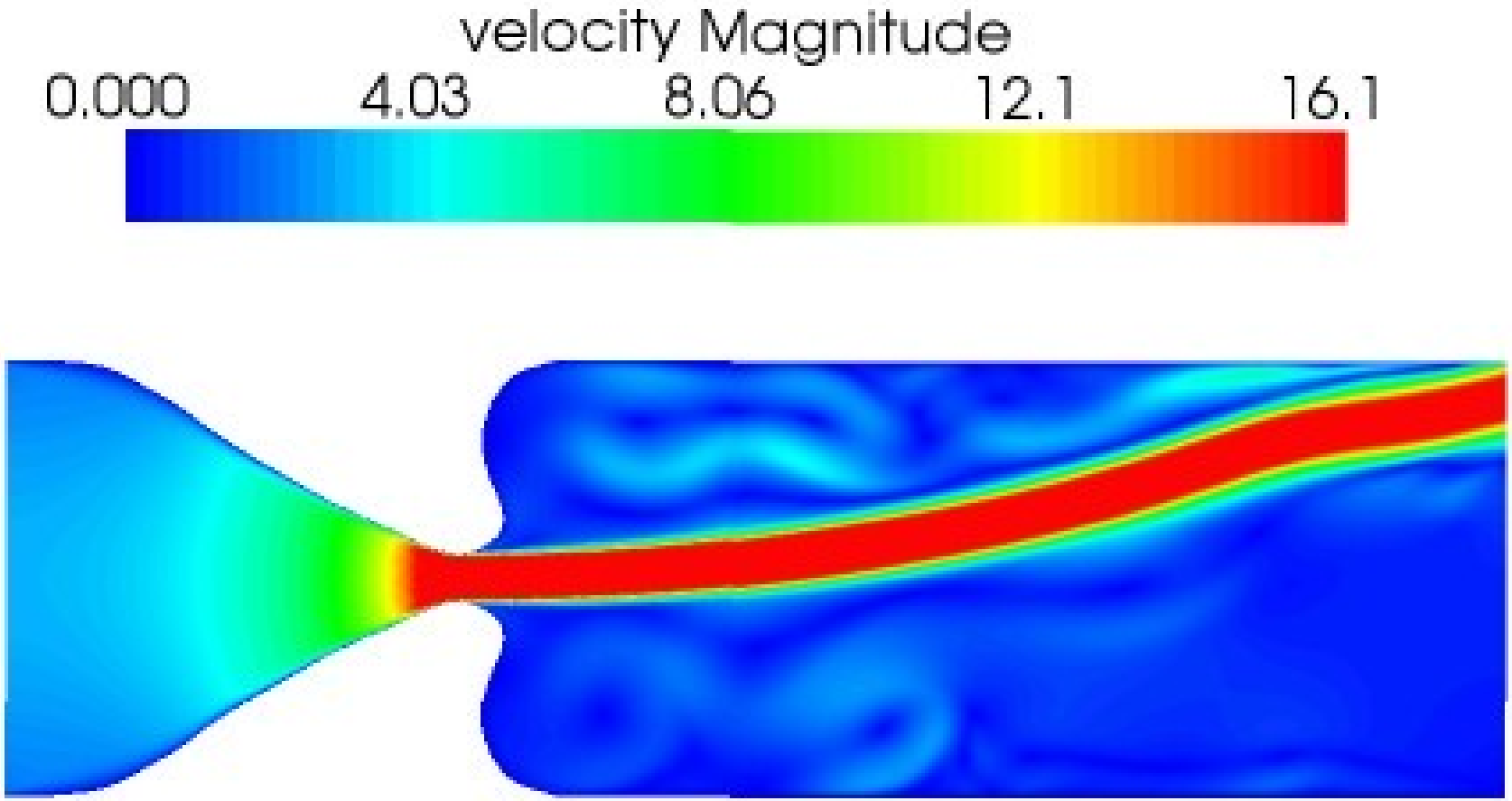

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

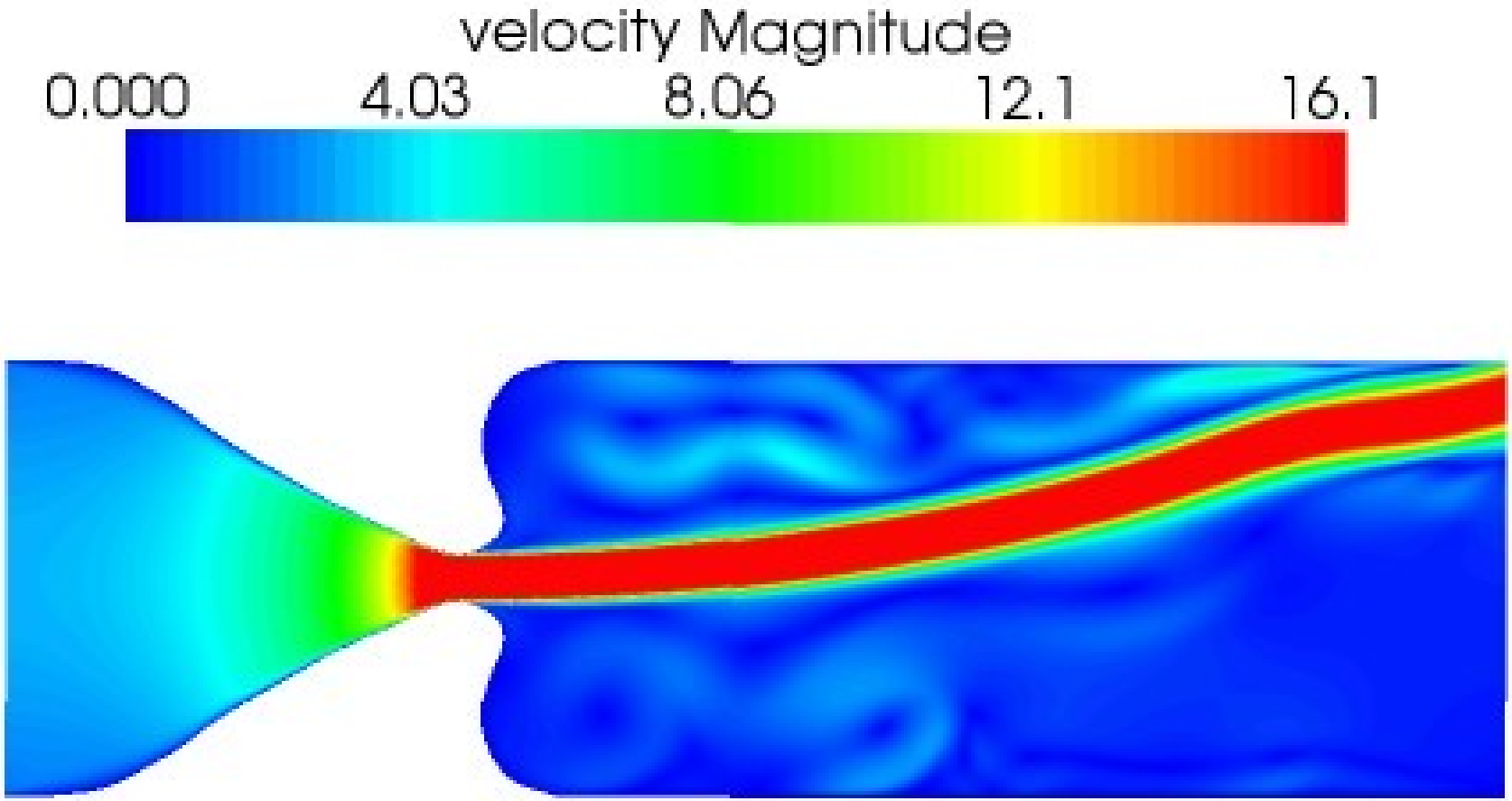

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

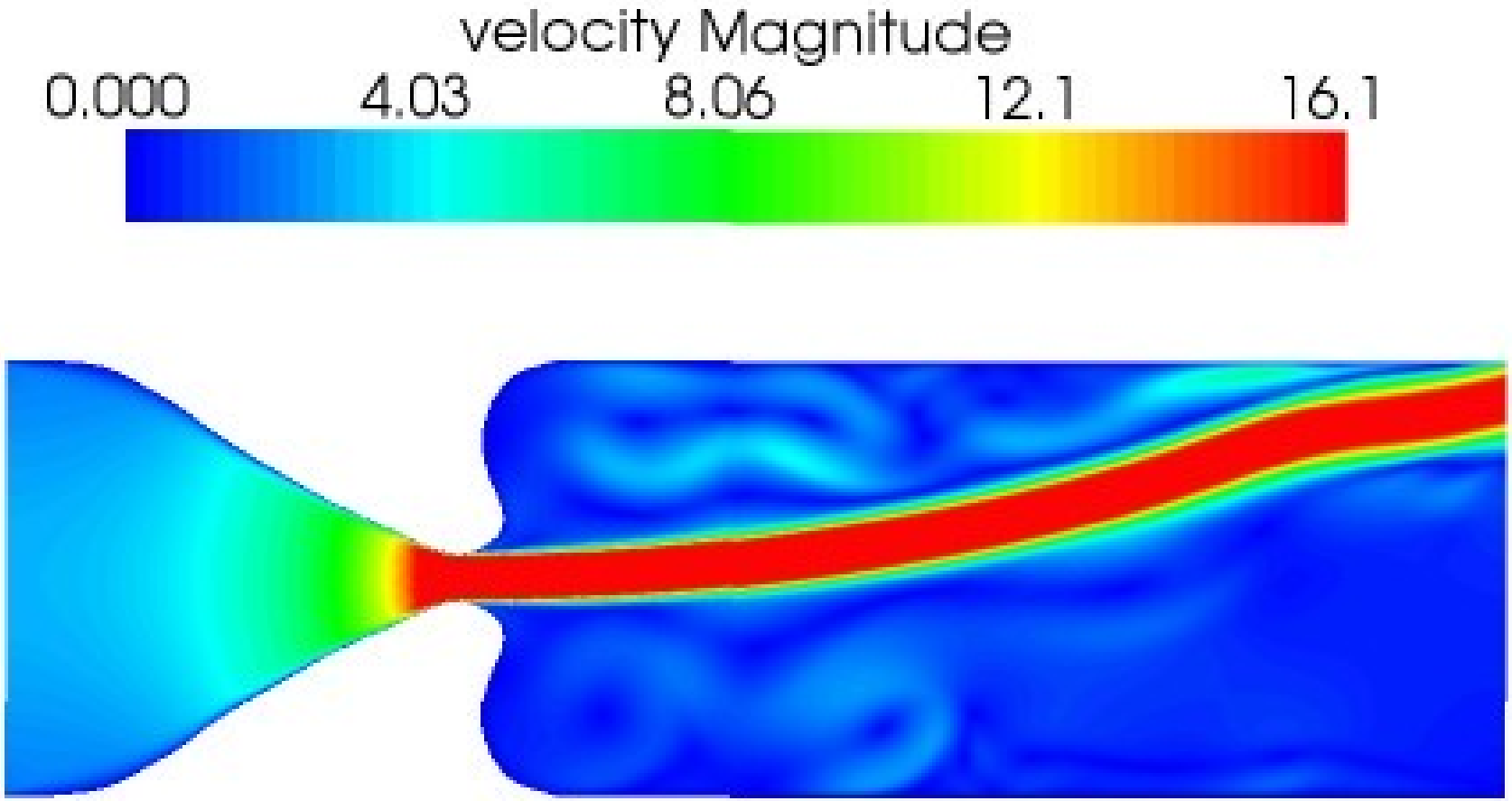

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

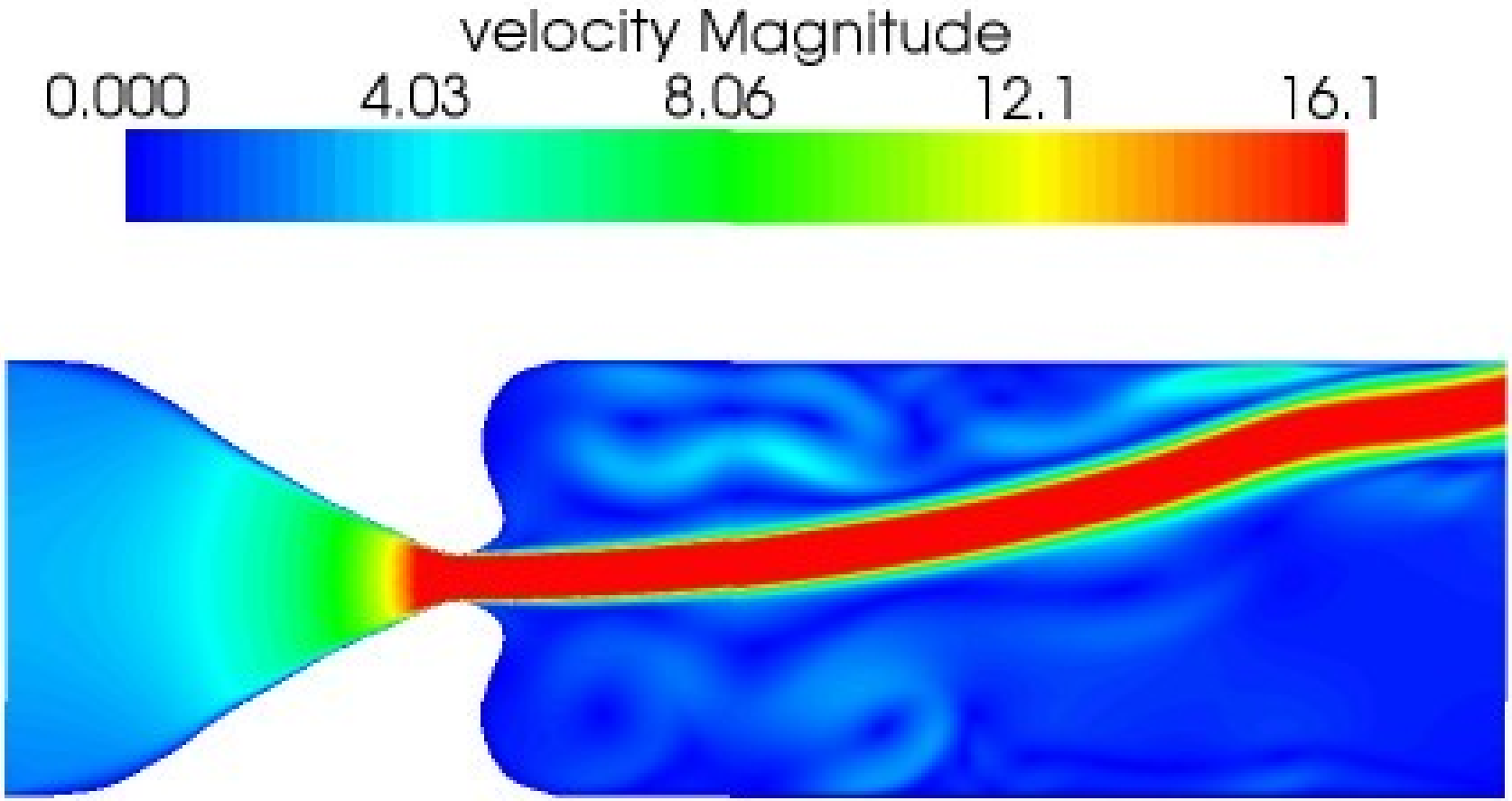

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

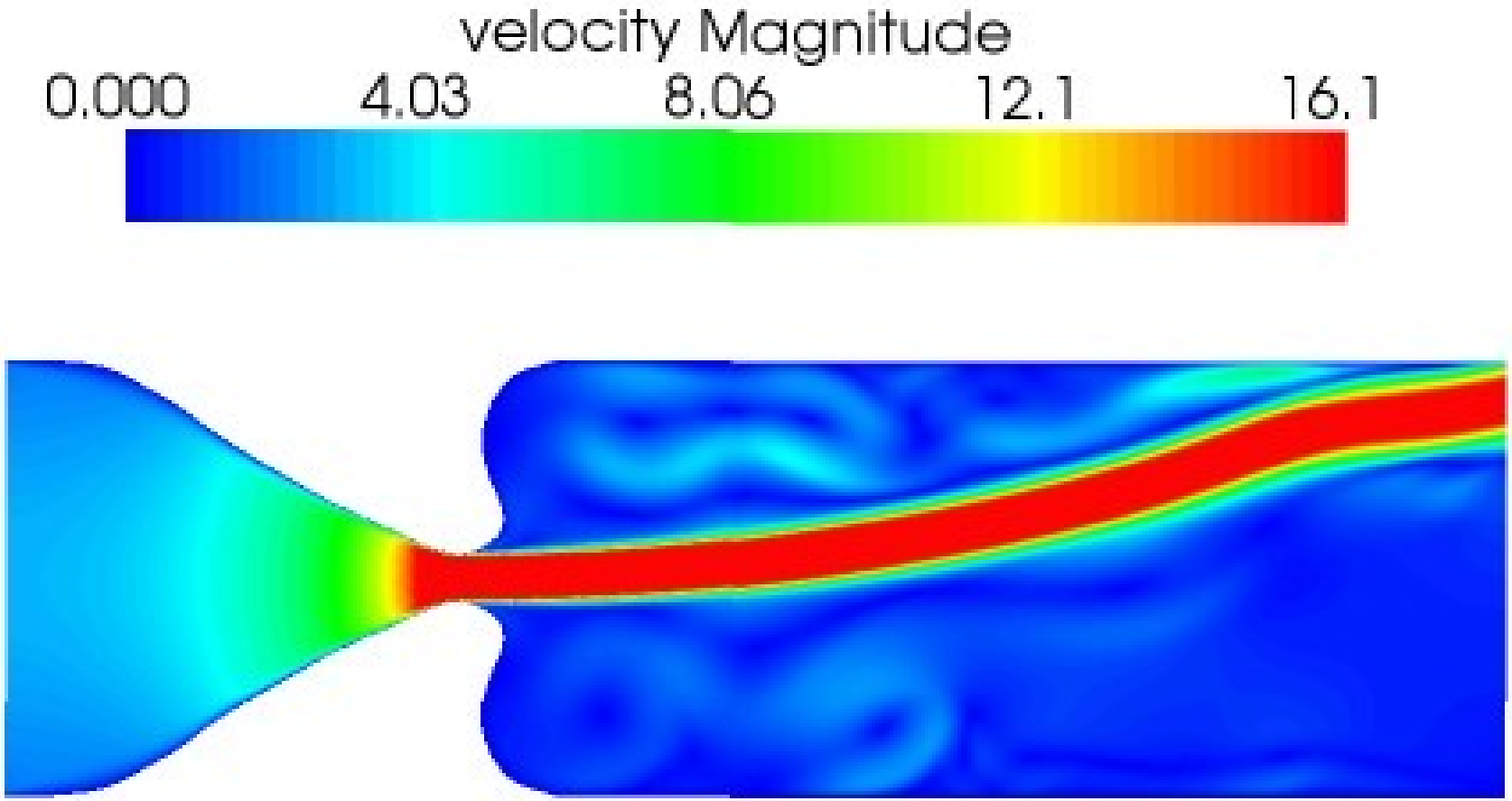

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

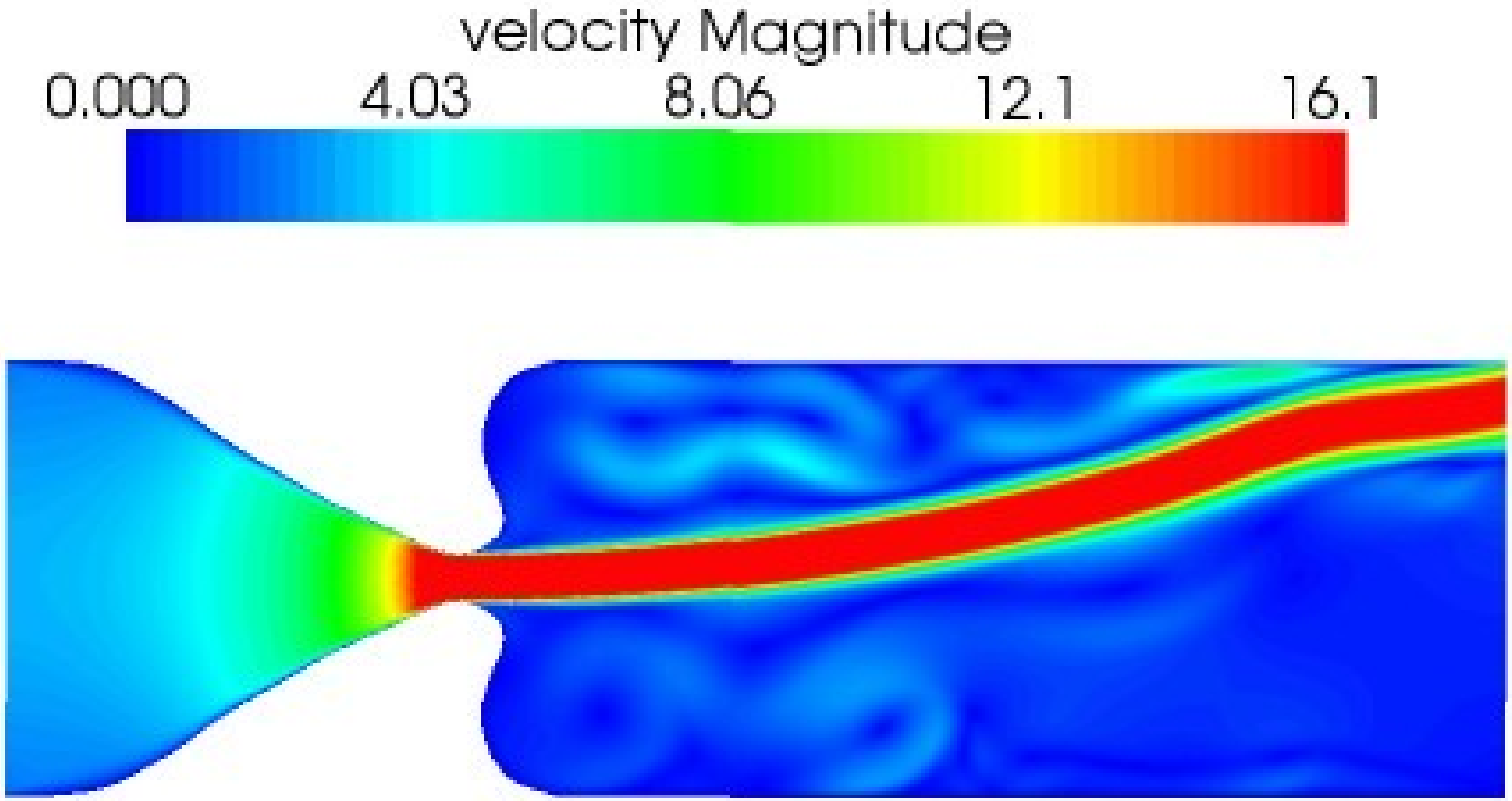

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

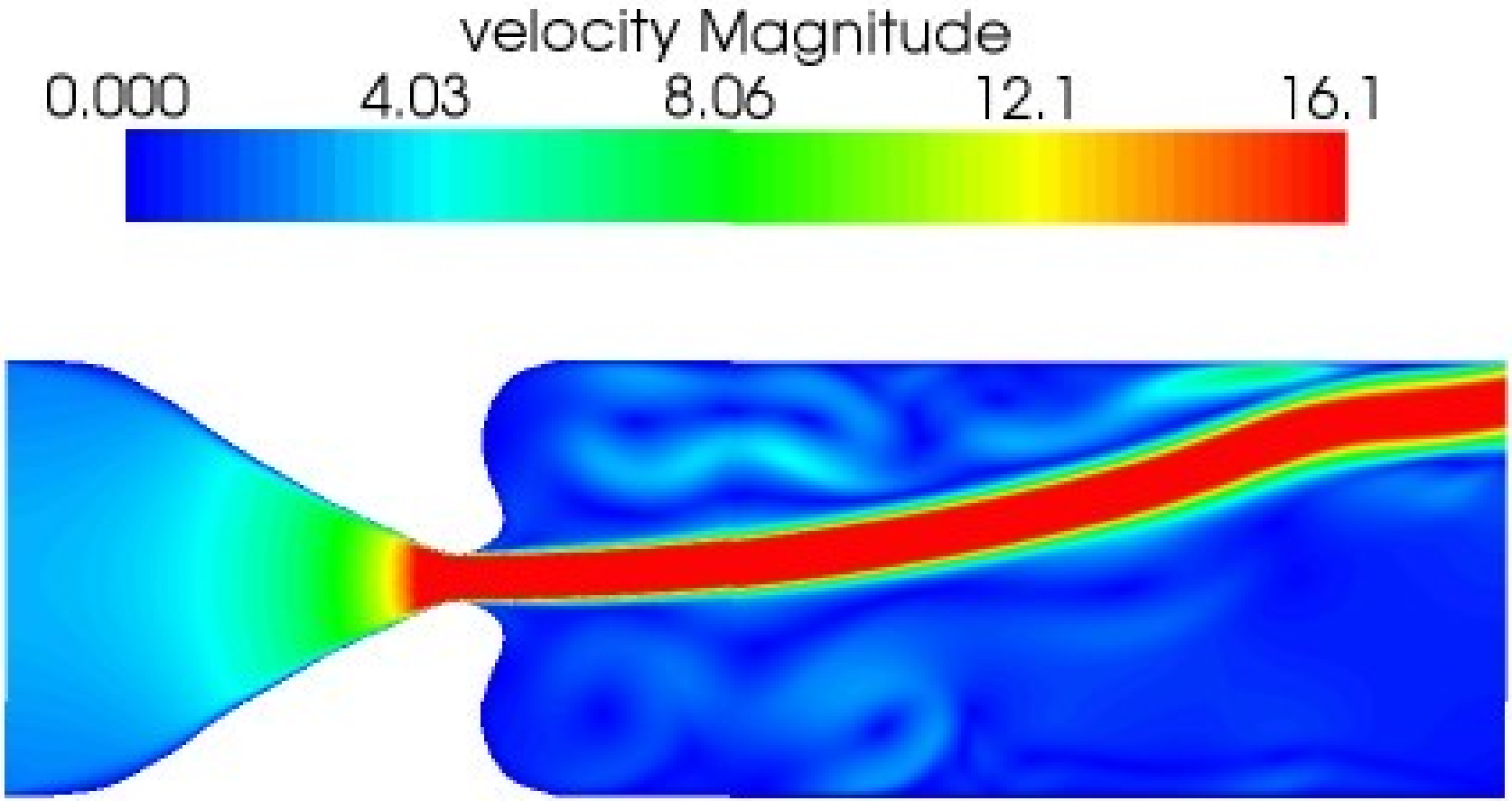

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

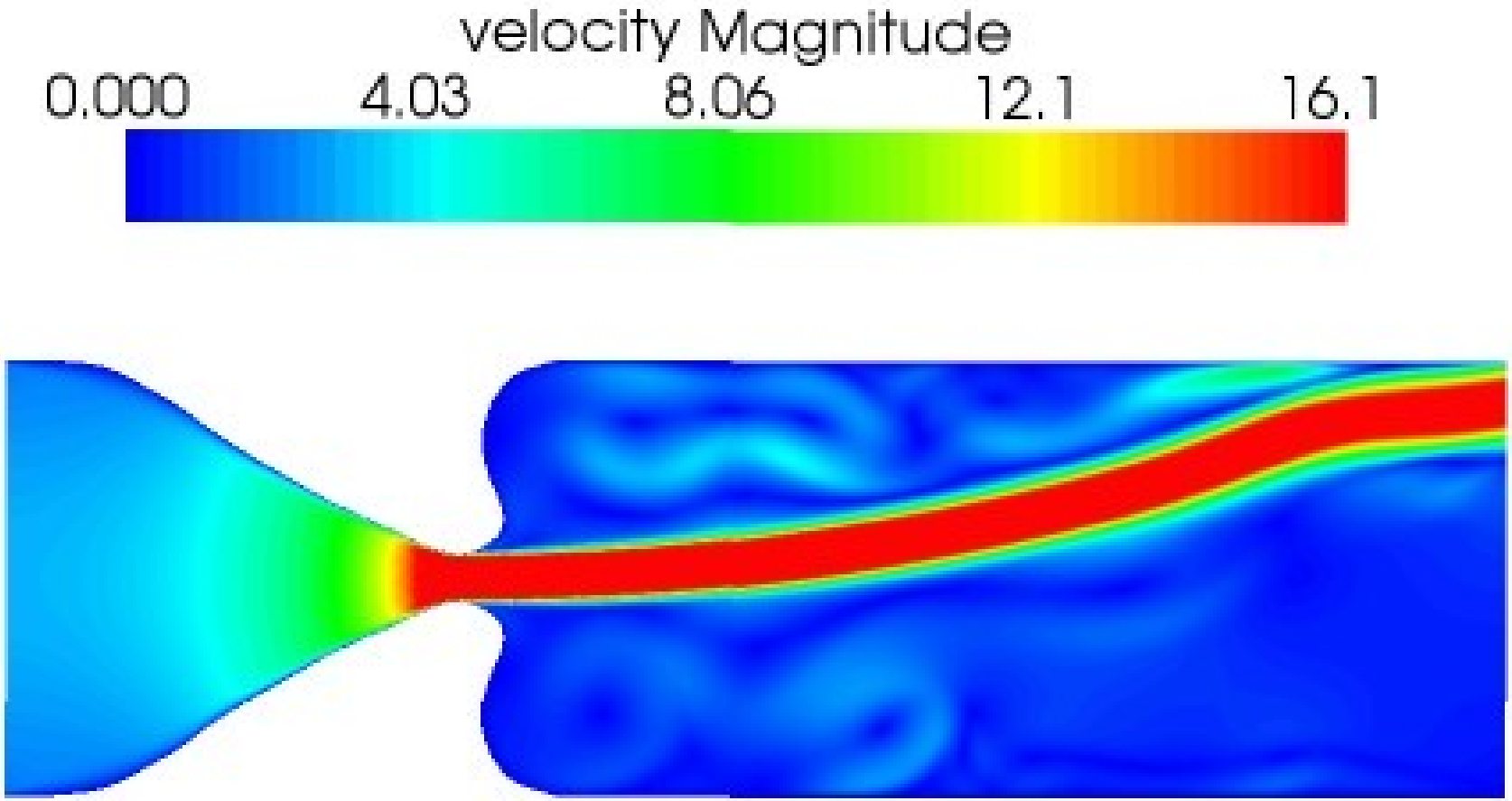

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

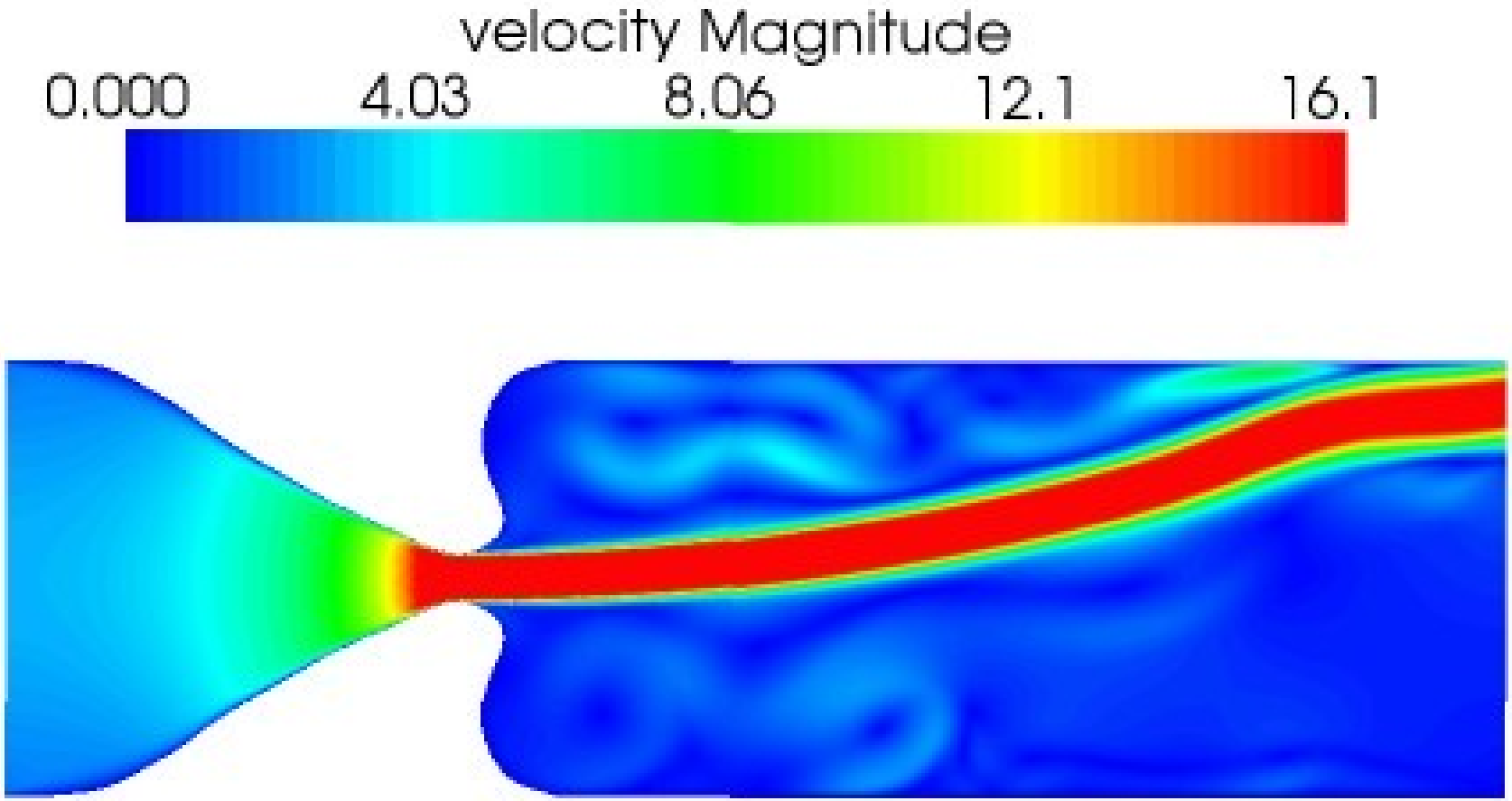

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

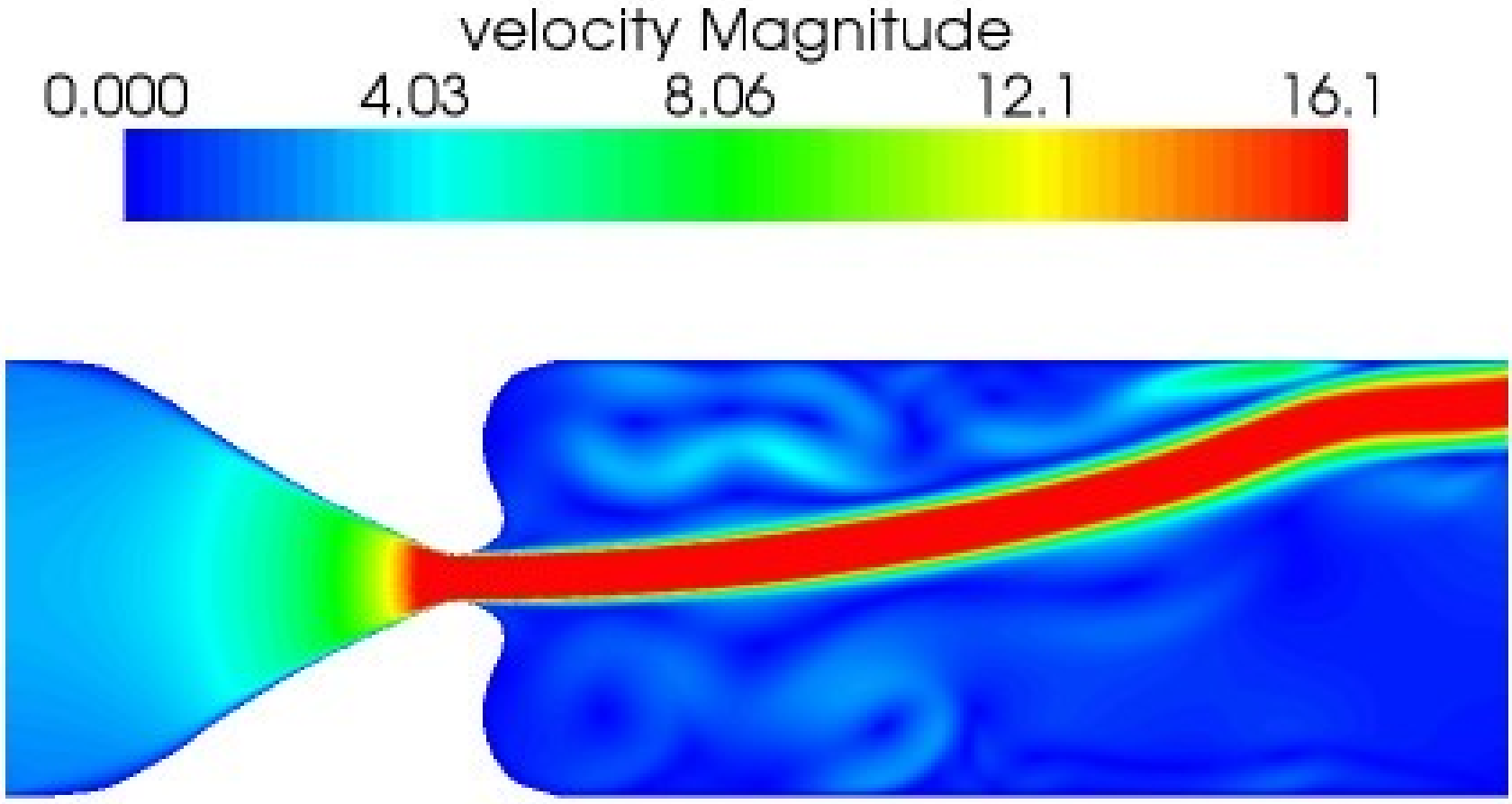

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

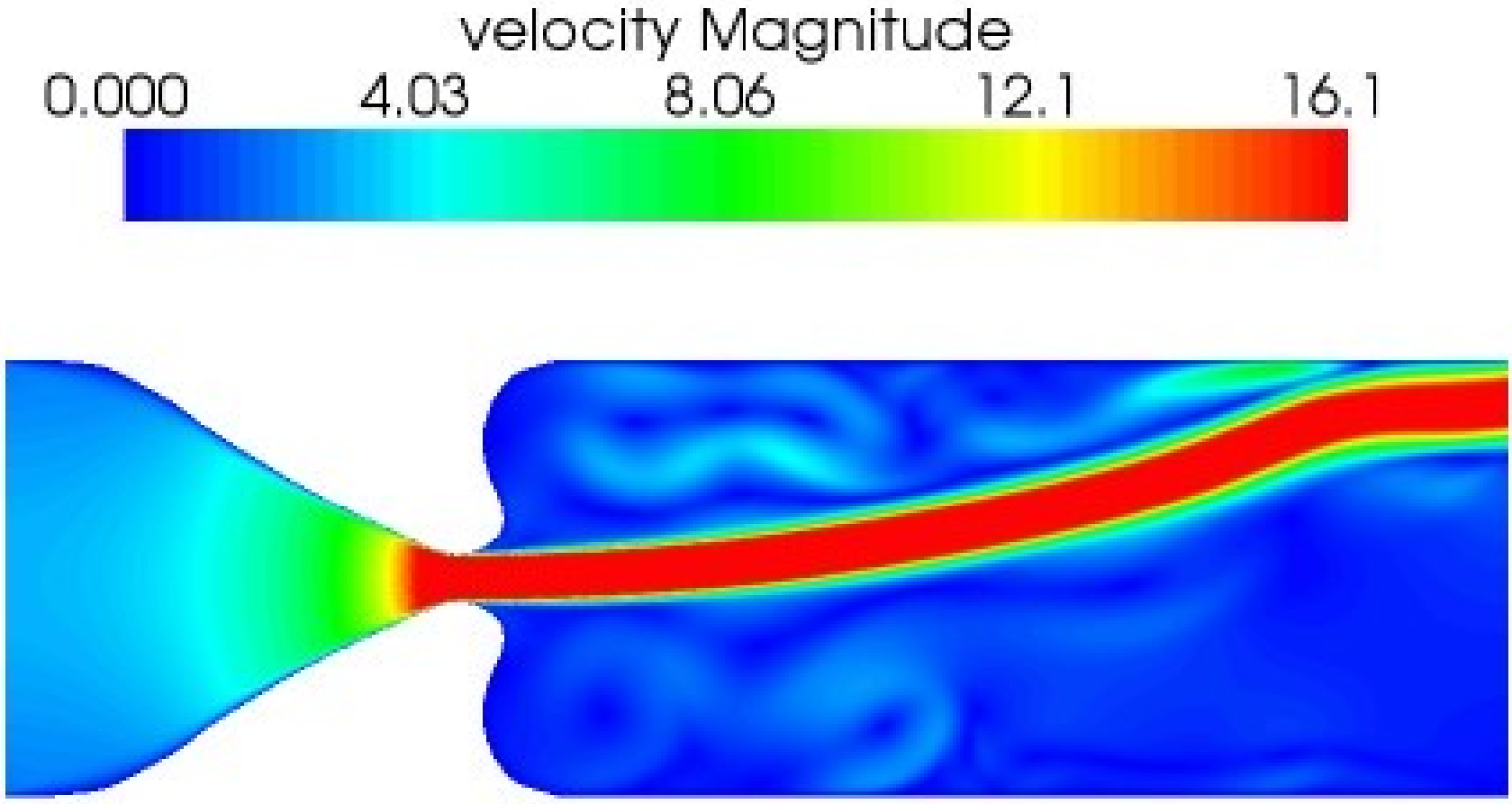

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

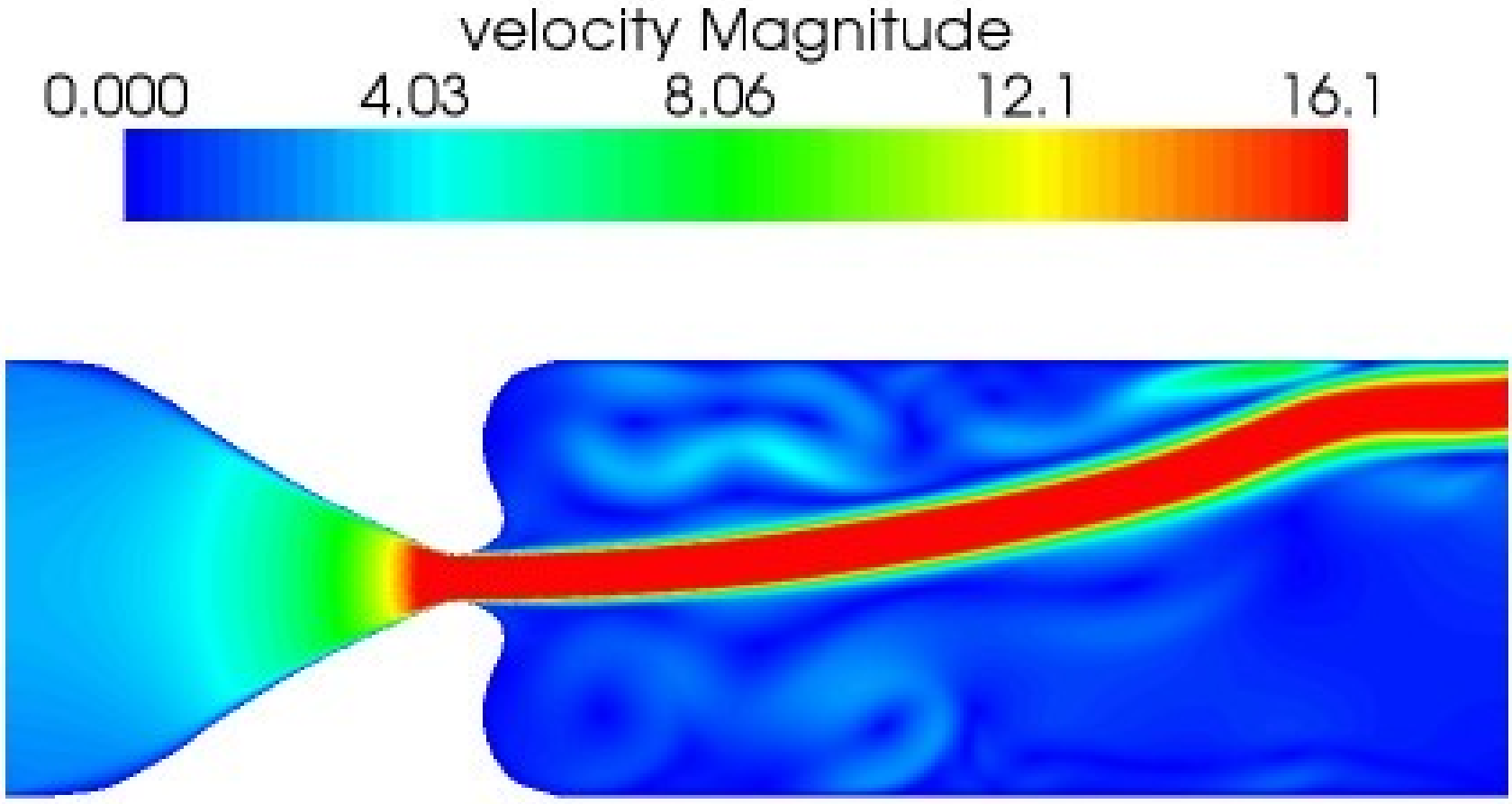

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

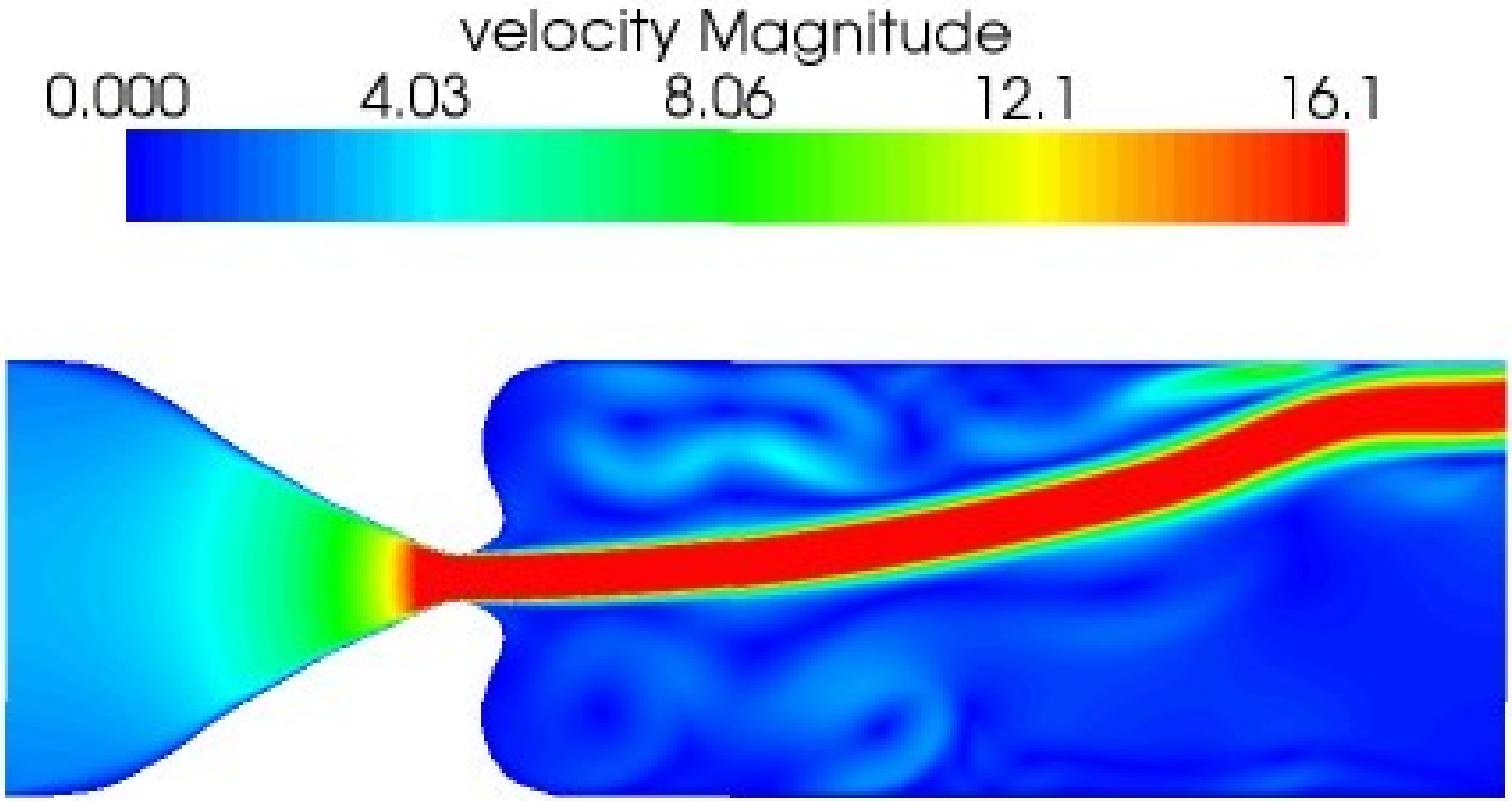

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

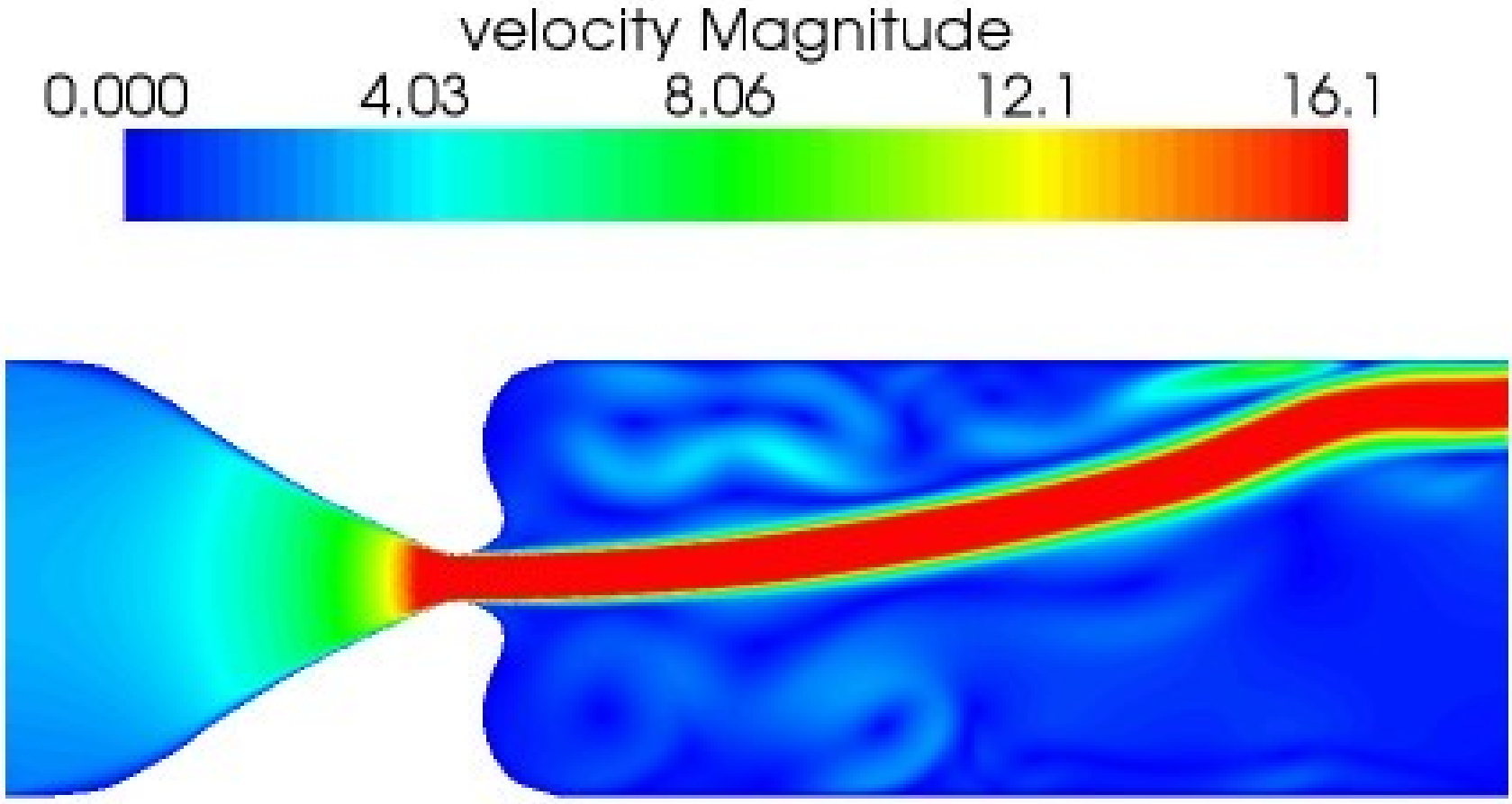

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

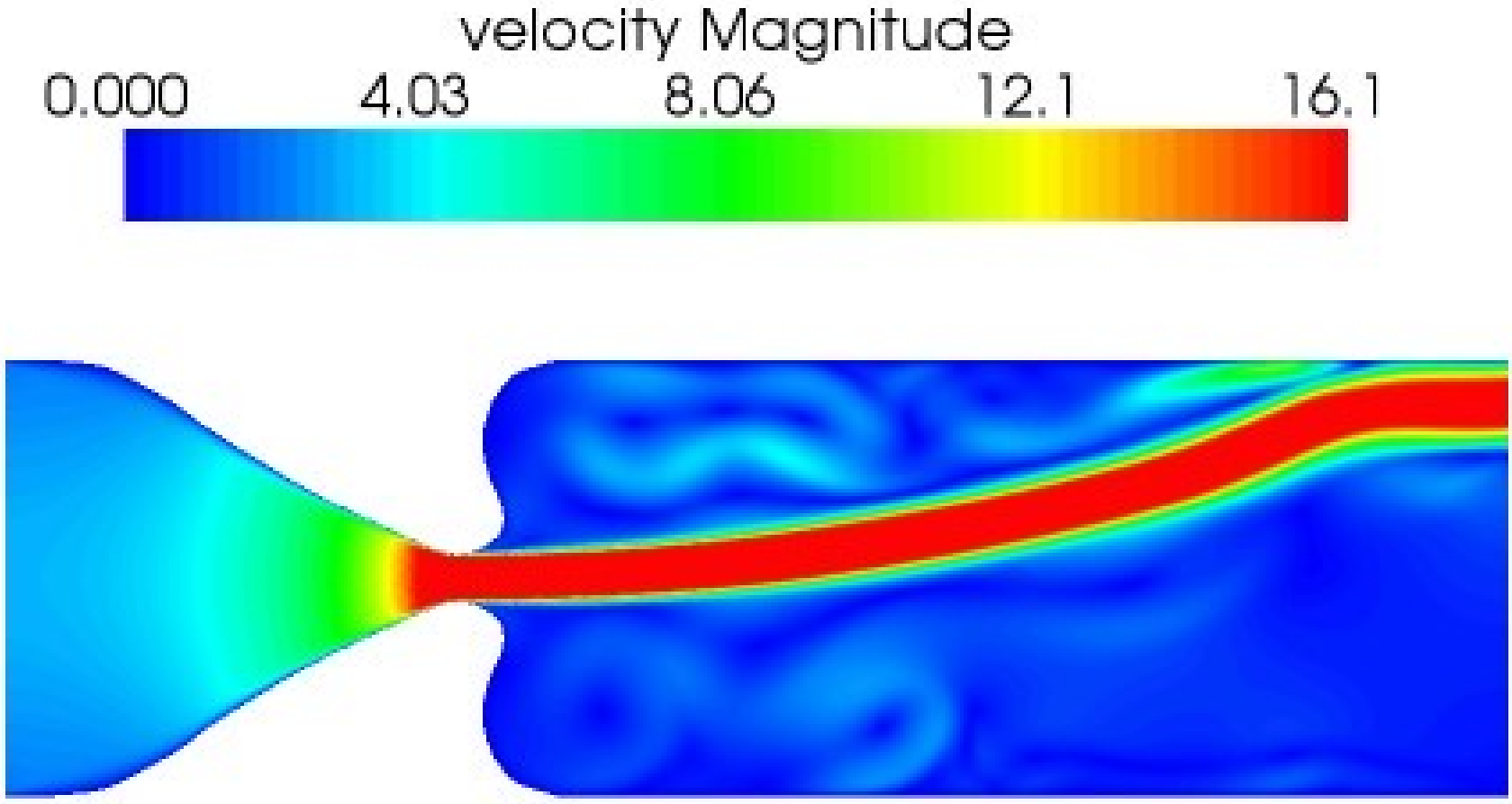

