Solve Kinetic Equations with Deep Learning

Zheng Ma

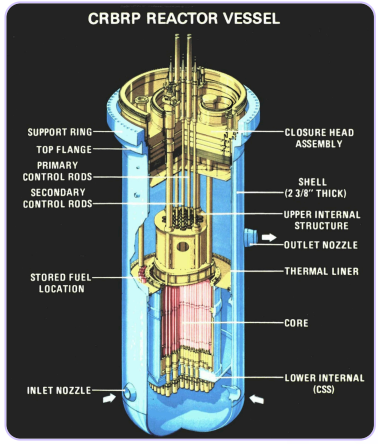
Shanghai Jiao Tong University

Nov. 12th 2024

Introduction

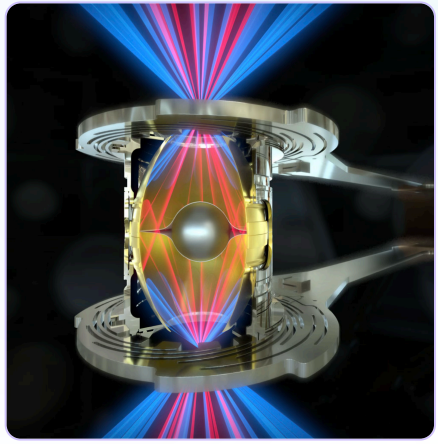
Kinetic equations are important in many areas

Neutron transport



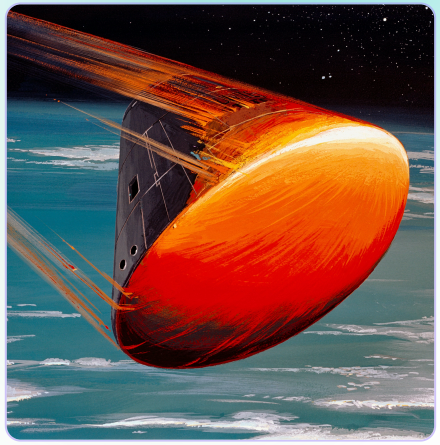
Fission reactor

Radiative transfer



ICF

Rarefied gas



Reentry

Key problem: numerical simulation of kinetic equations.

Multiscale Kinetic Equations

$$\partial_t f + v \cdot \nabla_x f = Q(f)$$

- f: distribution function of particles at time t in phase sapce (x, v)
- Q: collision operator
- ε: Knudsen number (ratio between mean free path and characteristic length)

Linear transport equation

$$\varepsilon \partial_t f + v \cdot \nabla_x f = \frac{1}{\varepsilon}\left(\frac{1}{2}\int_{-1}^{1} f \, \mathrm{d}v' - f\right)$$

BGK equation

$$\partial_t f + v \cdot \nabla_x f = \frac{1}{\varepsilon}\left(M(U) - f\right)$$

Multiscale problem: ε varies from $O(1)$ kinetic regime to 0 hydrodynamic regime.

Numerical Chanllenges

Curse of dimensionality

Dimension

- phase space + time: 6 + 1 = 7
- collision is a 5-fold integral
- evaluate collision at every phase point

Collision

- hard to determine the collision kernel
- highly non-linear for Boltzmann
- special velocity discretization is needed
- ray effect

Multiscale

- stability issues for small ε (stiffness)
- consistency of the scheme with limiting model as $\varepsilon \to 0$
- automatically capture the transition across regimes

Conventional Approaches

Probabilistic approach

- DSMC

Pros

- Easy for implementation
- Relatively efficient

Cons

- Low accuracy
- Converge slow
- Random fluctuations

Conventional Approaches

Probabilistic approach

- DSMC

Deterministic approach

- discrete velocity/ordinate methods — expensive, first or second order accuracy, maintain conservation.
- spectral methods — relatively expensive, spectral accuracy, do not maintain conservation.