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

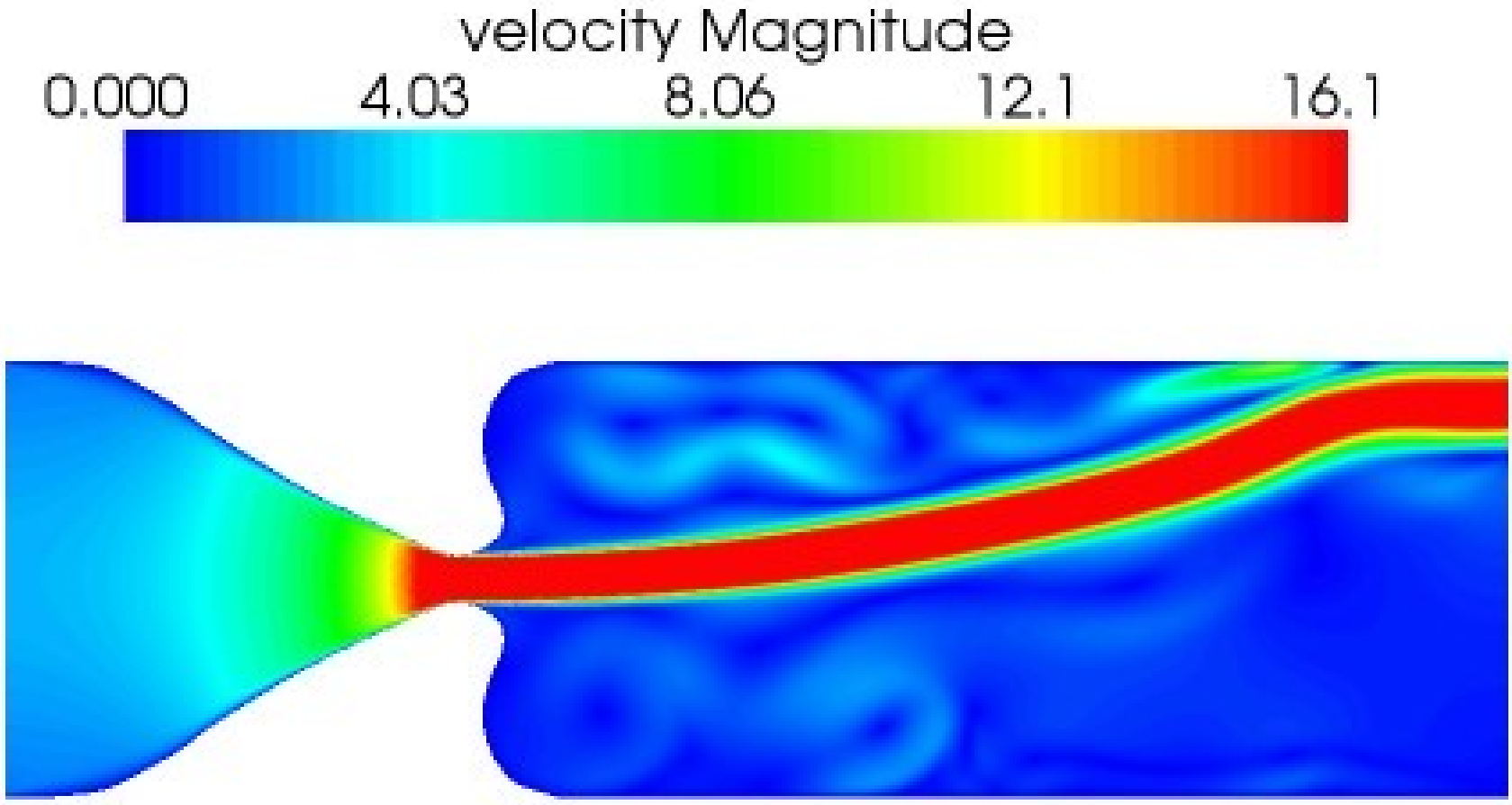

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

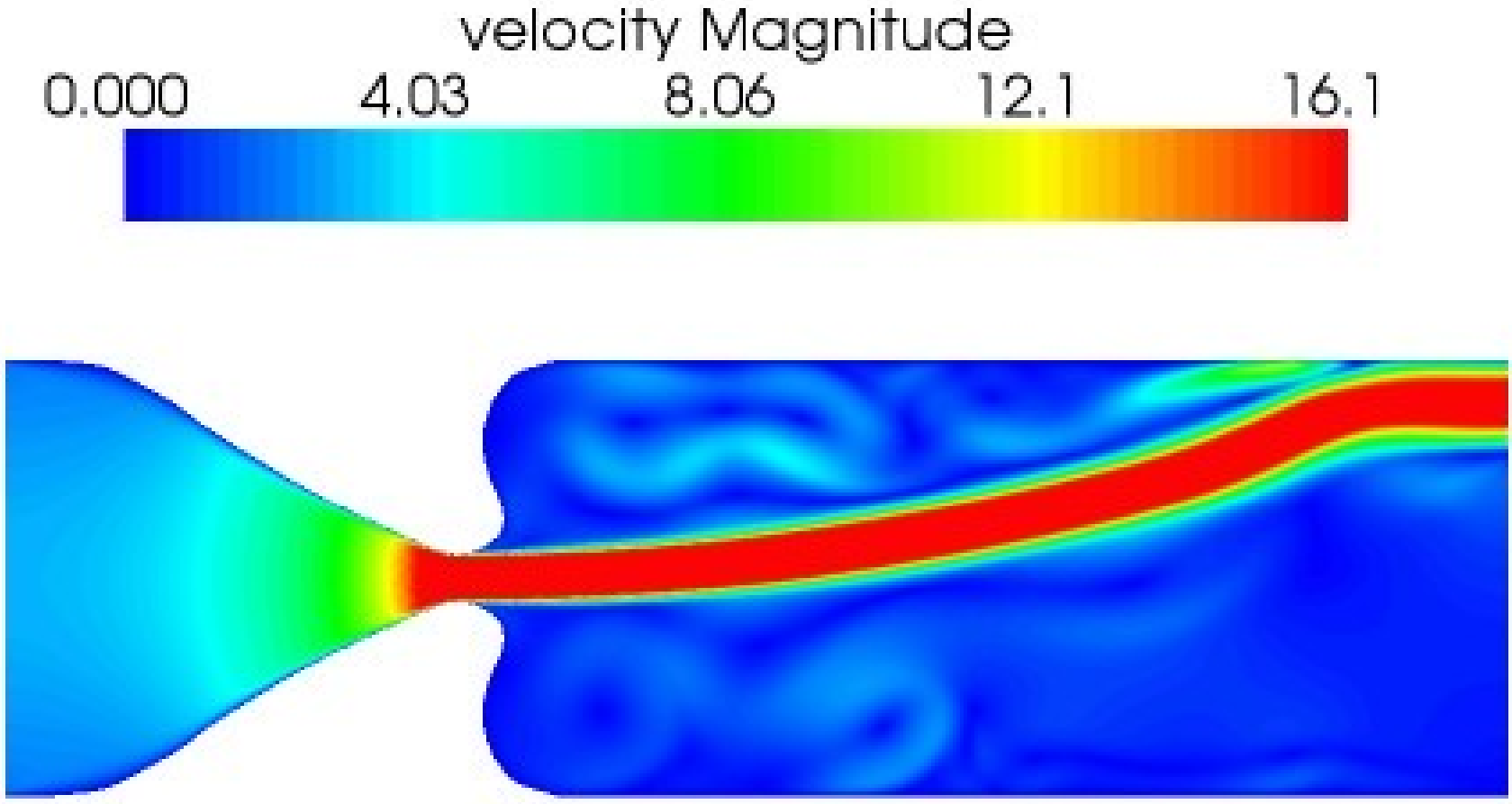

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

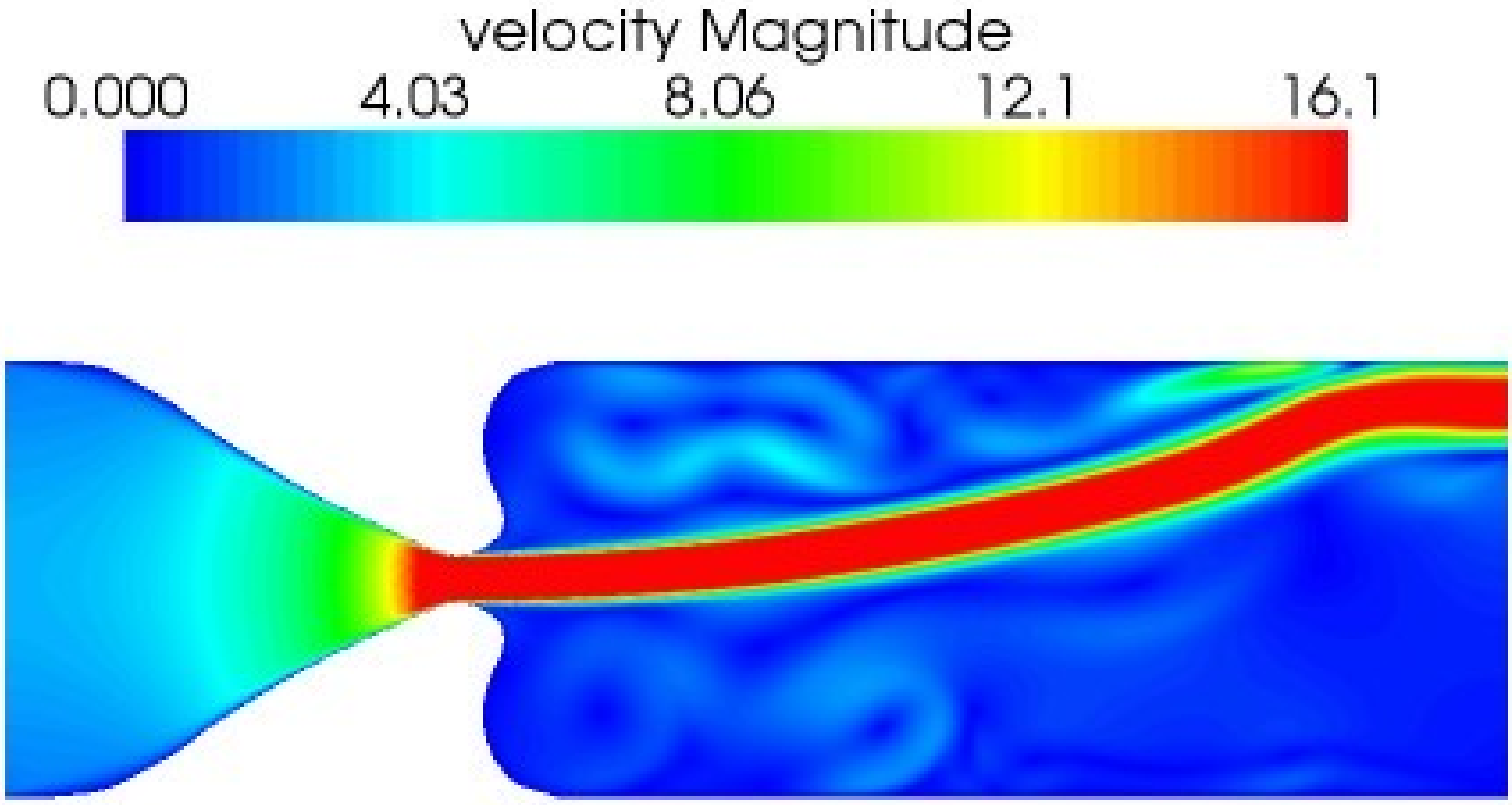

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

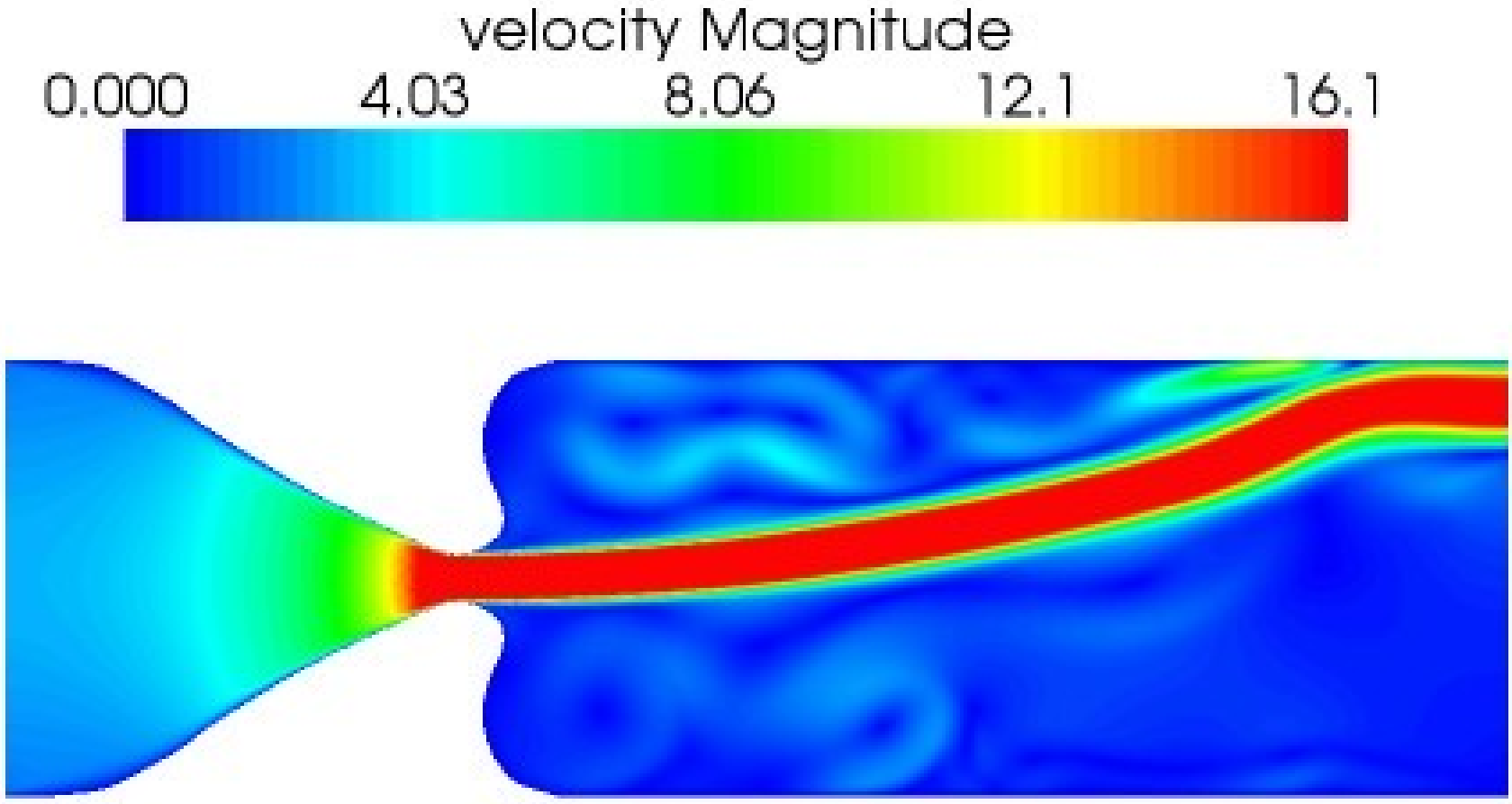

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

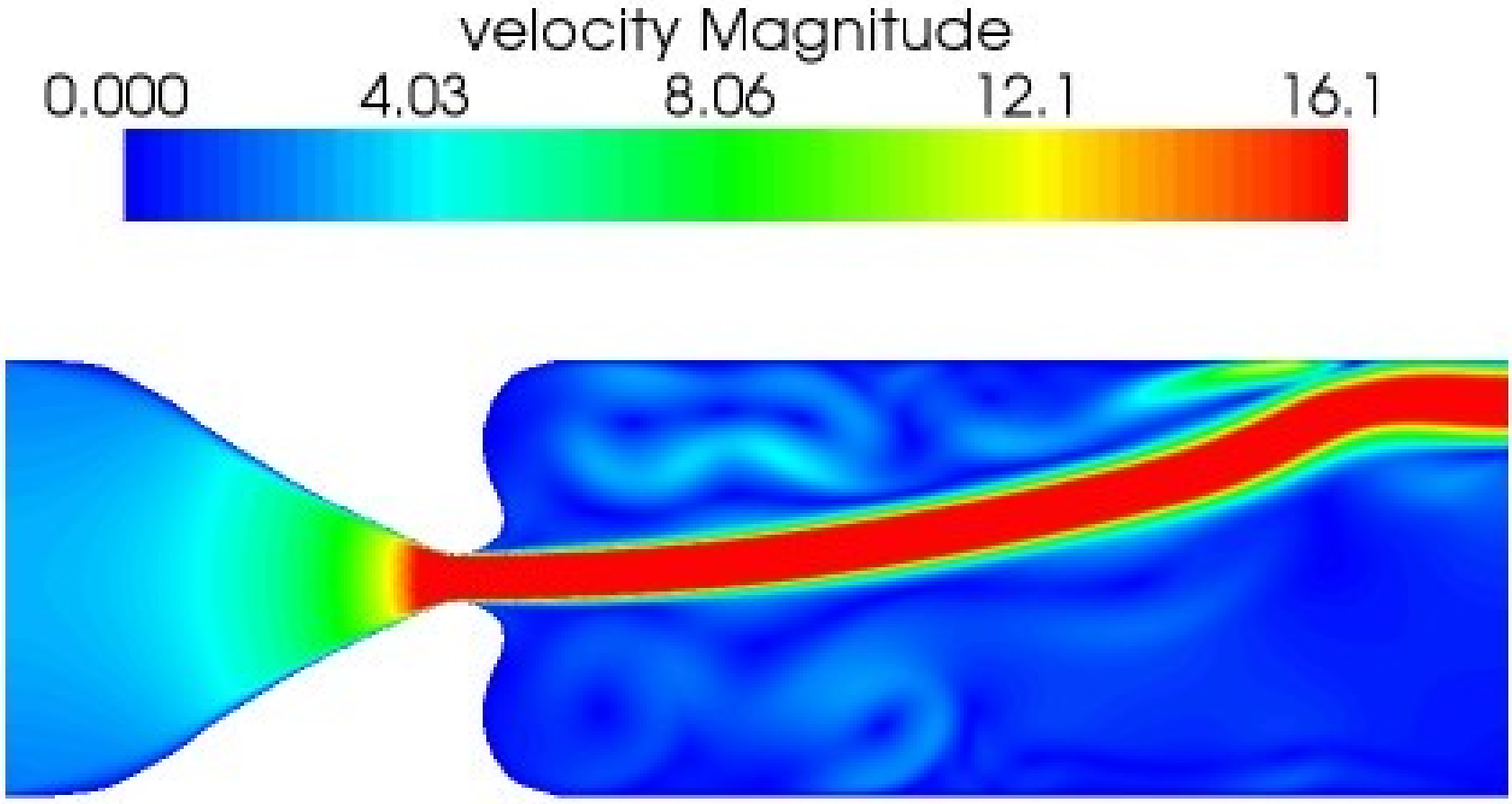

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

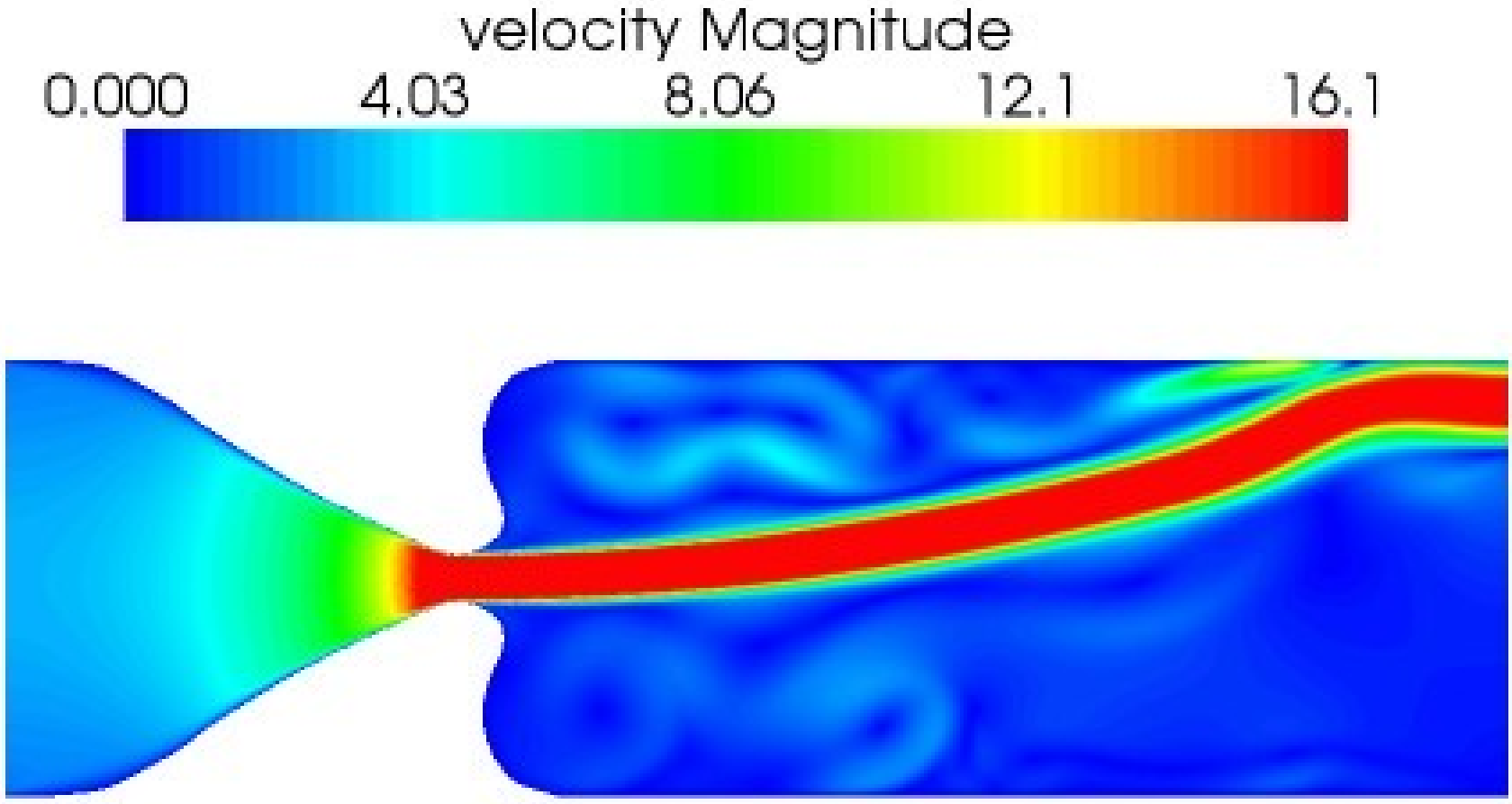

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

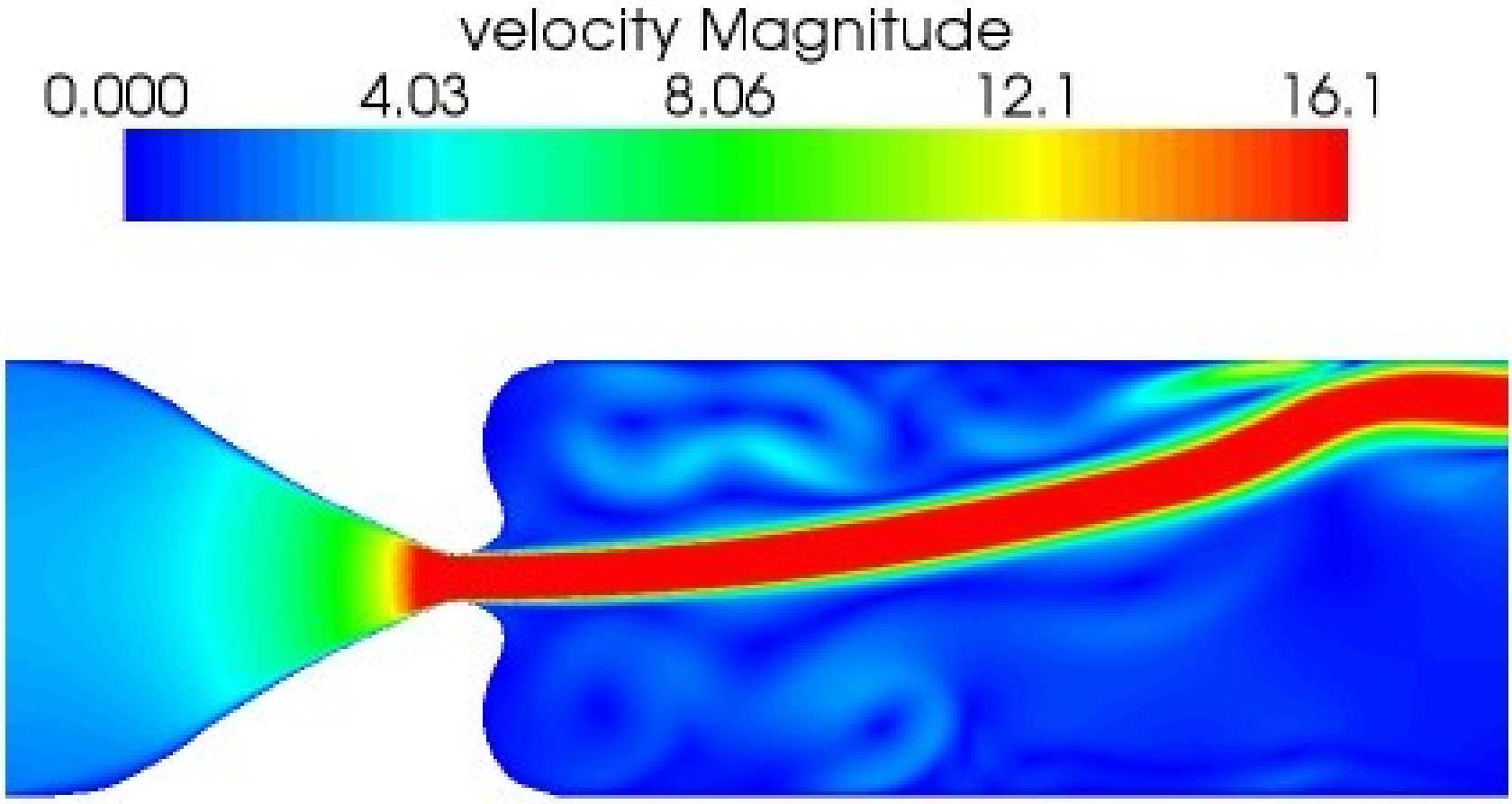

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

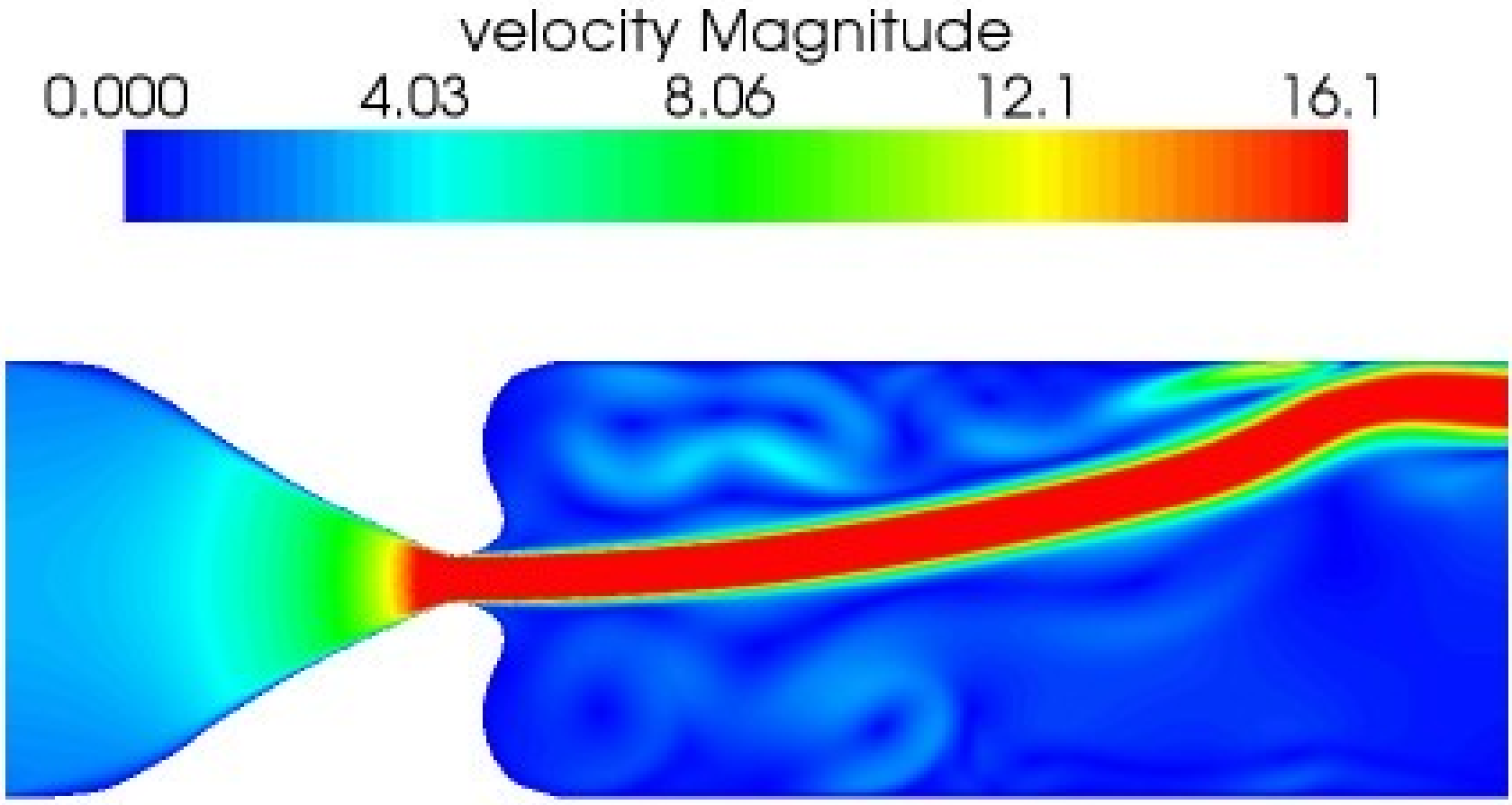

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

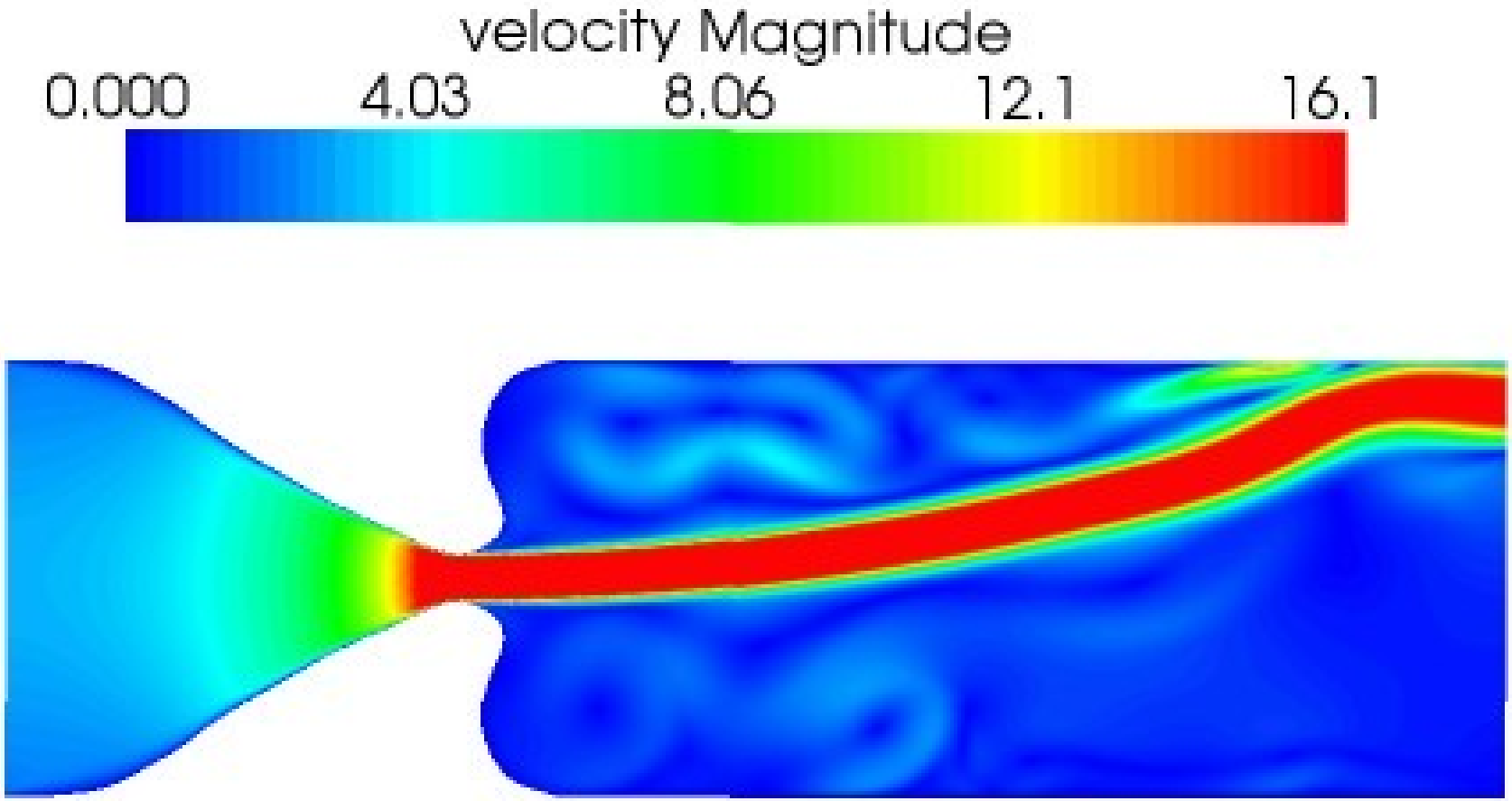

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

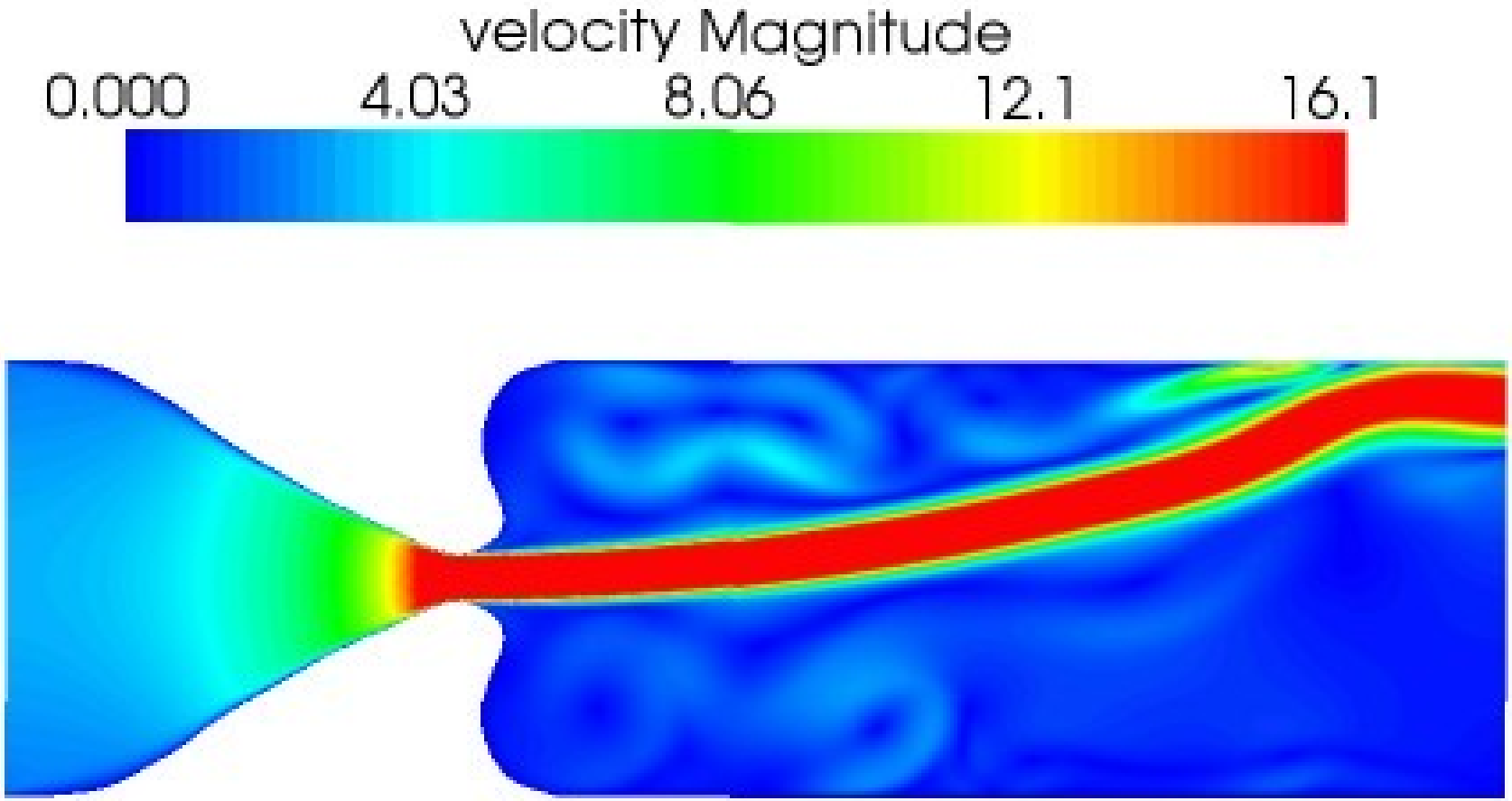

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

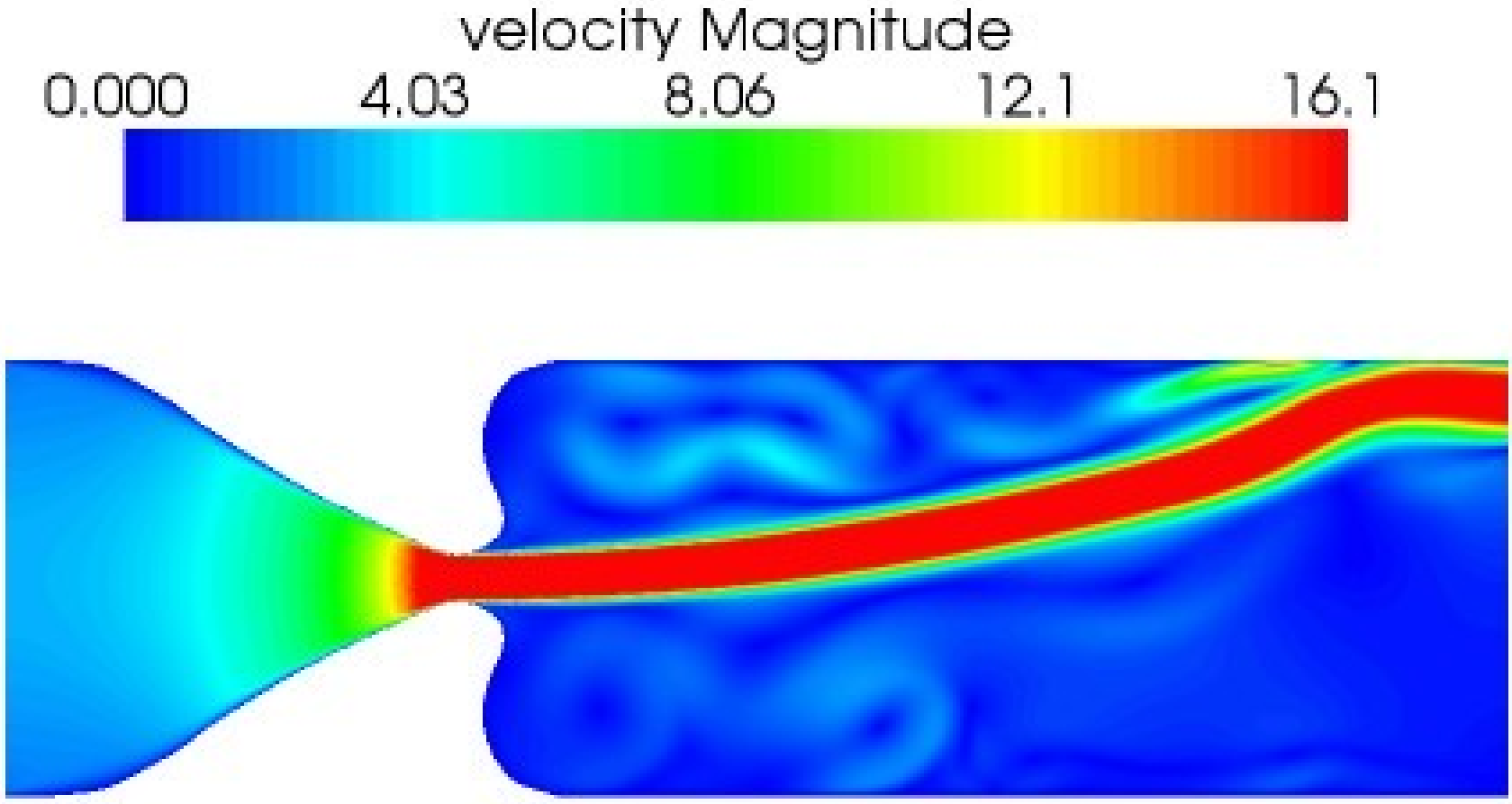

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

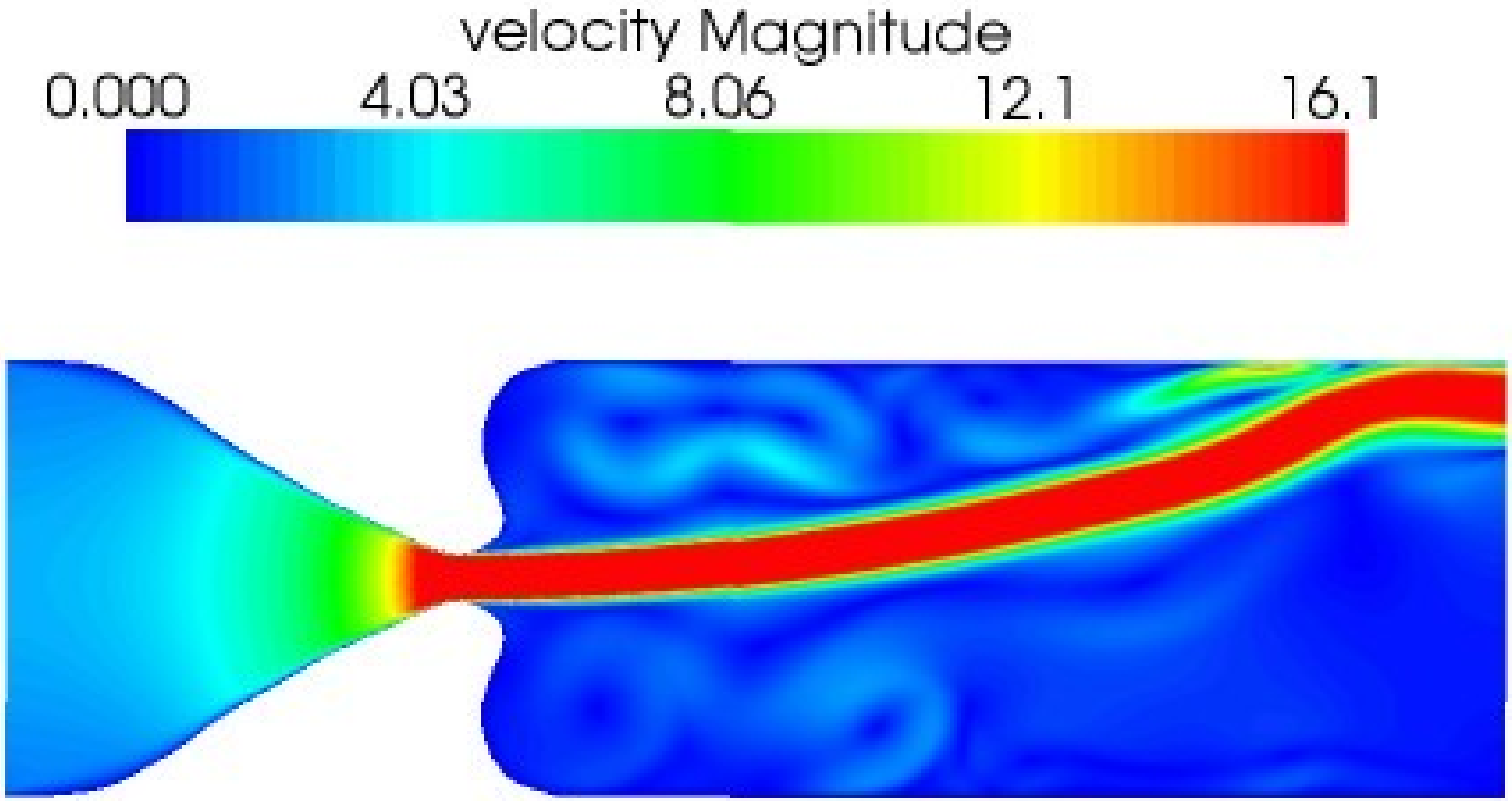

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

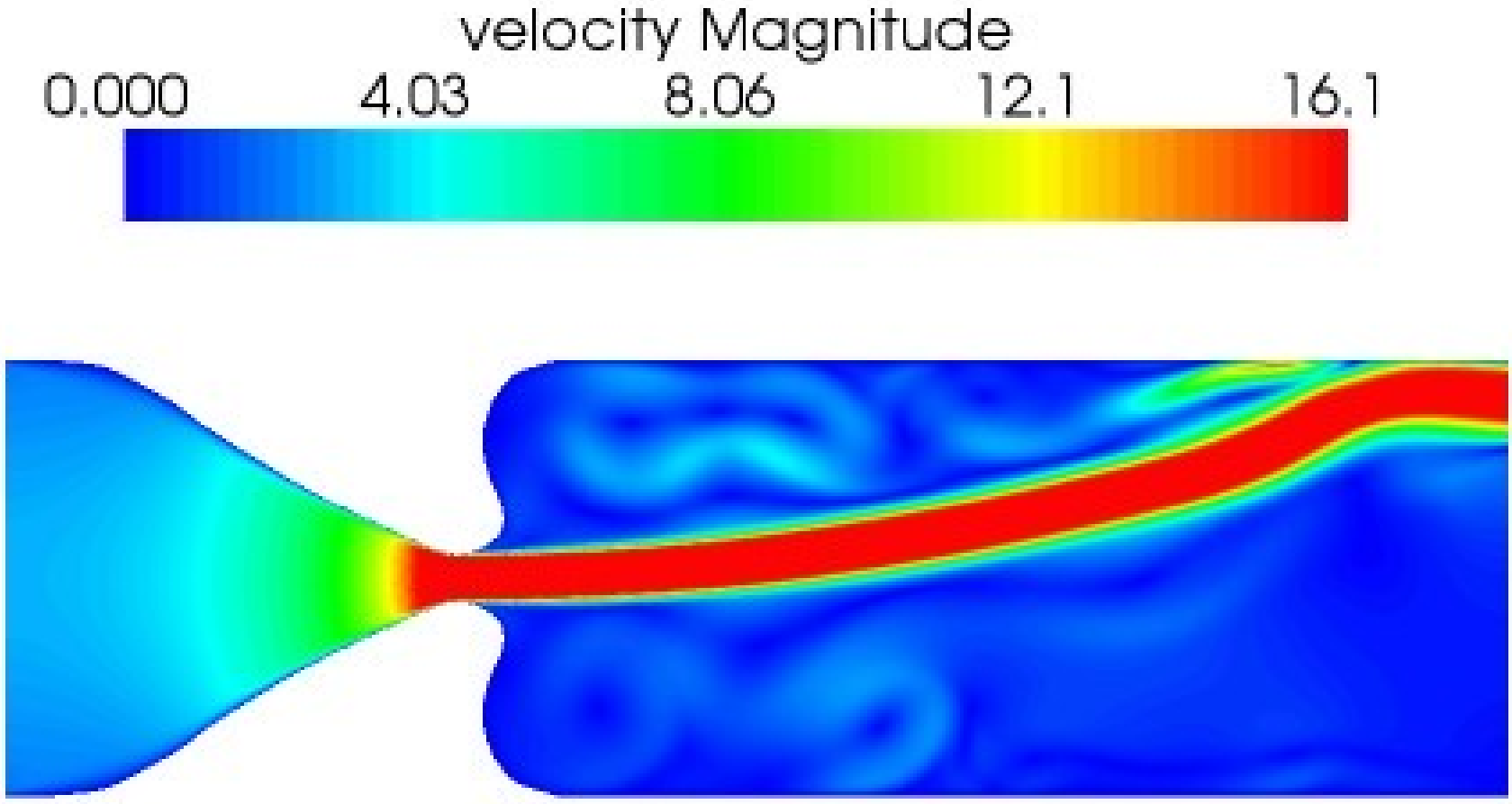

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

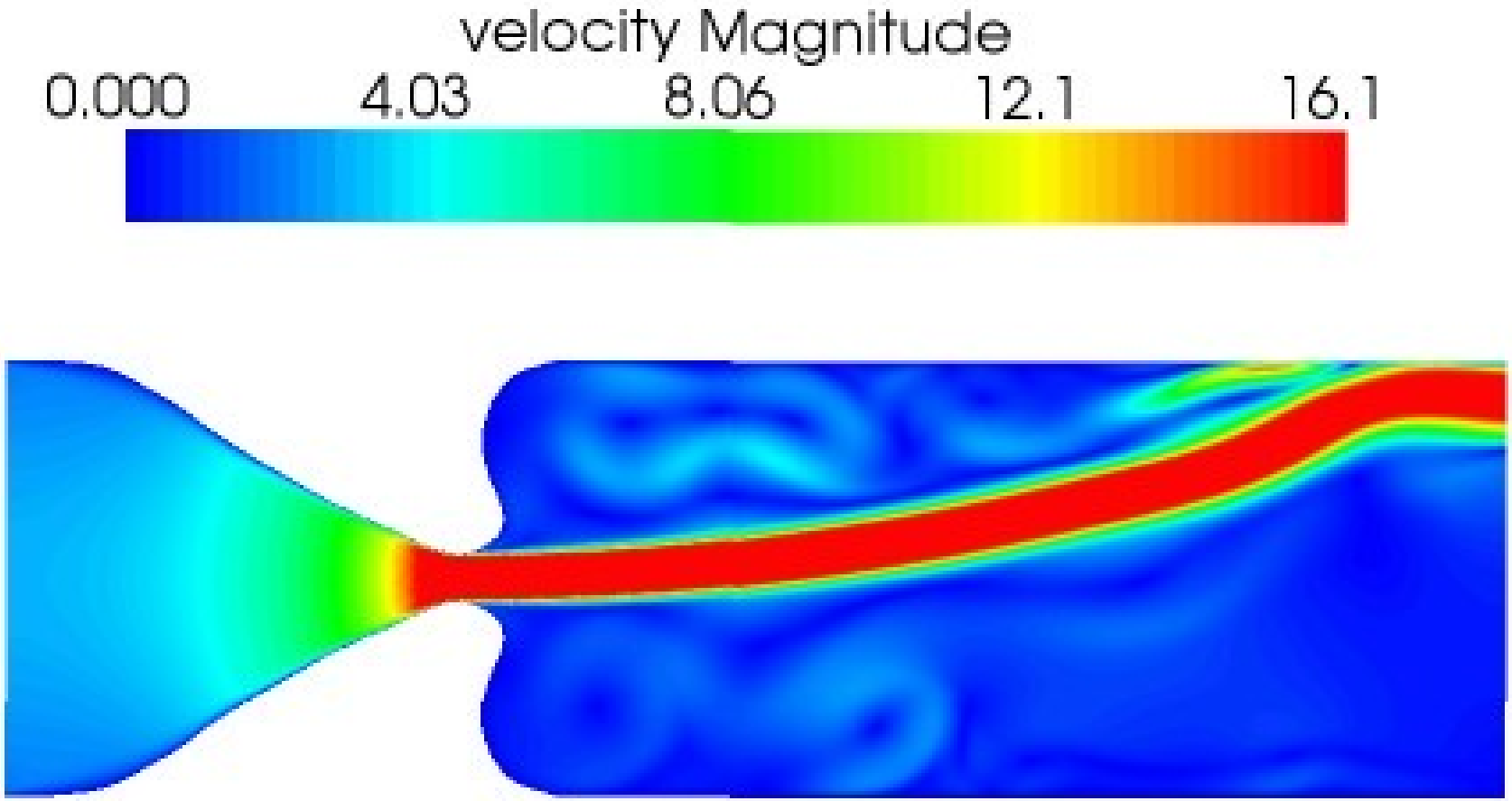

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

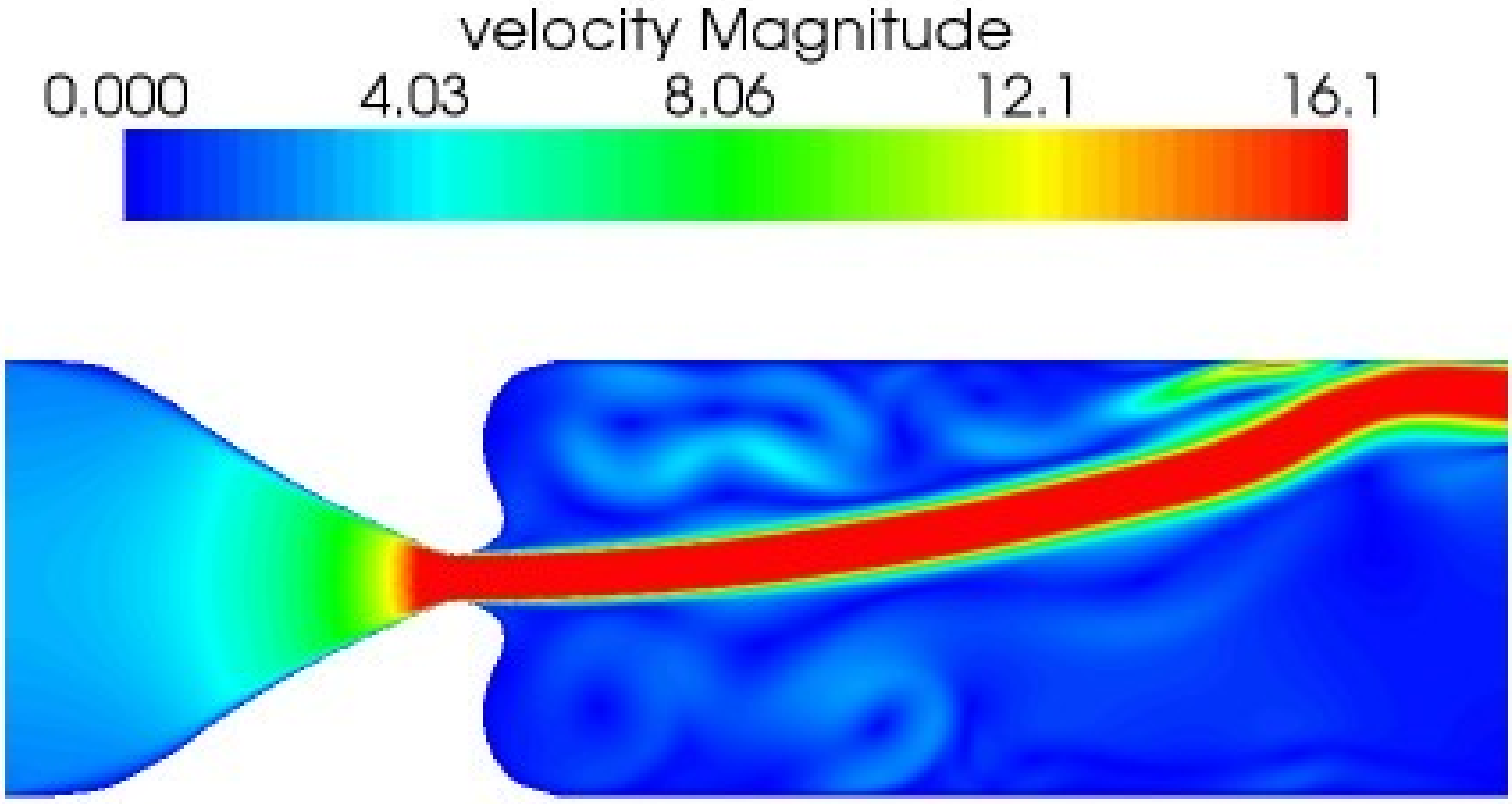

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

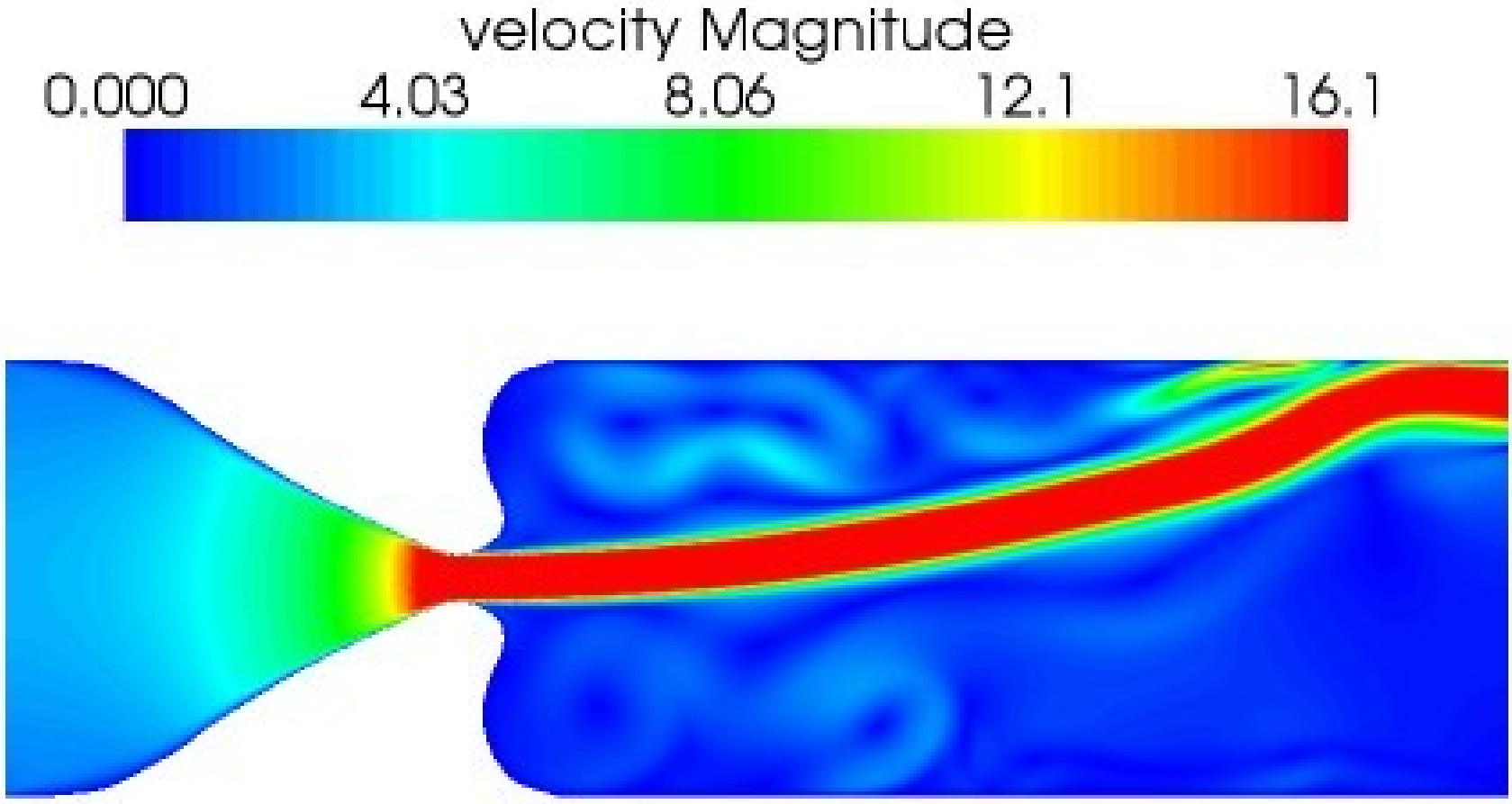

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

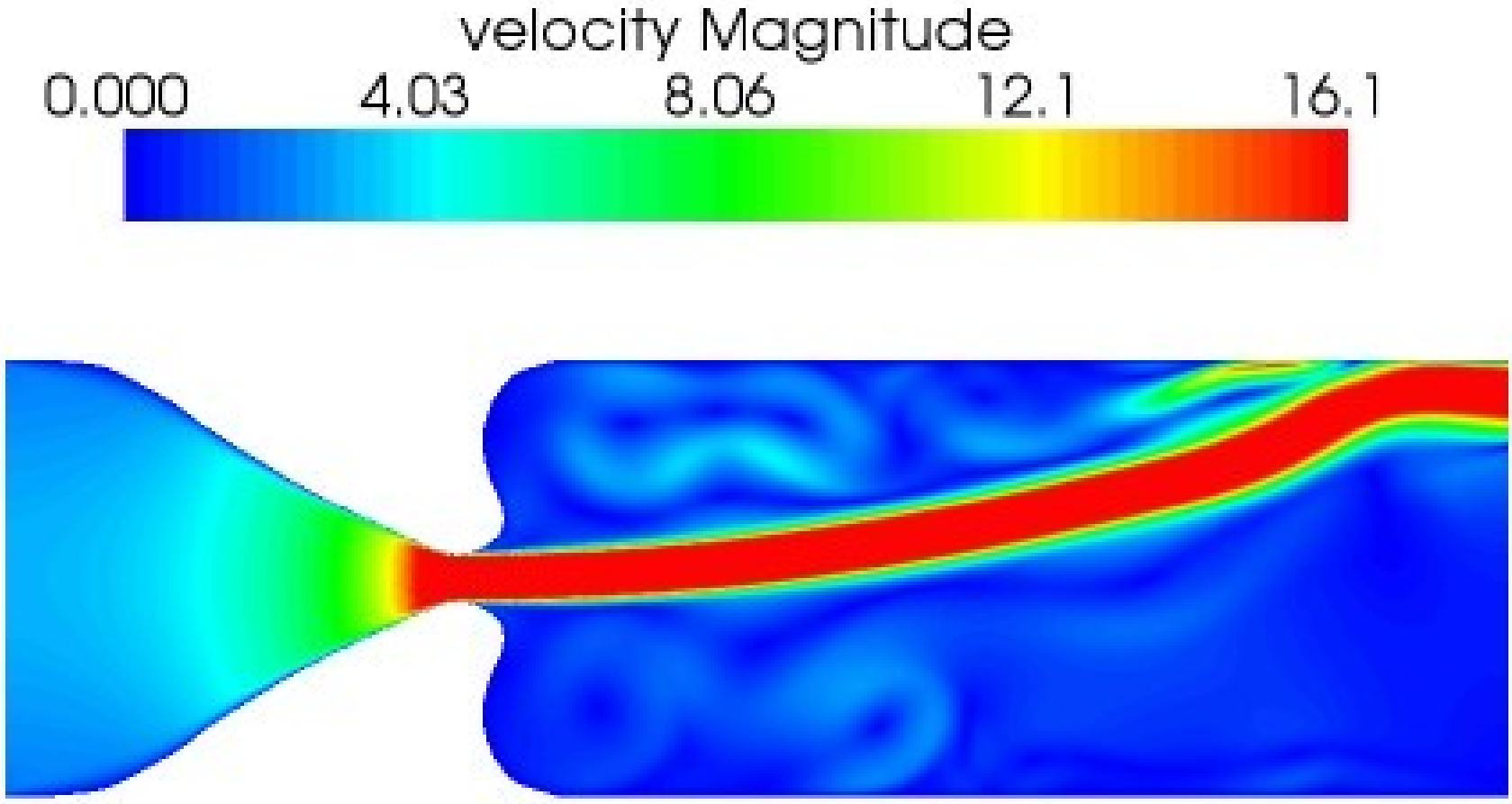

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

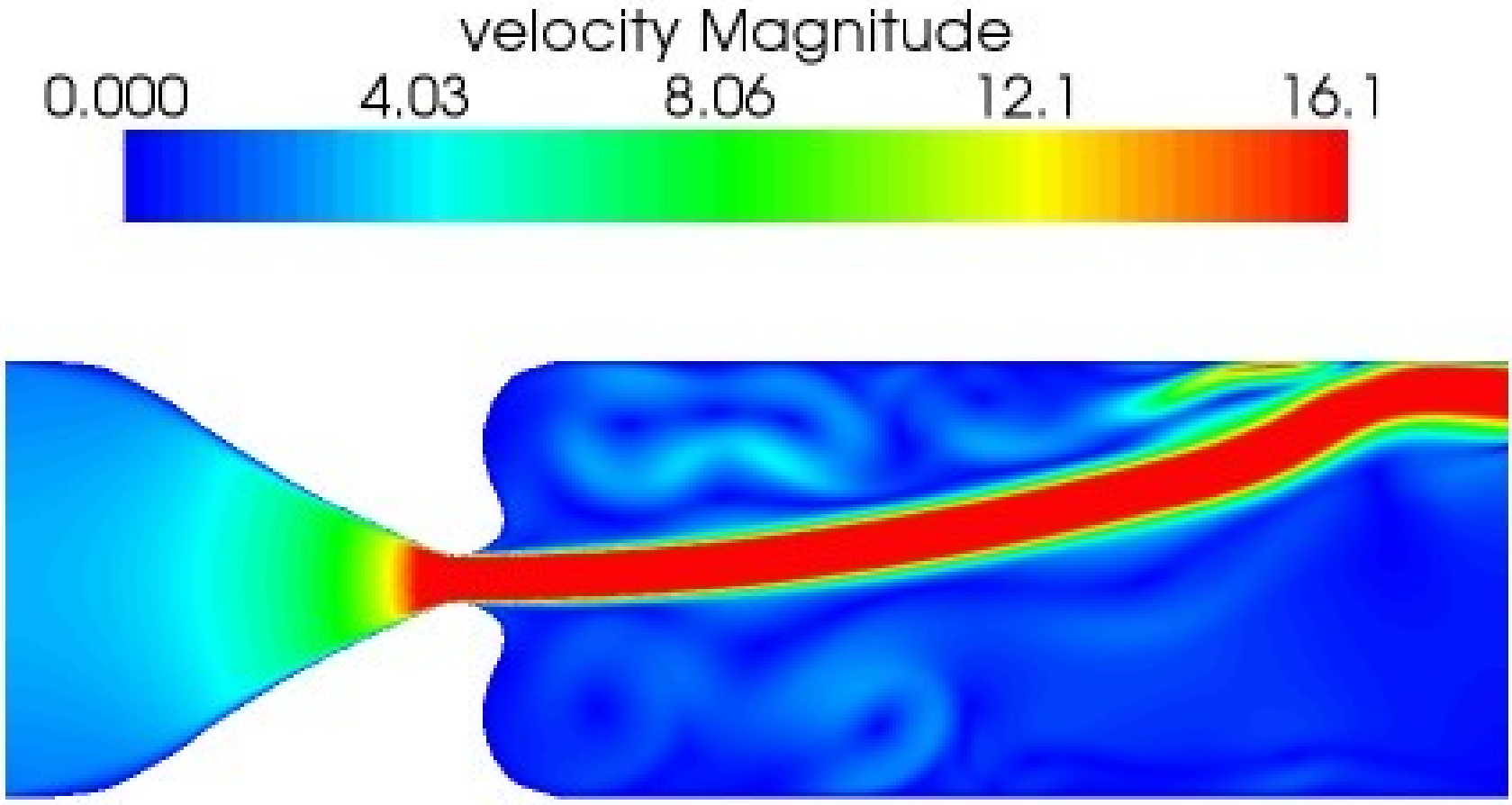

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

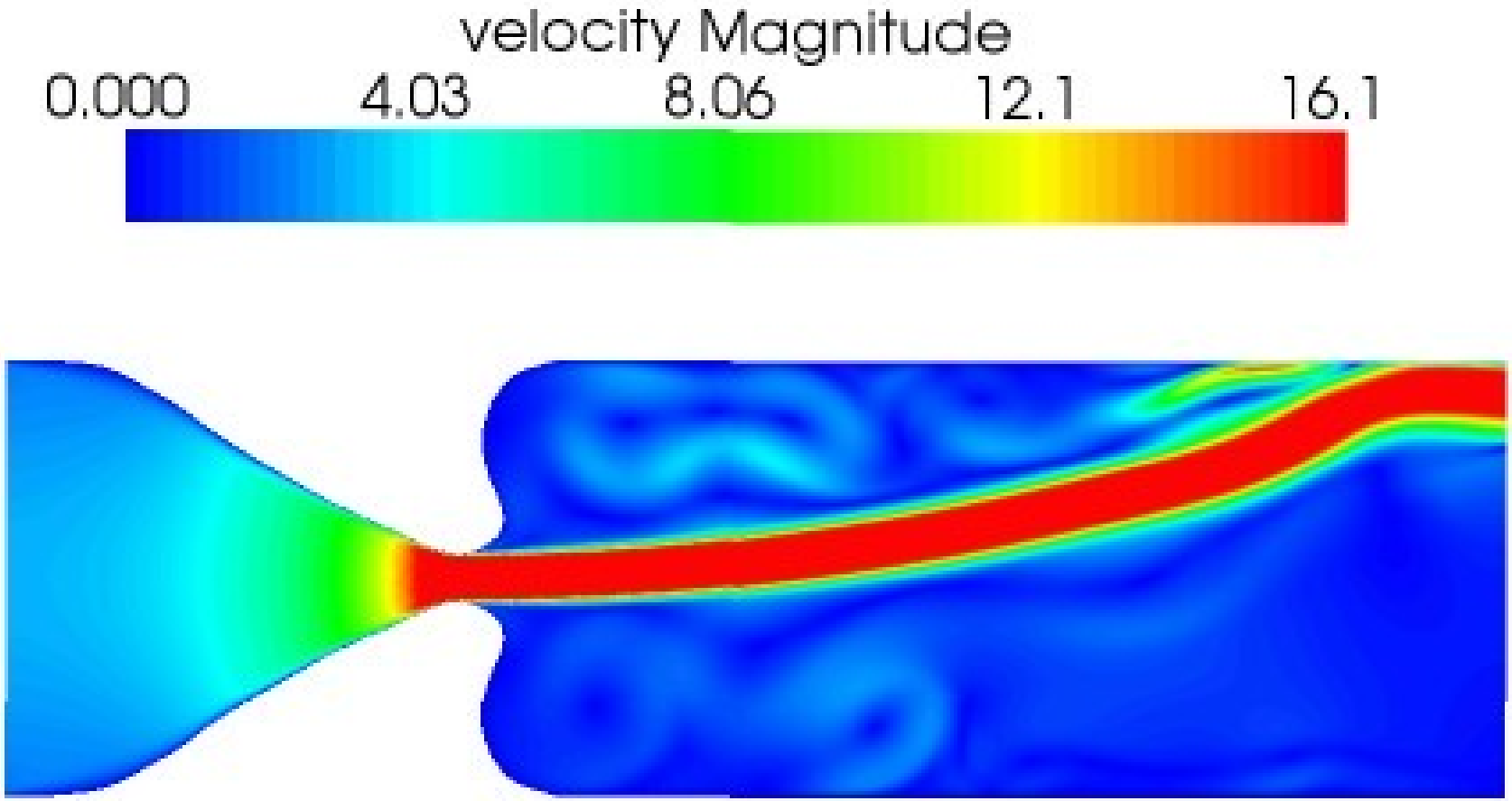

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

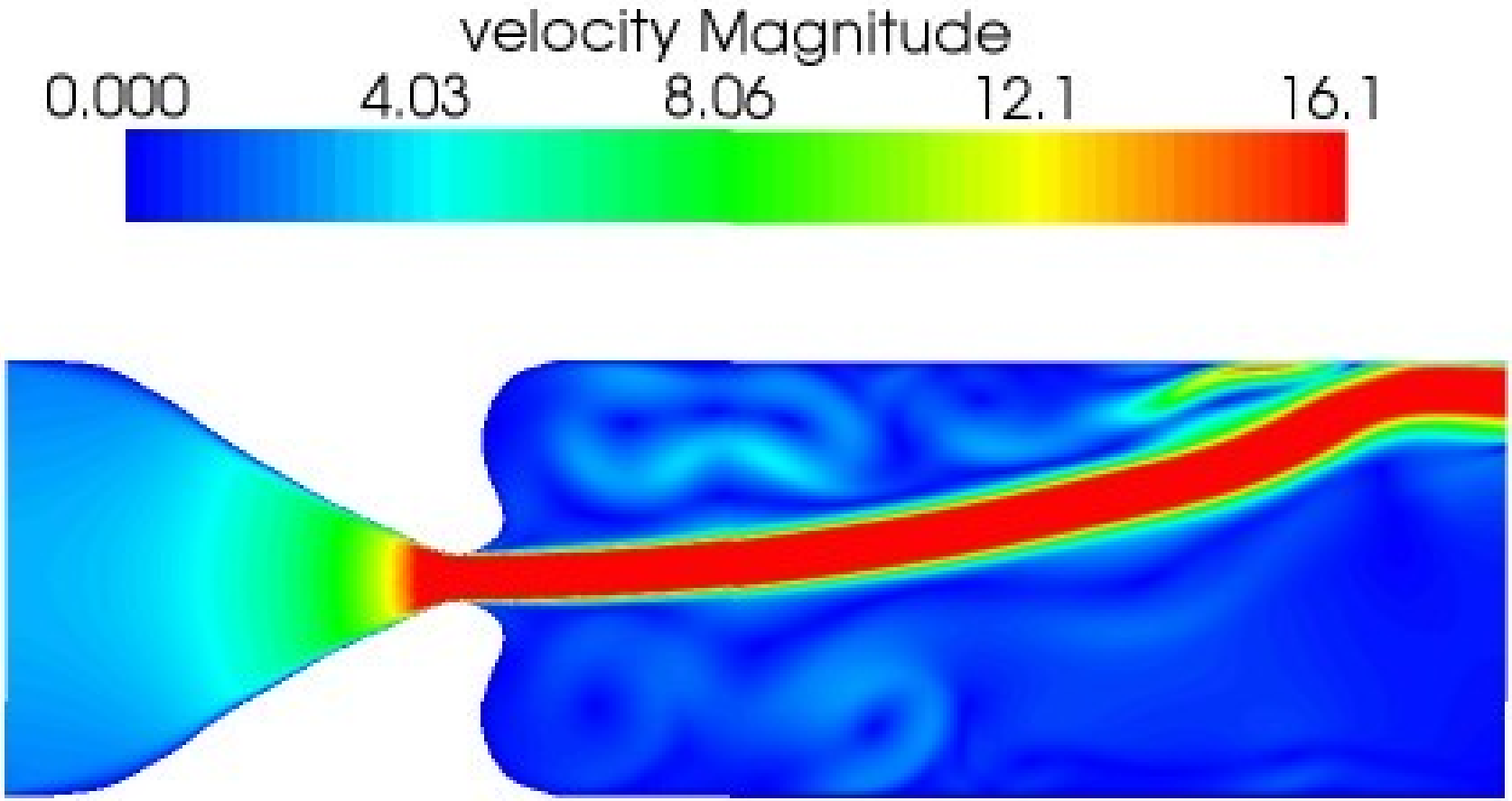

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

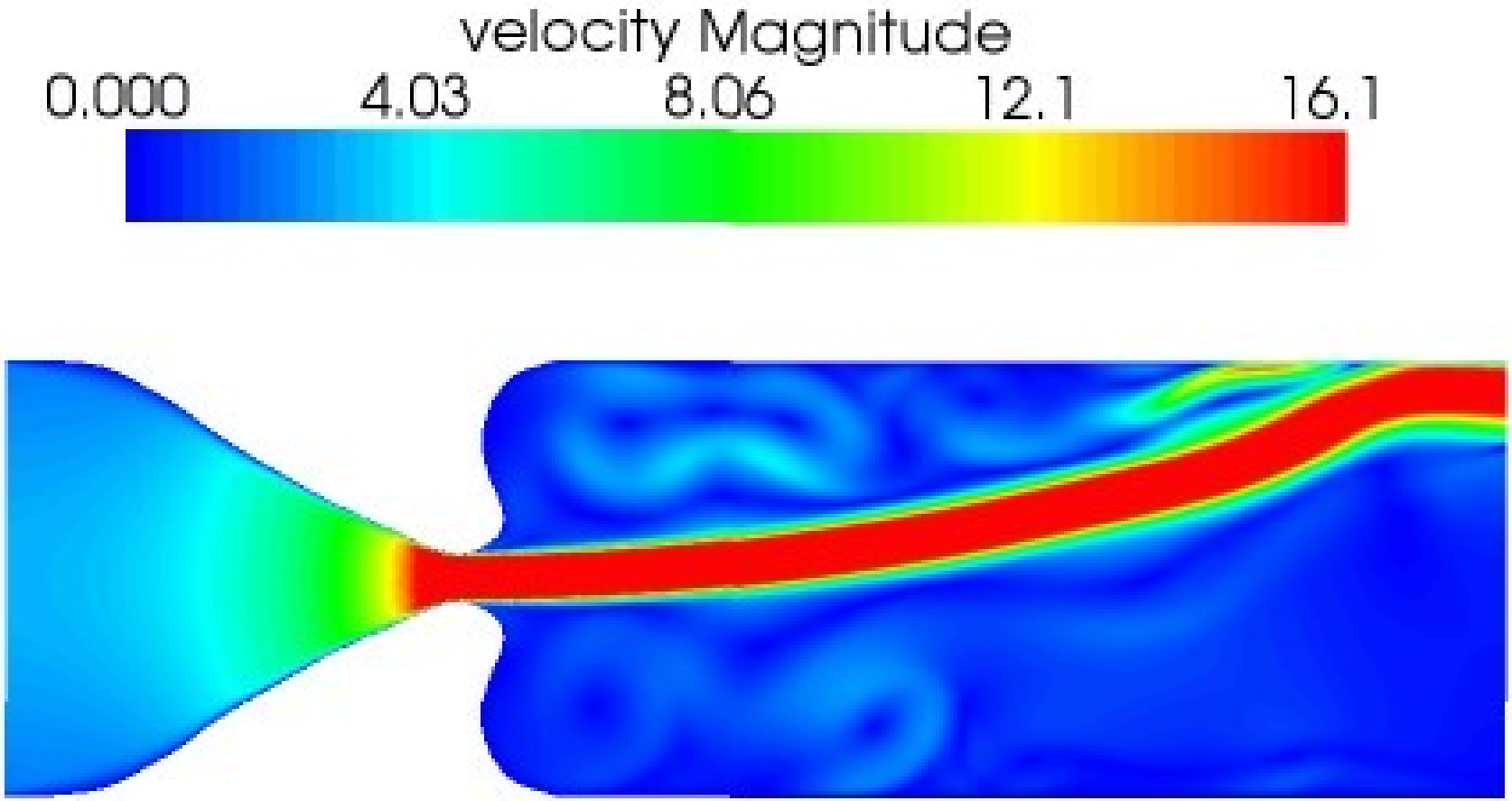

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

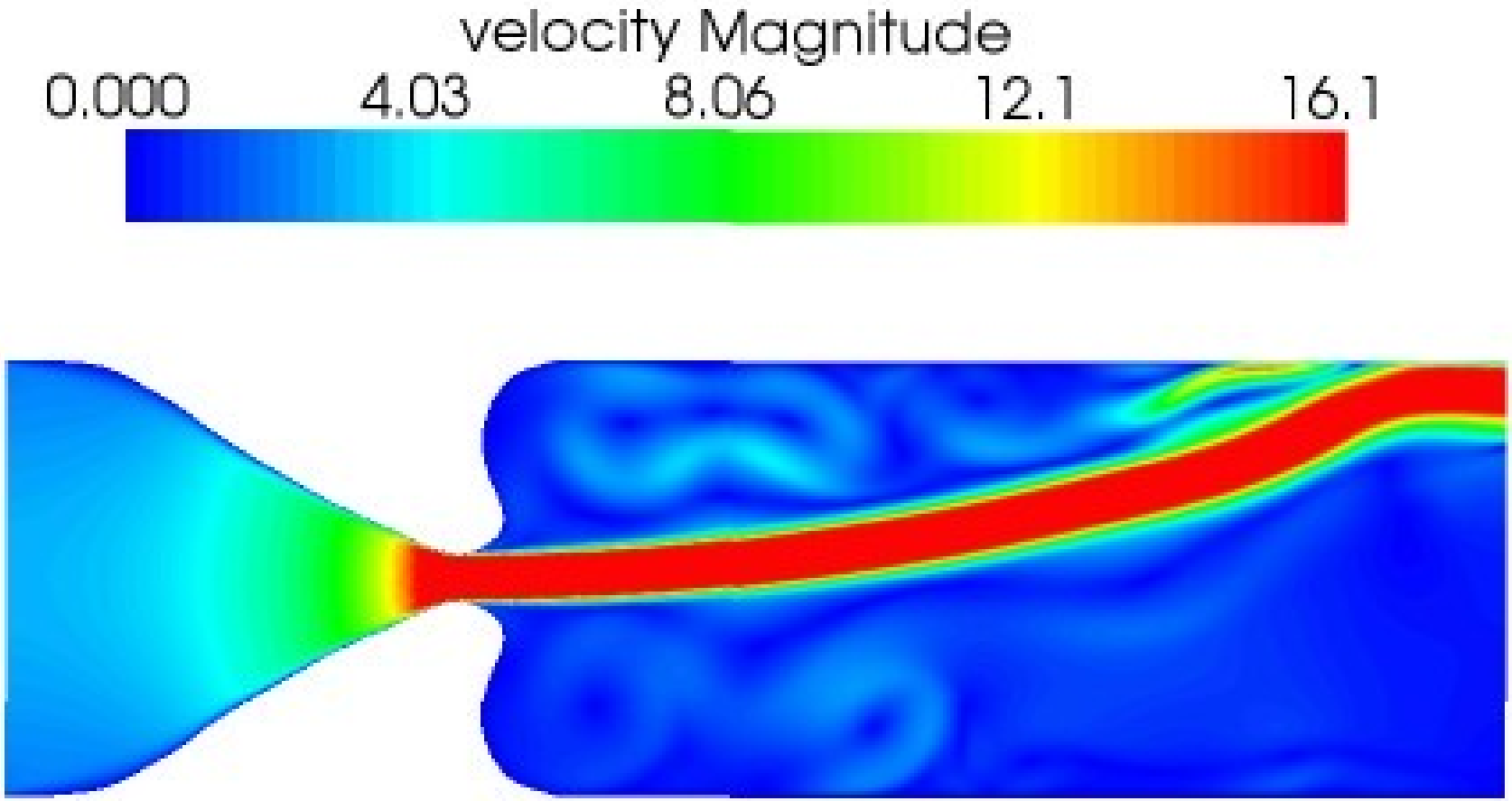

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

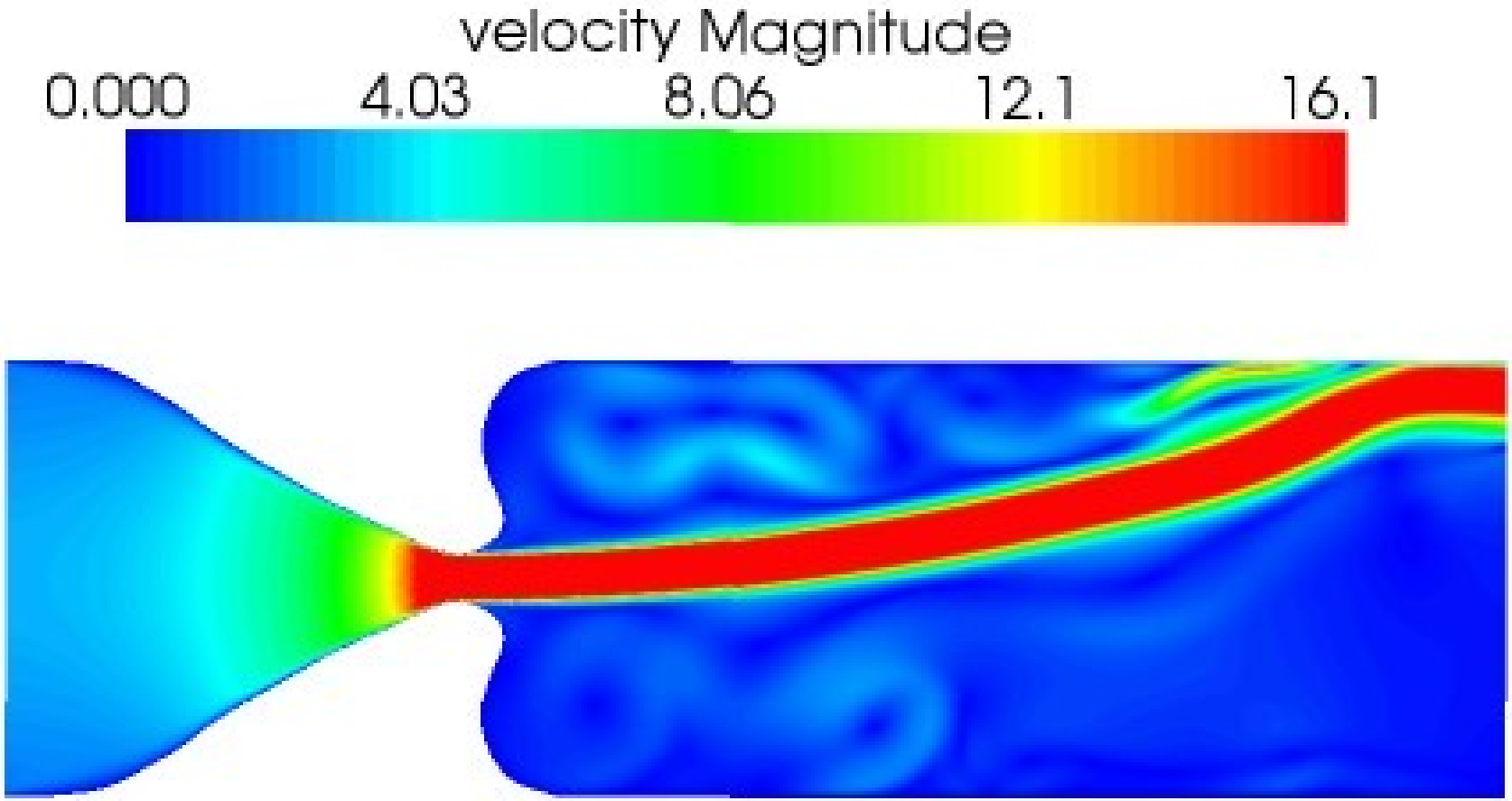

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

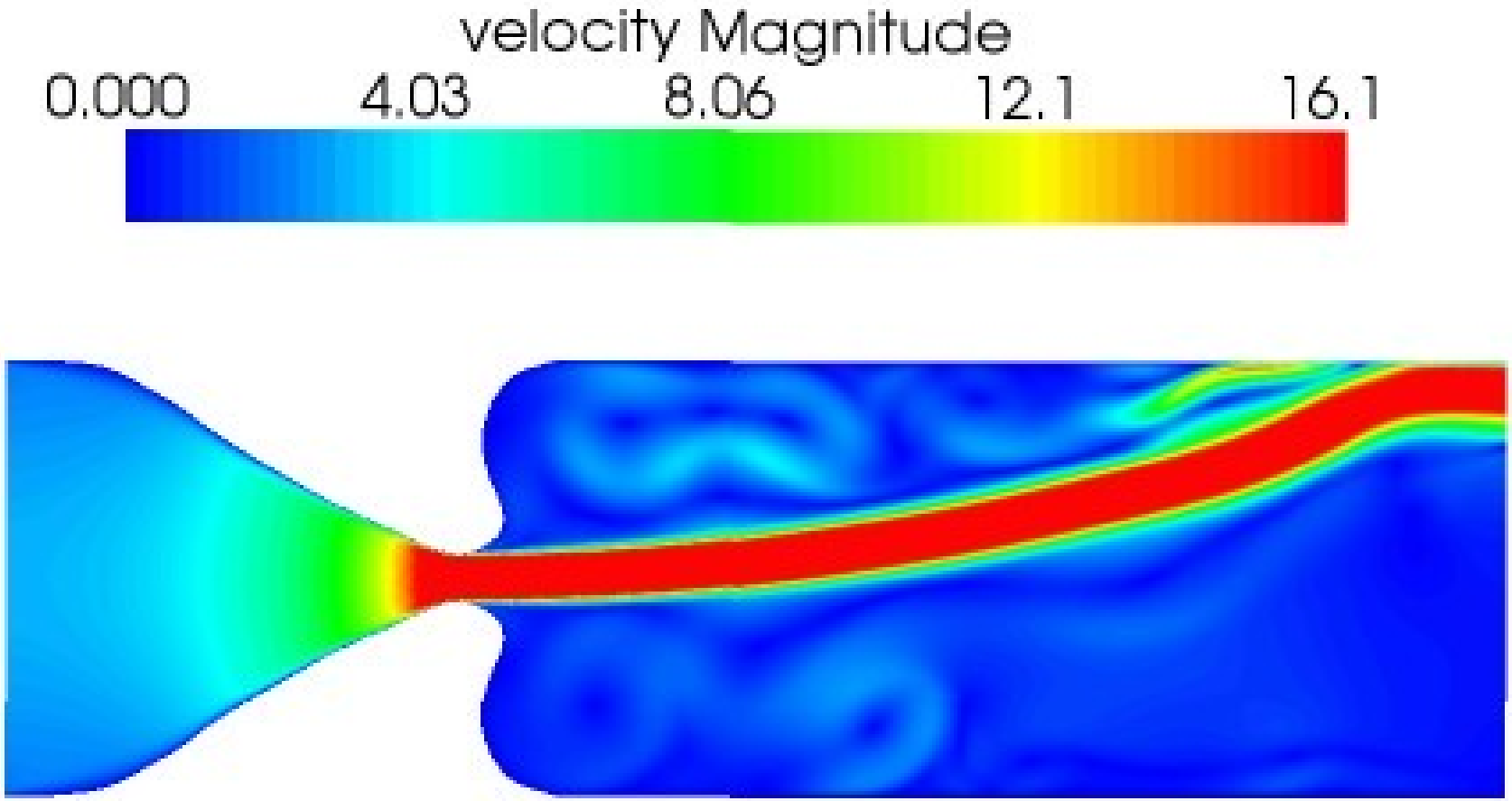

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

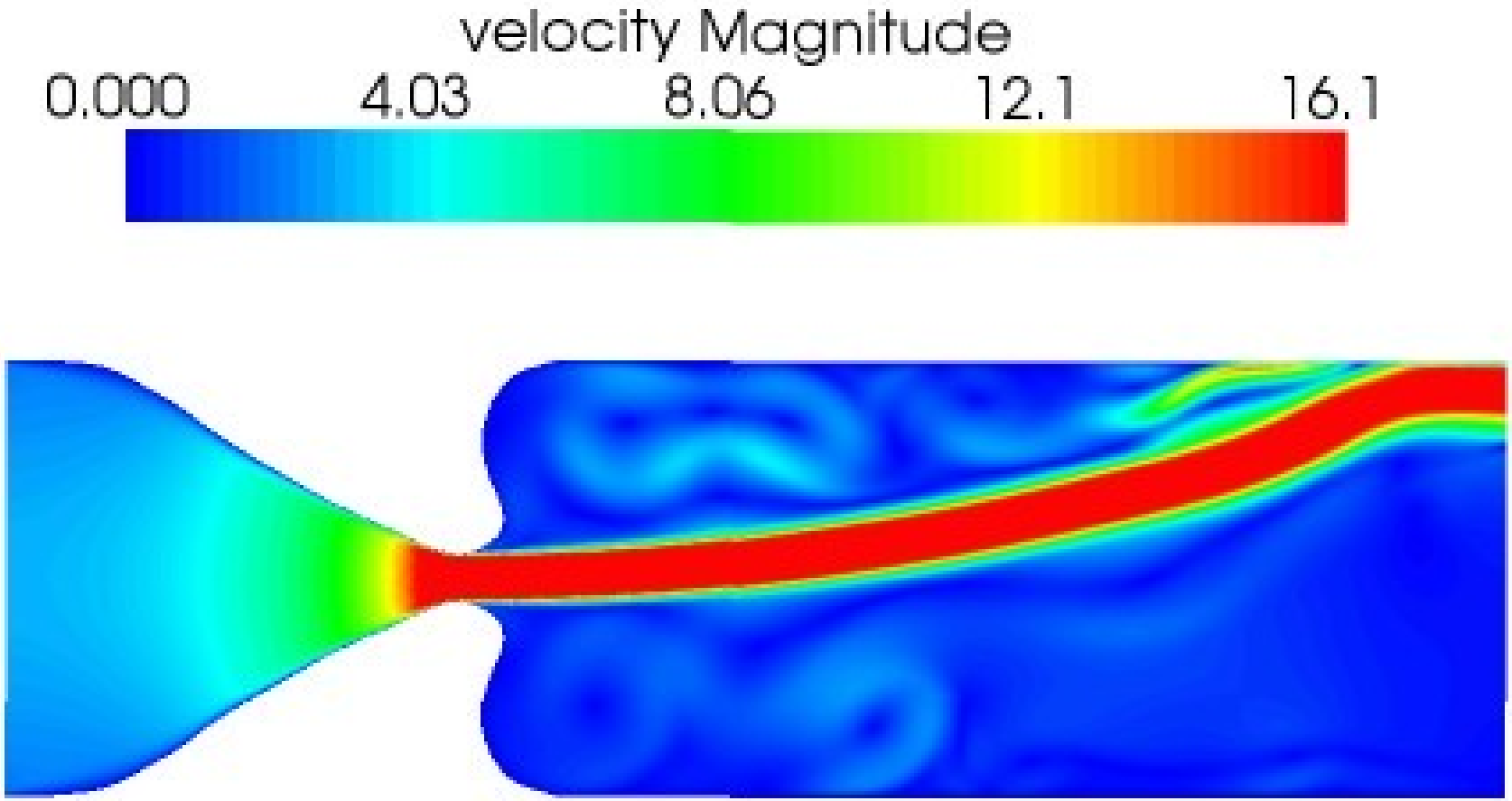

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

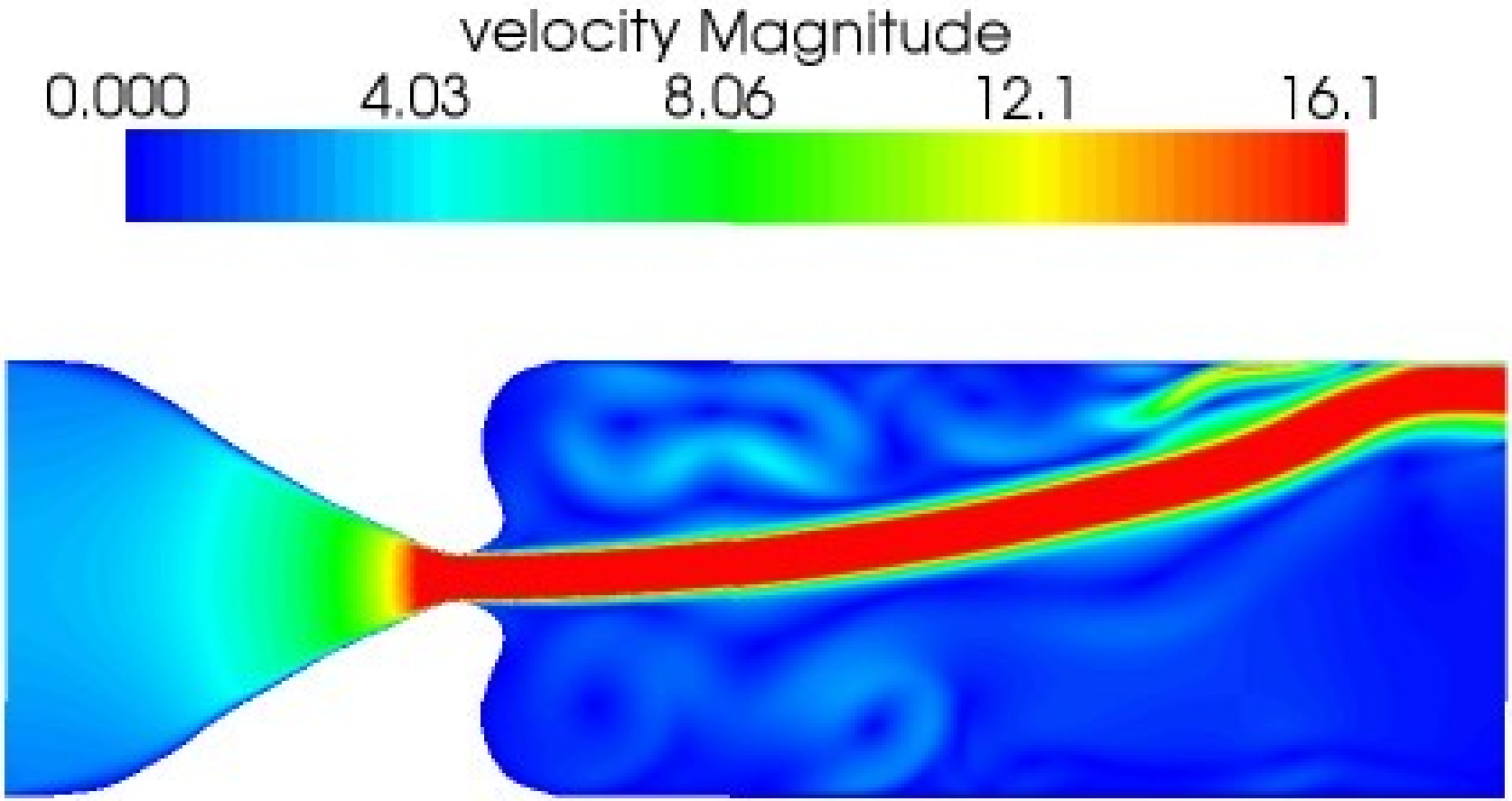

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

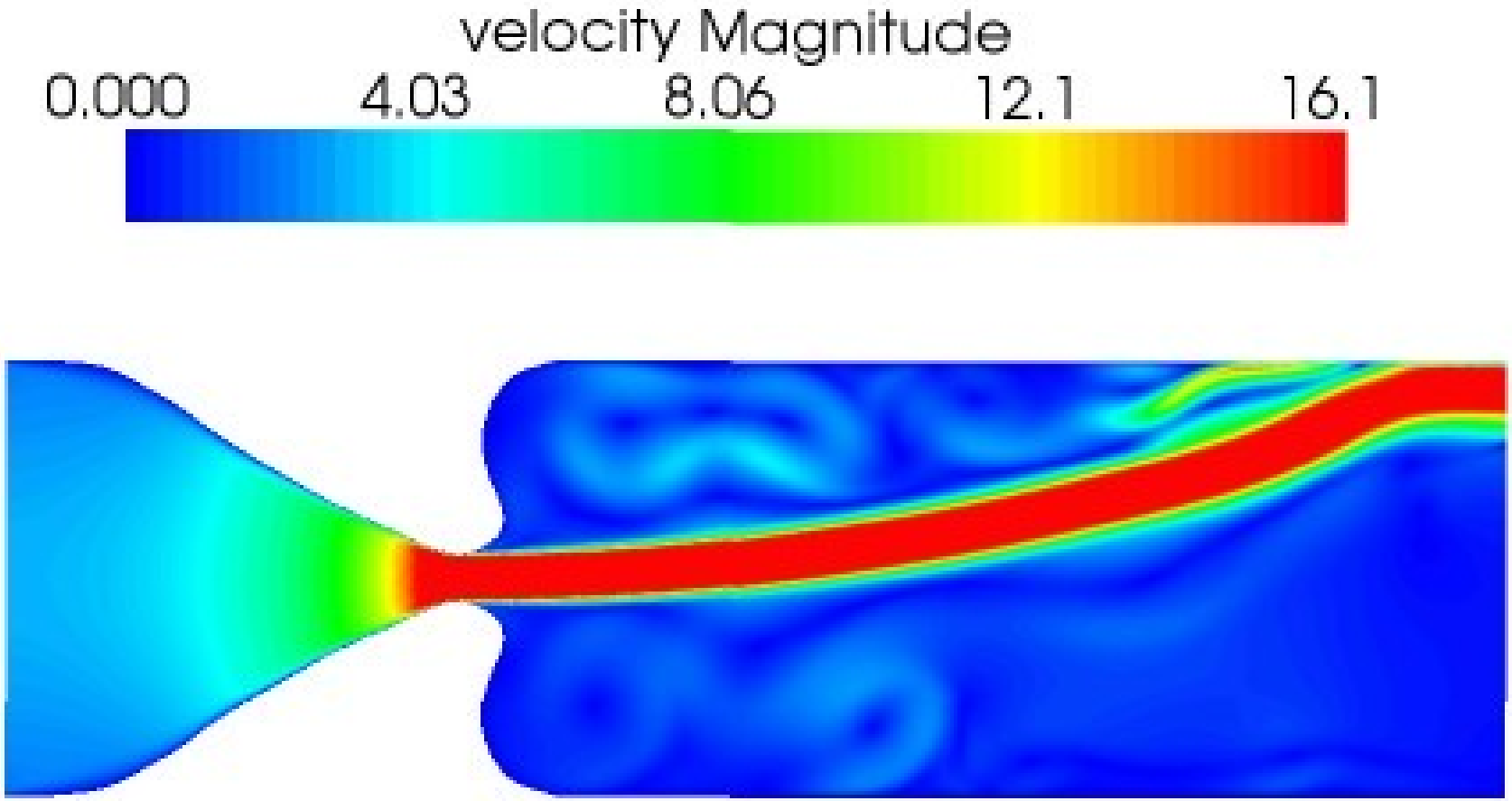

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

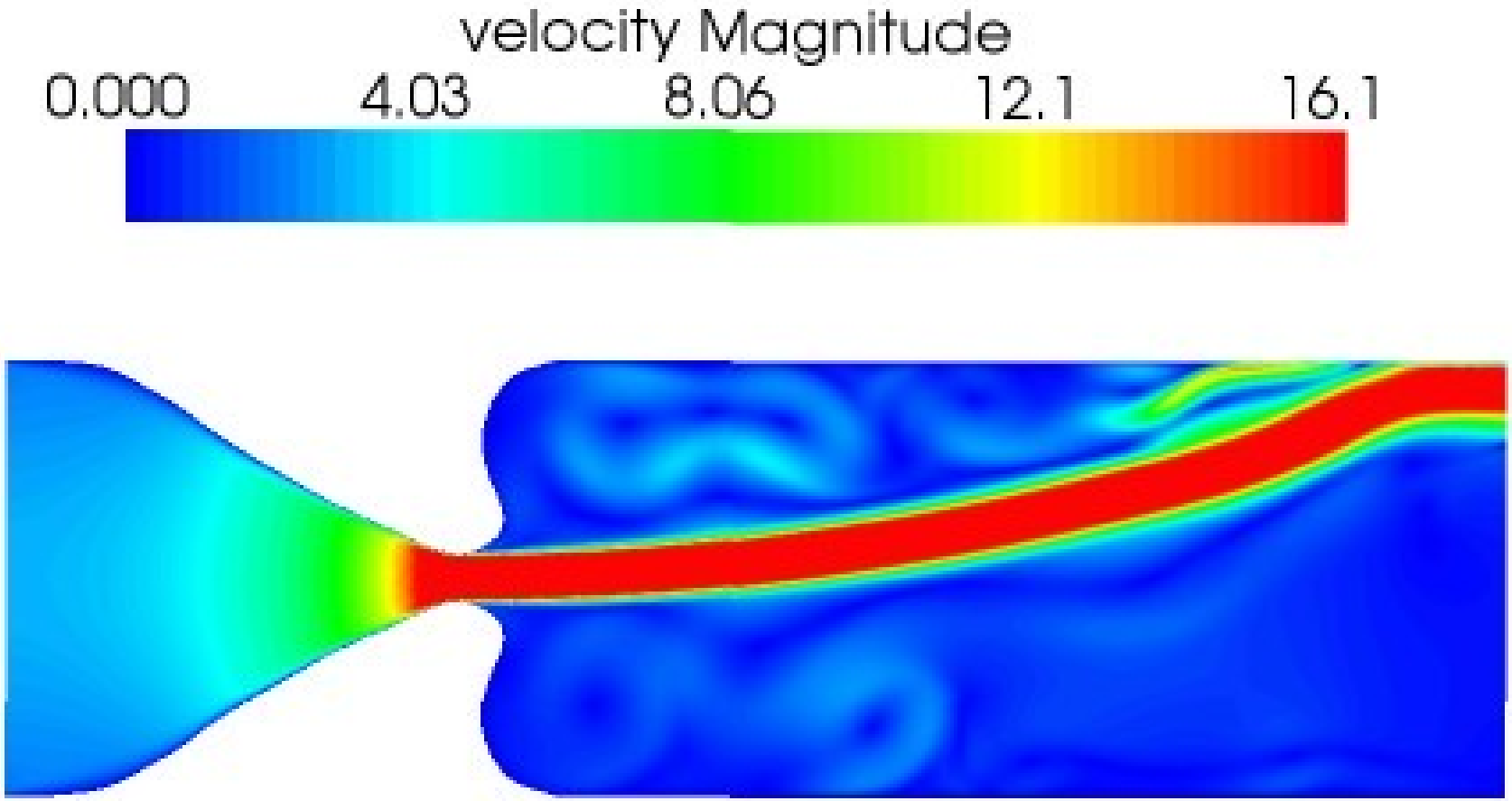

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

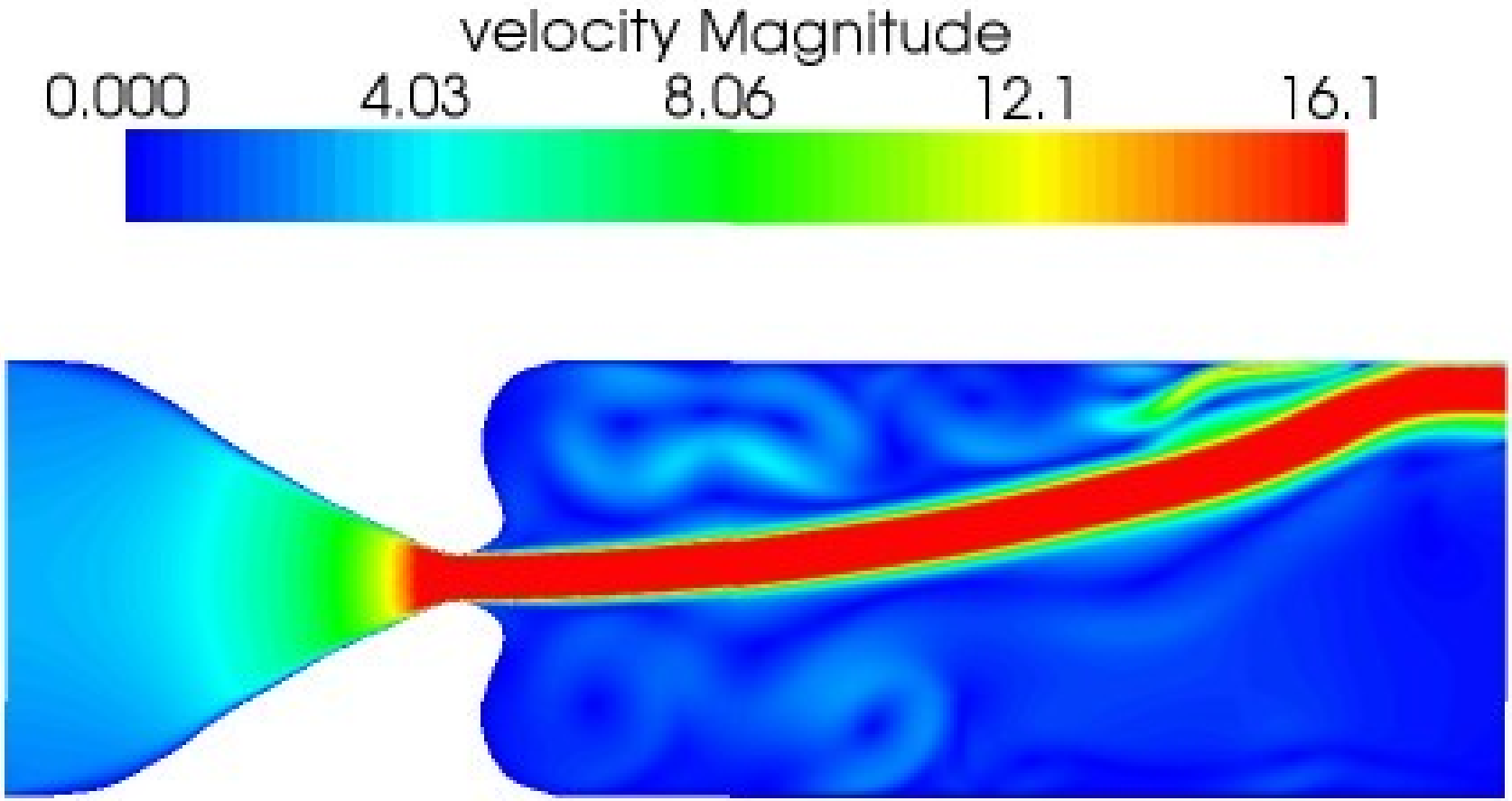