Pros

- Easy for implementation
- Relatively efficient

Cons

- ⚠ expensive
- ⚠ first or second order accuracy
- ⚠ not main conservation

Solve PDEs with Deep Learning

Key components and core ideas of solving PDEs by Deep Neural Networks

- **Architecture**: build a deep neural network (function class) as the trail function
  - approximate solution (PINNs)
  - approximate solution operator (DeepONet, FNO)
  - mapping from equations (as a computational graph) to solutions (PDEFormer)
- **Constraints (as Loss)**:
  - **Model**: PDE / physical information needed (e.g., PINNs, DeepRitz, Deep-Galerkin)
  - Data: pure supervised or as a priori information
  - IC (initial conditions) and BC (boundary conditions)
  - Other constraints: **conservation**, symmetry, etc.
- **Optimization**: minimize loss over the parameter space, usually SGD, Adam, LBFGS, etc.

Numerical Challenges

Good icons need to be:

Colorable

Adapt the color on the fly

Scalable

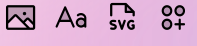
Resize as ease

Large Amount

Manage hundreds of icons

Bundling

Optimize bundle



Loading

Fetch icons on-demand

Dynamic

Icons not known at compile time

Anthony Fu

Core team member of Vite Vue and Nuxt

Creator of Vitest Slidev UnoCSS Type Challenges Elk

Maintainer of ESLint Stylistic Shiki Twoslash

Working at NuxtLabs

antfu.me antfu antfu@webtoo.ls antfu7

Icons

Simple to define, but complicated to engineer

Engineering Challenges

Good icons need to be:

Colorable

This is a text

Scalable

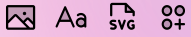
Resize as ease

Large Amount

Manage hundreds of icons

Bundling

Optimize bundle



Loading

Fetch icons on-demand

Dynamic

Icons not known at compile time

Solution 1: `<img>`

Icons as images

```
<img src="/icons/fugue/gear.png" alt="Gear" width="16" height="16" />
```

Fugue Icons by Yusuke Kamiyamane

Pros

- Simple, mo additional setup
- Native browser support
- No runtime dependency

Cons

- No color control
- Pixelated on scaling
- Large amount of requests
- Flash of invisible icons
- Verbose to write
- Assets & paths management

Solution 2: Web Fonts

Font Awesome 4 for example:

SVG icons are converted to font glyphs and assigned a special unicode character for each icon.

Use CSS classes to apply the icon.

```
<i class="fa fa-address-book-o" aria-hidden="true"></i>
```

```
/* font-awesome.css */
@font-face {
  font-family: 'FontAwesome';
  src: url('../fonts/fontawesome-webfont.eot?v=4.7.0');
}
.fa {
  font: normal normal normal 14px/1 FontAwesome;
}
/* class for each icon */
.fa-address-book-o:before {
  content: '\f2ba';
}
```

Pros

- Easy to use
- Colorable
- Scalable, SVG based
- Single request

Cons

- Large font file, all icons loaded upfront
- Hard to customize
- Flash of invisible icons

Solution 3: Components

Example of using MDI icons:

```vue
<script setup>
import SvgIcon from '@jamescoyle/vue-icon'
import { mdiAccount } from '@mdi/js'
</script>

<template>
  <SvgIcon type="mdi" :path="mdiAccount" />
</template>
```

Example of using Tabler icons:

```vue
<script setup>
import { IconHome } from '@tabler/icons-vue'
</script>

<template>
  <IconHome />
</template>
```

Pros

- Colorable, Scalable
- Full features of SVG
- No requests, no flash of invisible icons
- SSR friendly

Cons

- Creates a lot of SVG DOM elements
- Bundle size
- Requires specific integration support
- Vendor lock-in, hard to switch

Iconify

- One solution for all popular icon sets
- Unified collections of data for consuming
- 100+ icon sets, 200,000+ icons
- Iconify API for dynamic icons

Sponsor Iconify

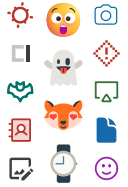
Home of open source icon sets

All popular icon sets, one framework.

Over 200,000 open source vector icons.

Iconify makes it easy to avoid vendor lock-in.

You can use many open source icon sets with a large choice of open source icon components. Thousands of high-quality icons from 100+ icon sets, all validated, cleaned up, optimised and always up to date.



Material Design, Phosphor, Remix, Carbon, Bootstrap, Tabler, Feather, Fluent, IconPark, Octicons and many other icon sets. Twitter Emoji, Fluent Emoji, EmojiOne, Noto Emoji are also available as icon sets.