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

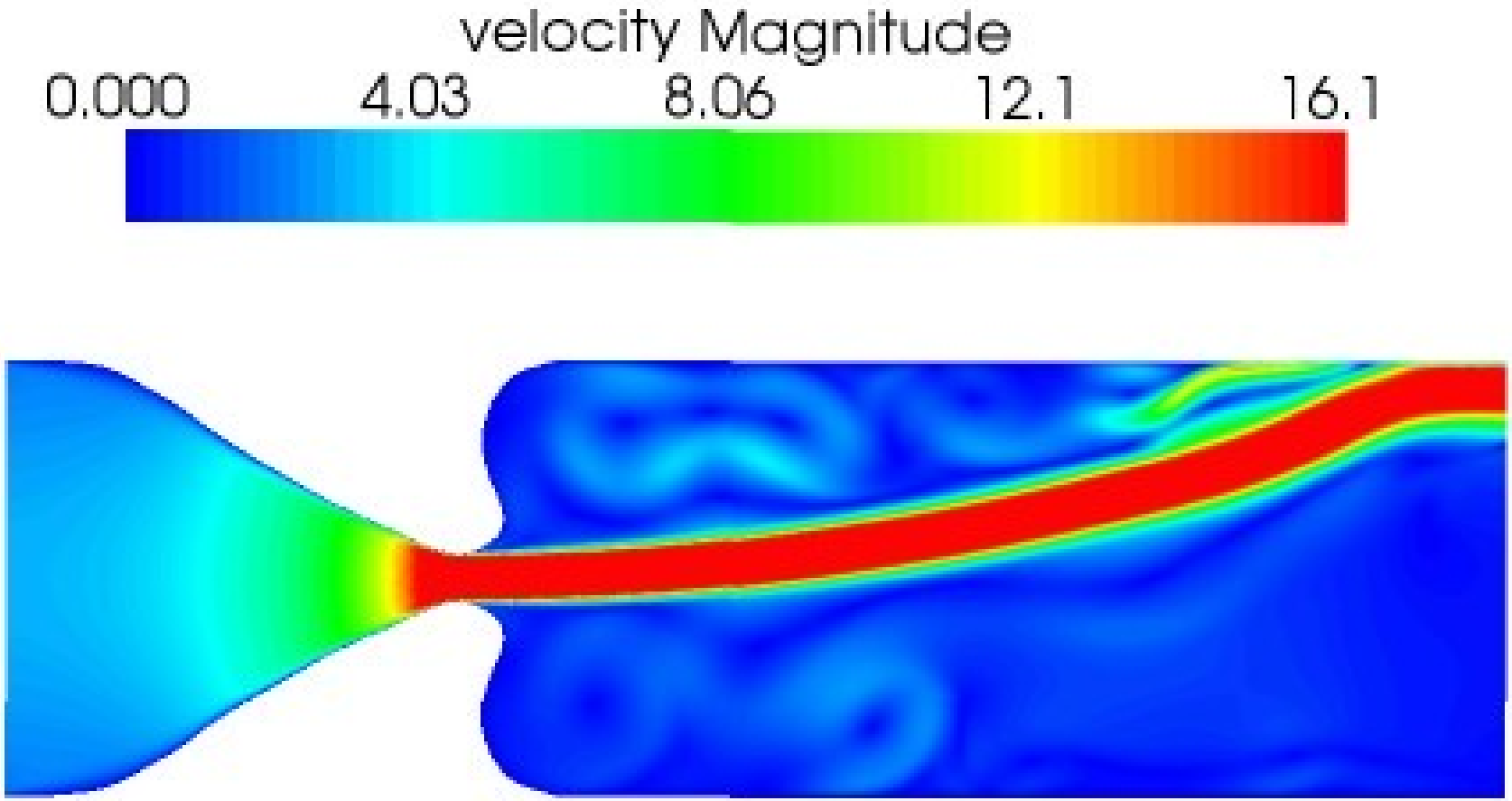

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

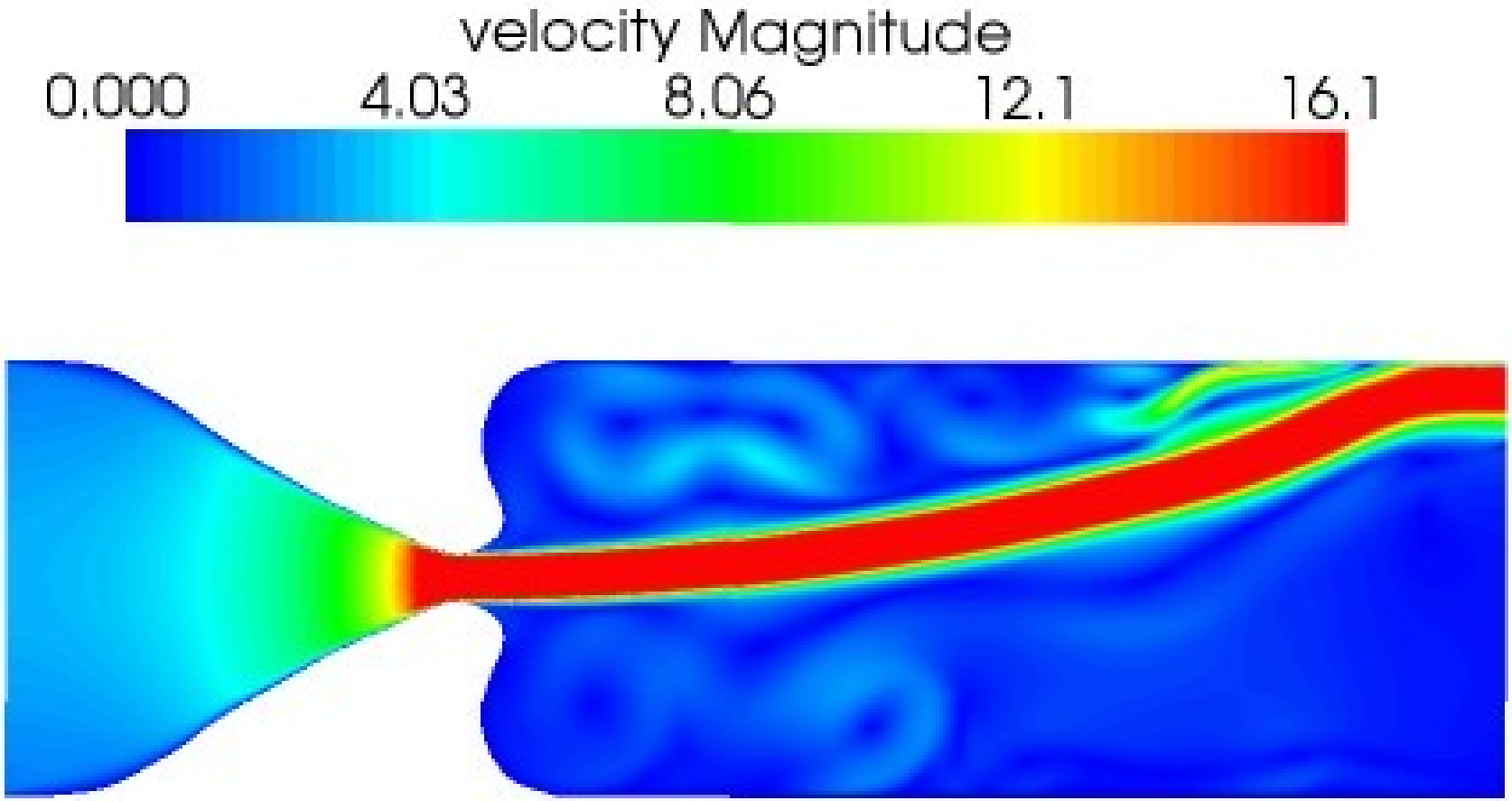

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

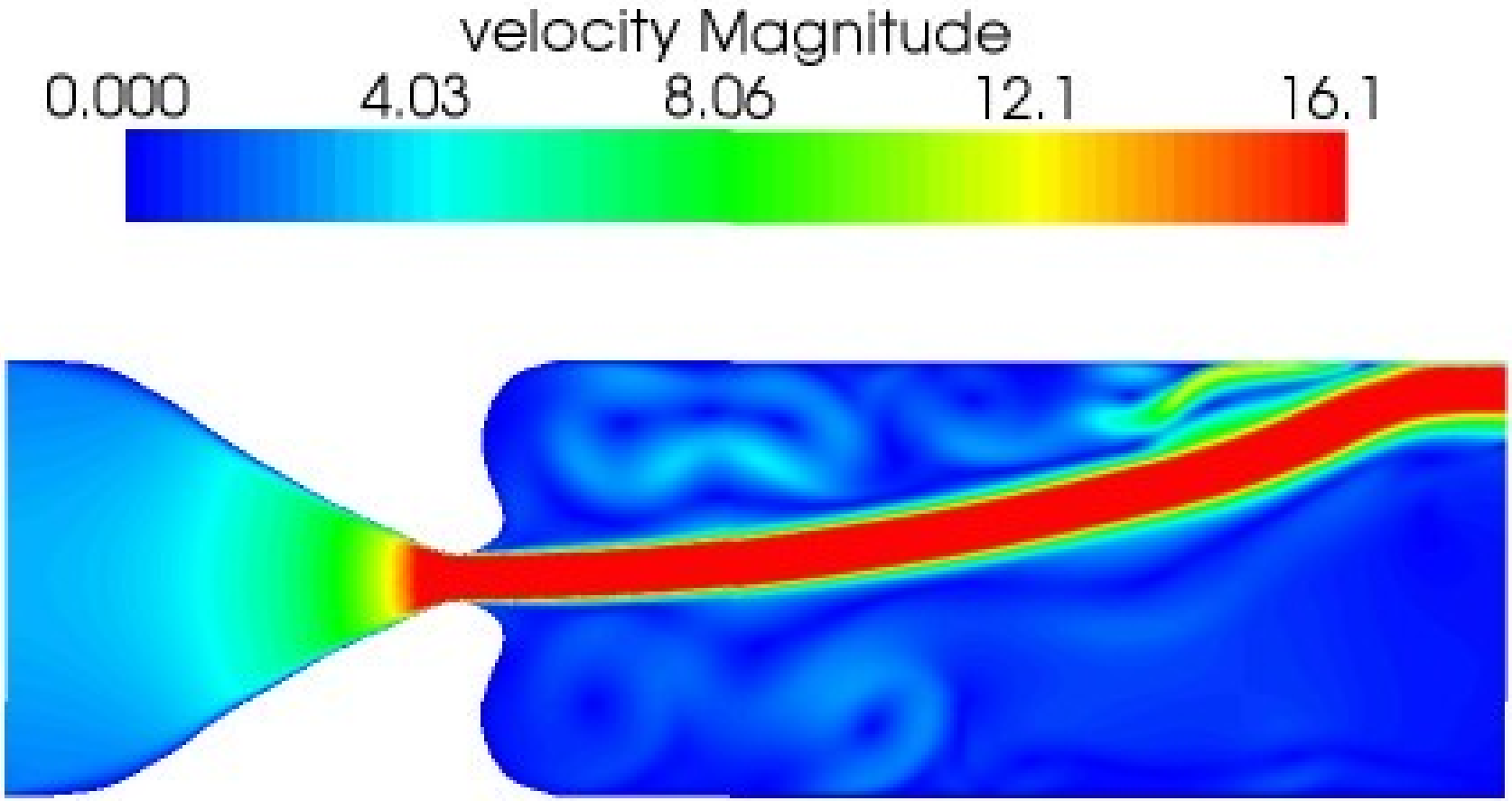

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

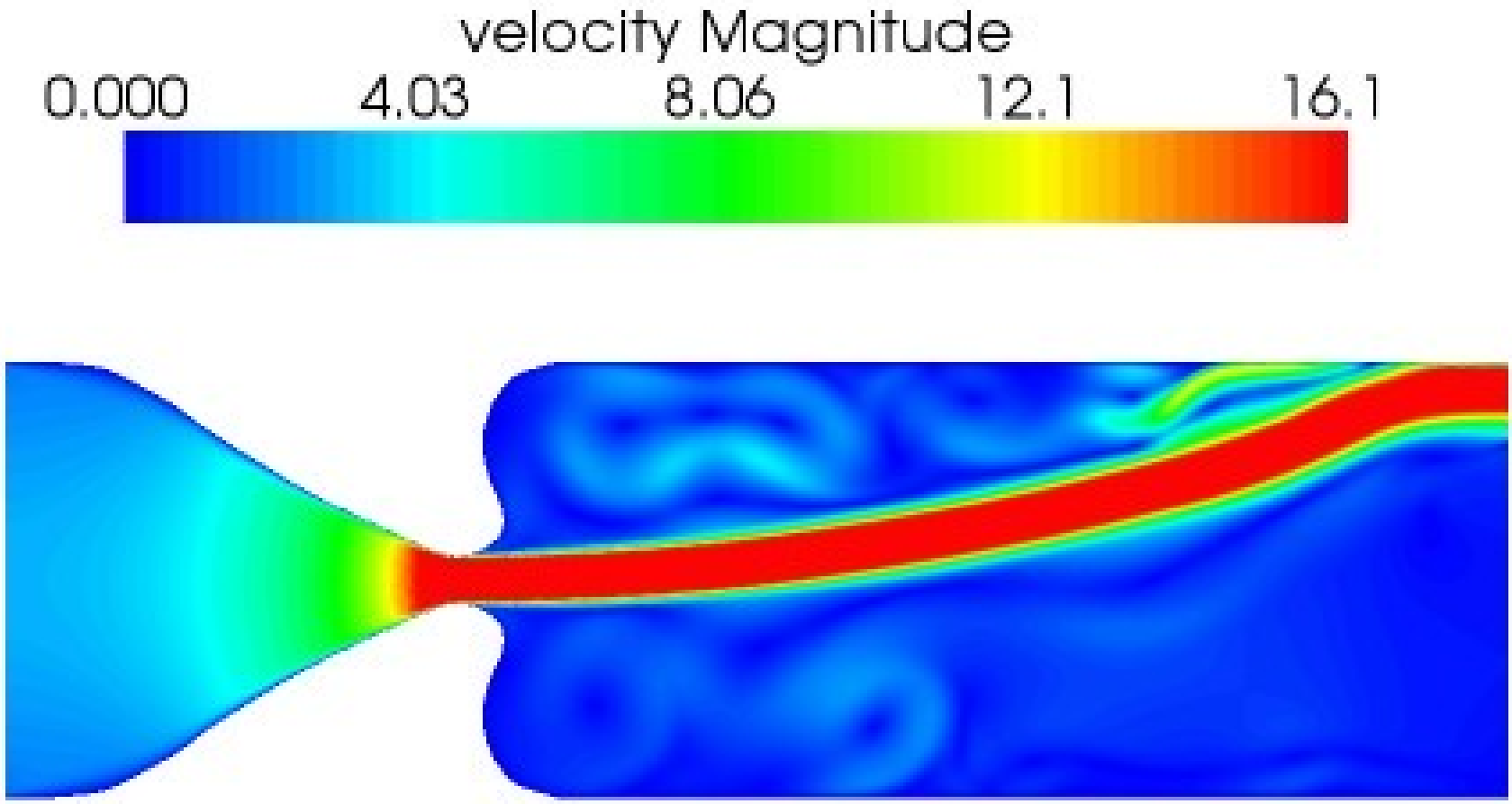

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

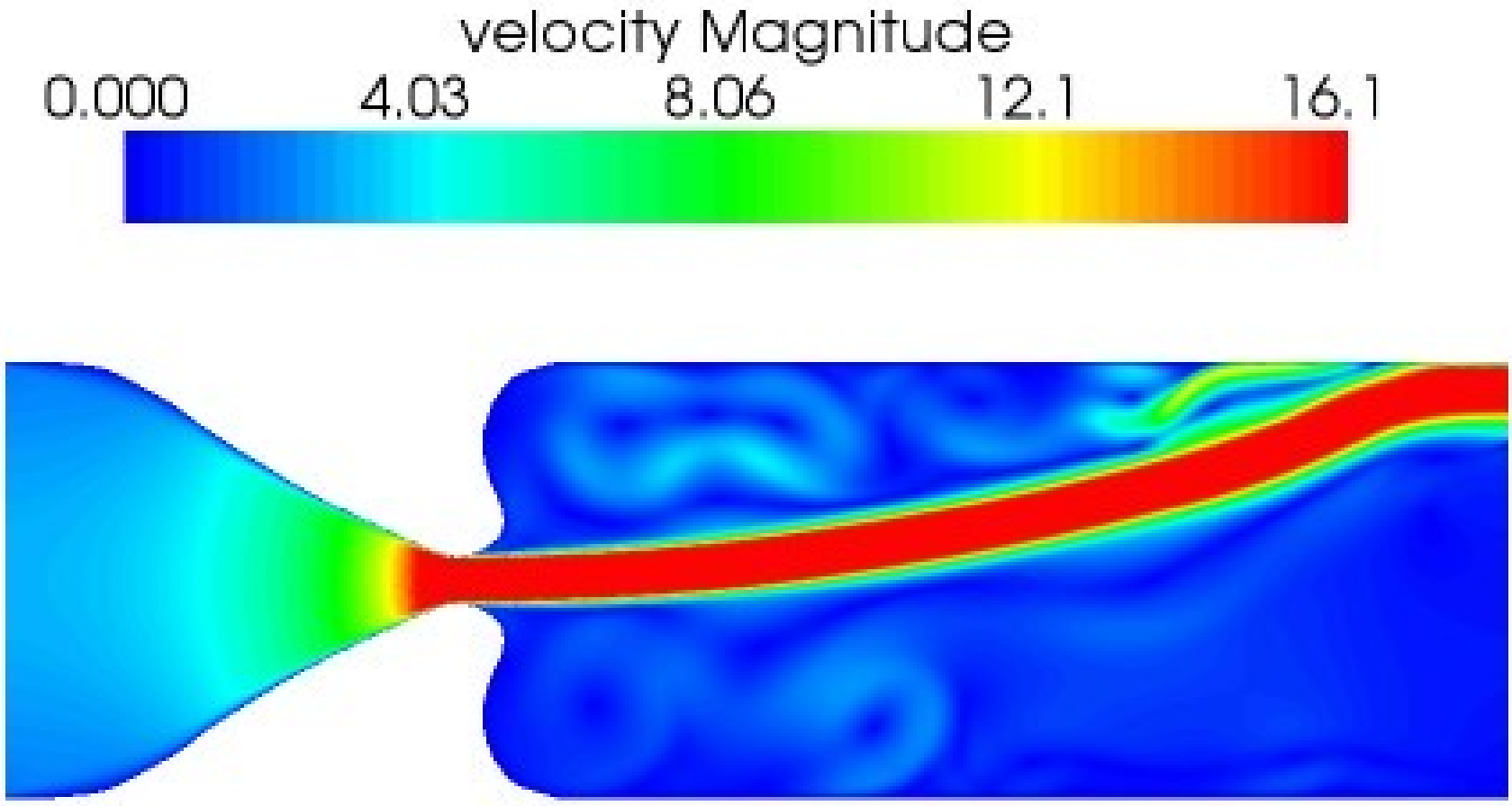

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

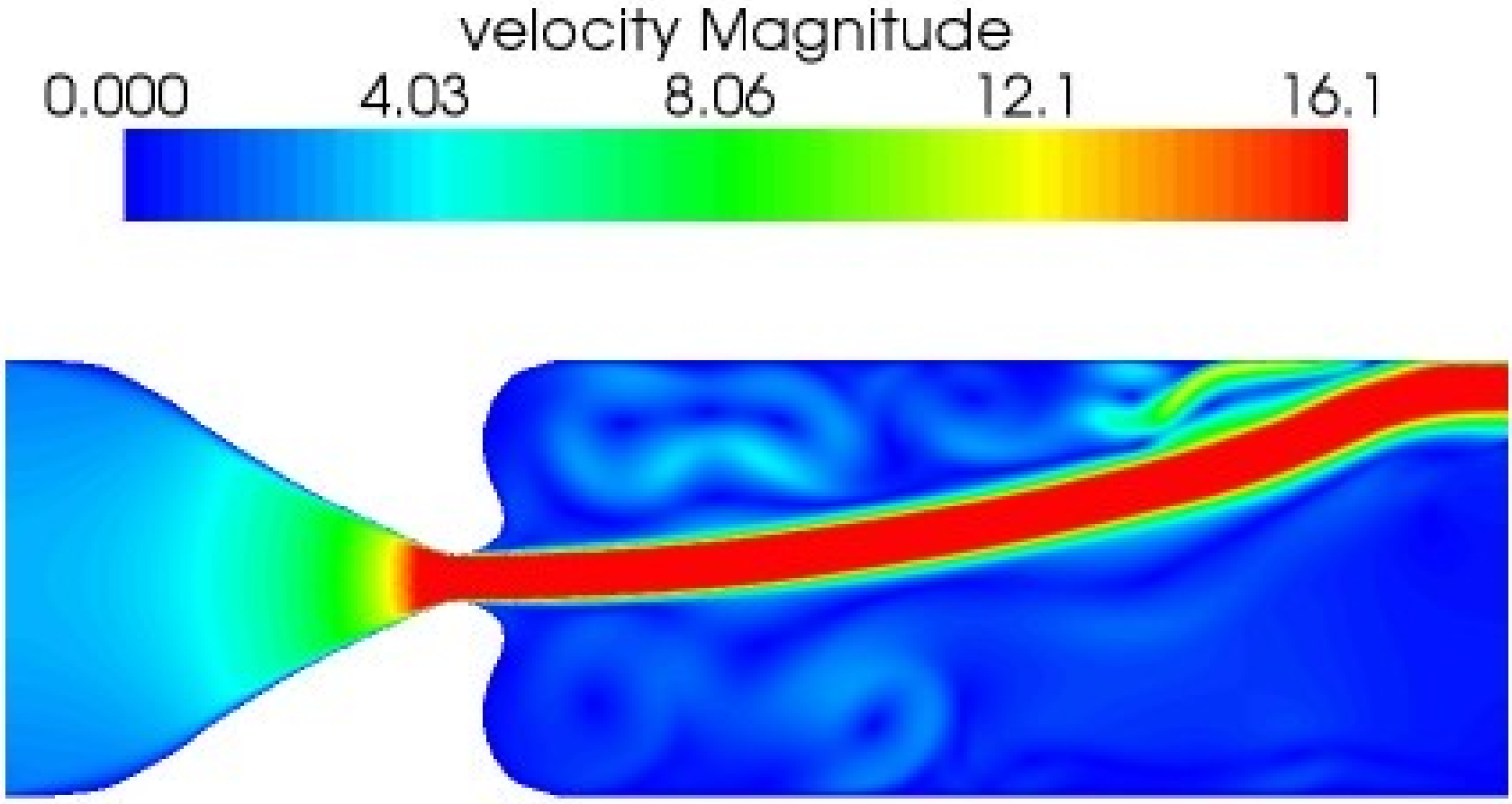

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

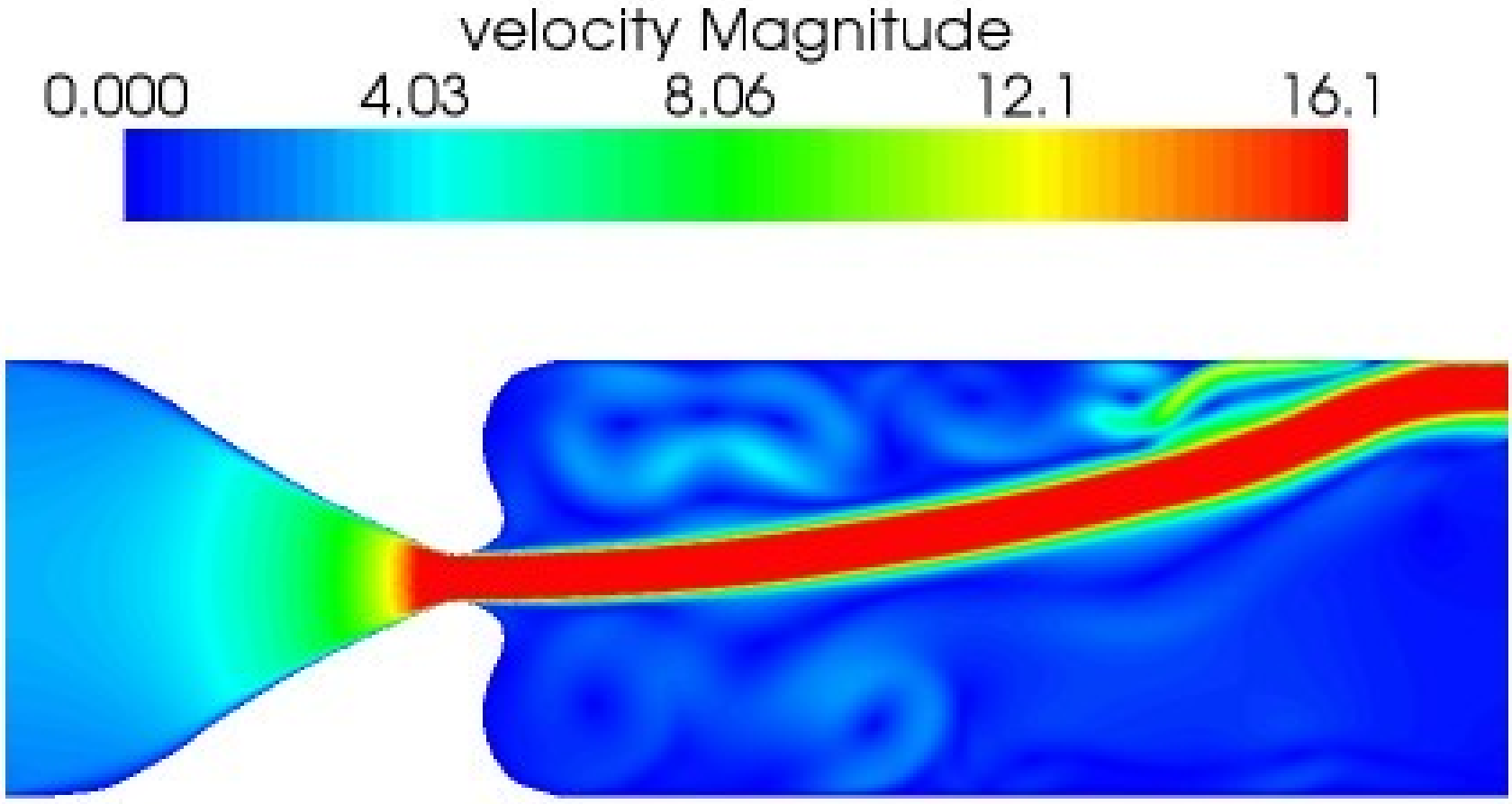

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

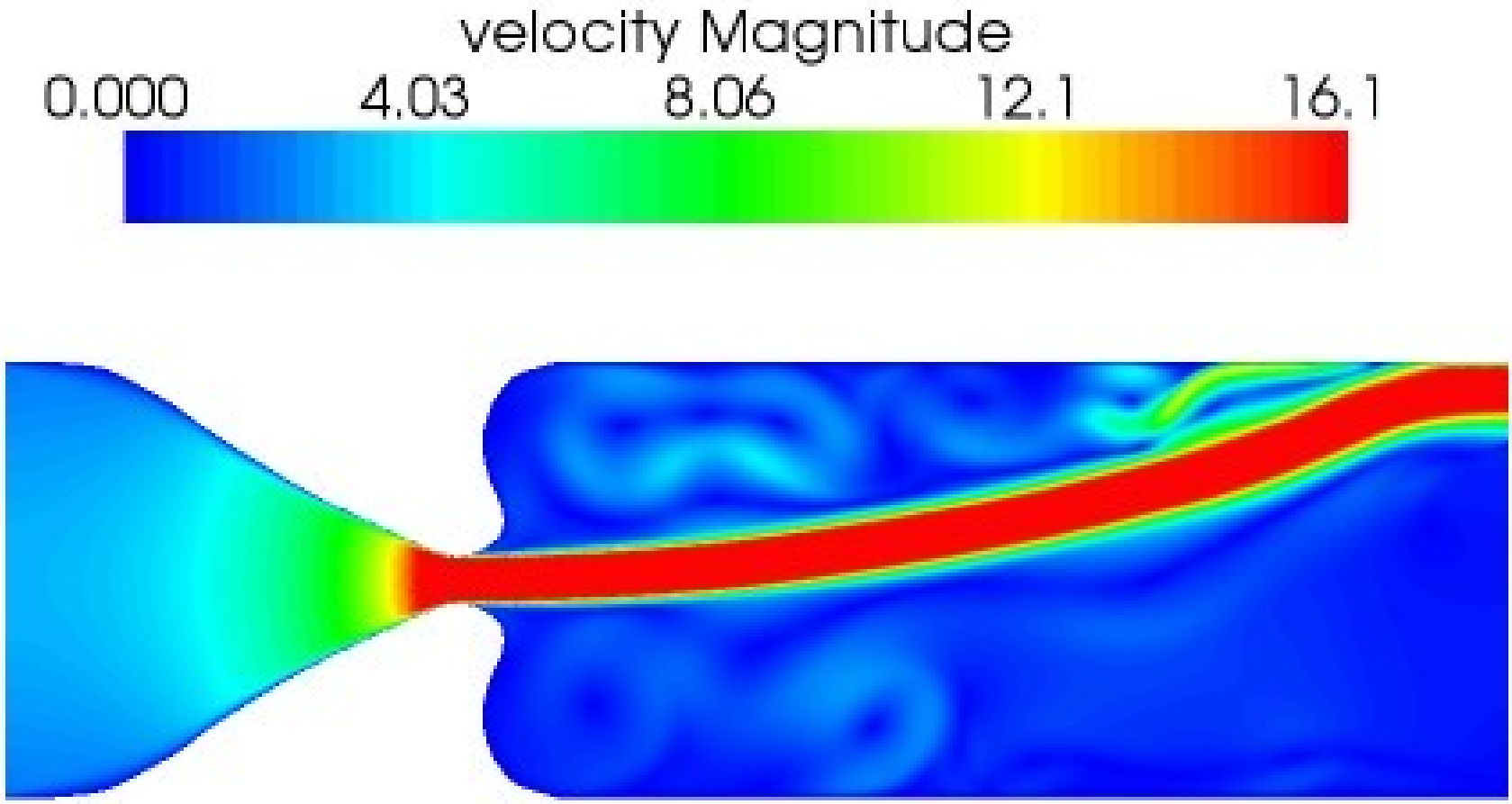

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

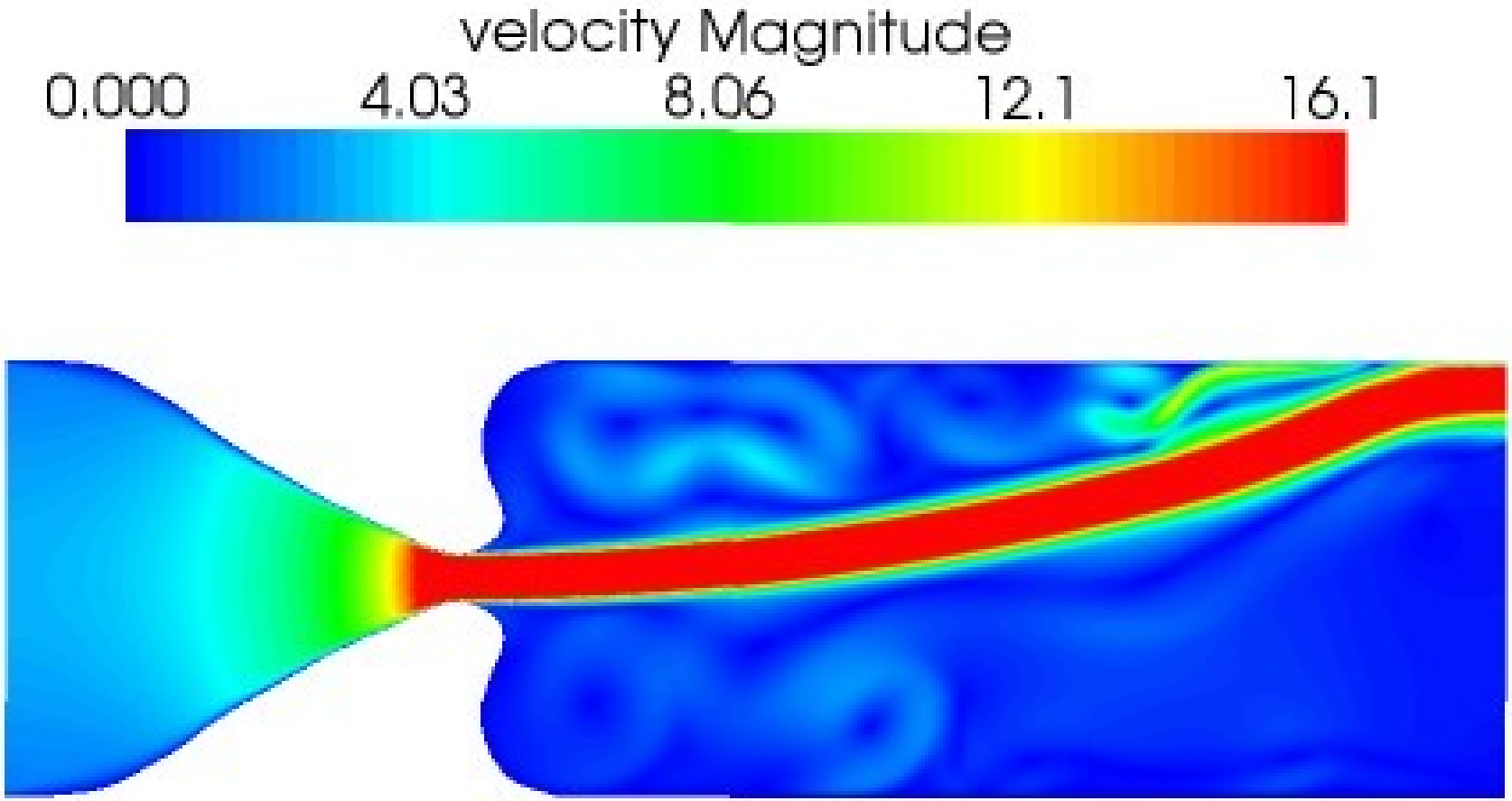

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

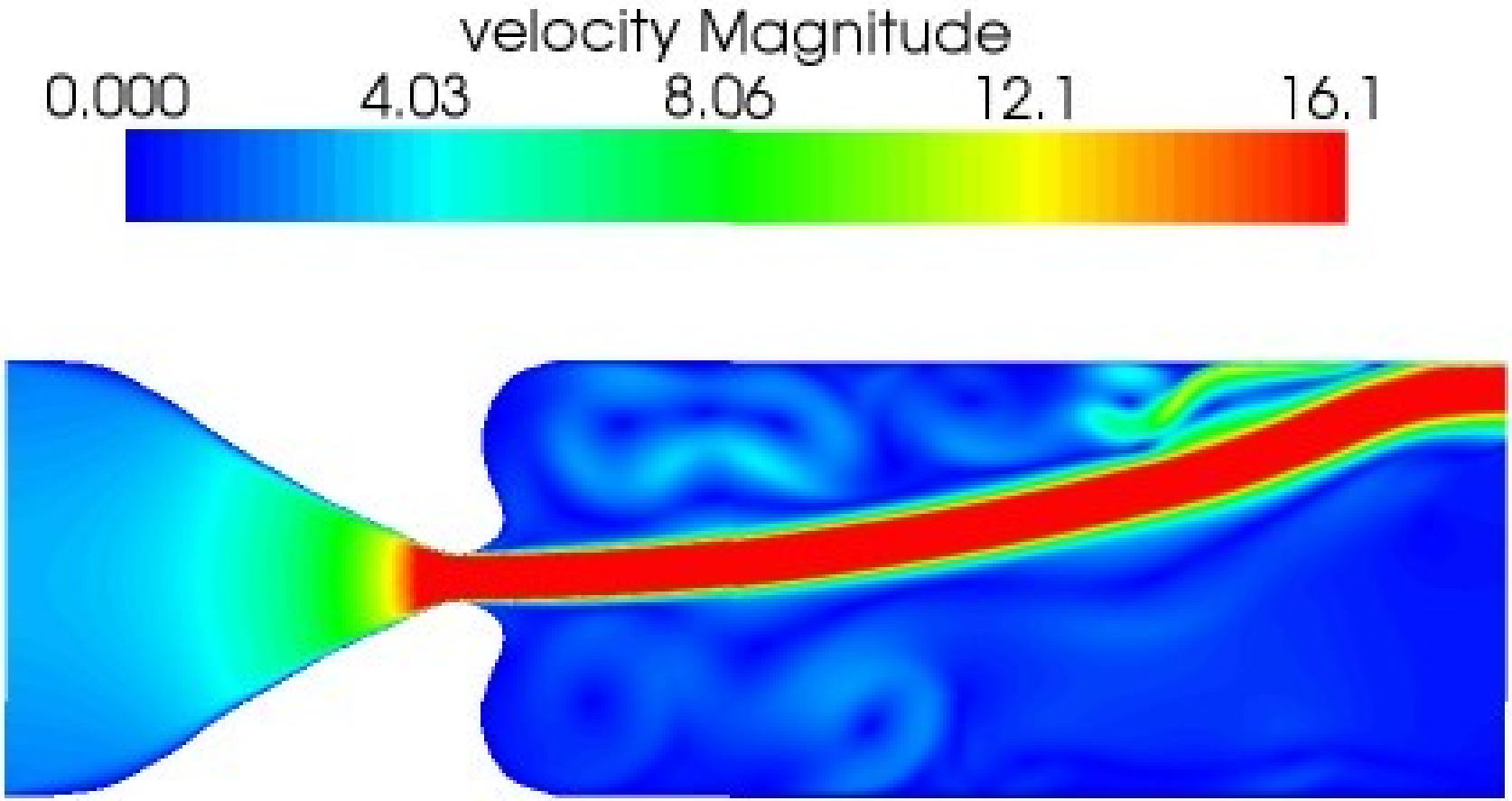

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

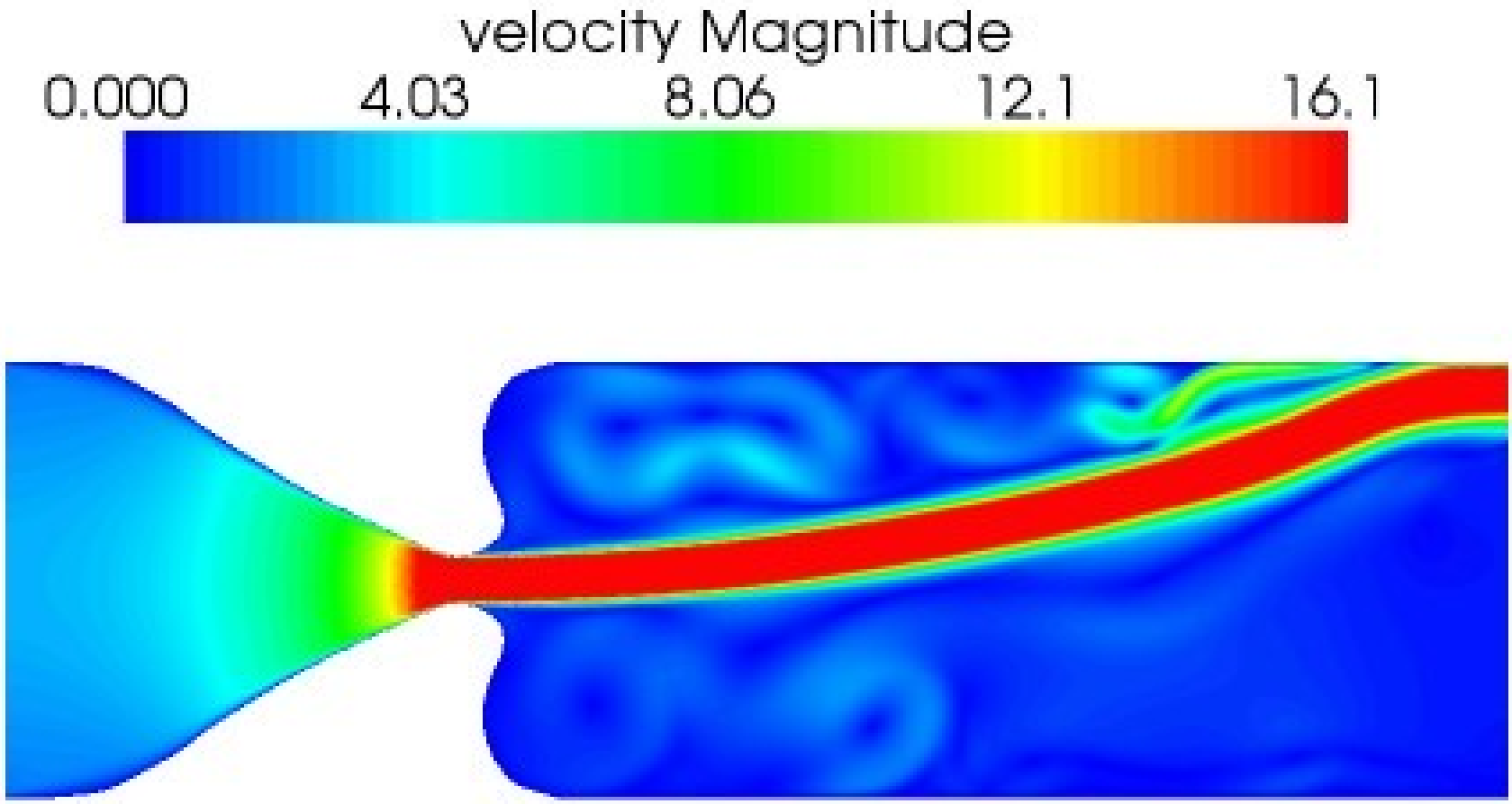

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

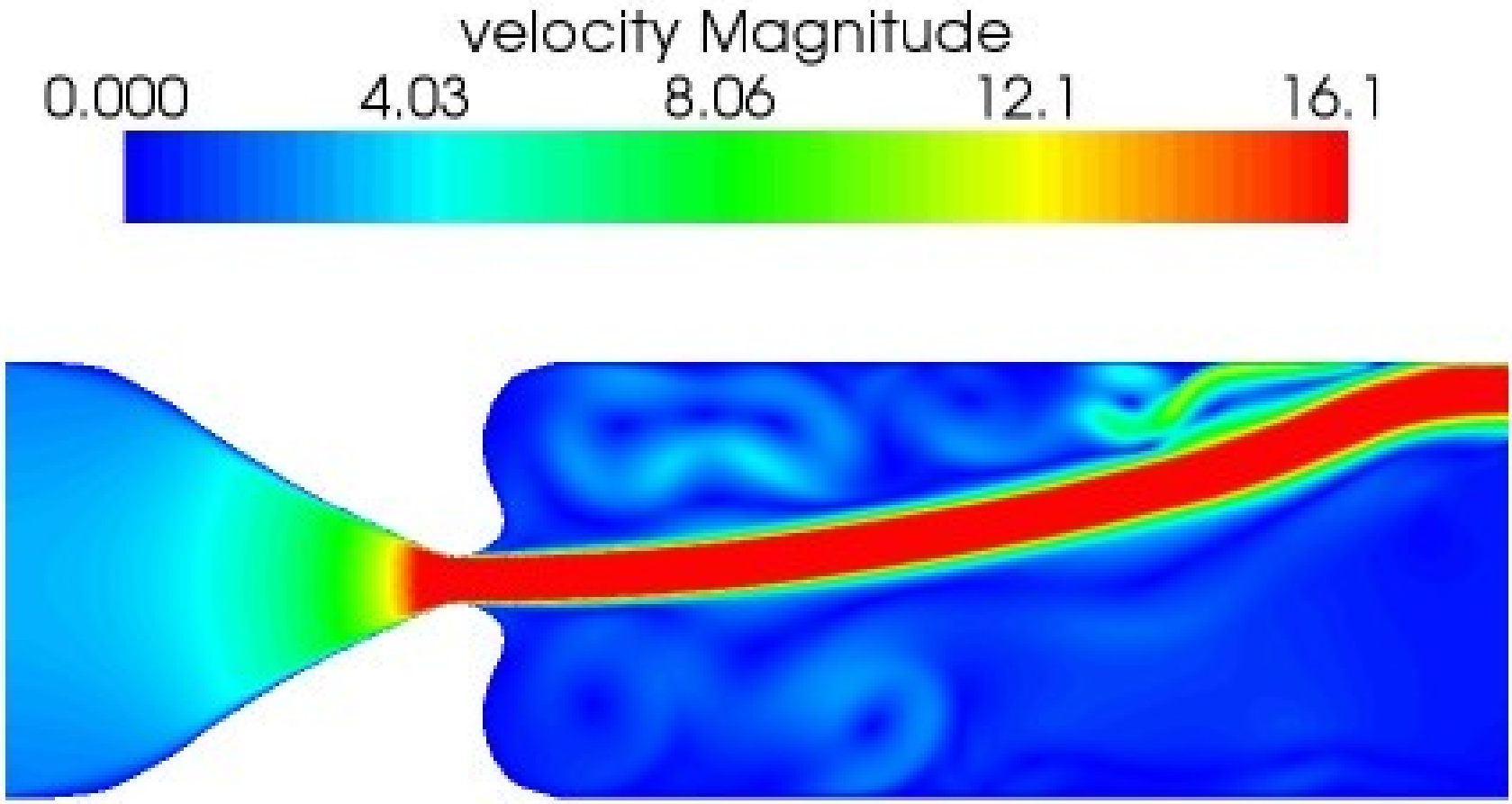

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

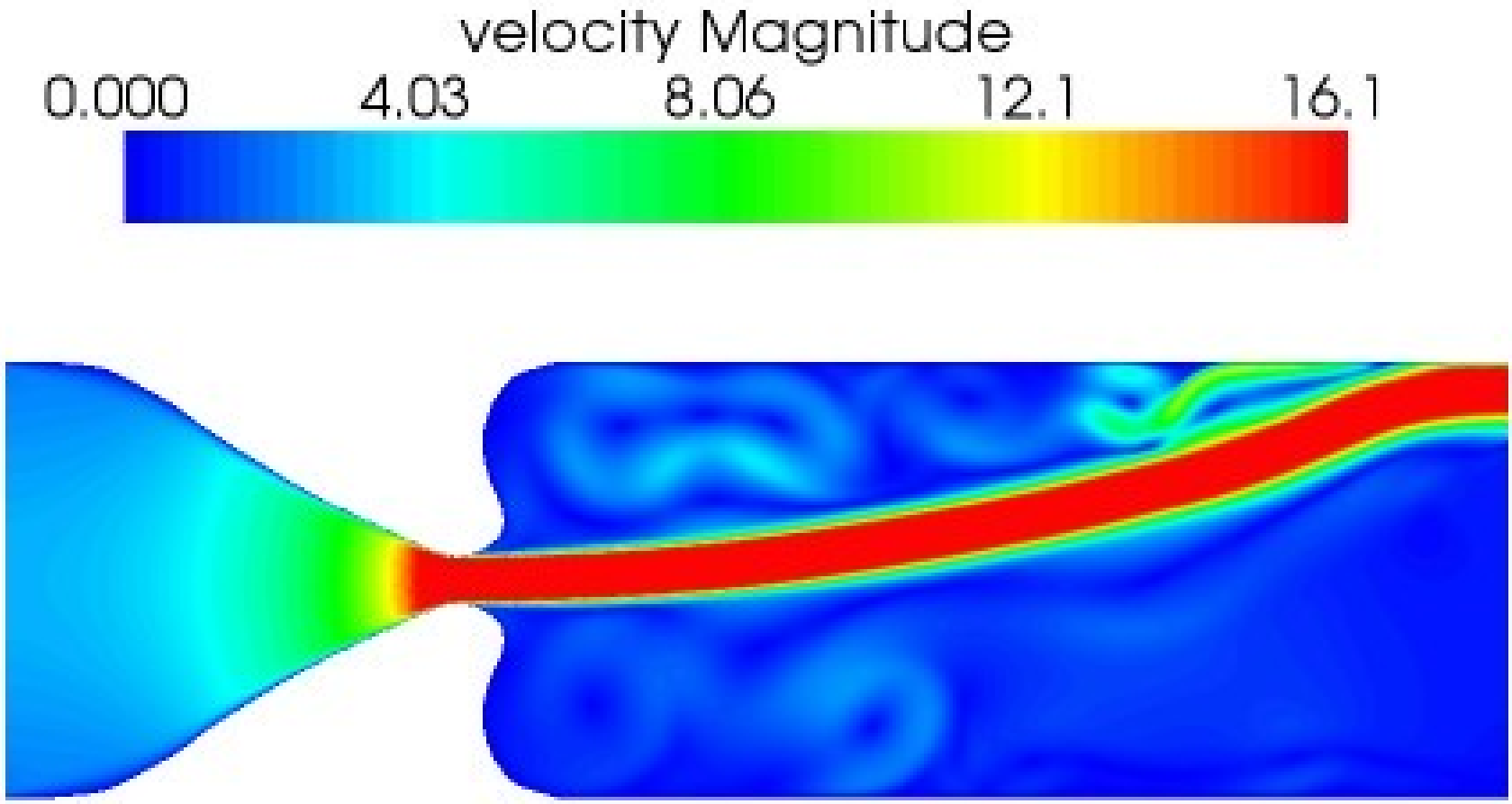

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

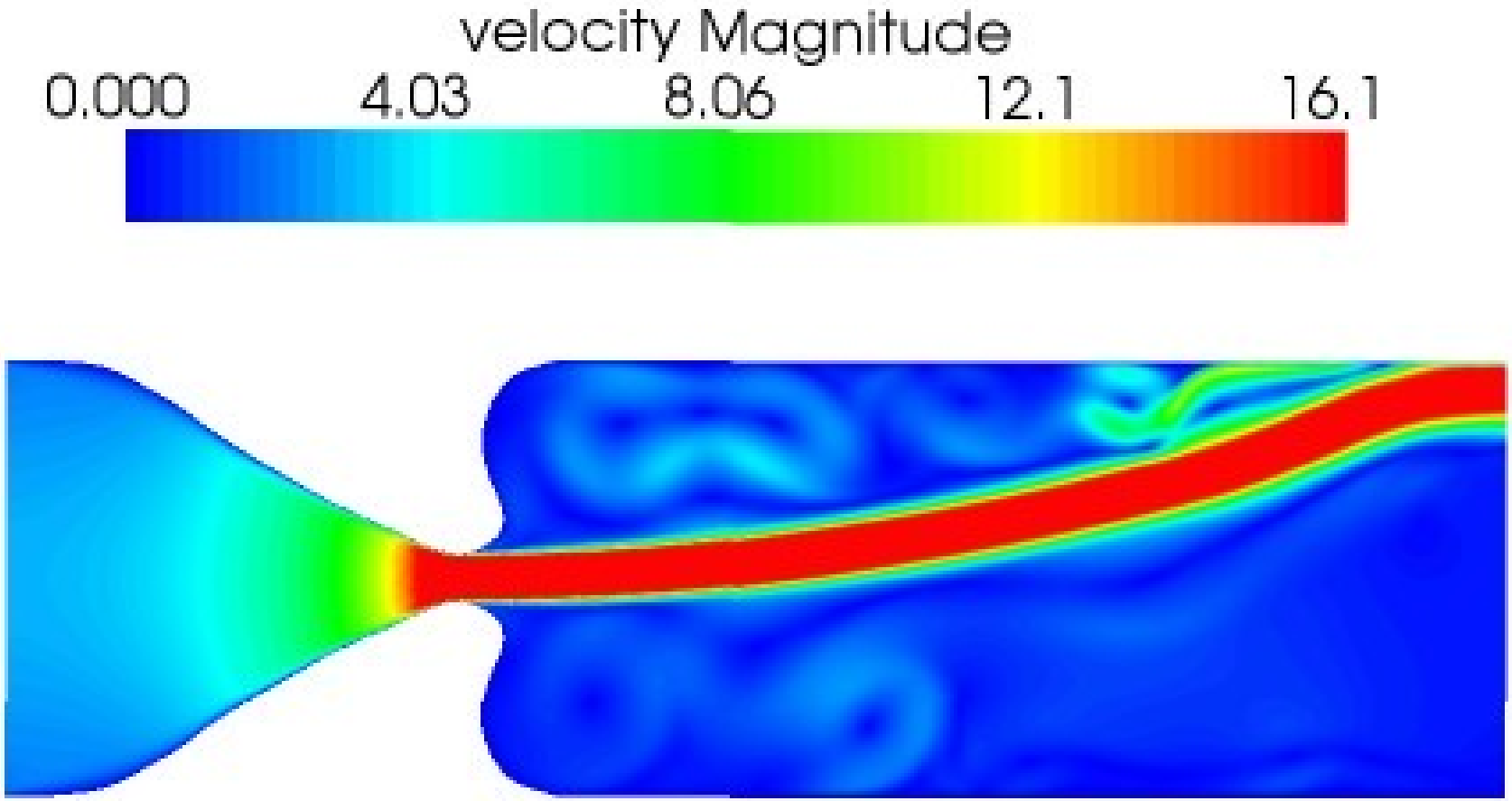

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

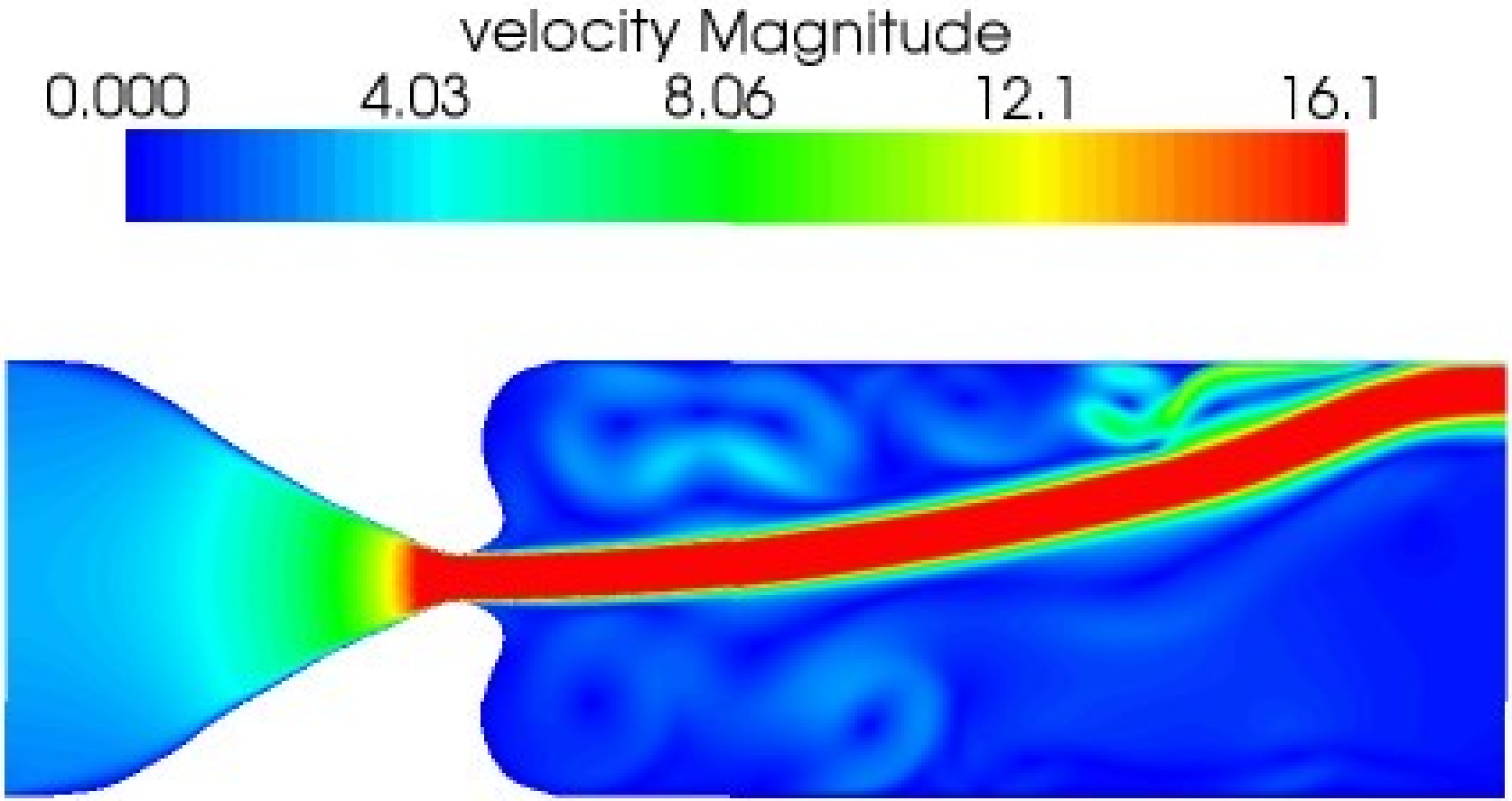

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

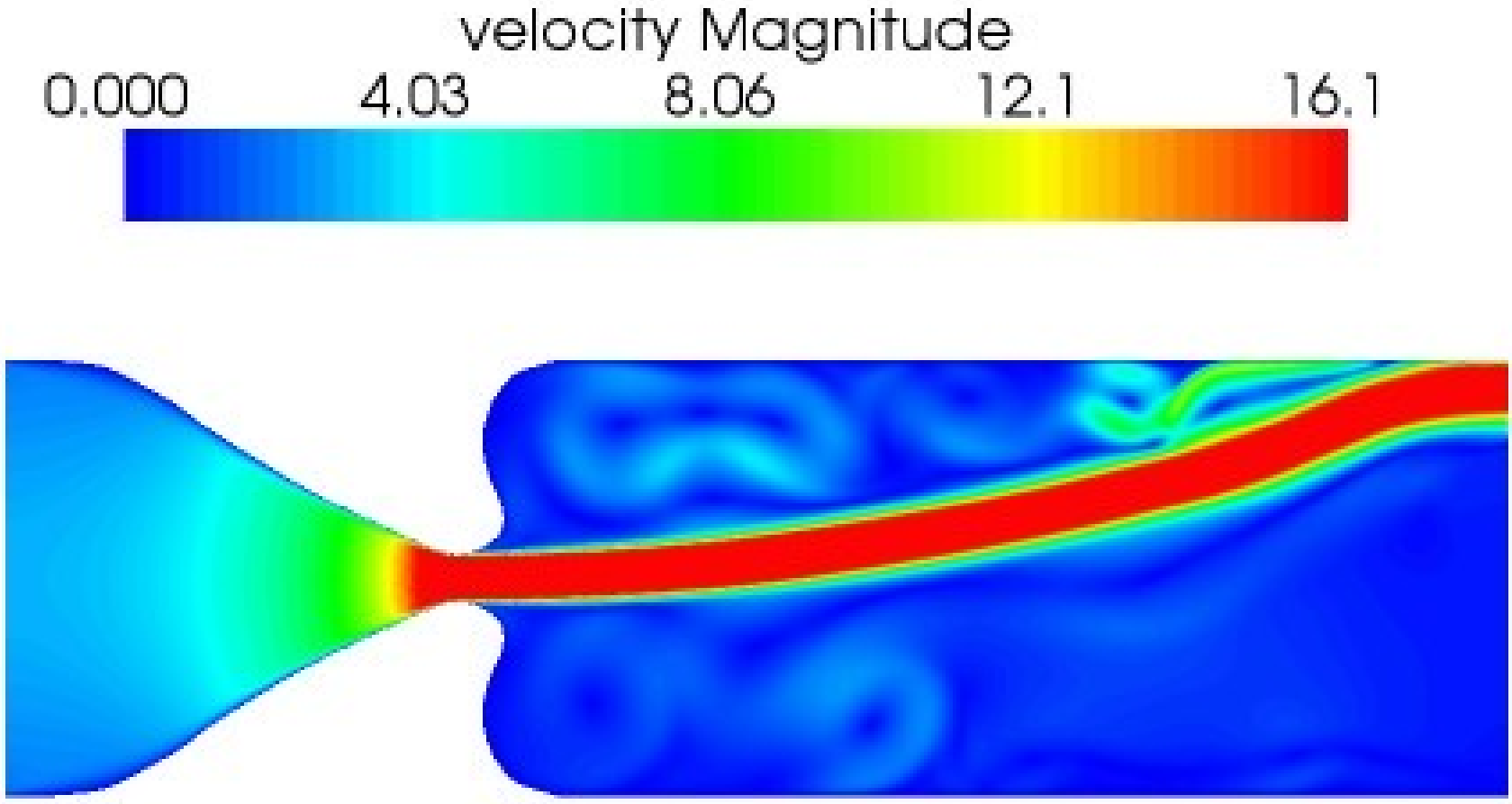

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

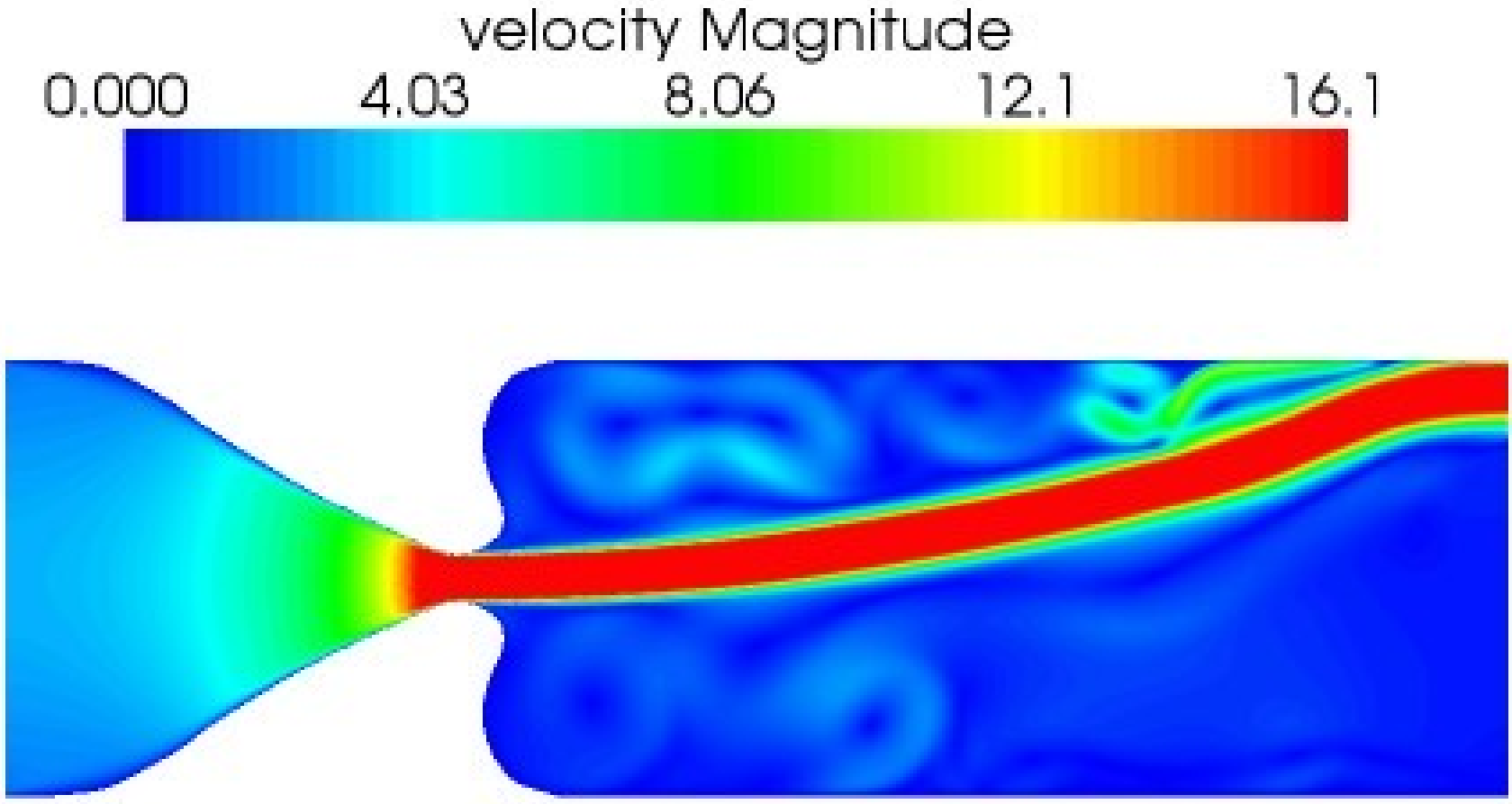

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

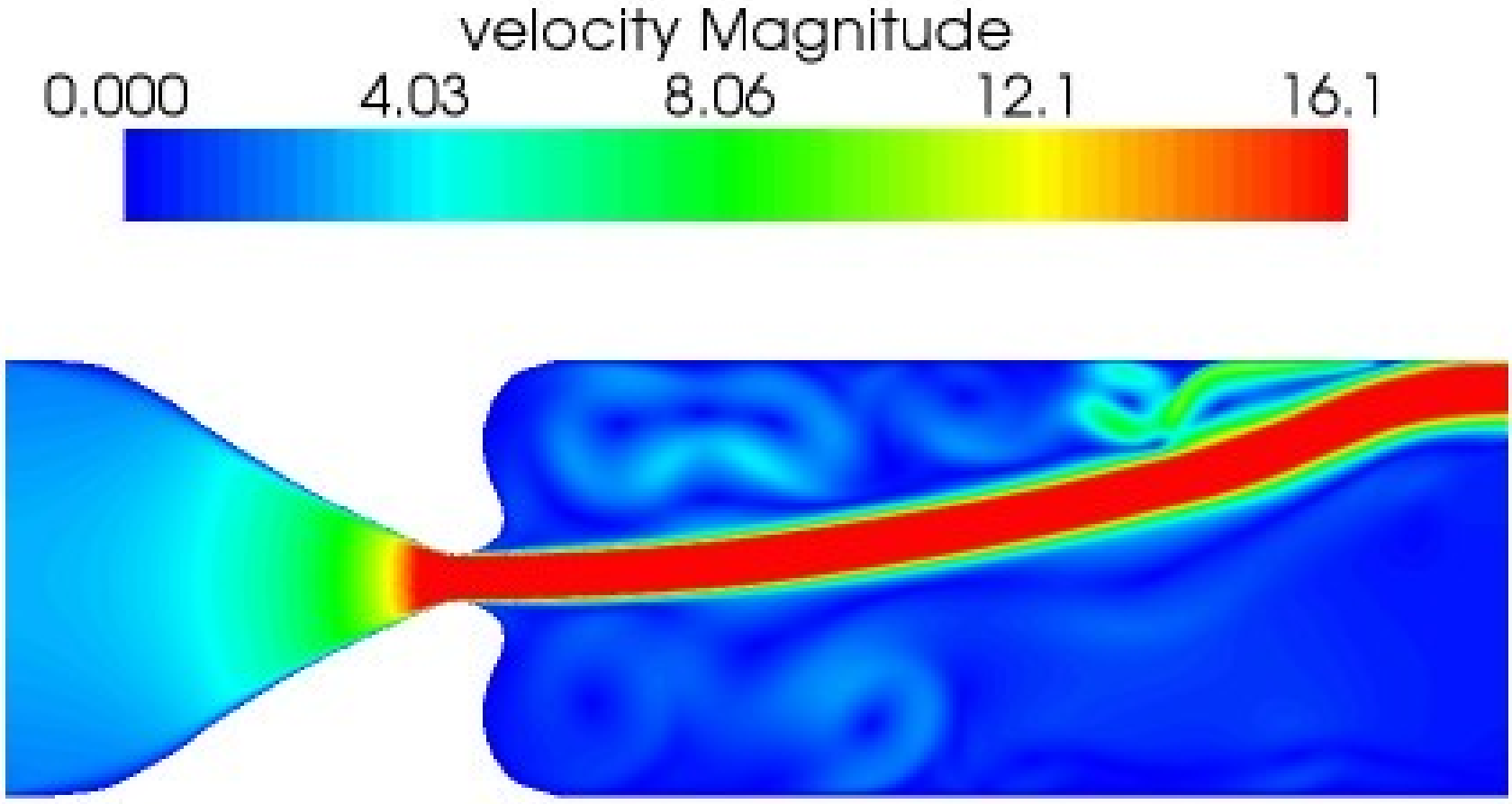

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

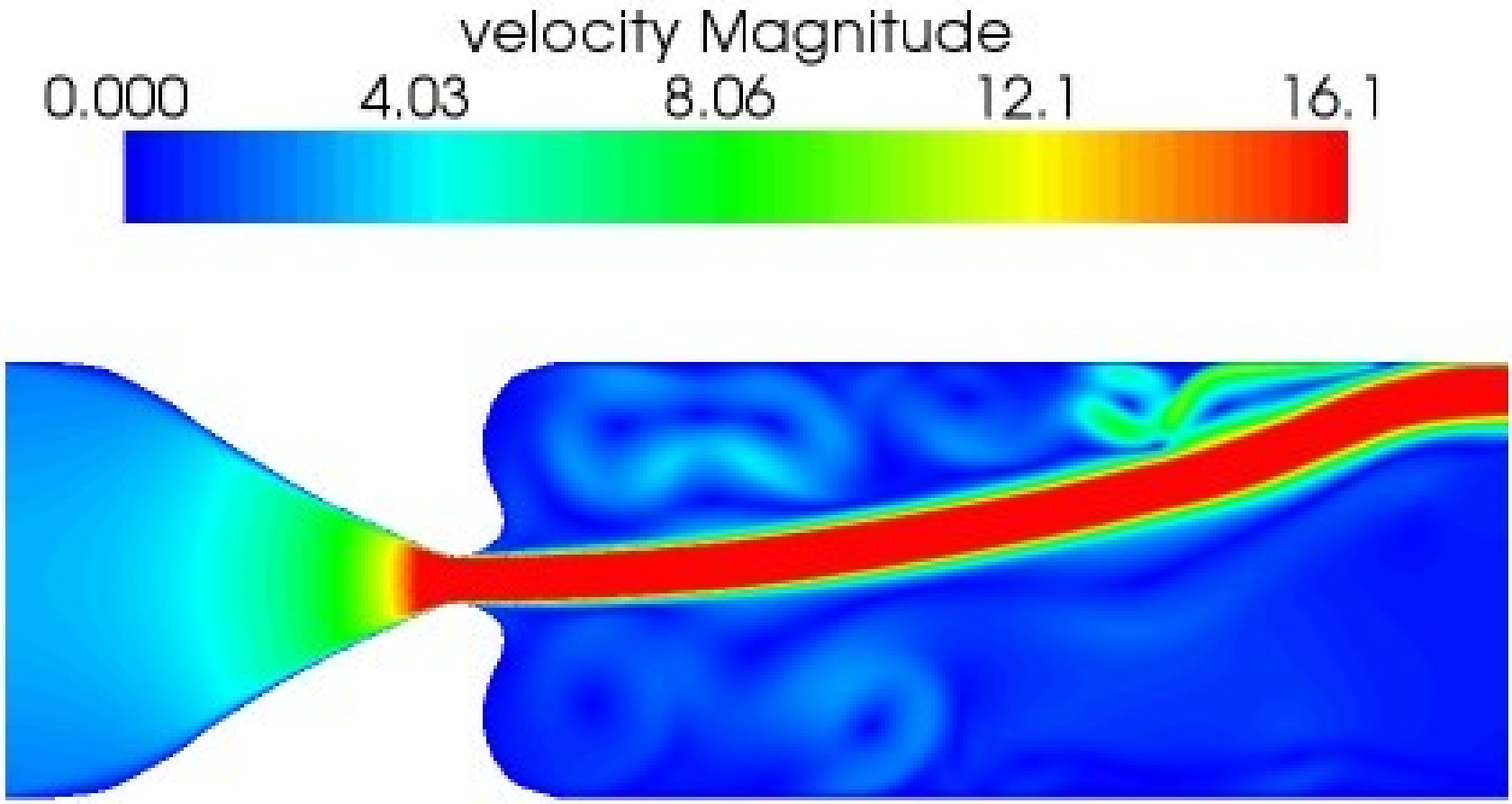

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

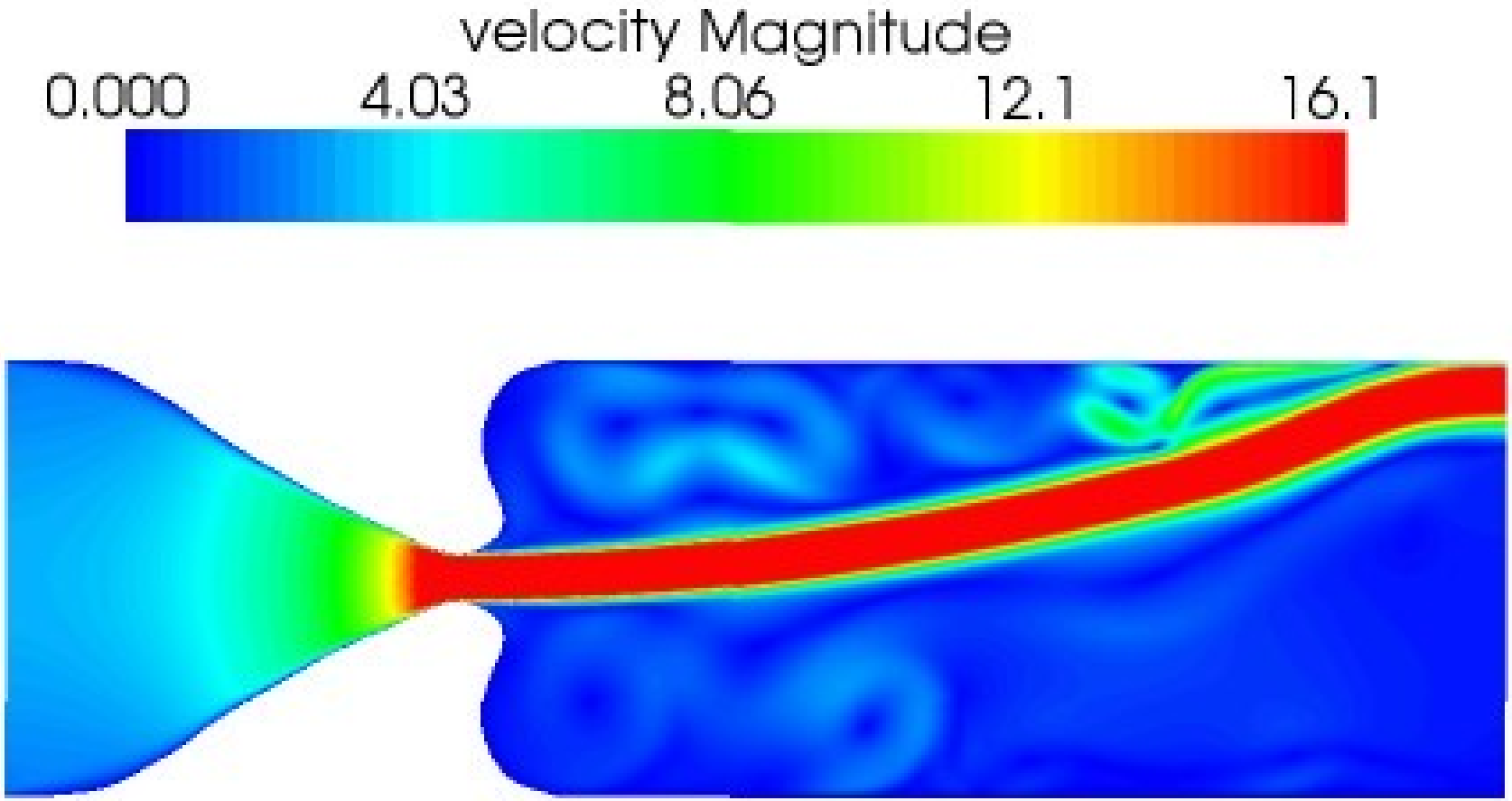

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

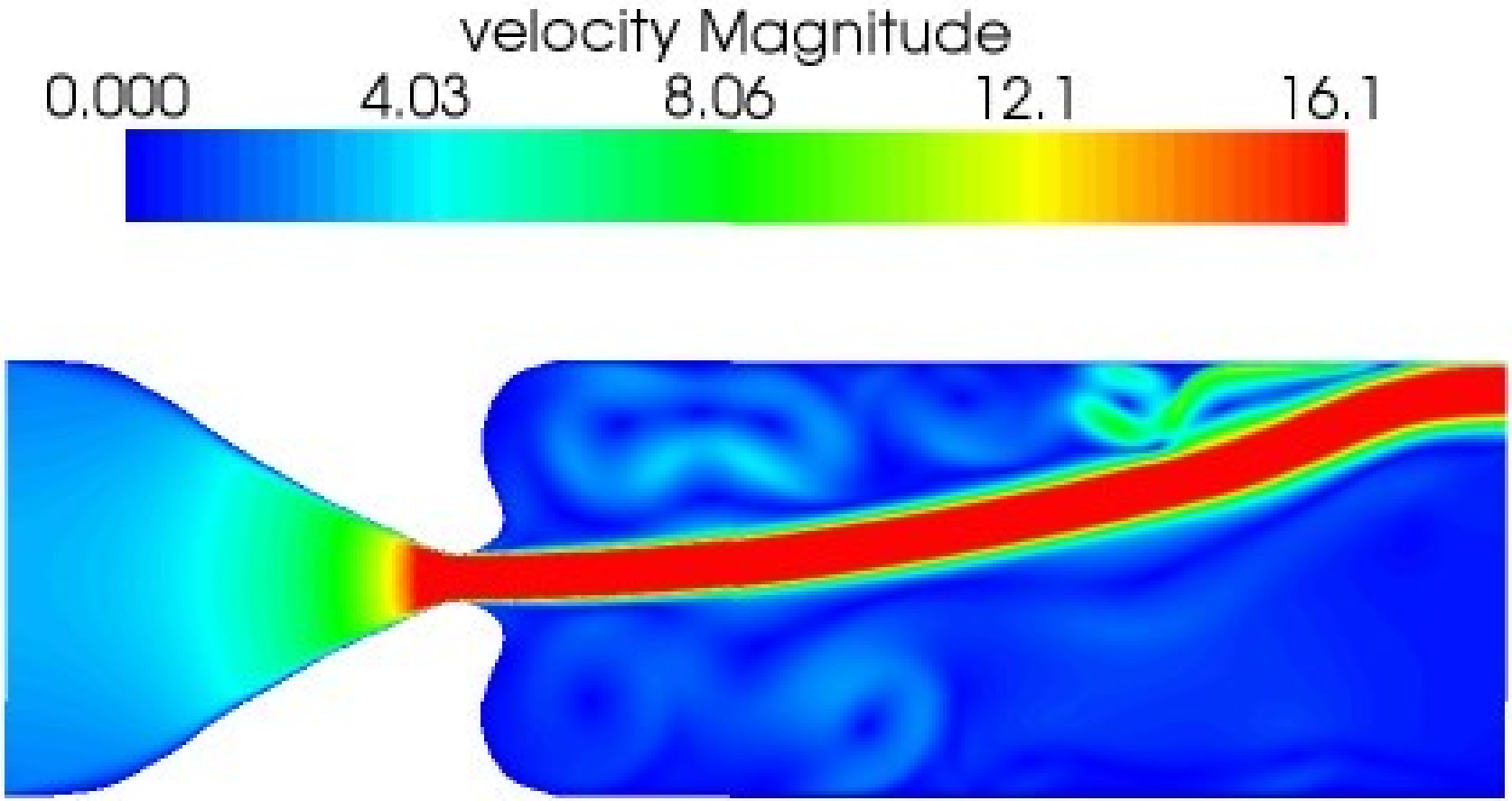

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

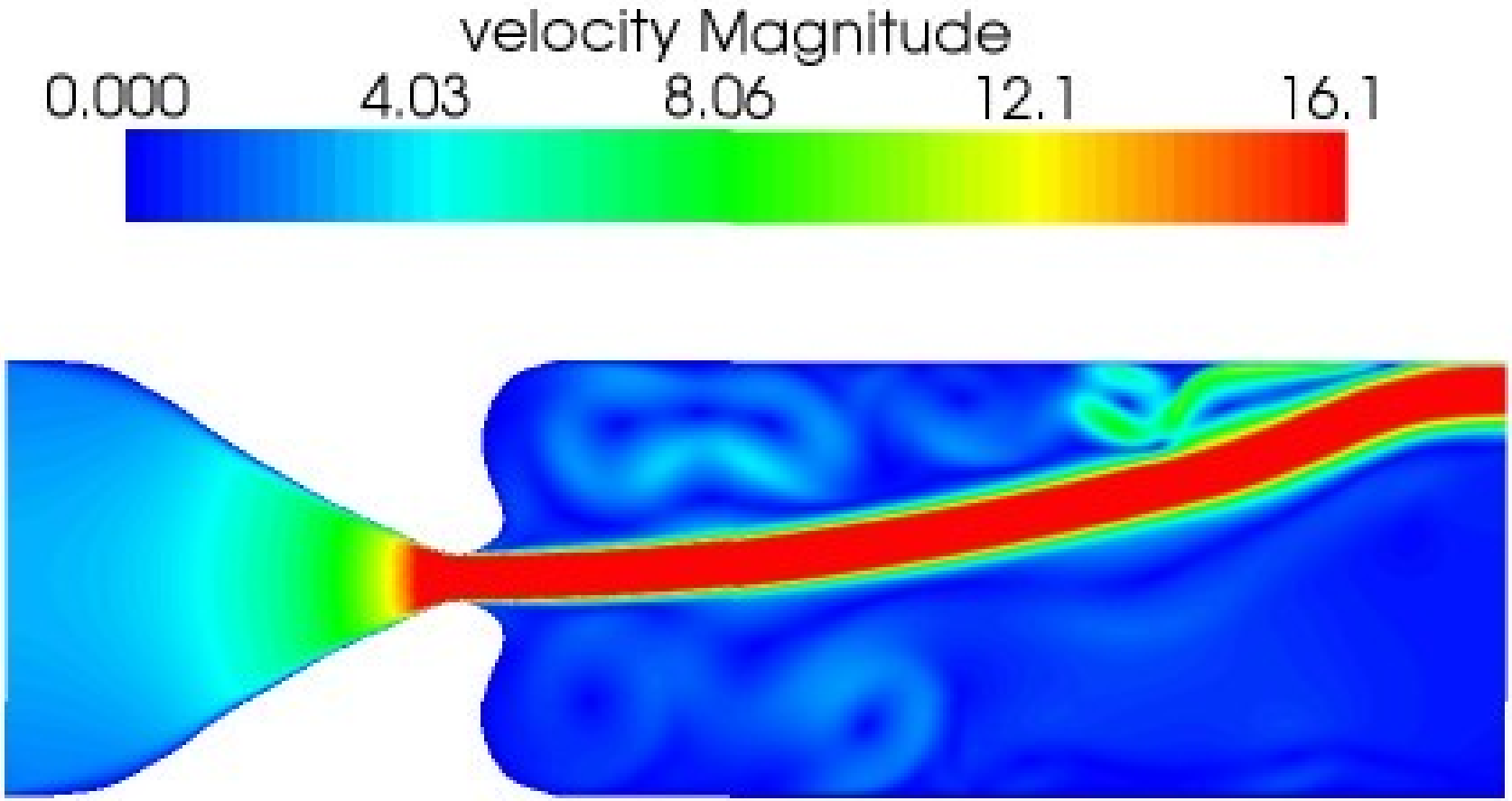

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

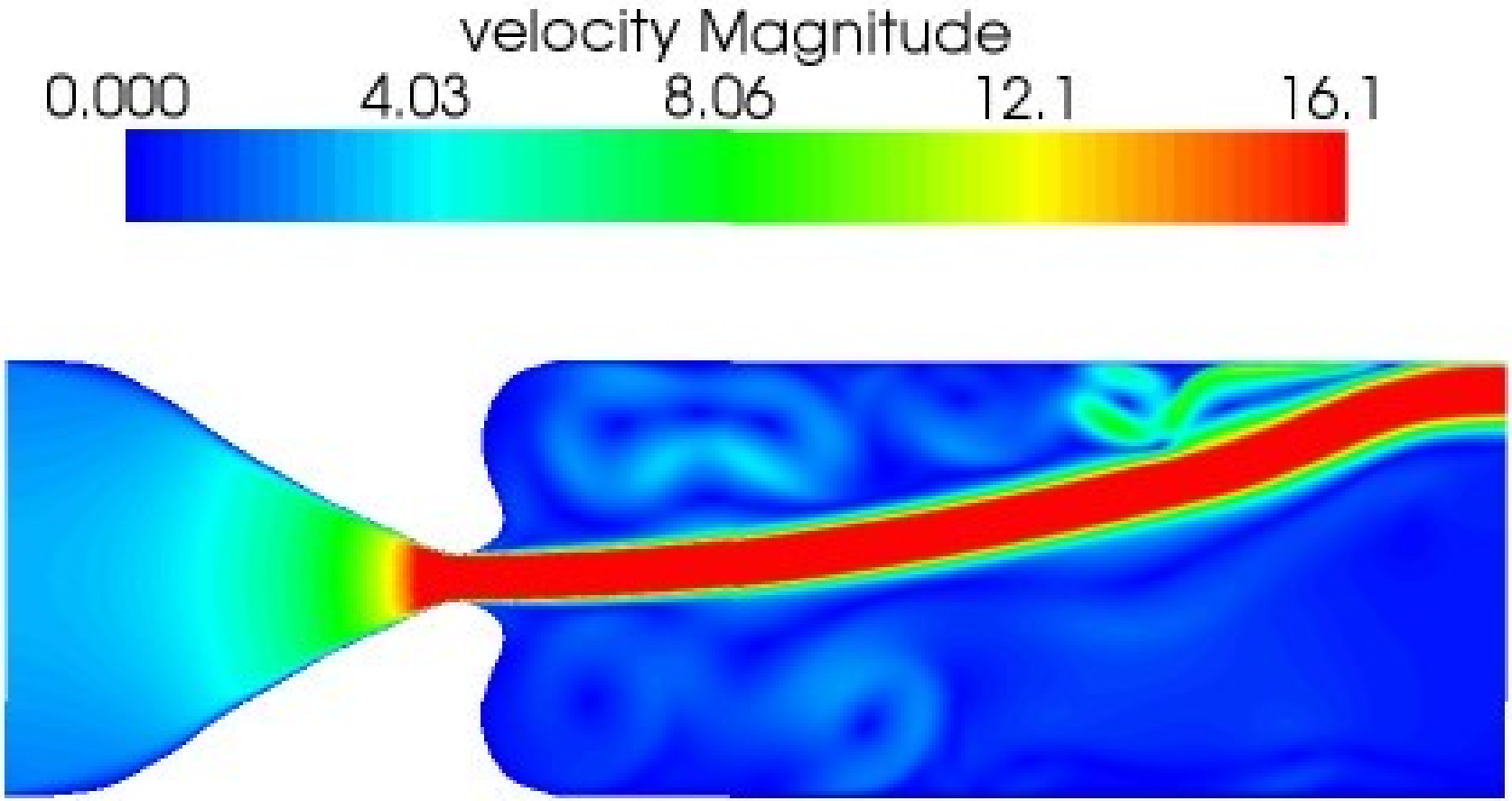

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

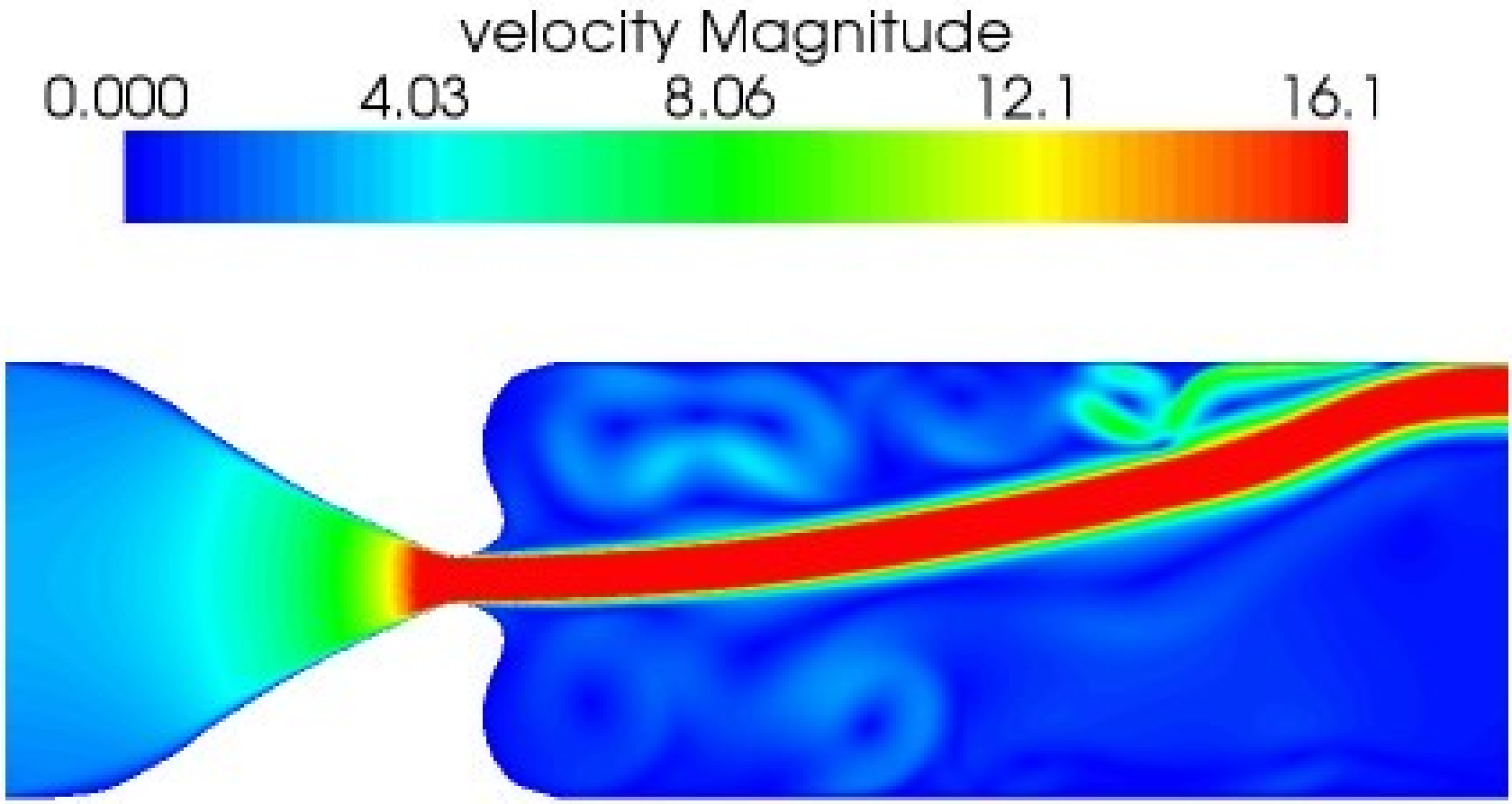

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

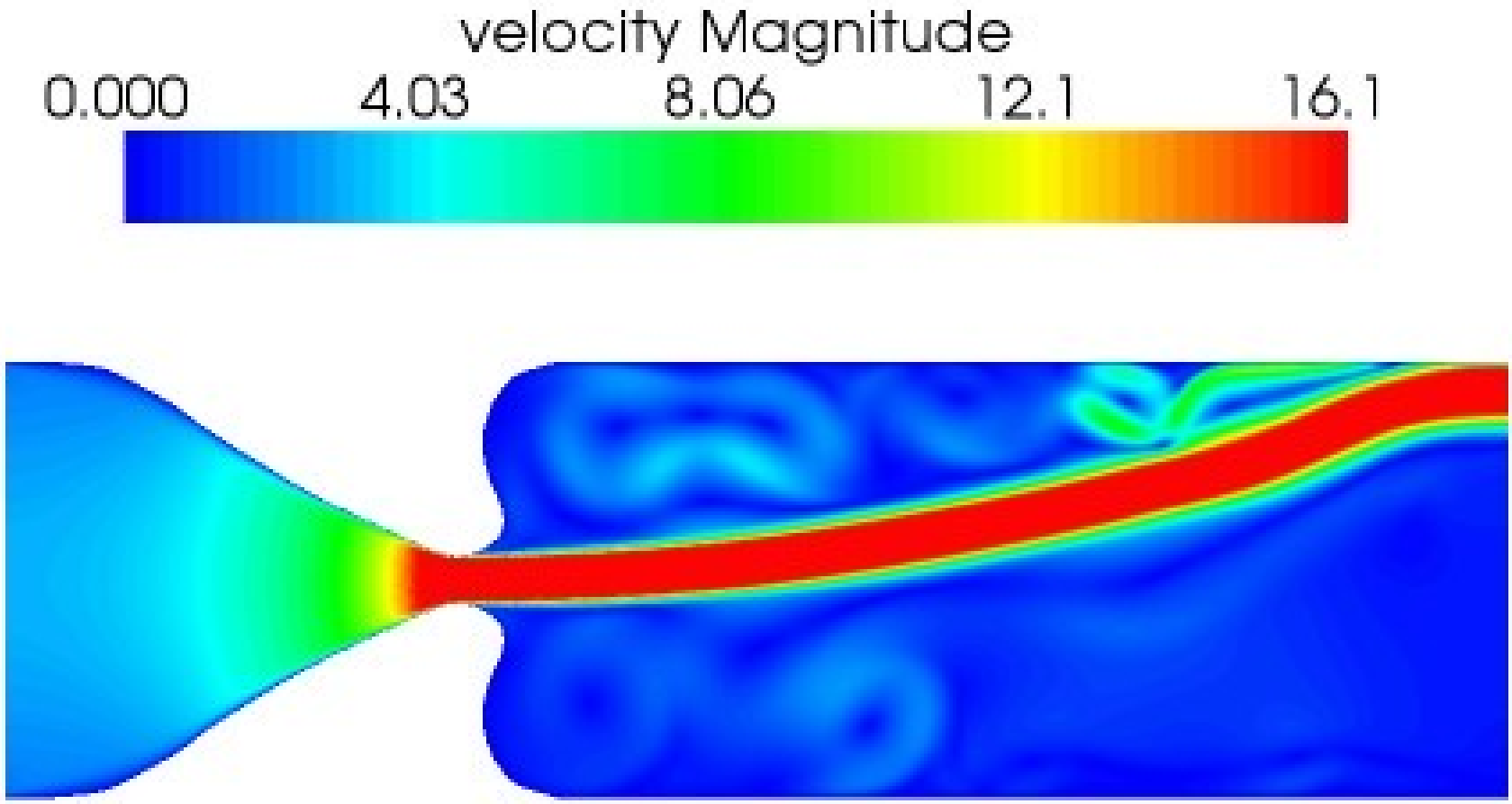

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

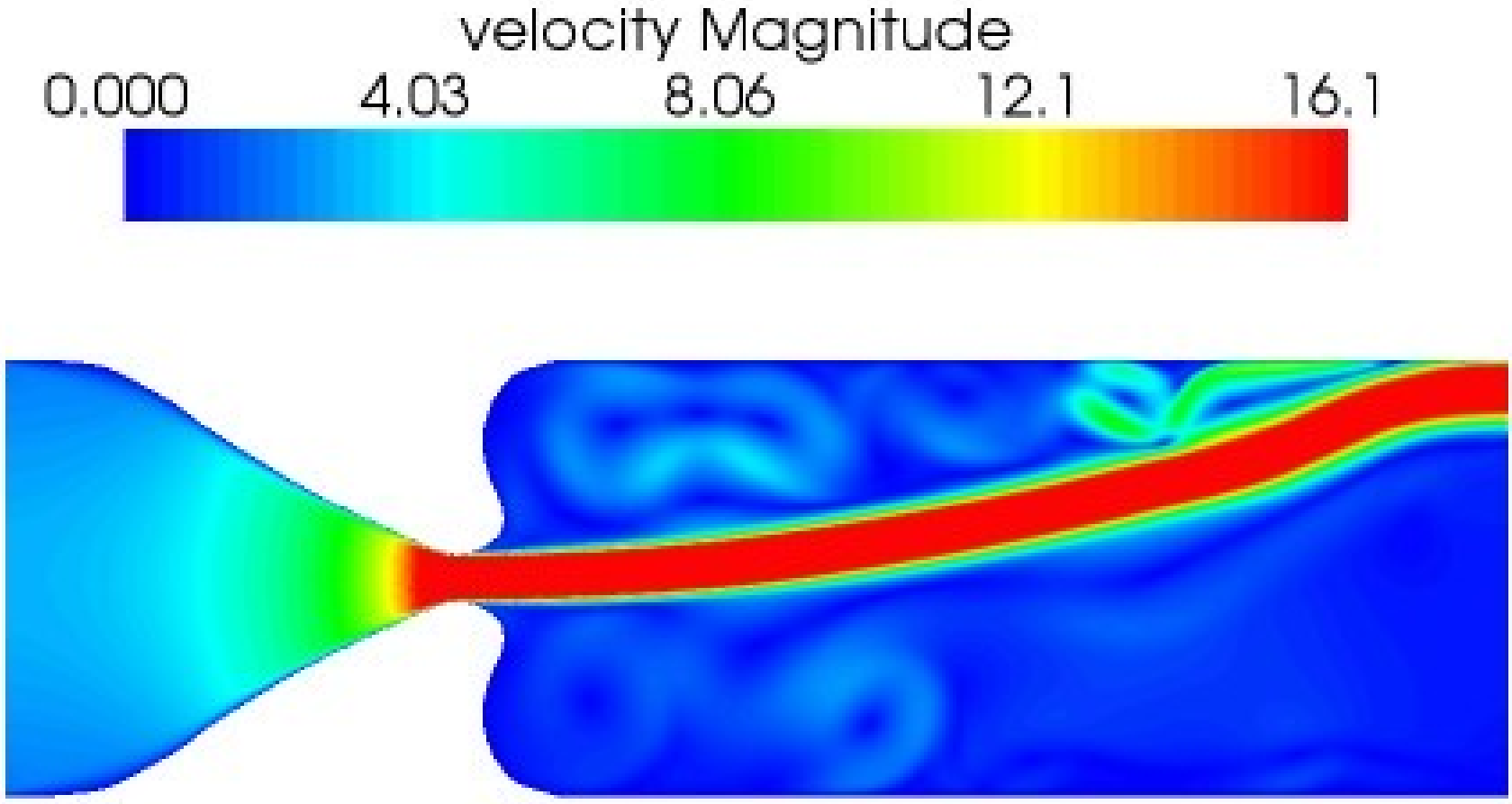

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

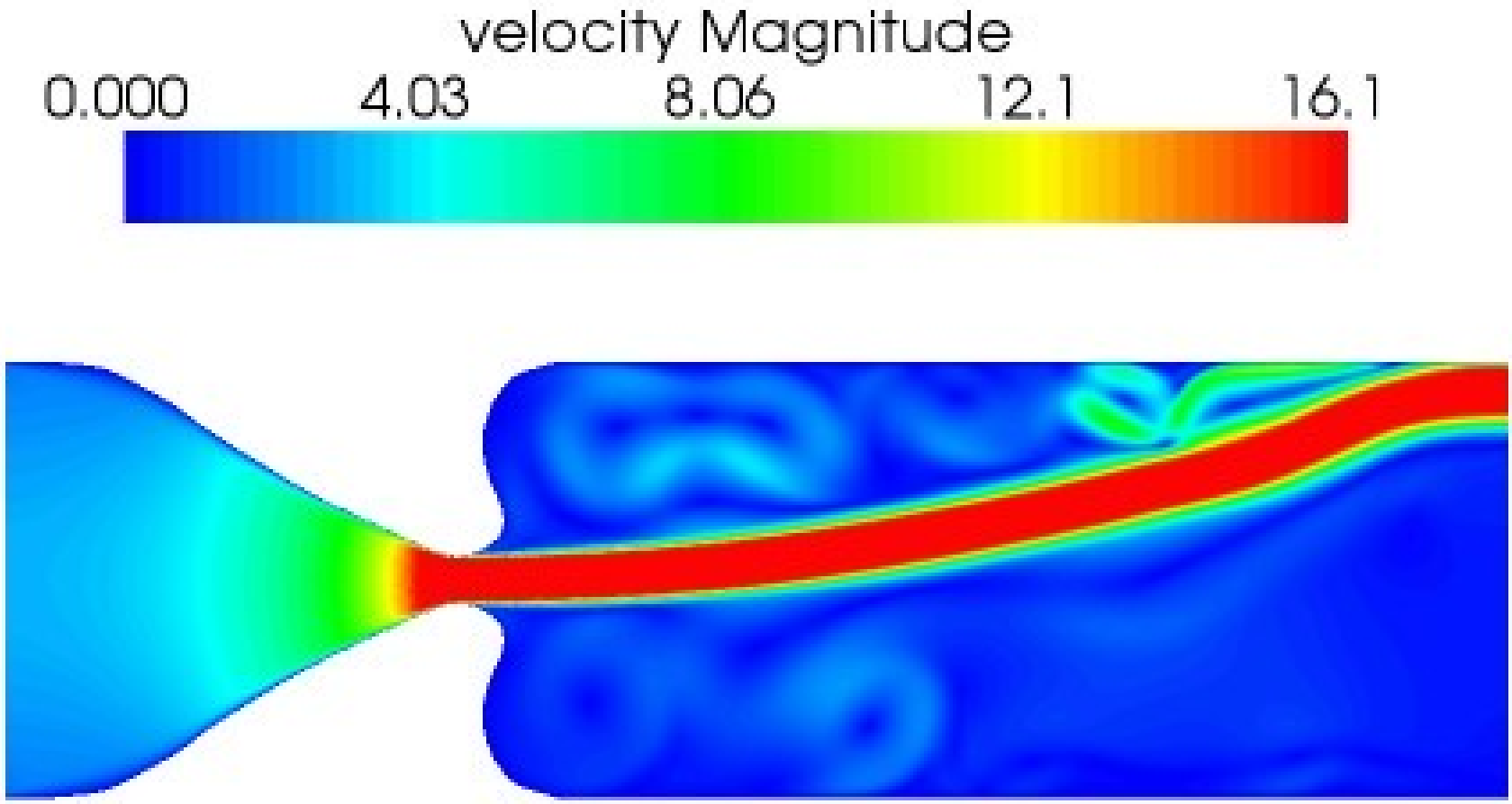

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

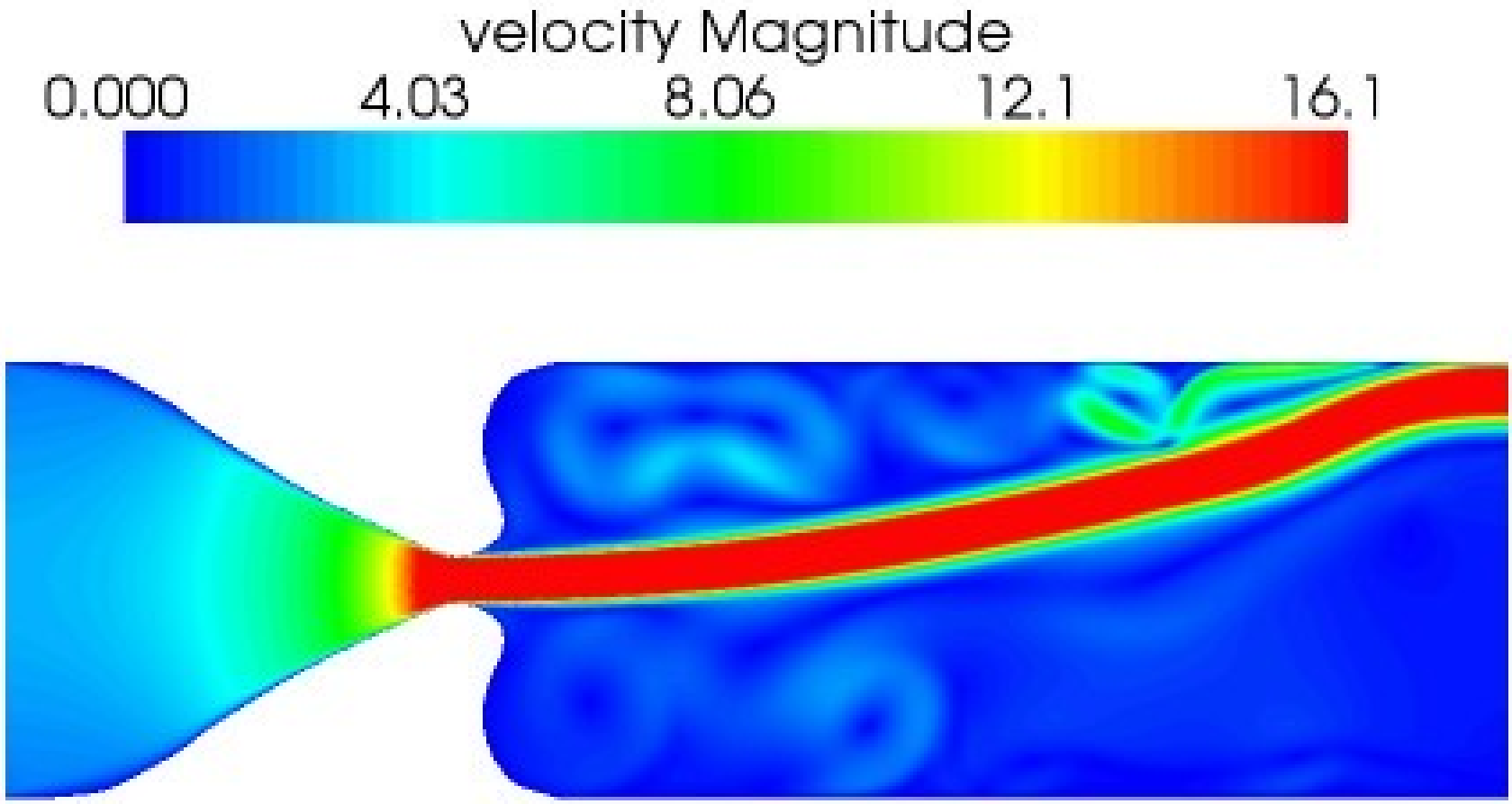

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

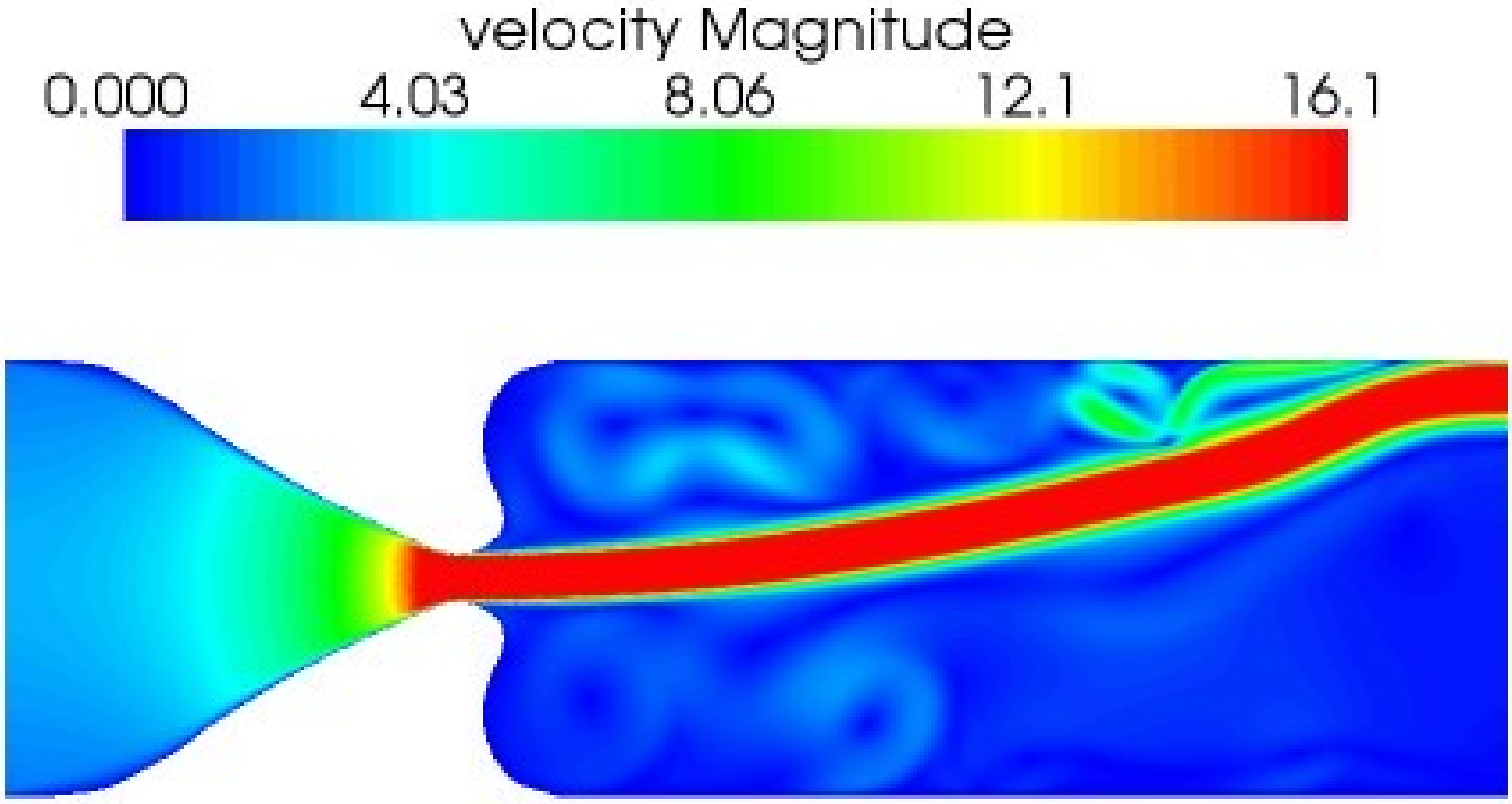

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

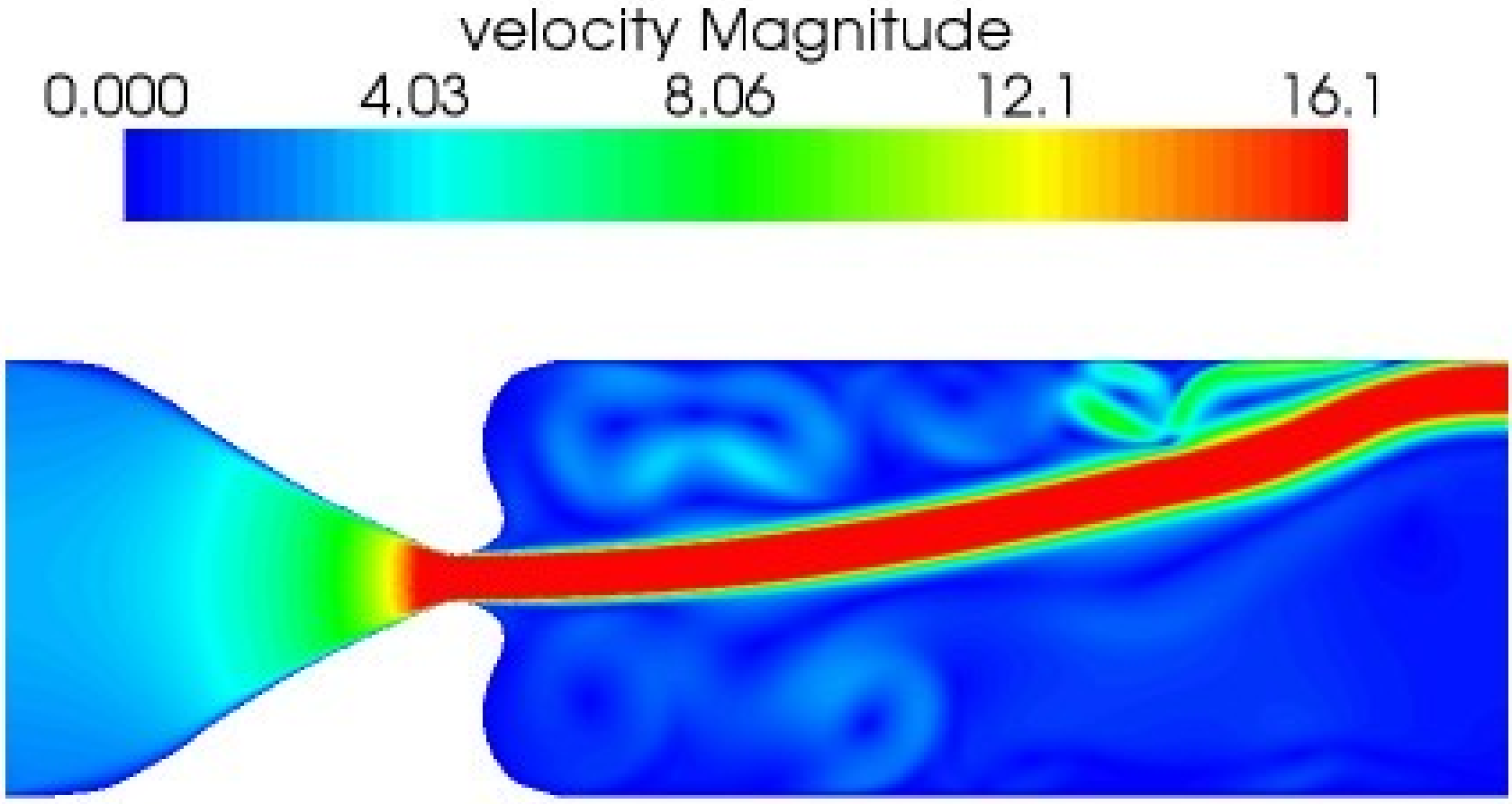

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

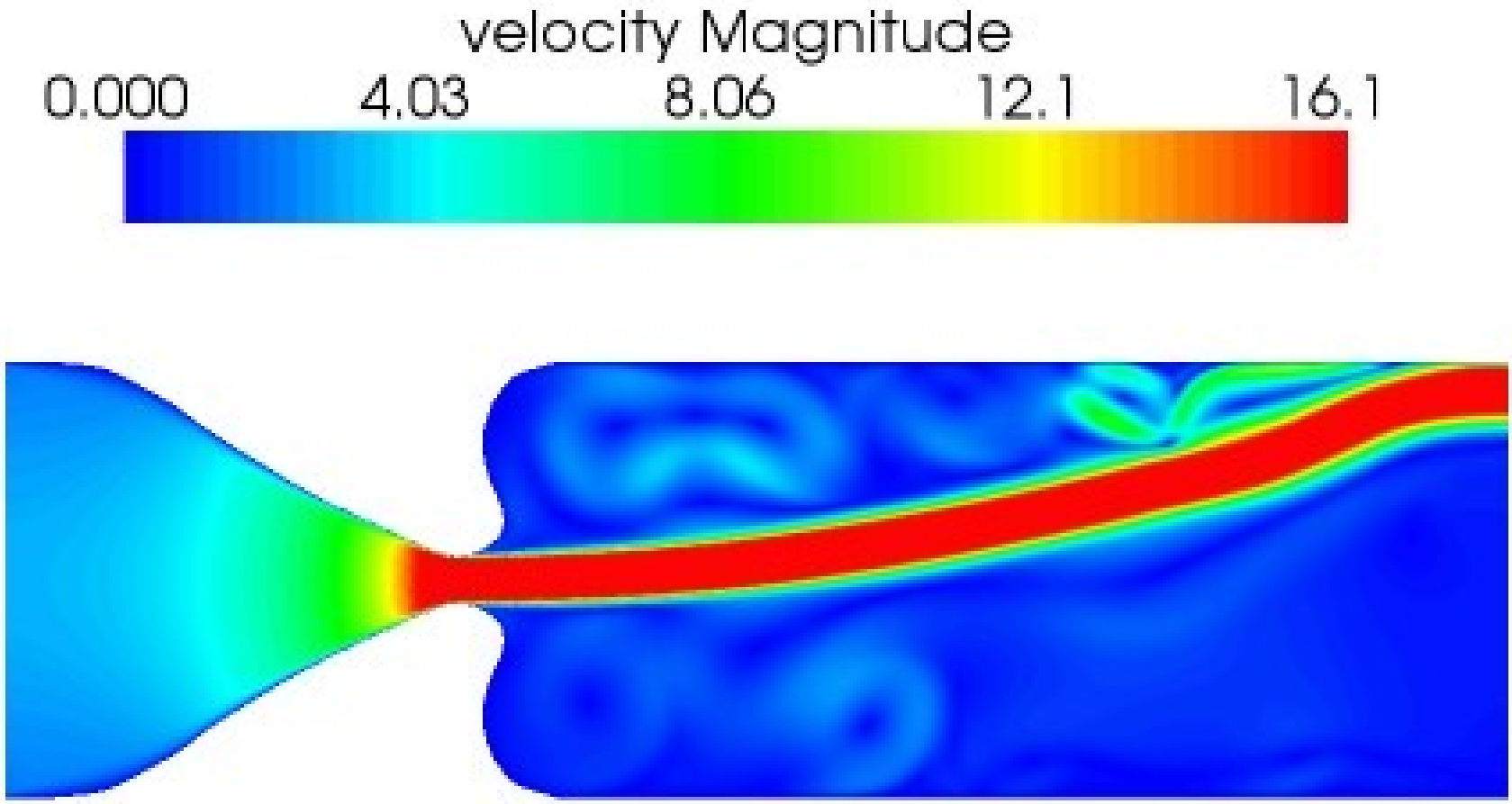

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

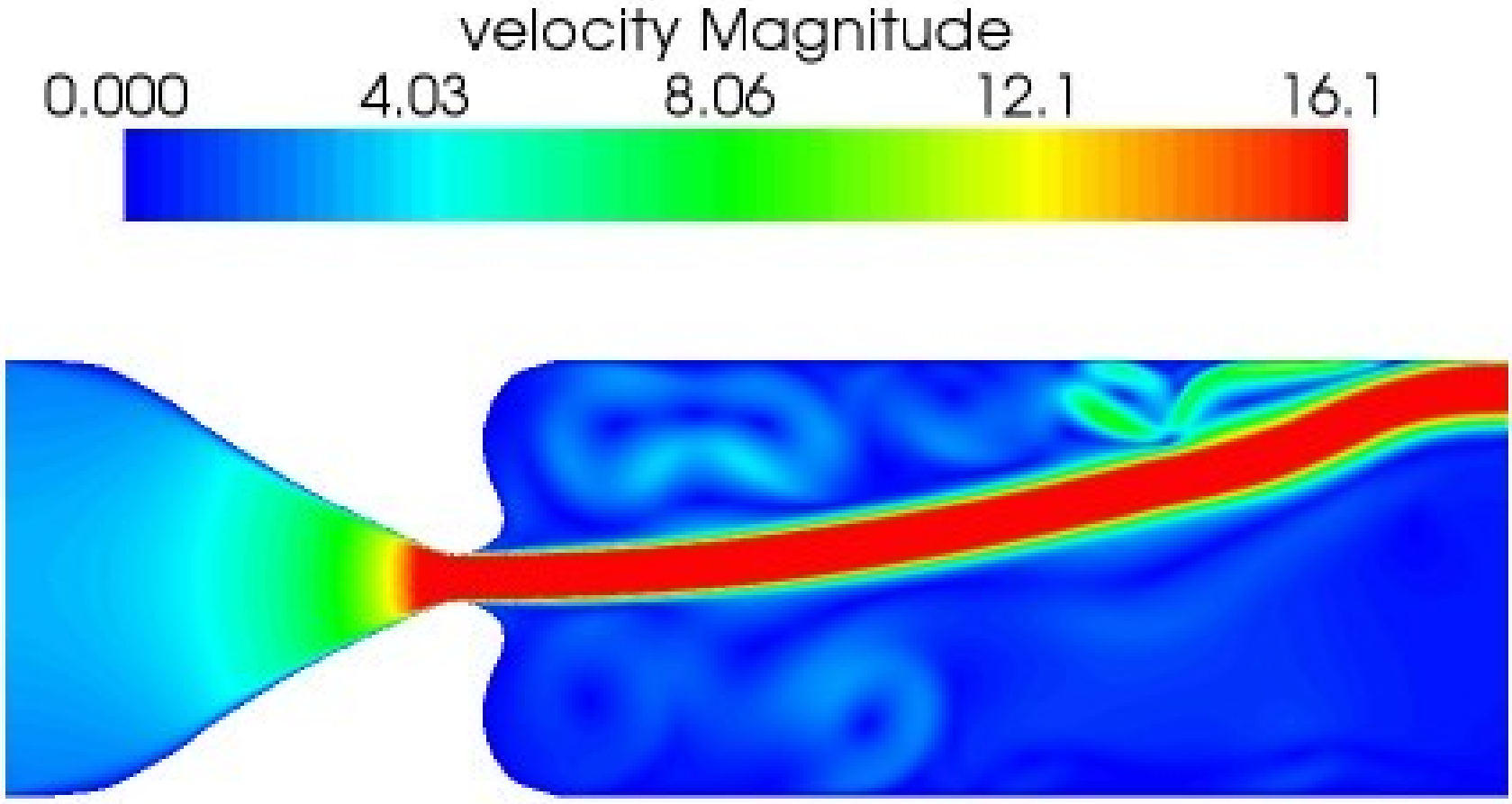

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

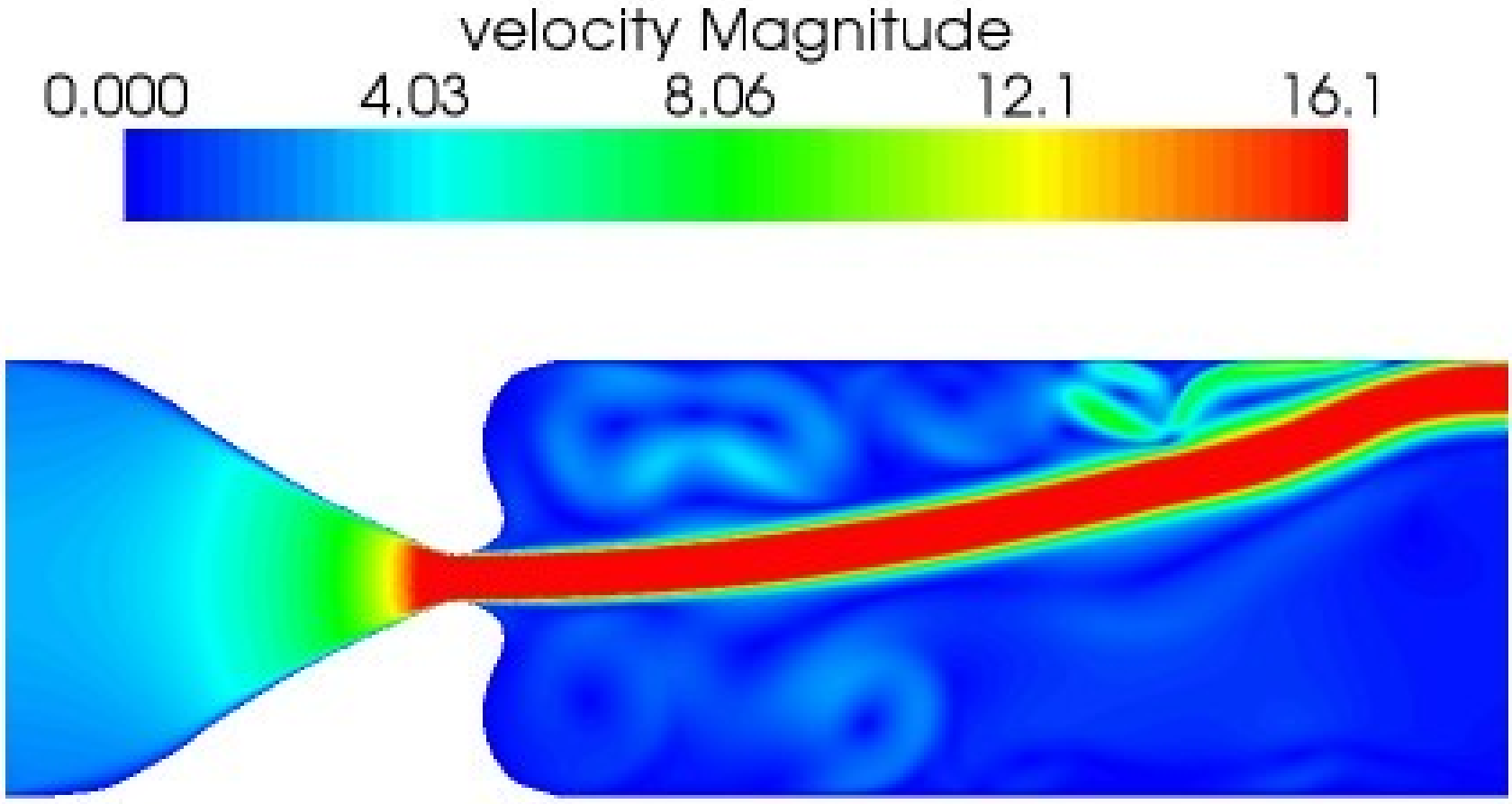

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

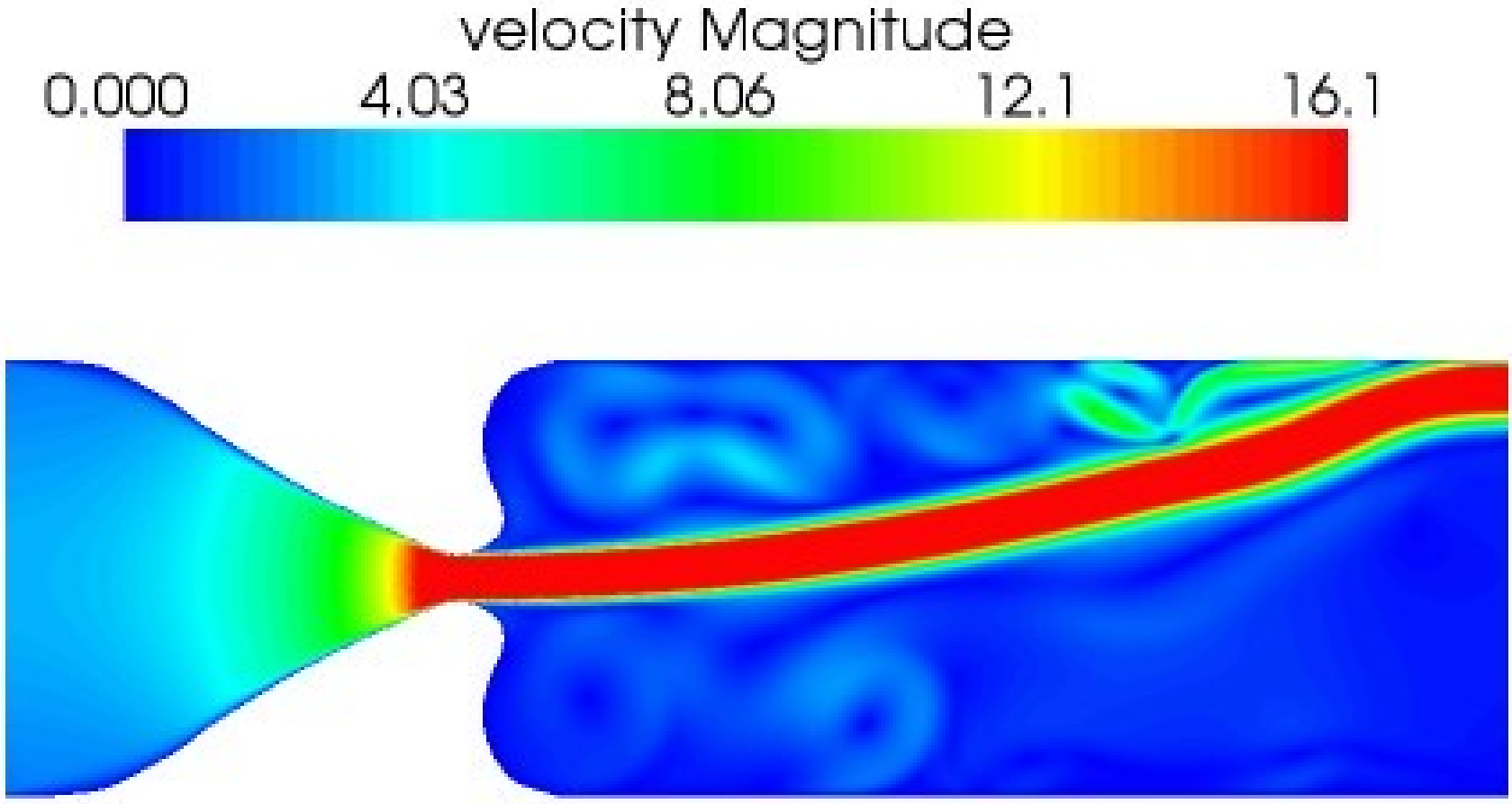

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

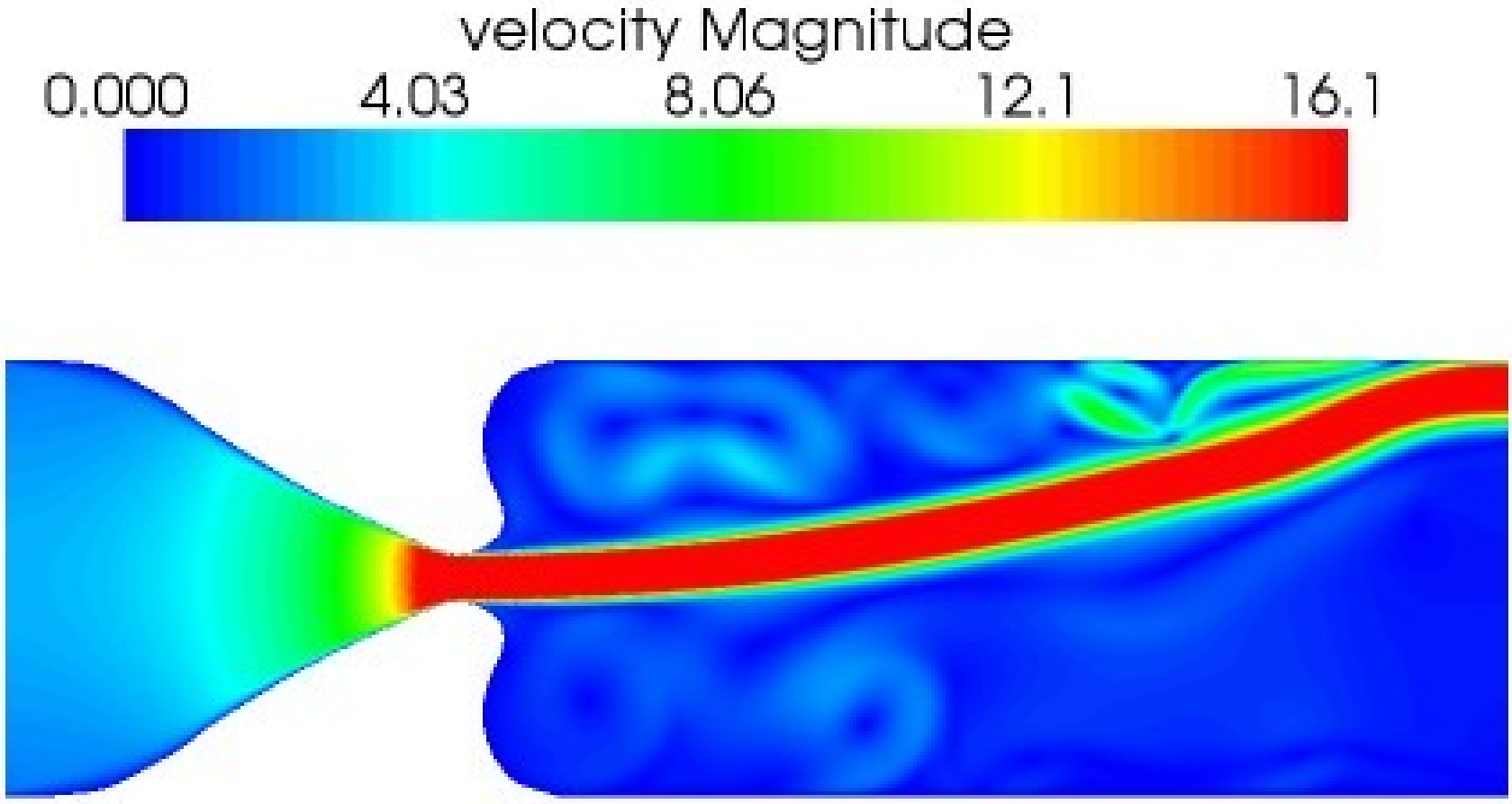

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

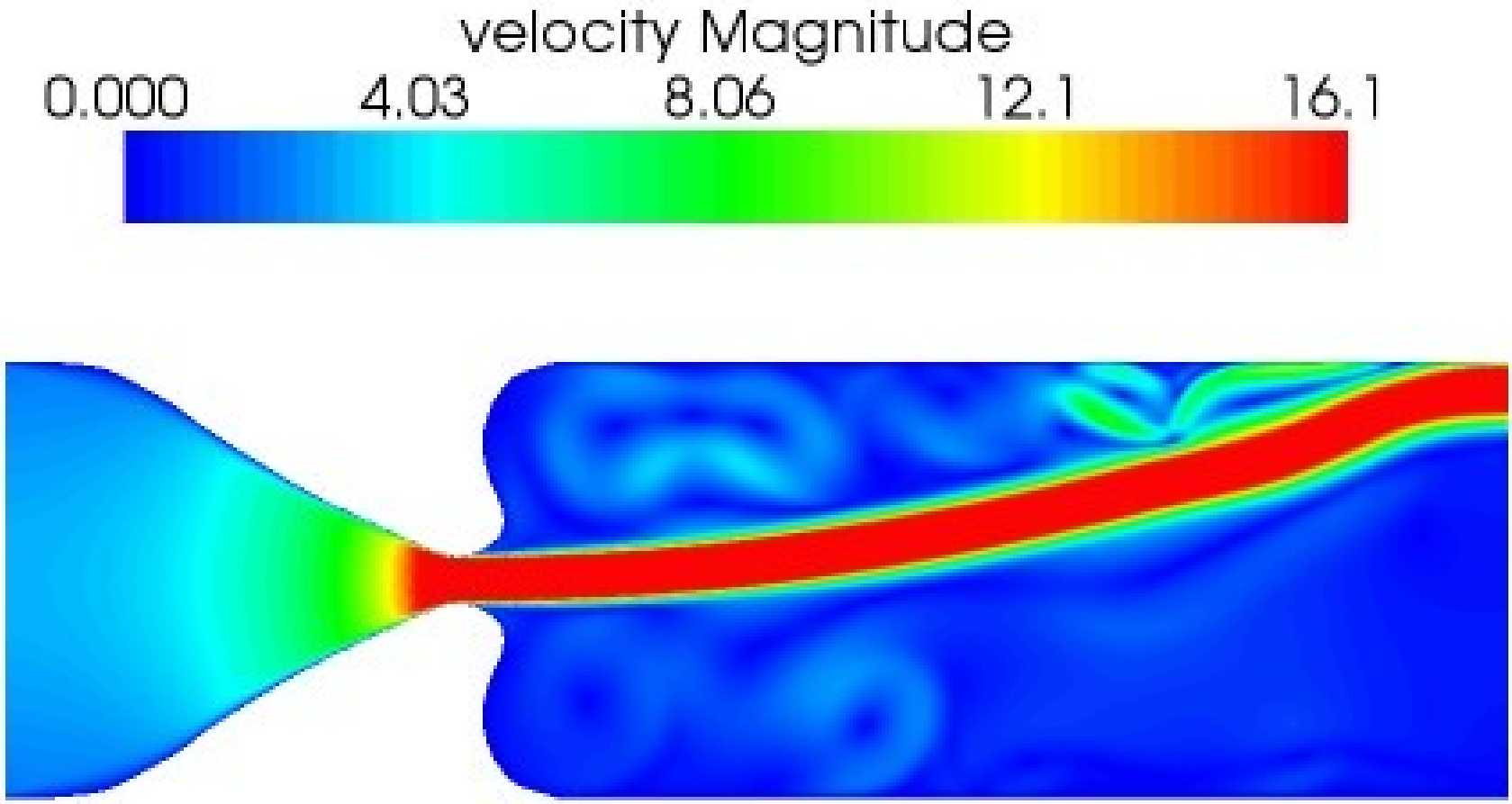

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

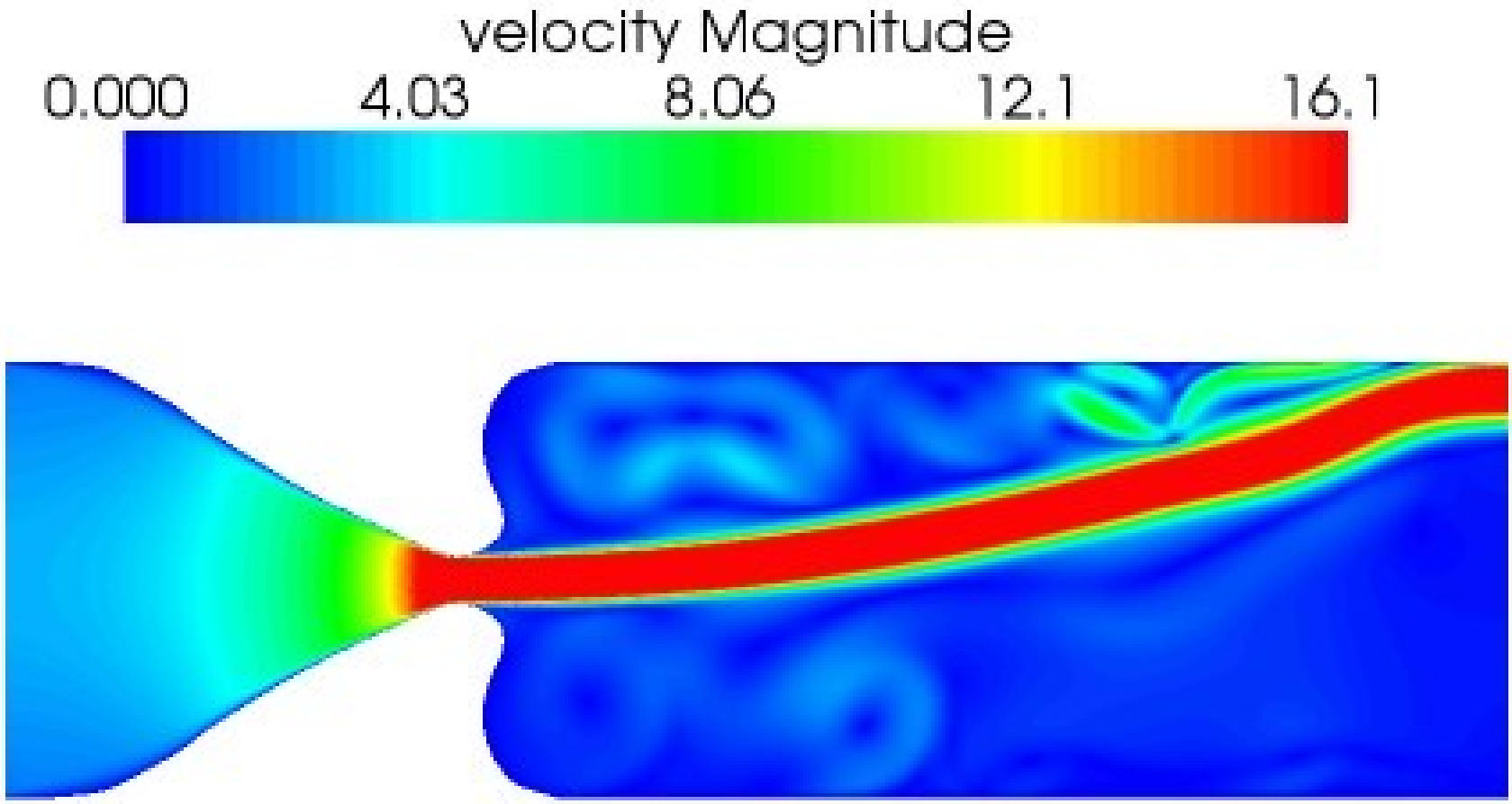

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

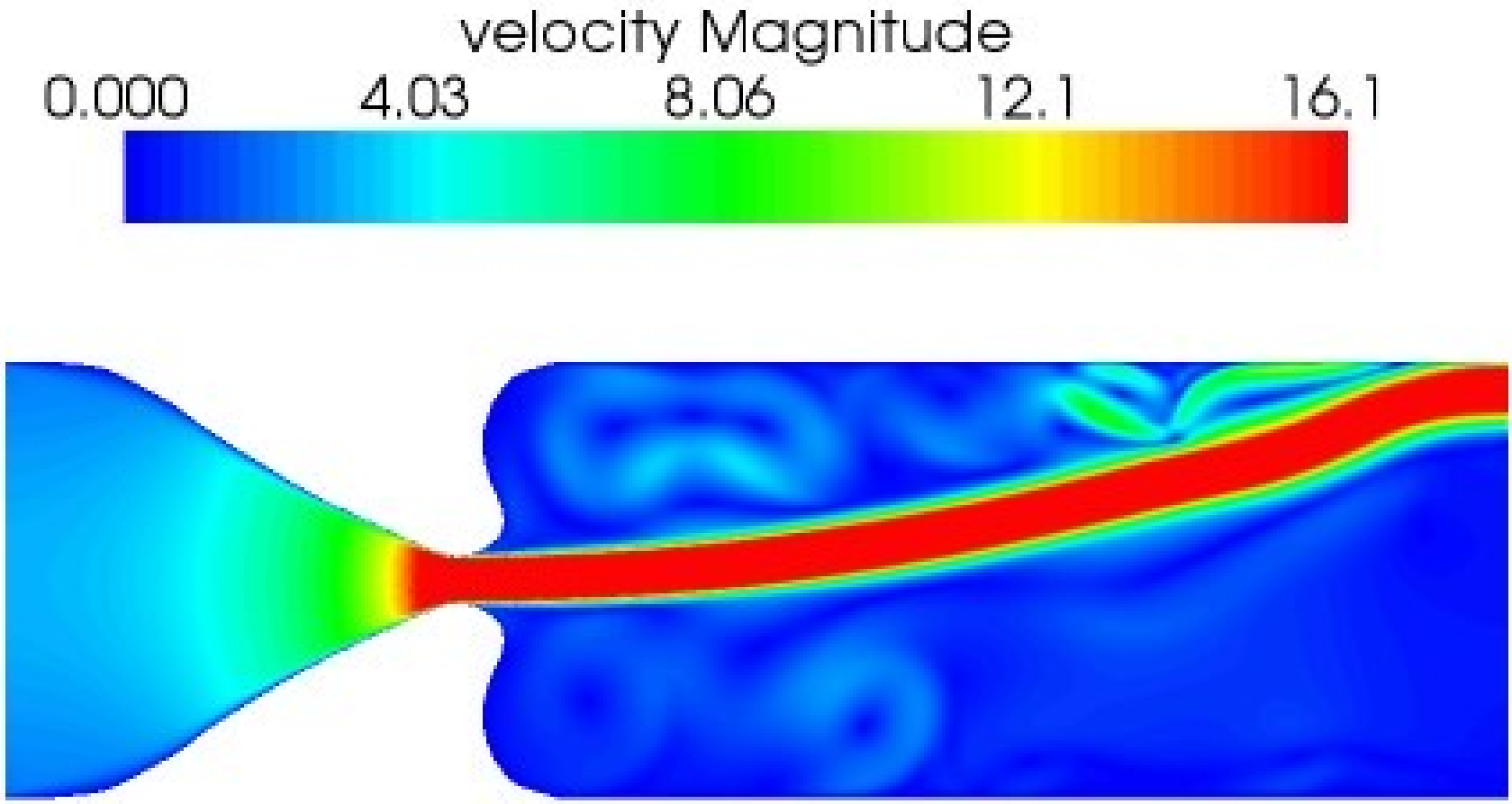

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

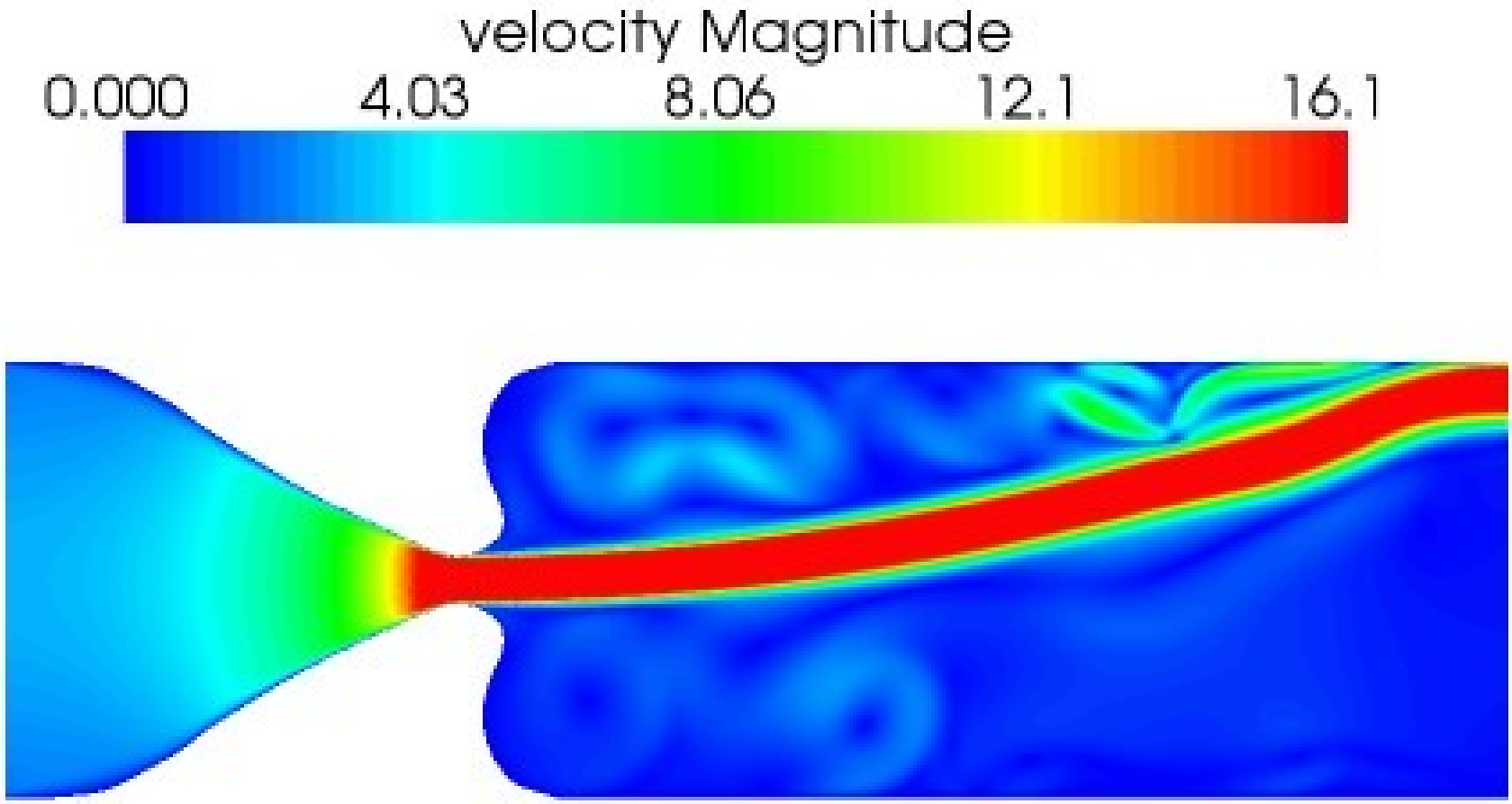

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

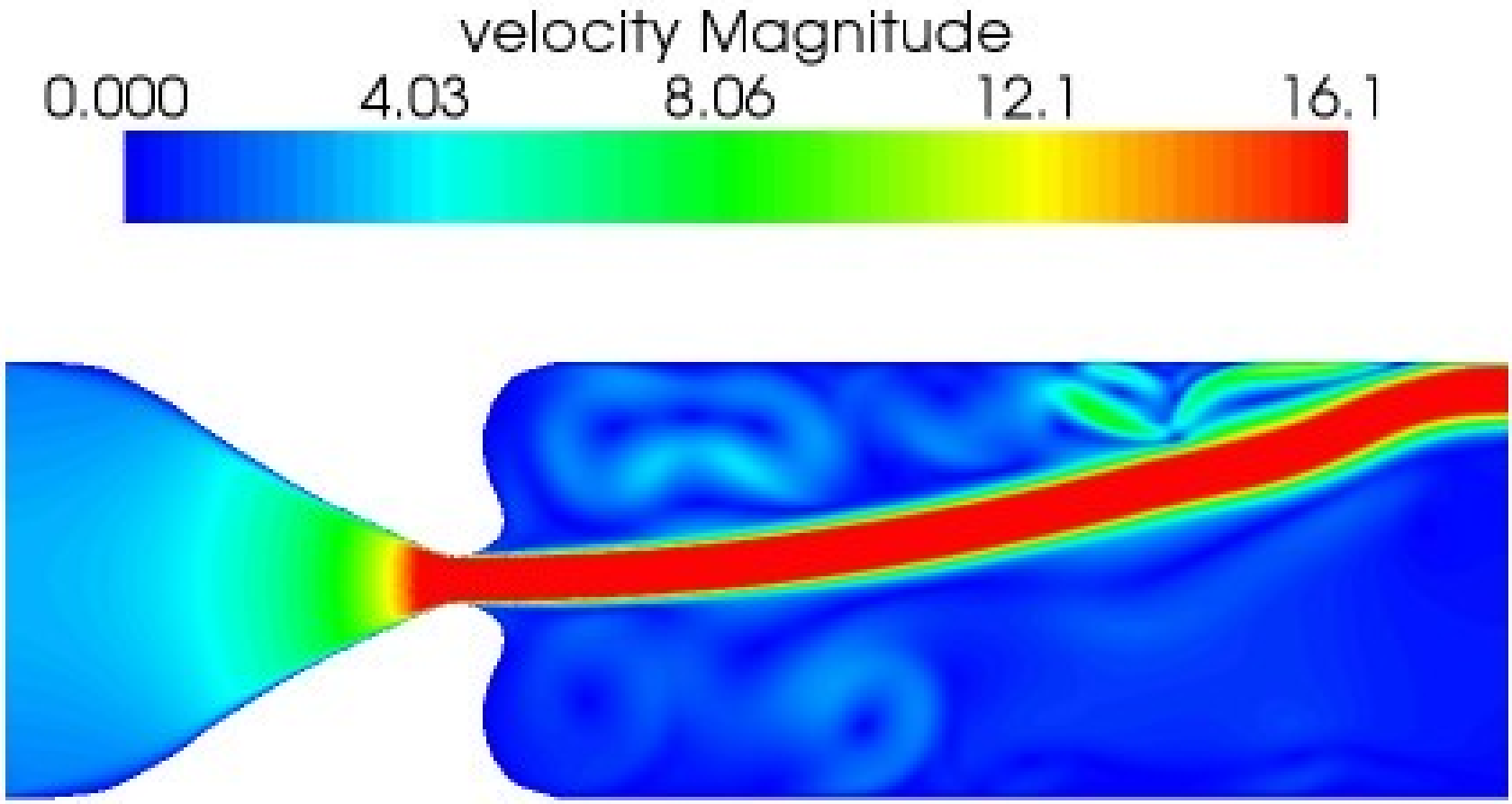

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

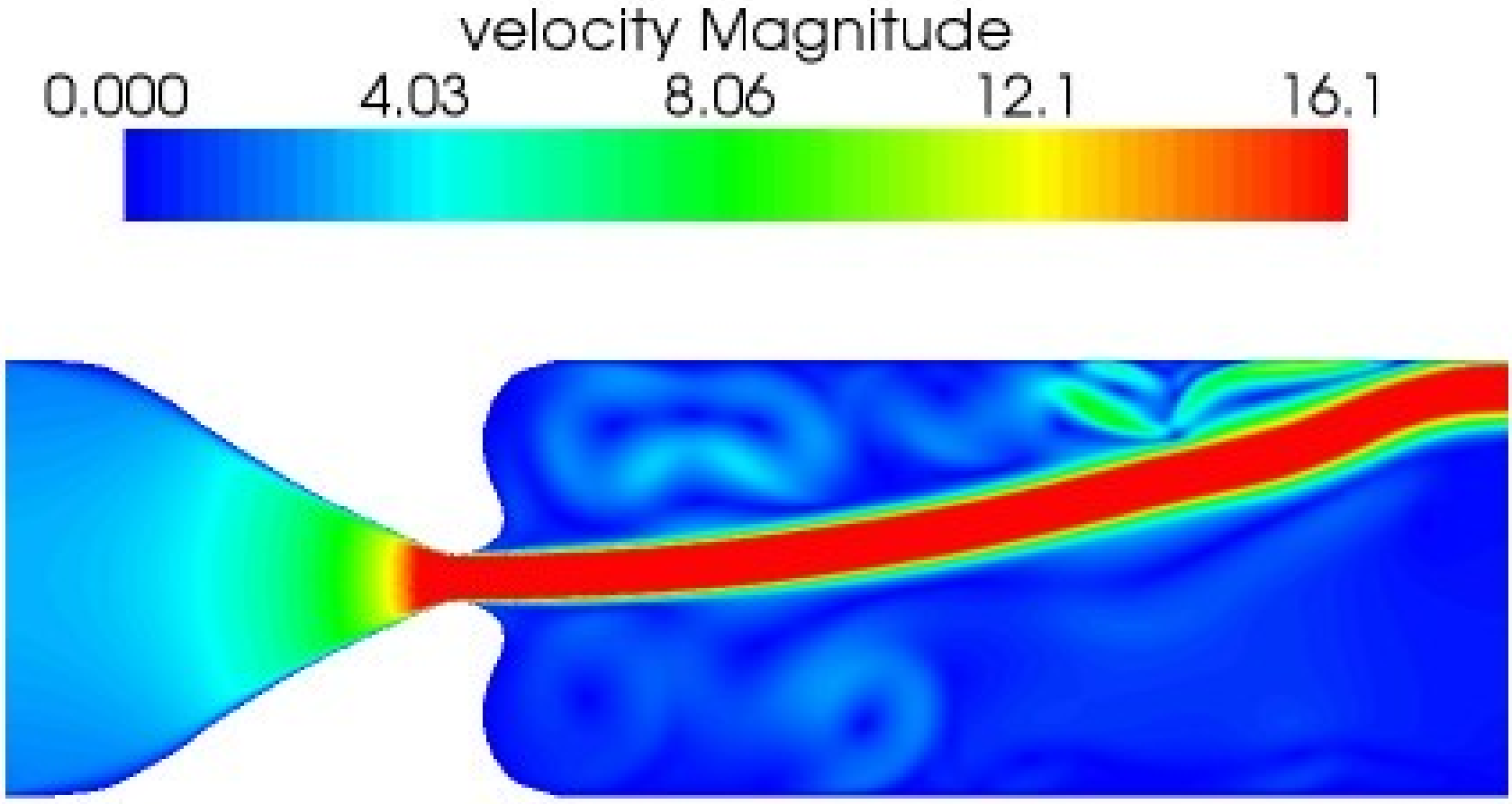

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

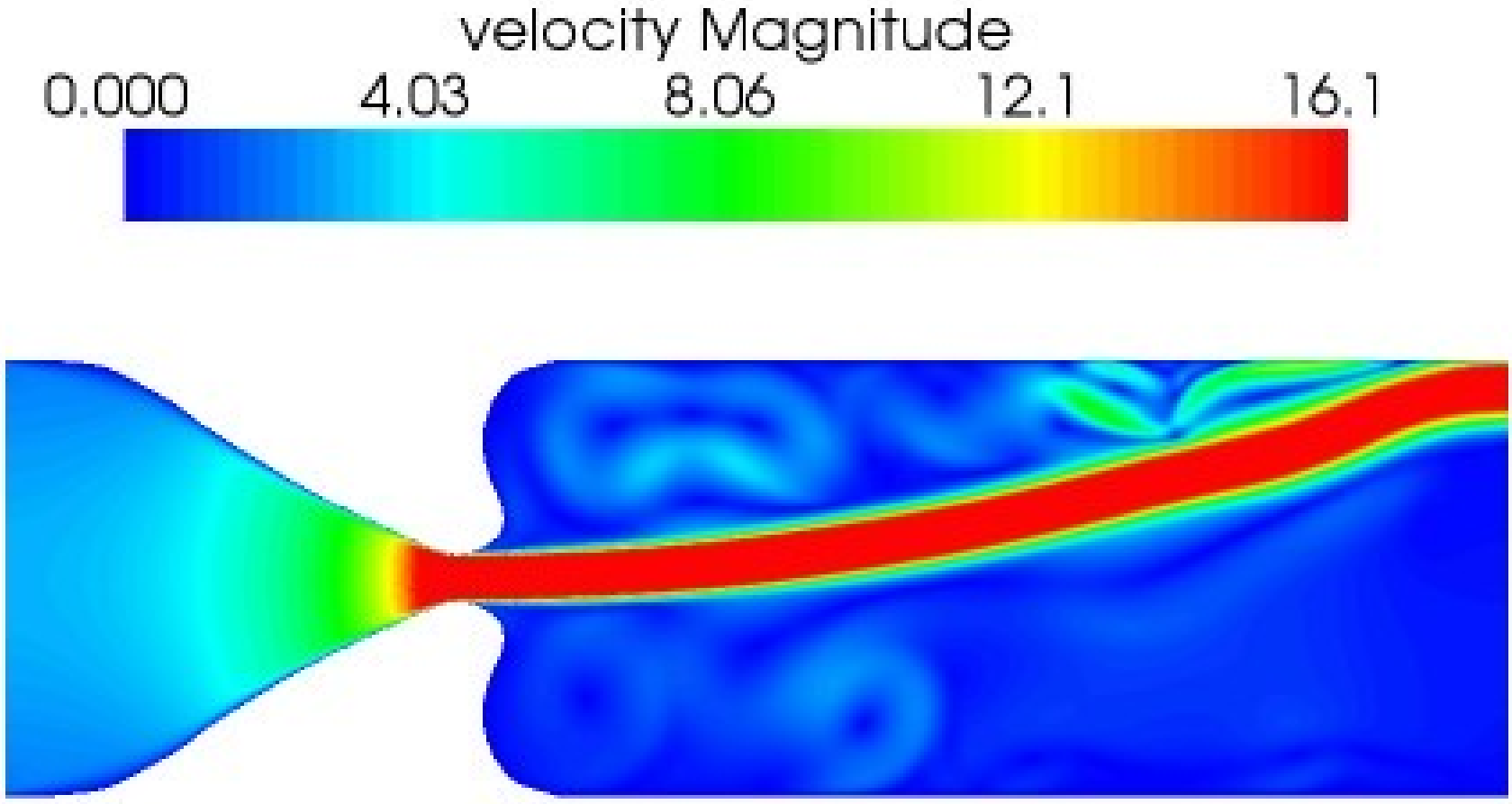

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

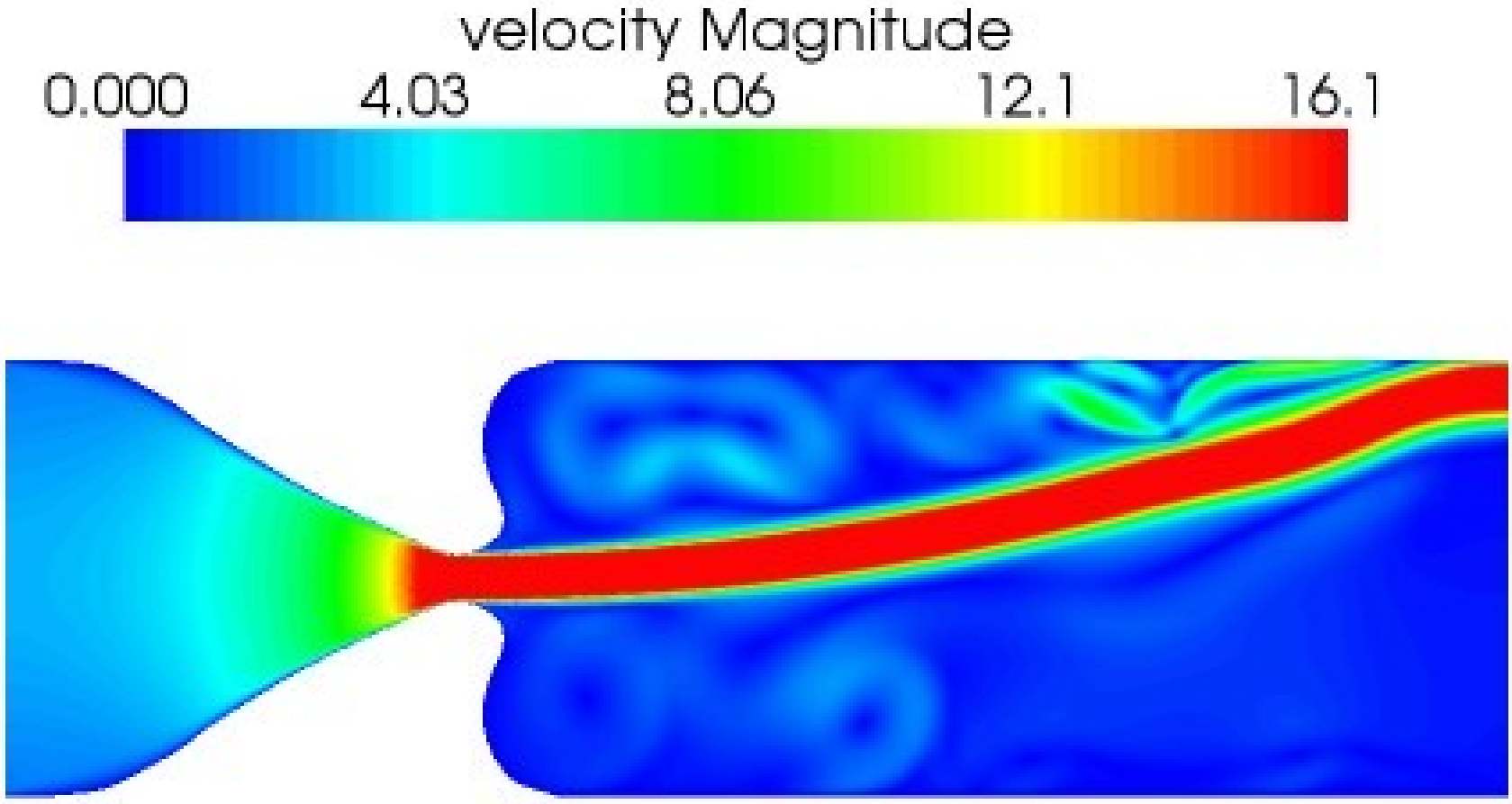

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

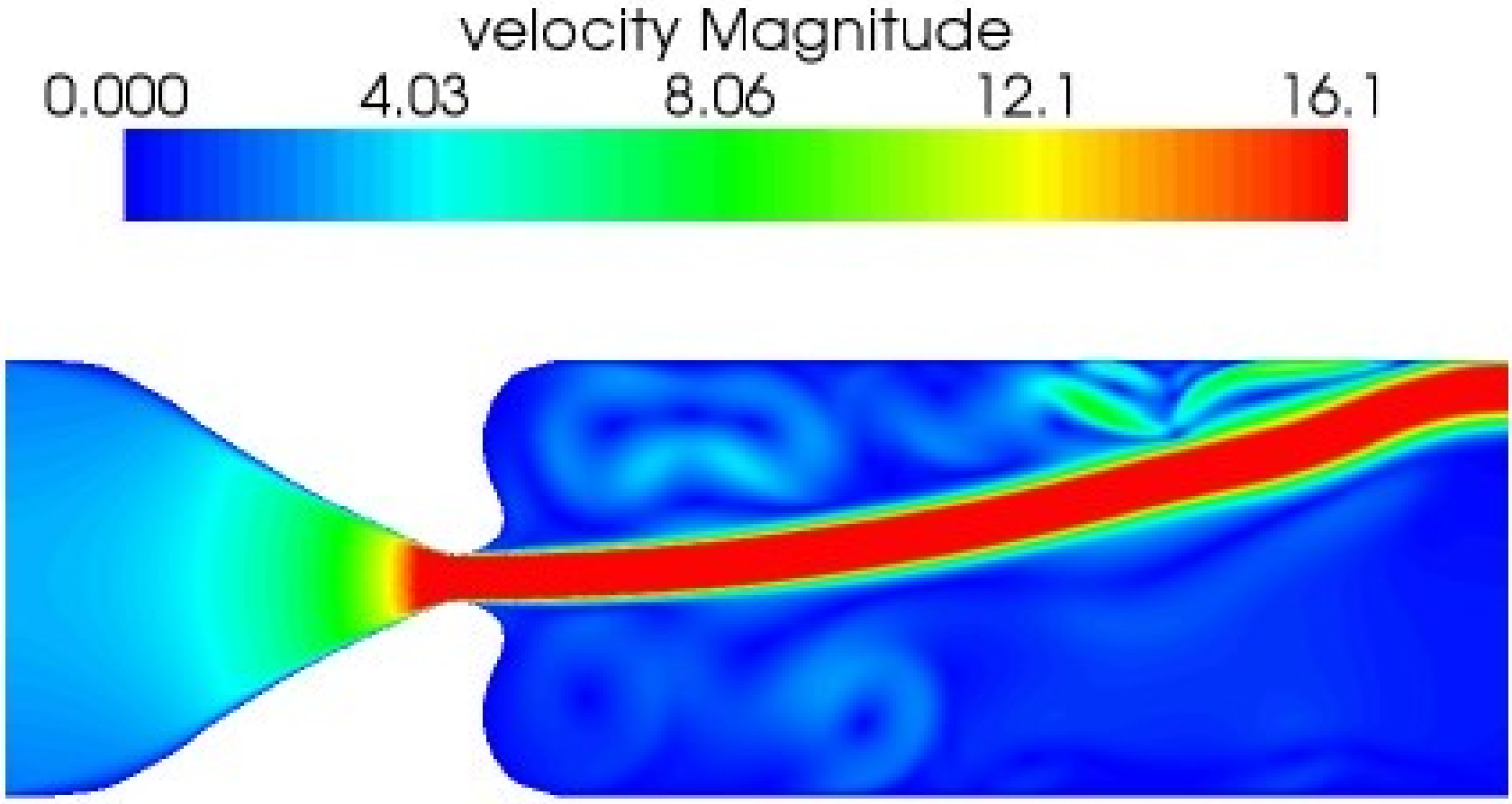

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

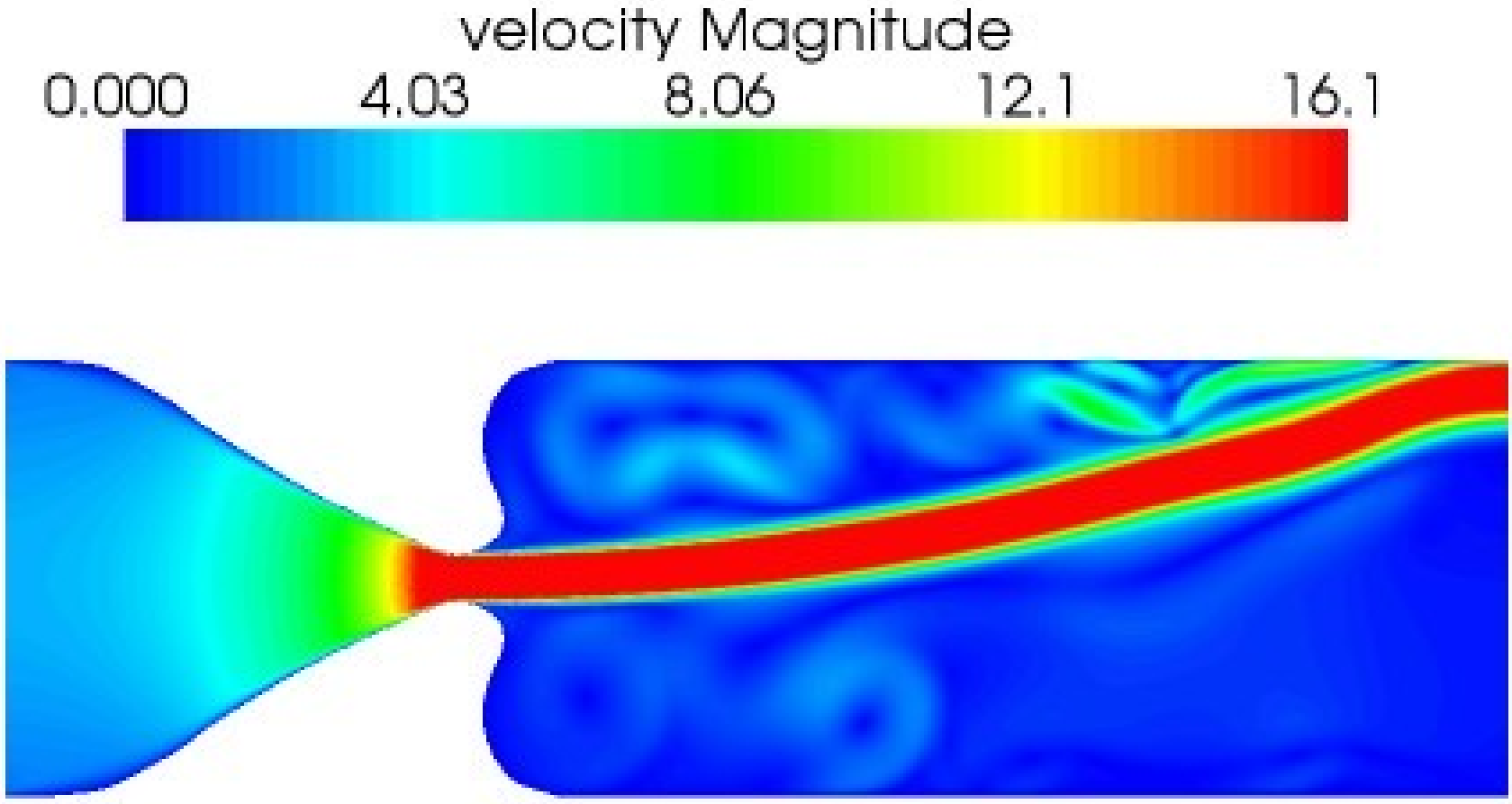

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

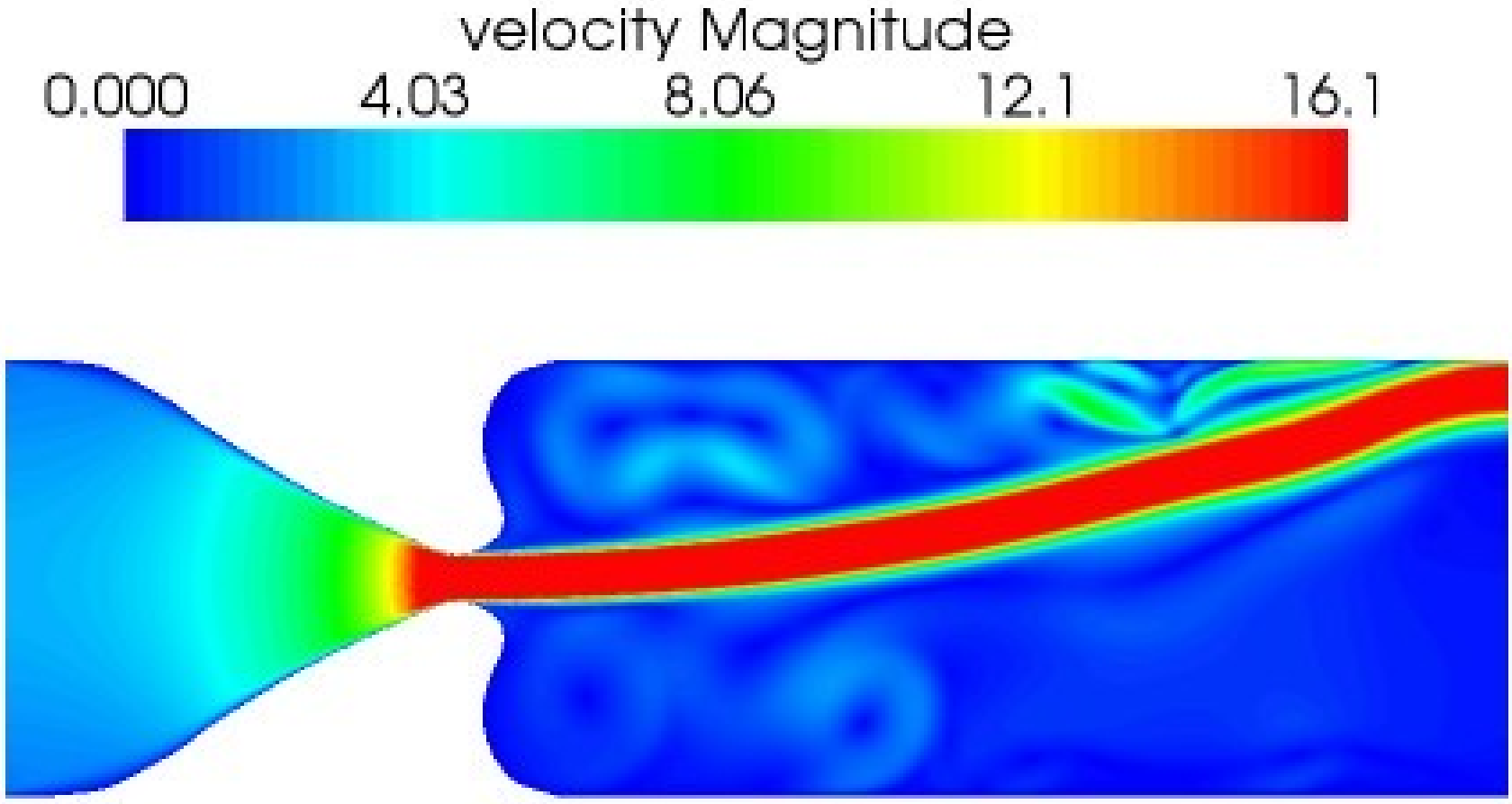

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

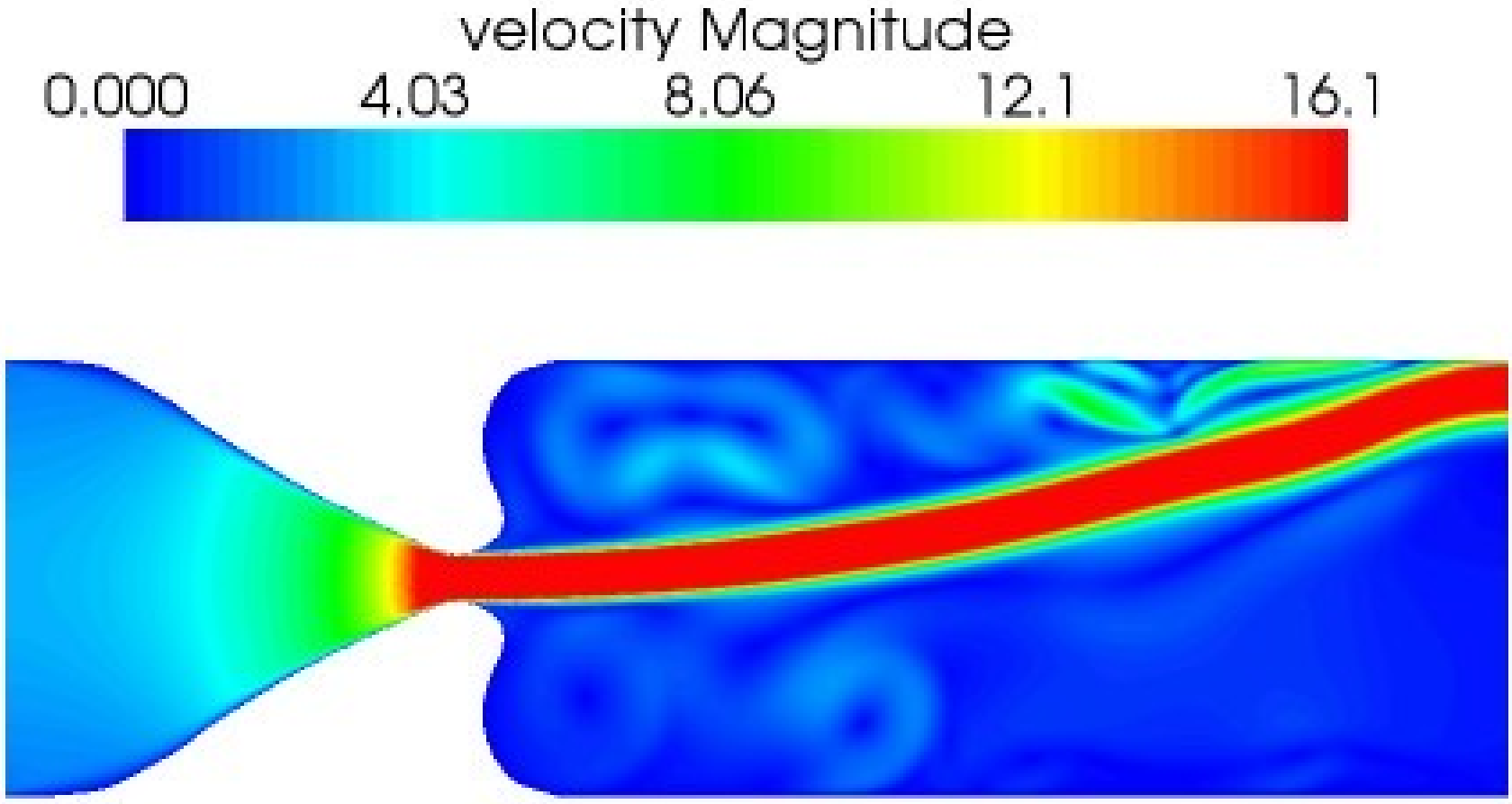

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

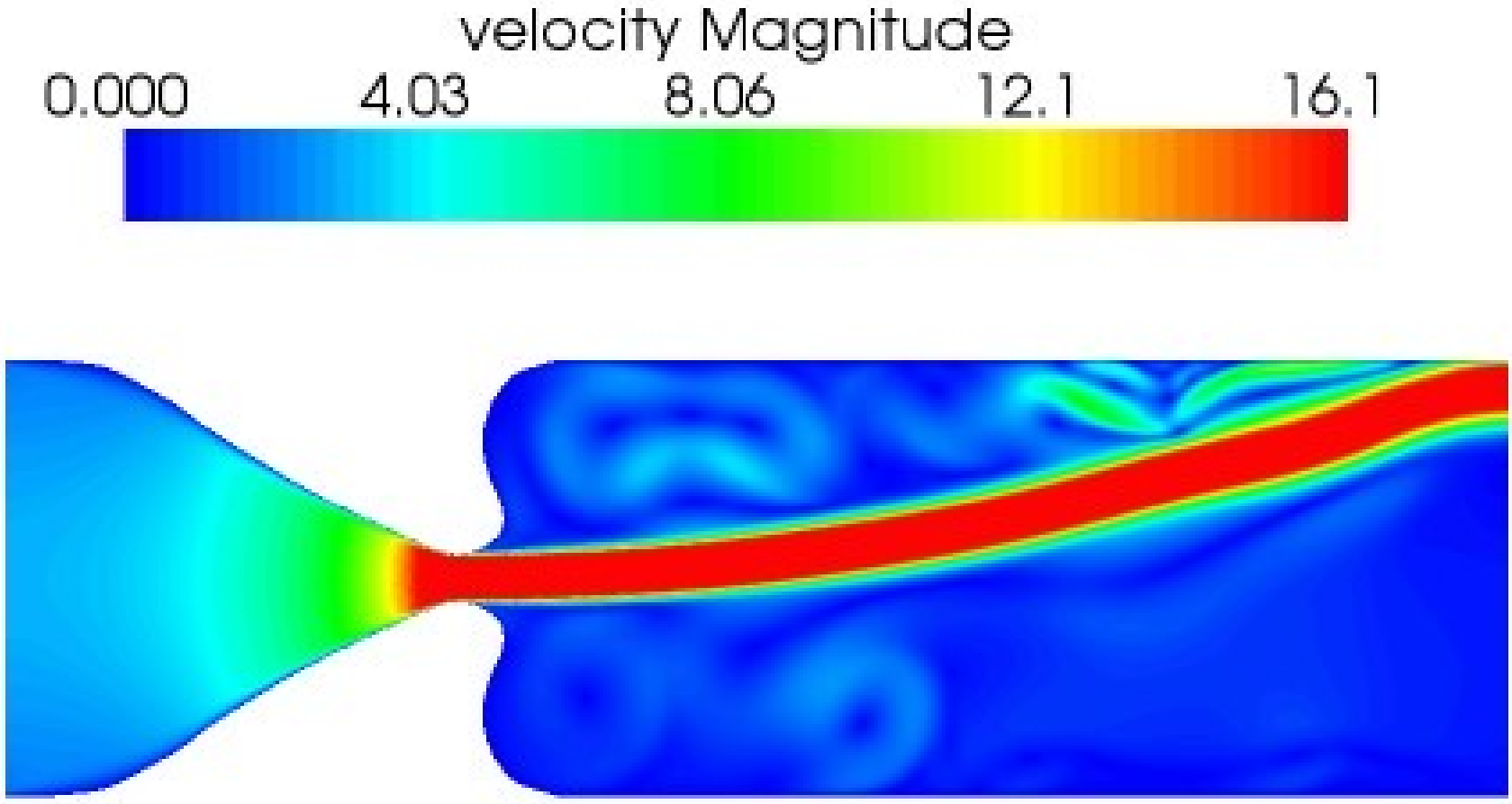

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

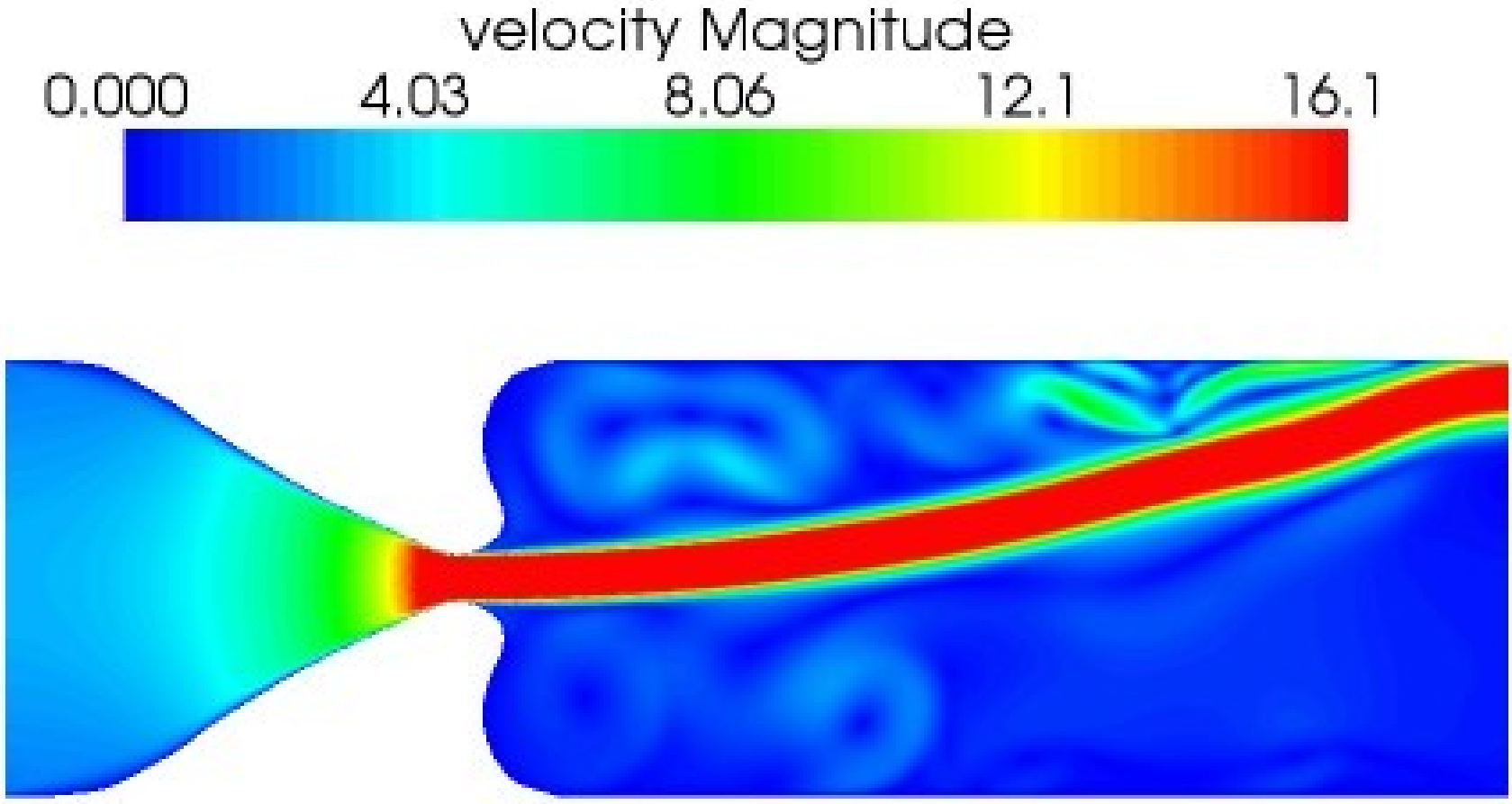

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

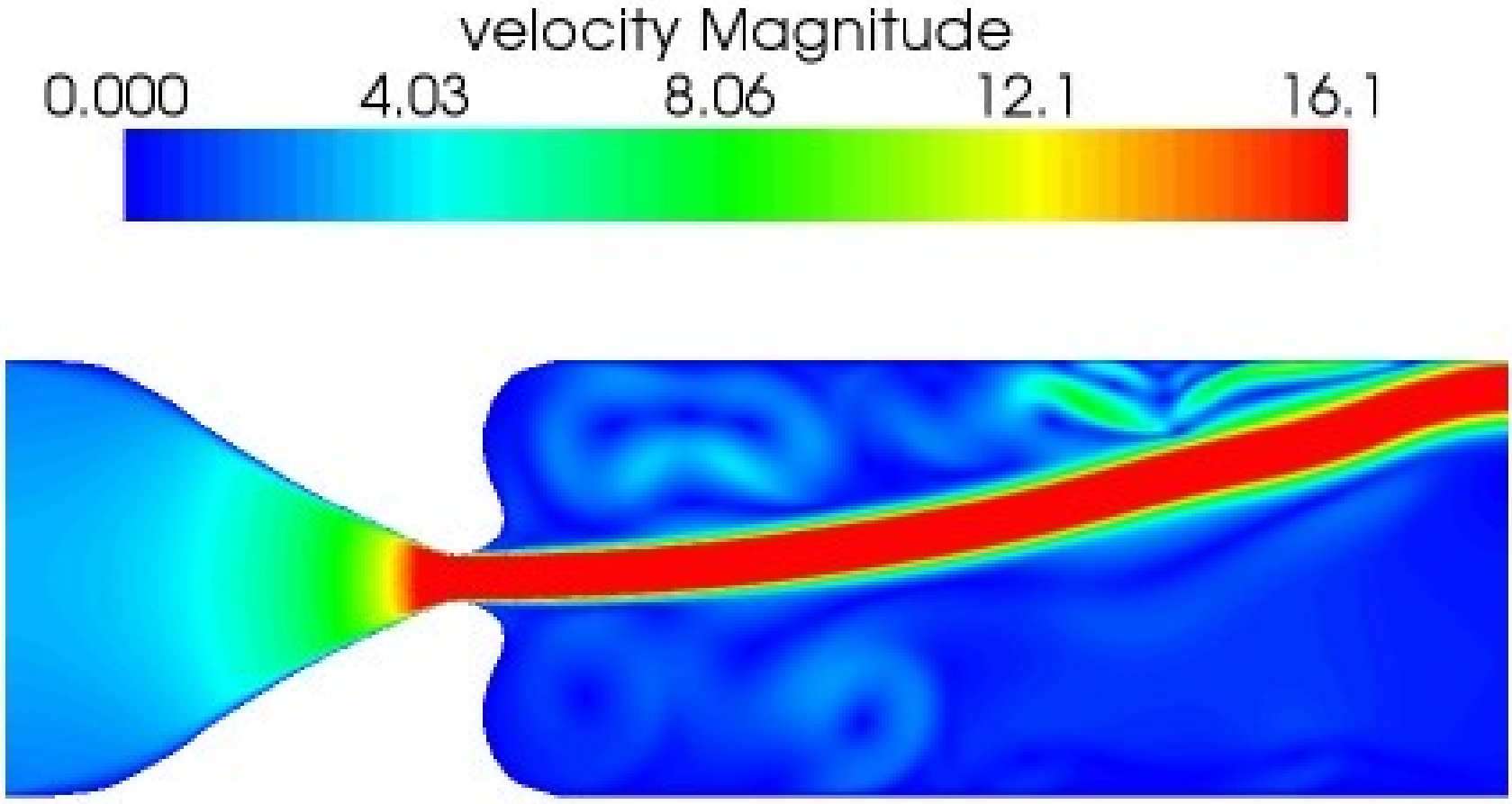

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

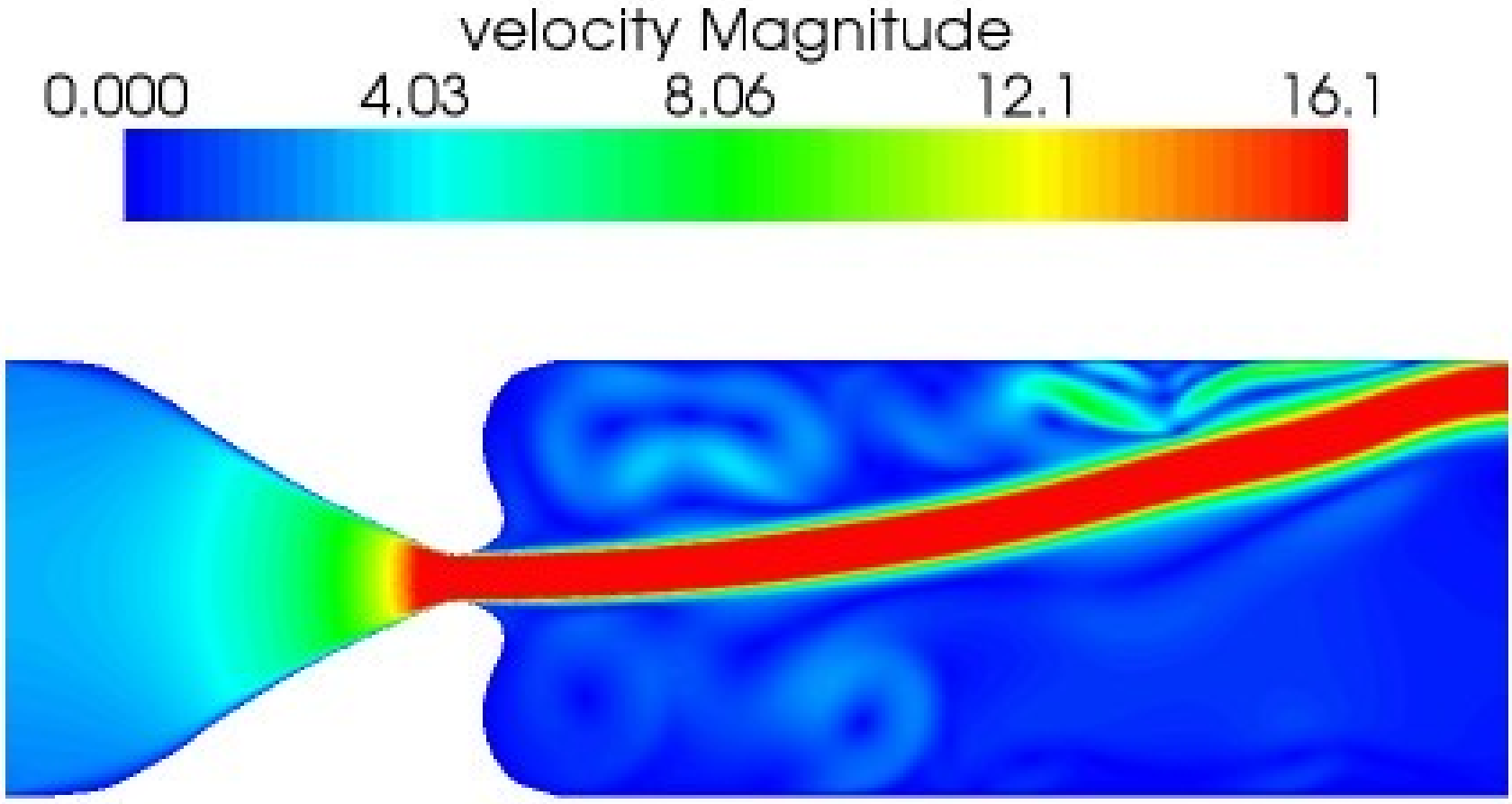

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

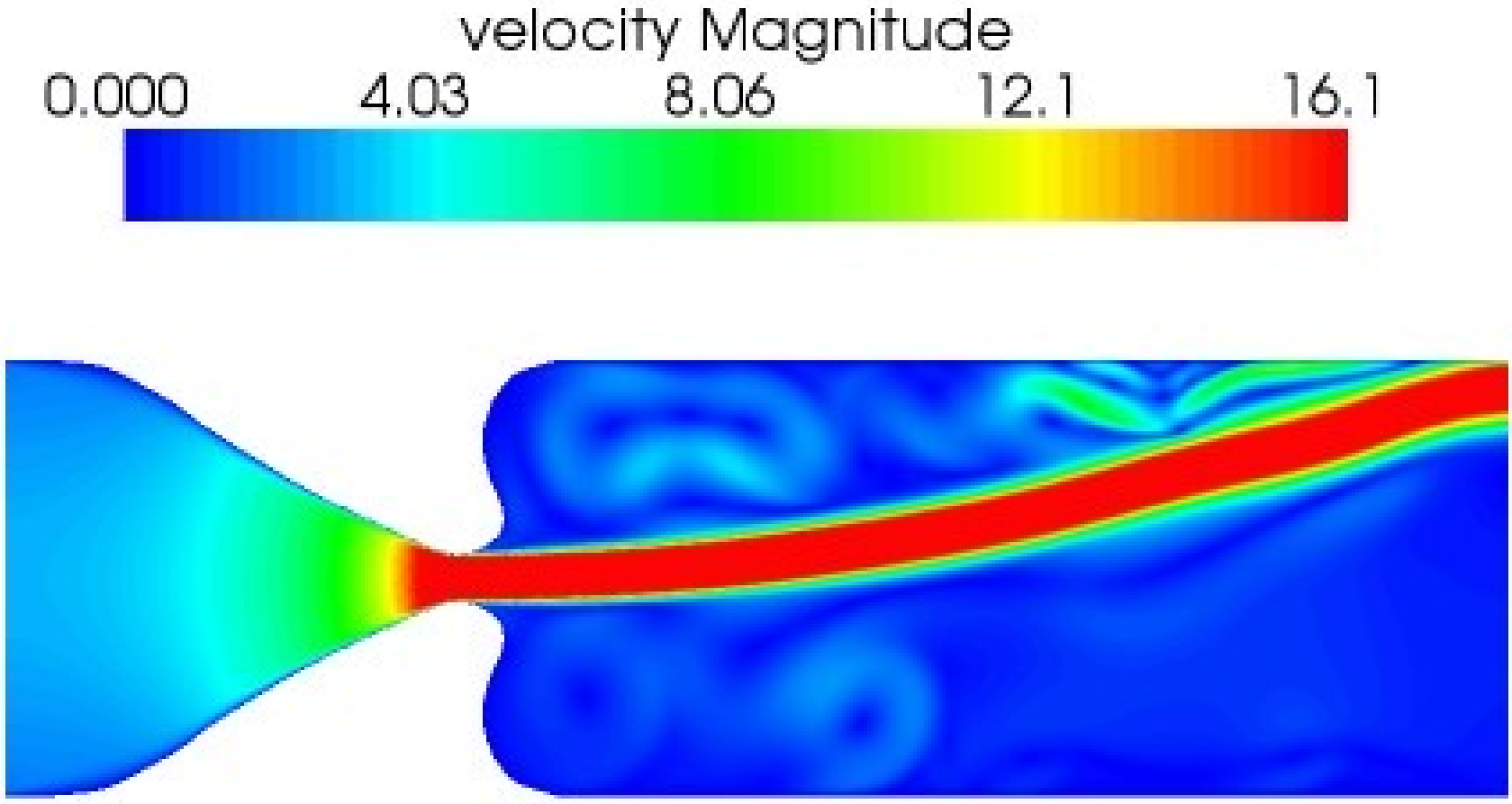

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

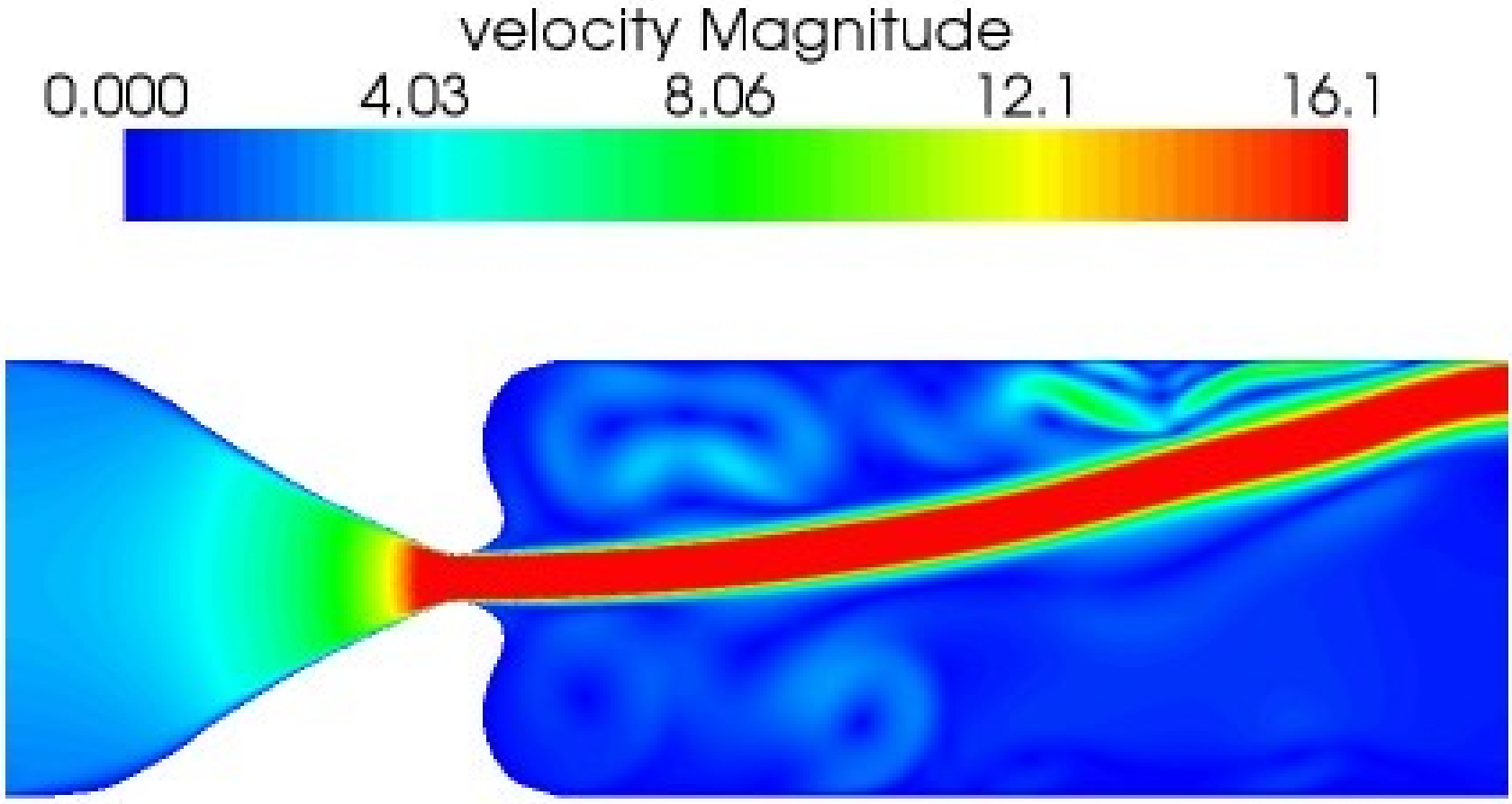

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

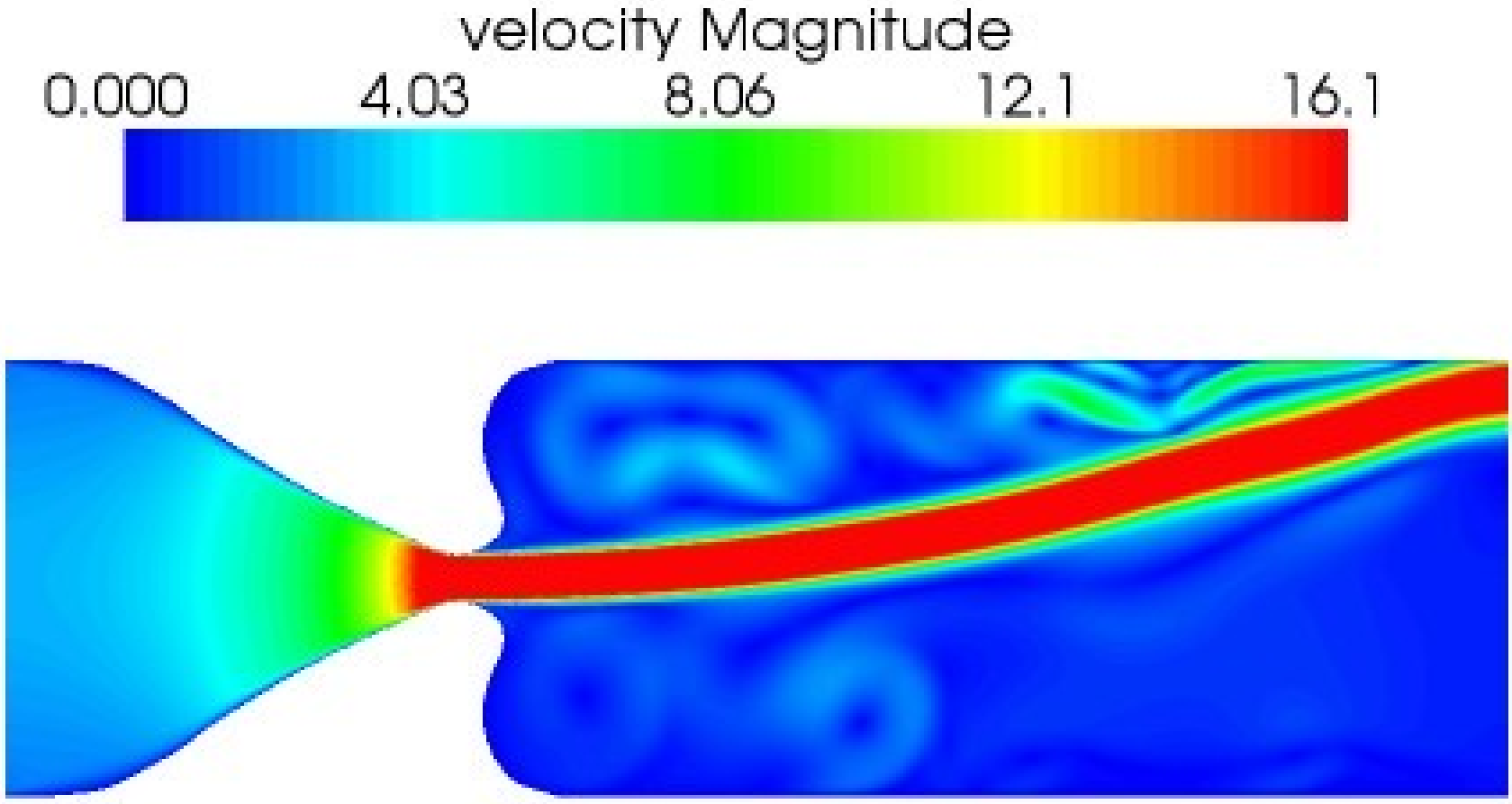

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

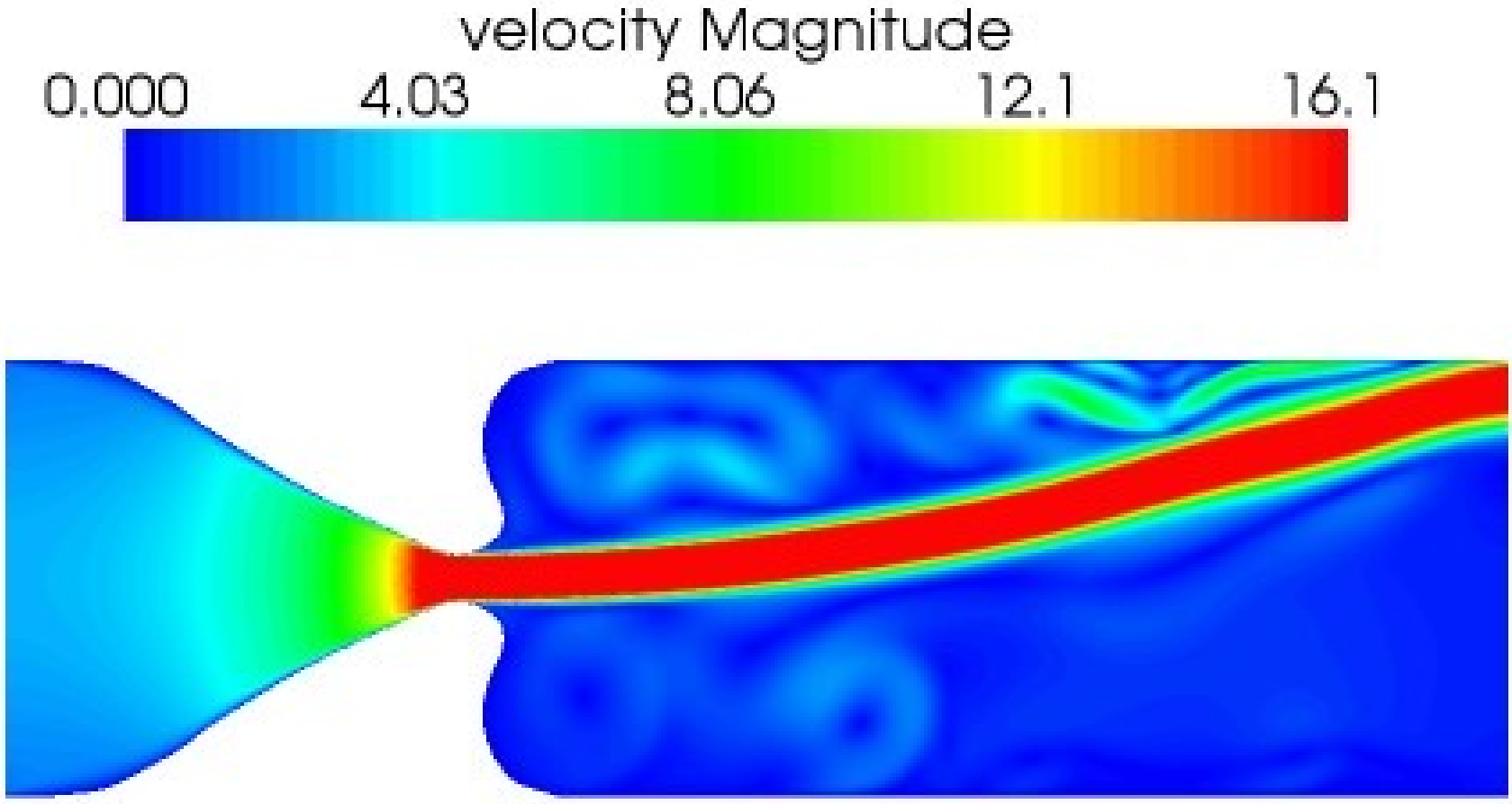

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

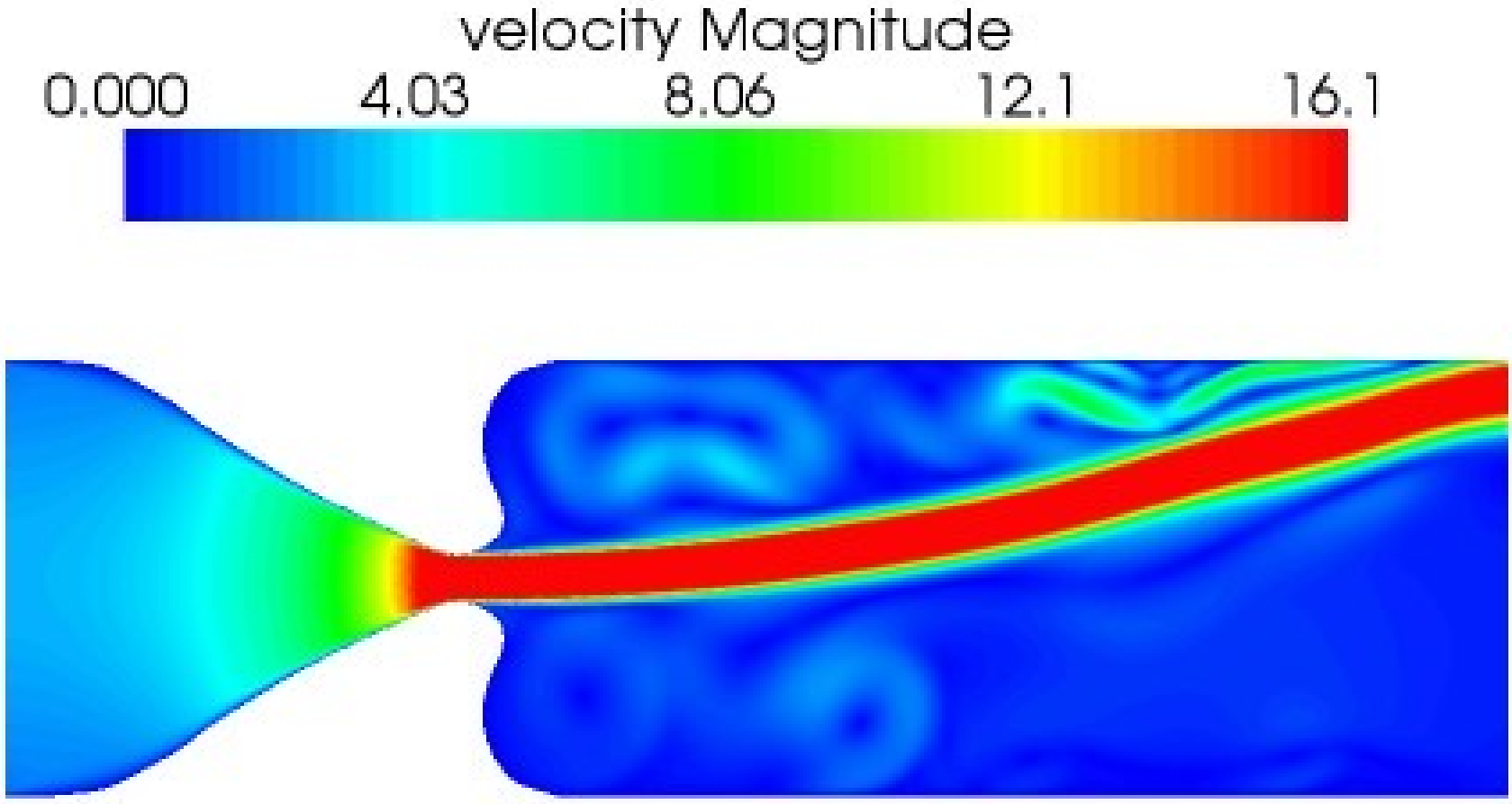

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

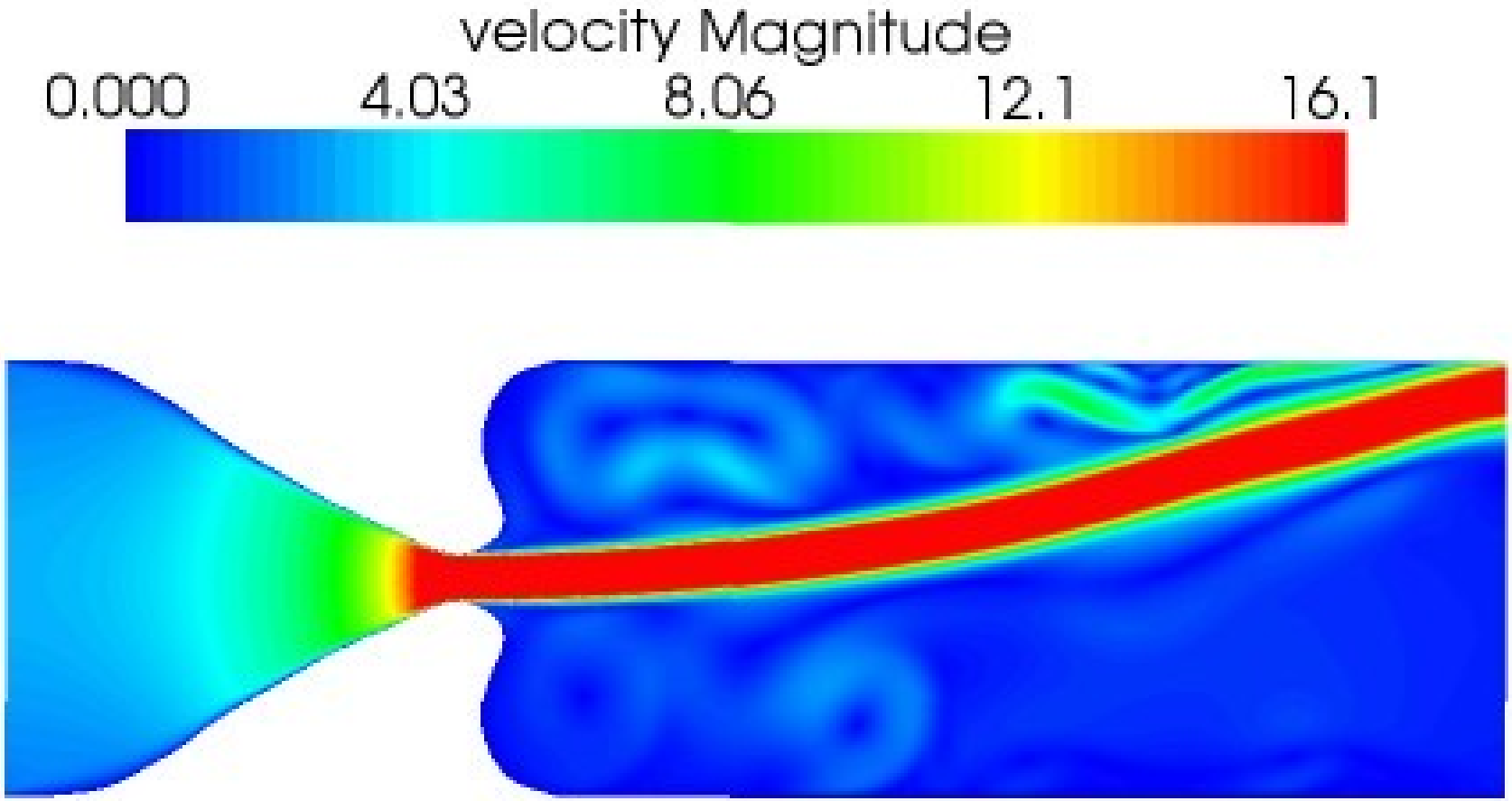

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

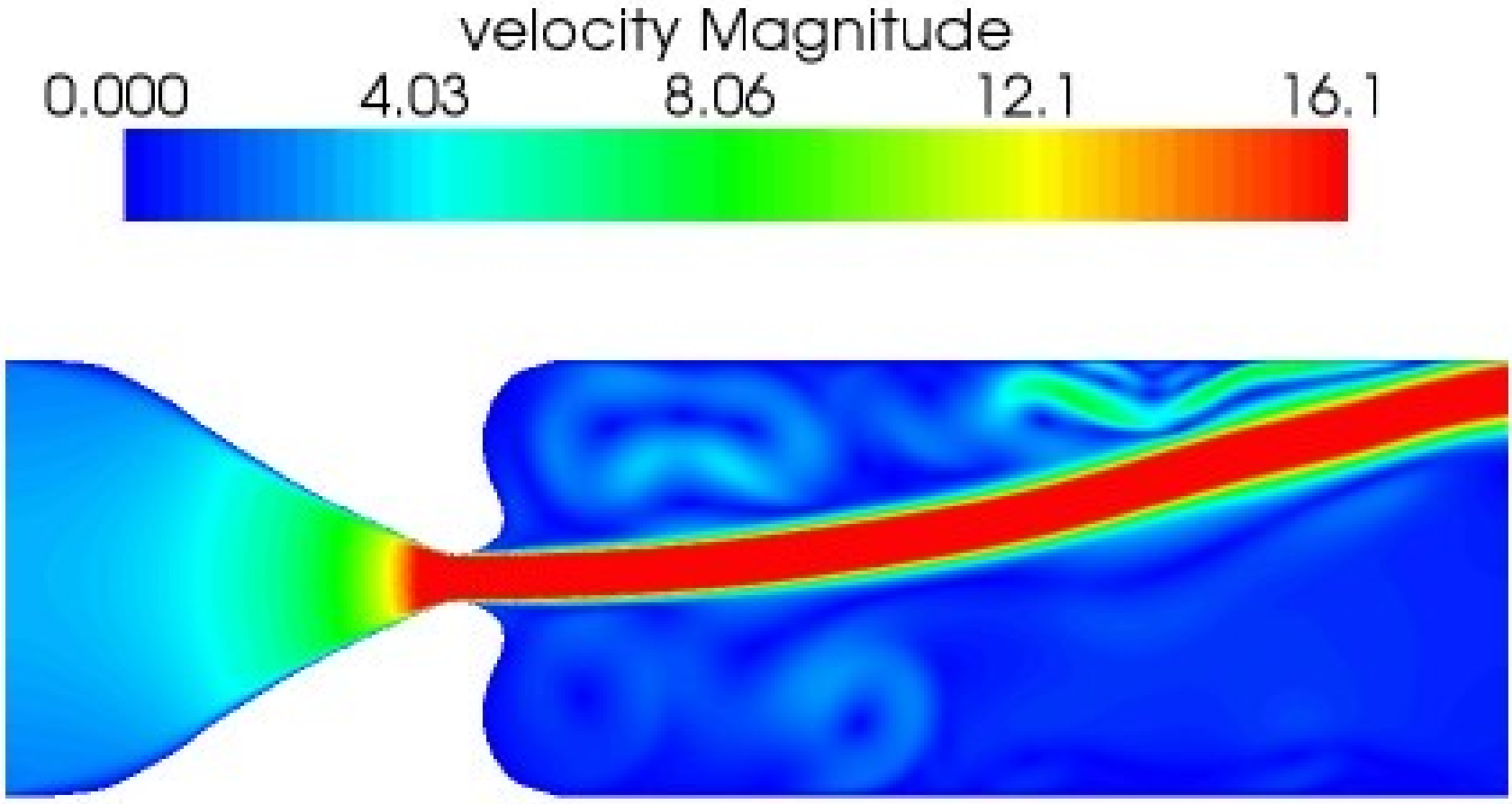

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

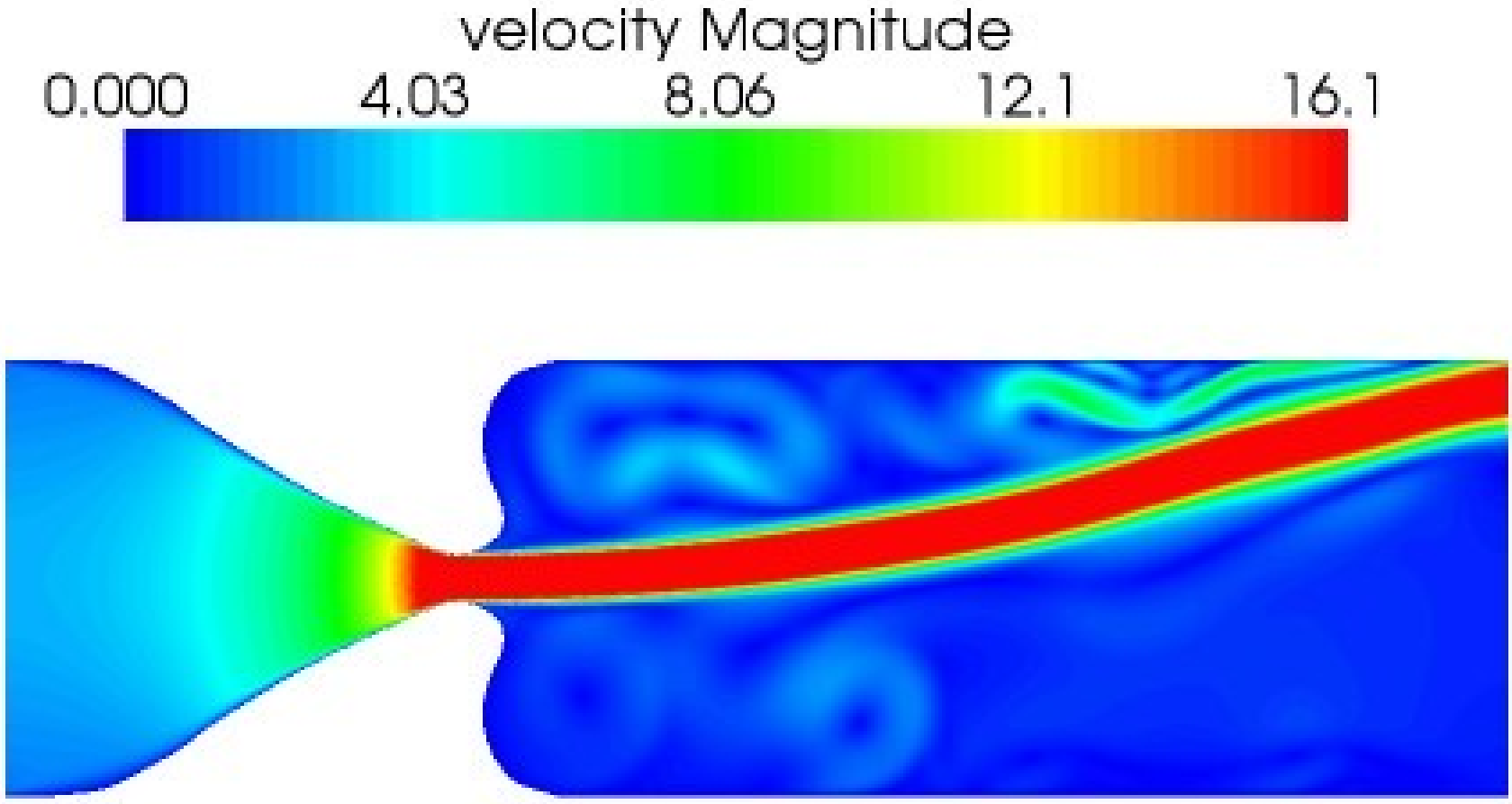

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

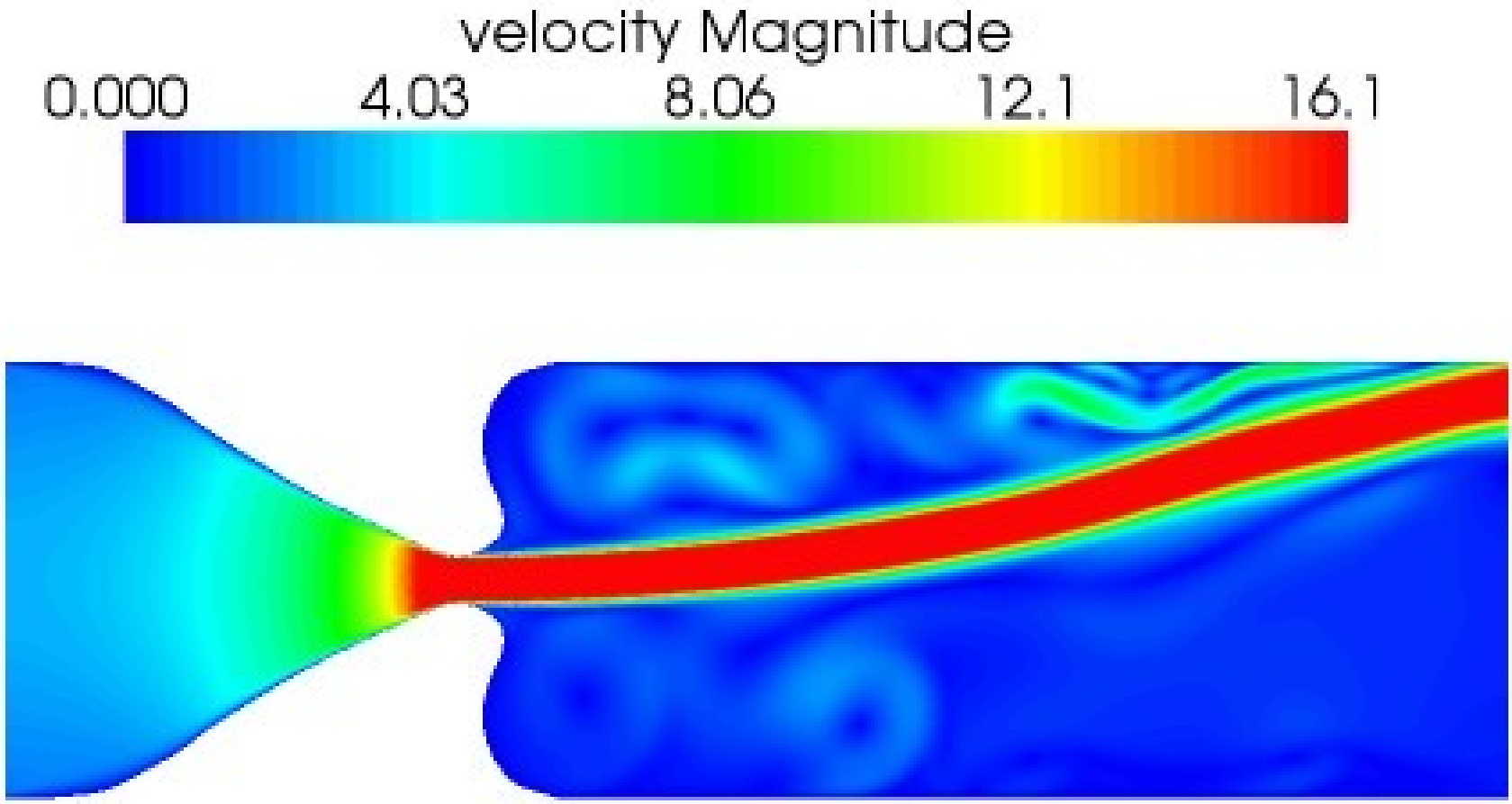

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

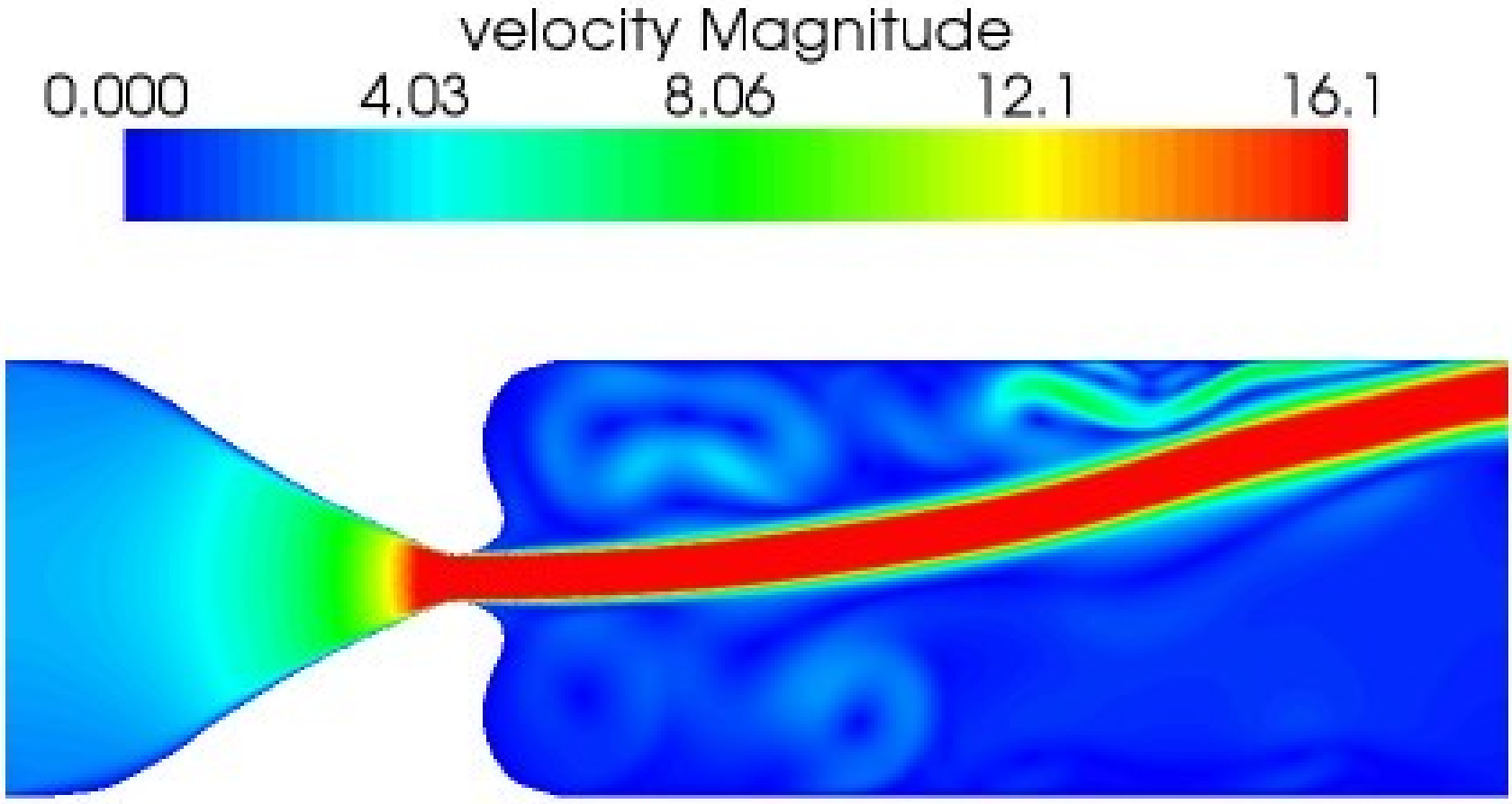

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

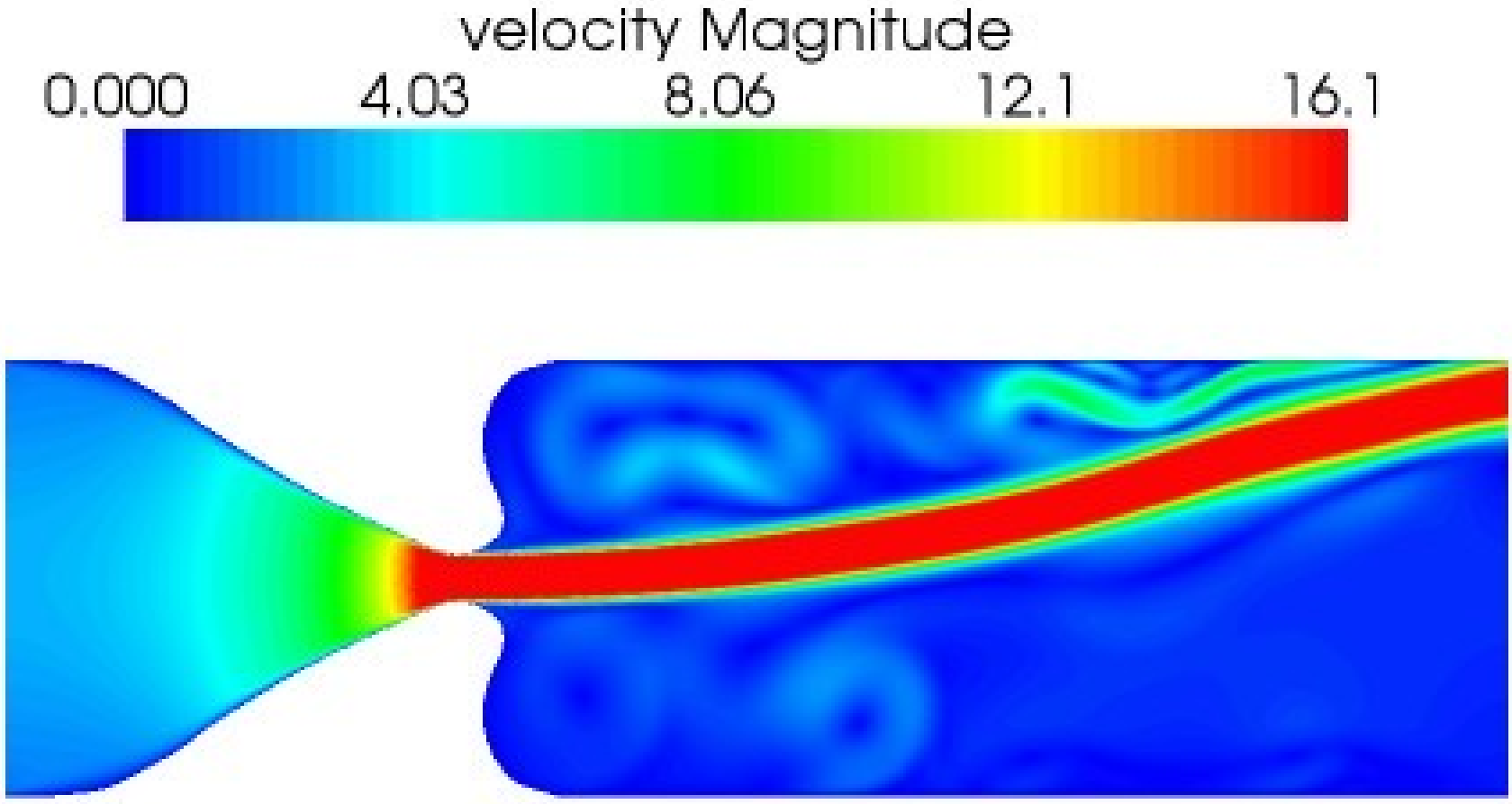

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

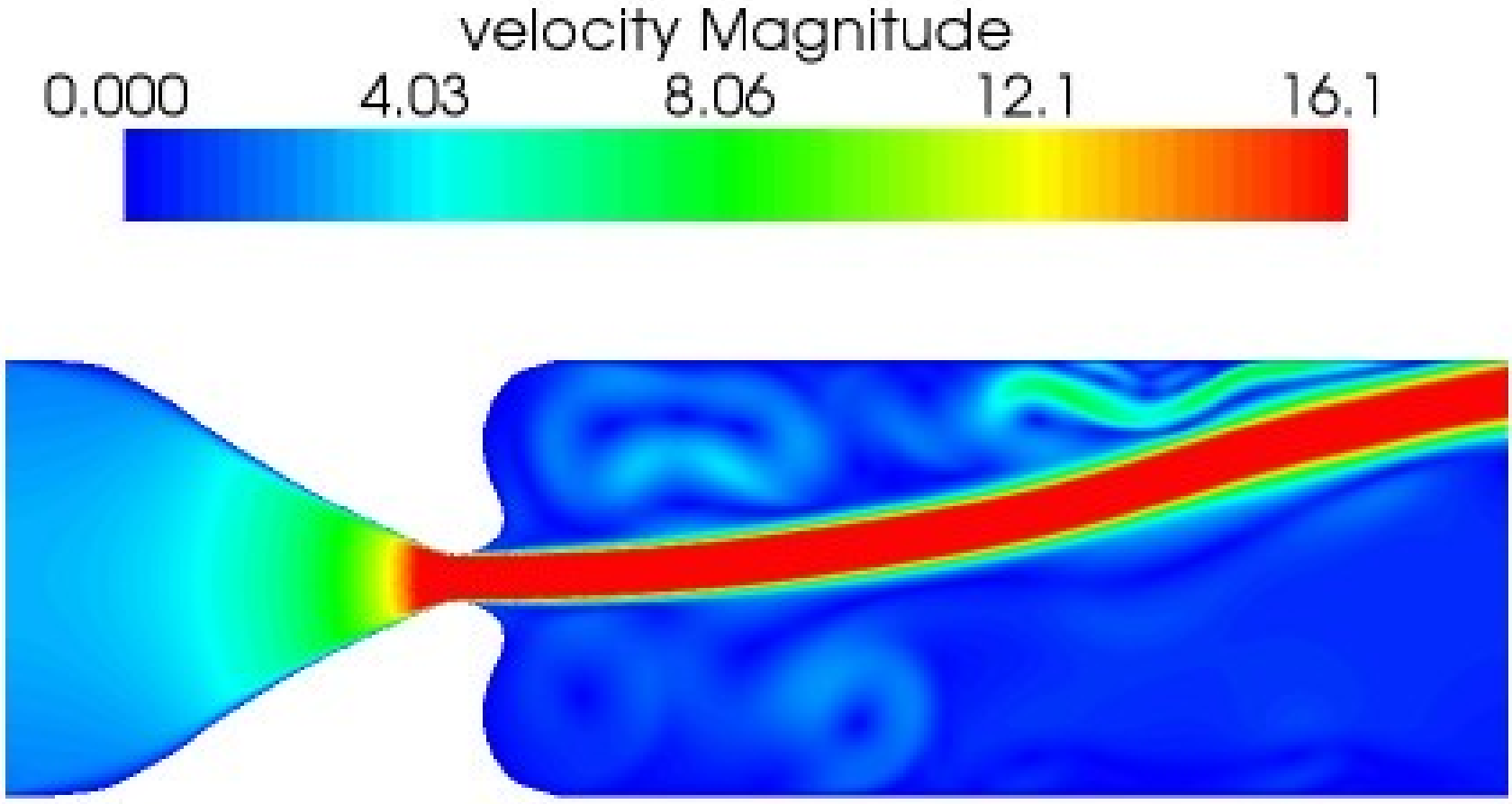

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

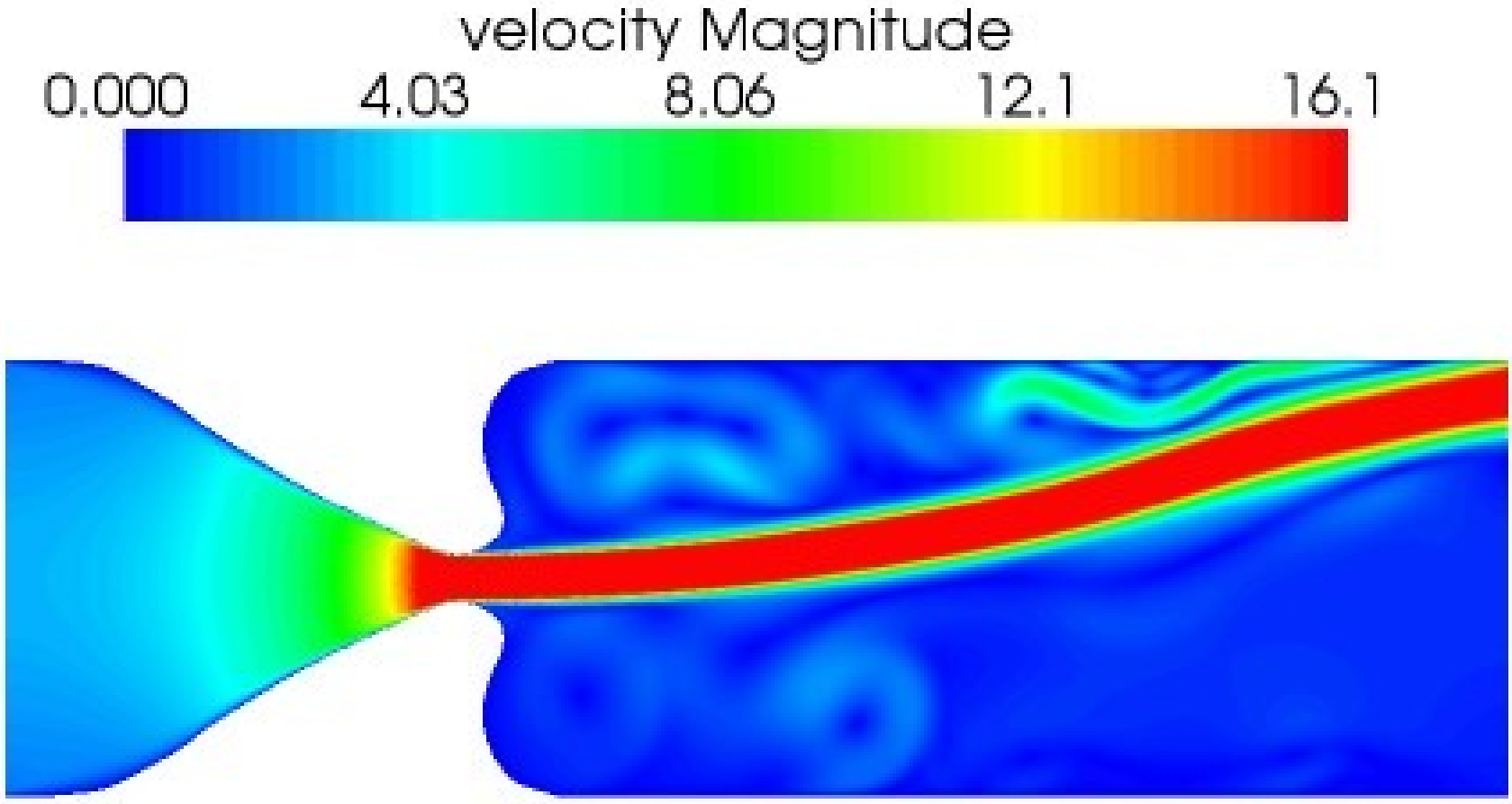

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

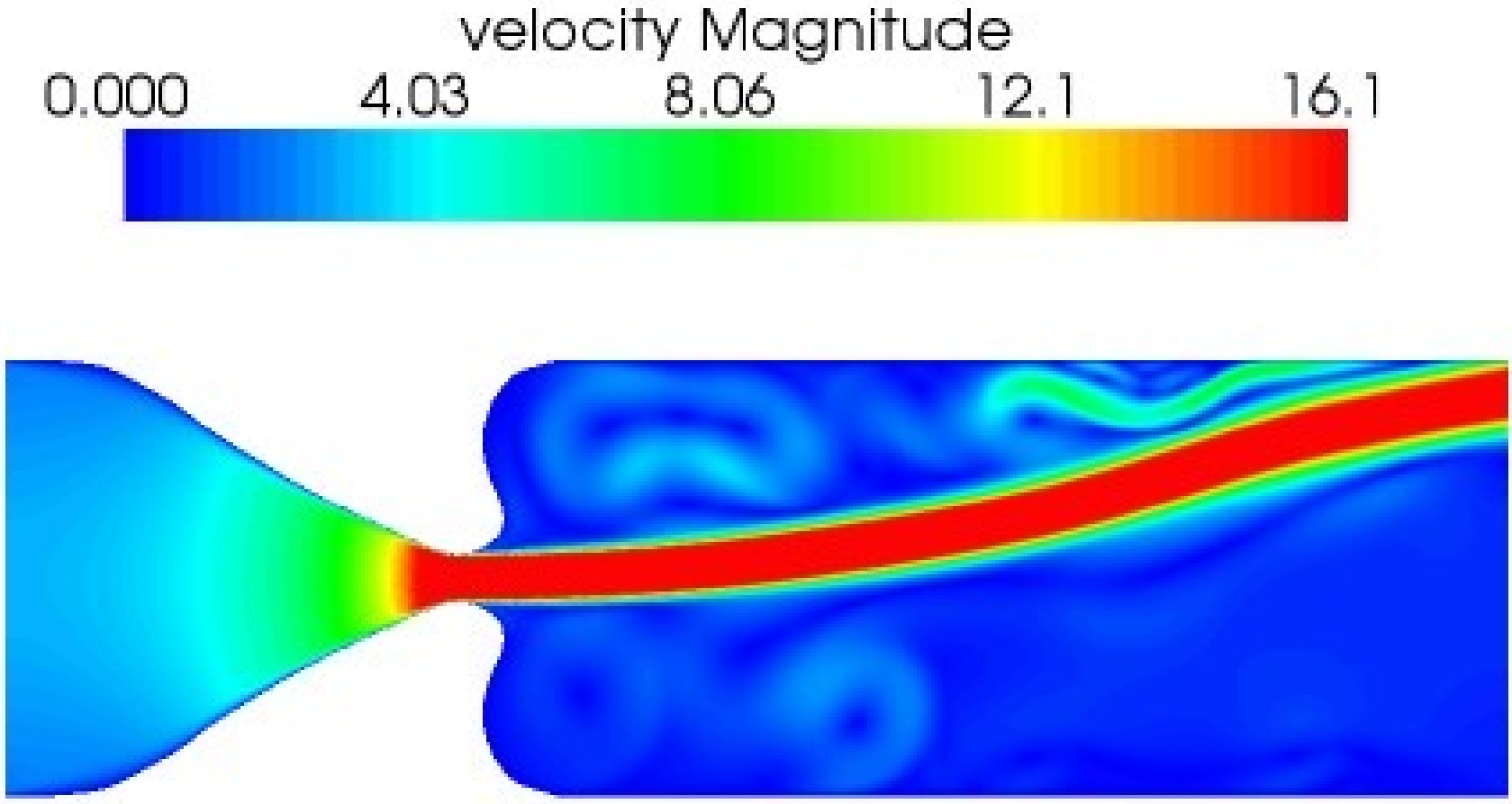

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

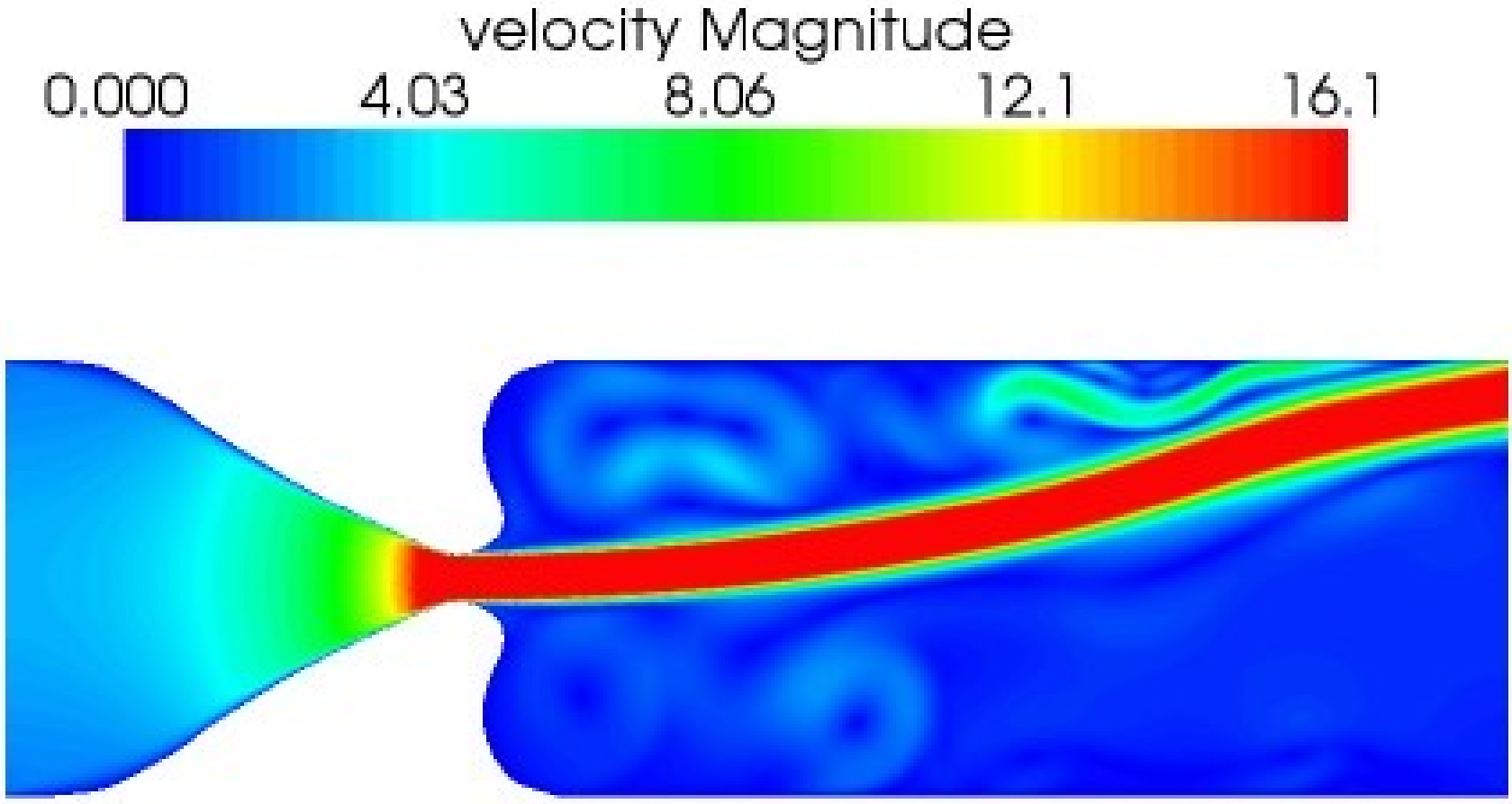

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

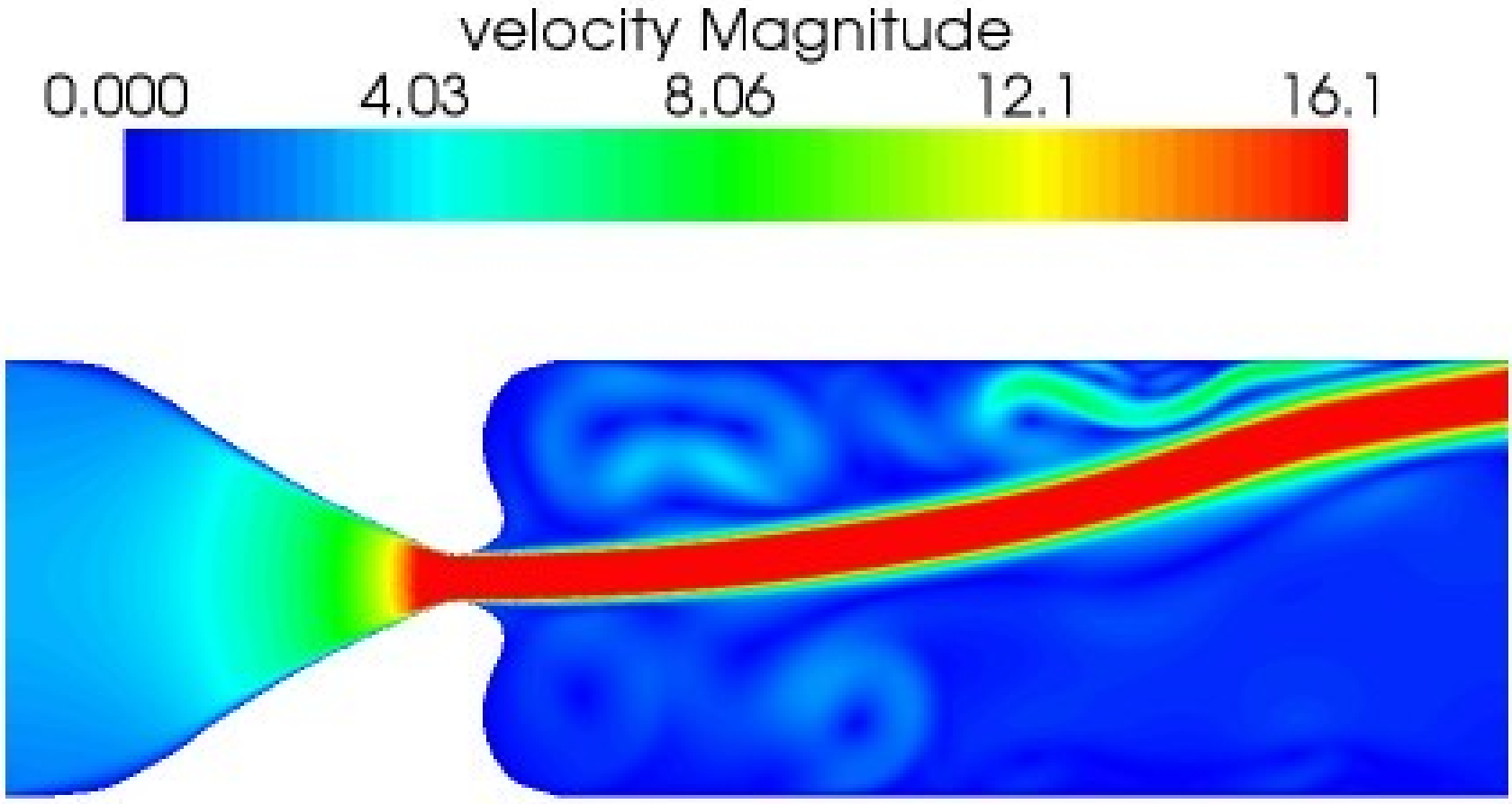

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

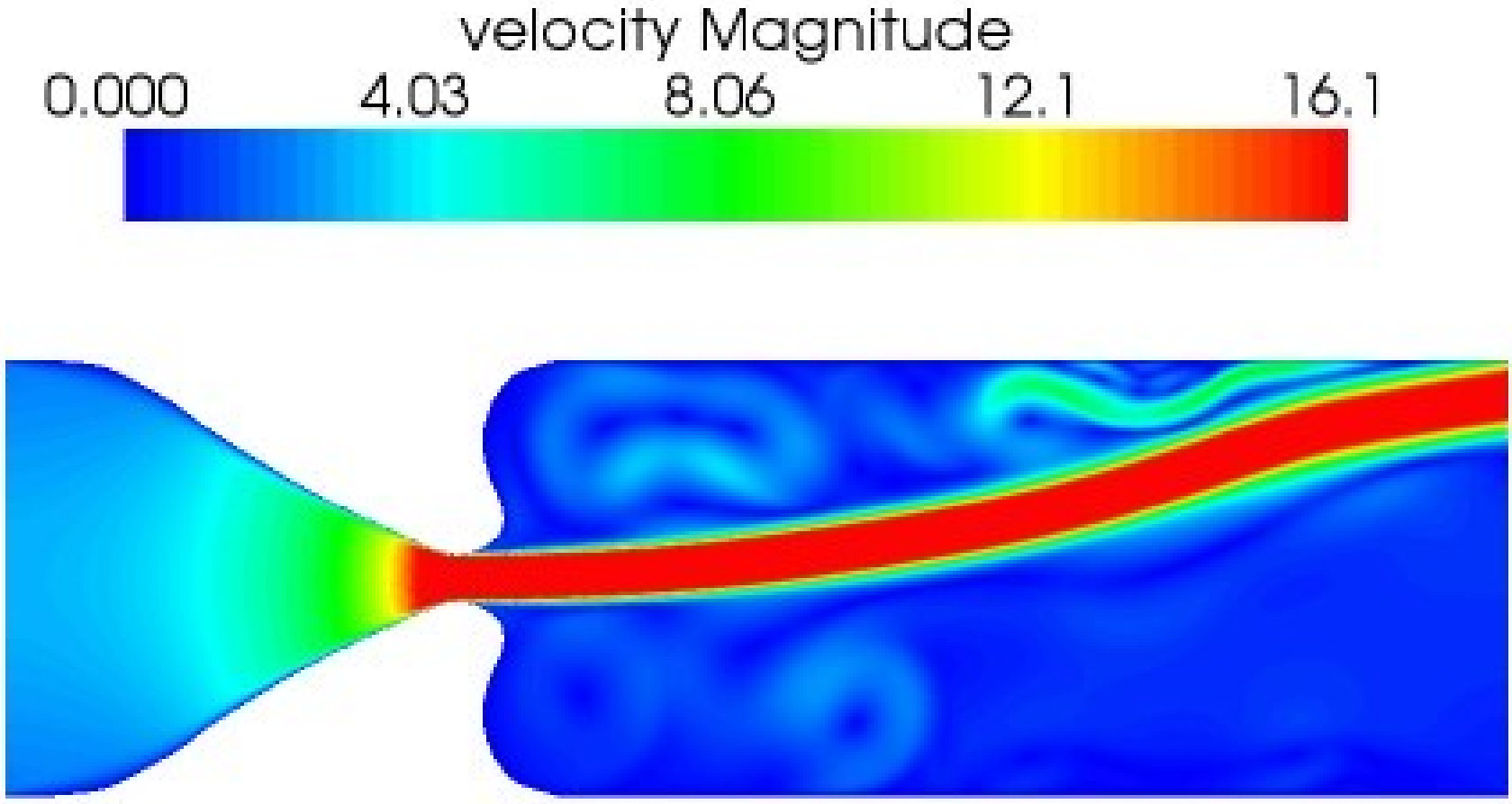

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

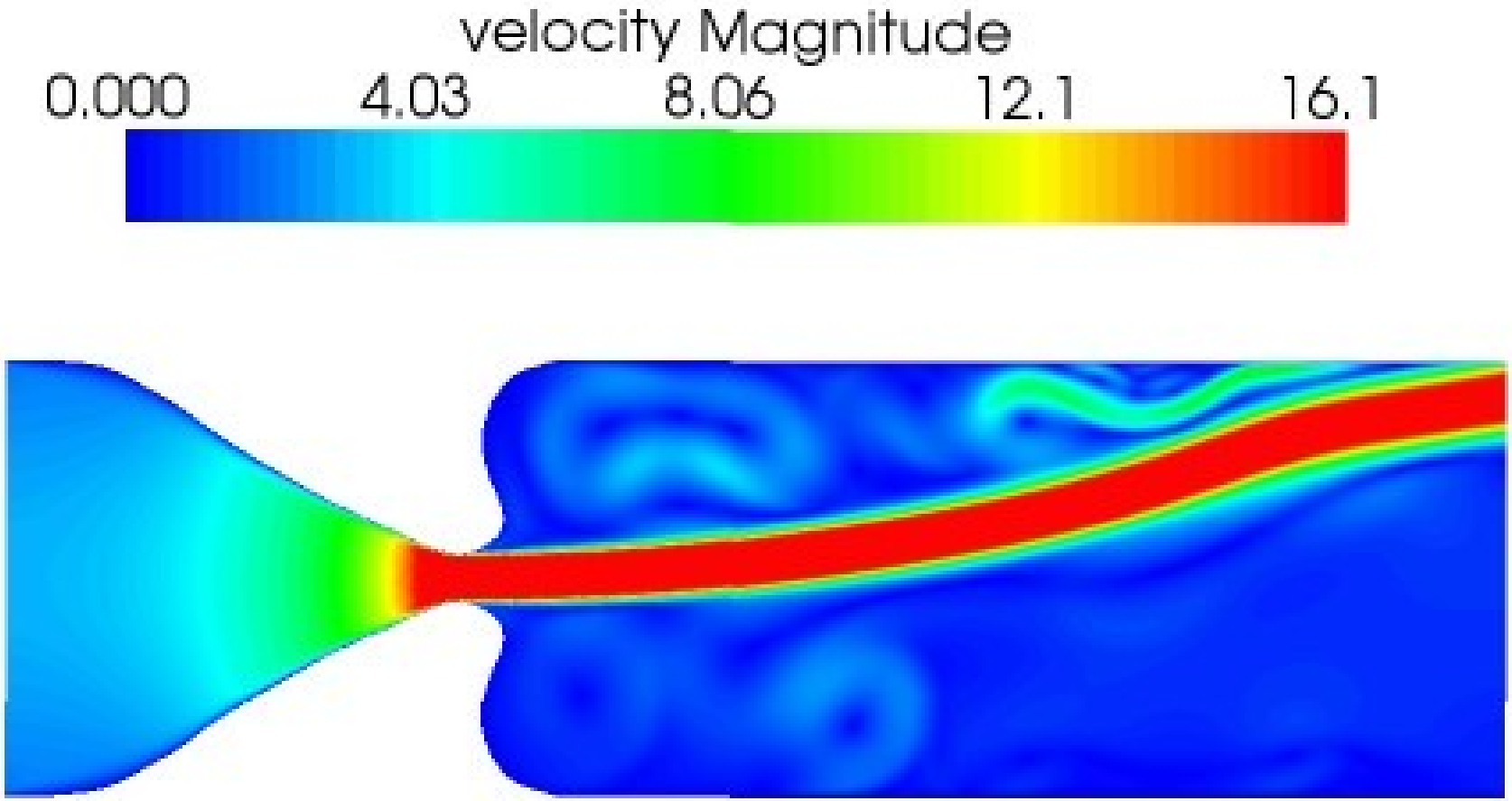

velocity Magnitude
0.000
4.03
8.06
12.1
16.1

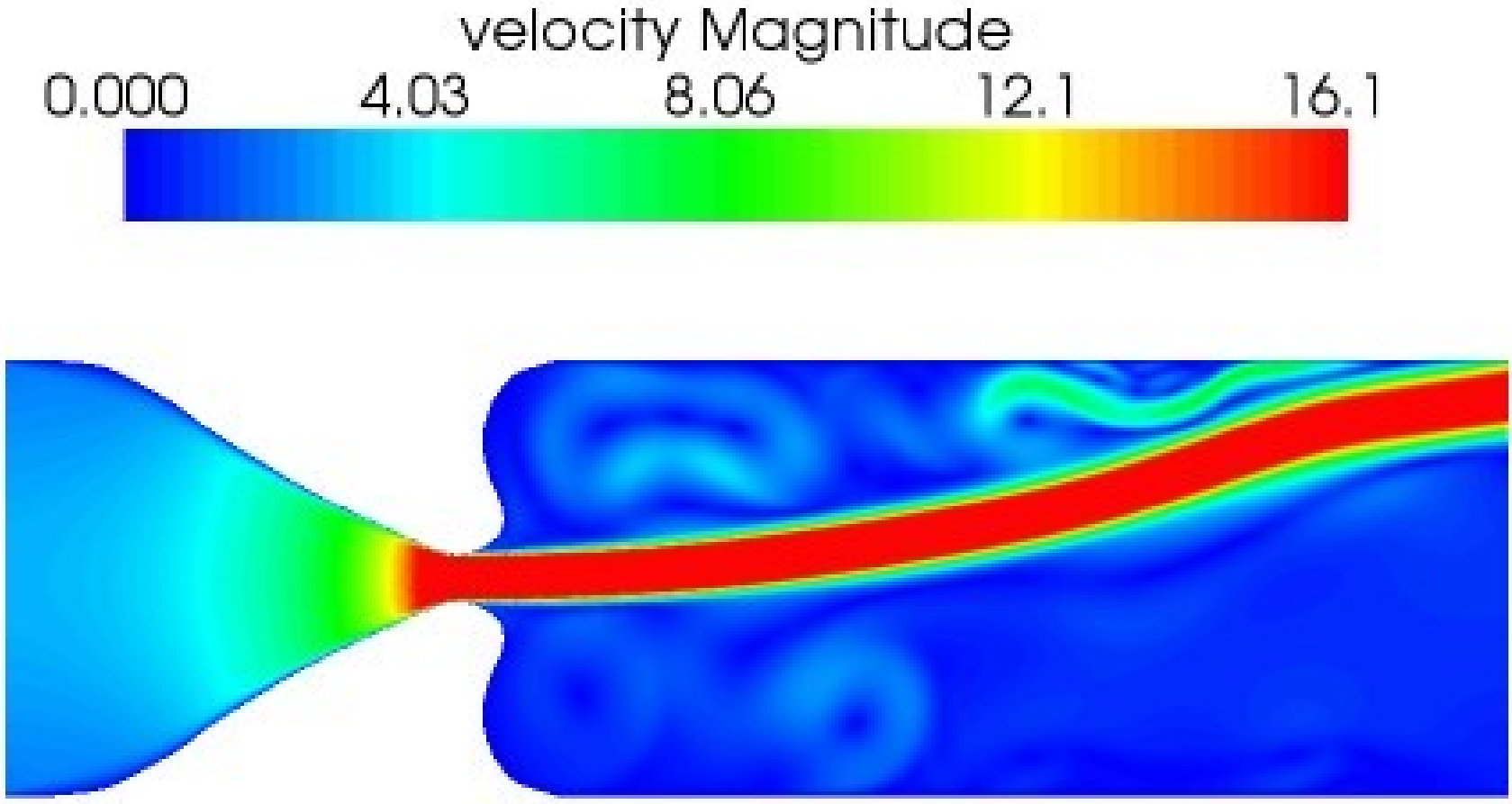

velocity Magnitude
0.000
4.03
8.06
12.1
16.1


velocity Magnitude
0.000
4.03
8.06
12.1
16.1


velocity Magnitude
0.000
4.03
8.06
12.1
16.1


velocity Magnitude
0.000
4.03
8.06
12.1
16.1


velocity Magnitude
0.000
4.03
8.06
12.1
16.1


velocity Magnitude
0.000
4.03
8.06
12.1
16.1