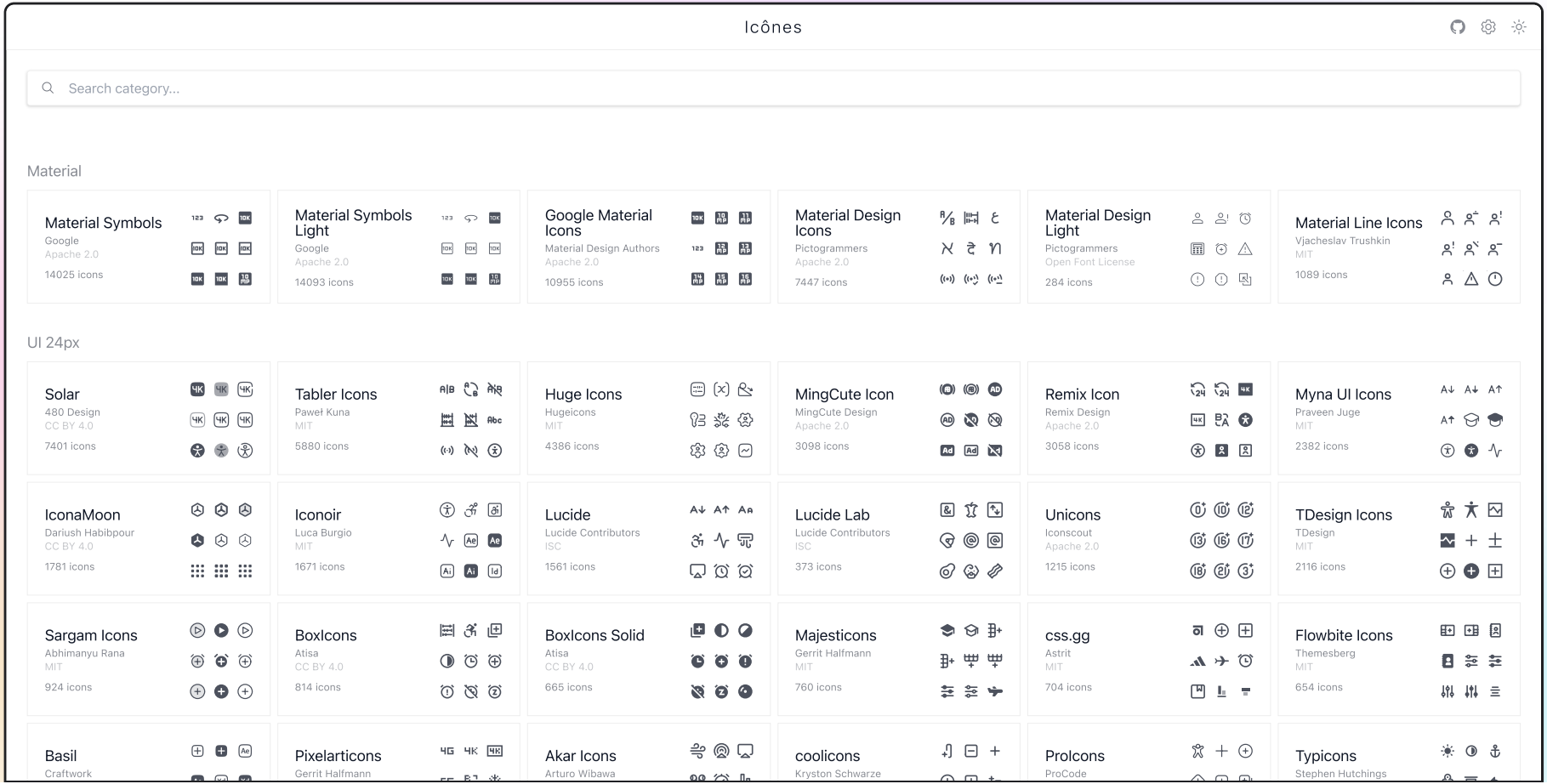
Use them with the same tools, same reusable and customisable icon components:

- Iconify Icon web component (HTML with or without UI frameworks. Works great with SSR).
- Iconify framework native components (React, Vue, Svelte, Ember).

You can also get raw SVG using various tools and embed them in your pages without any components.

Iconify is supported by a growing community. In addition to Iconify icon components, there are now more ways to use icons, created by amazing open source developers:

Icônes icones.js.org



Solution 4: Iconify Runtime

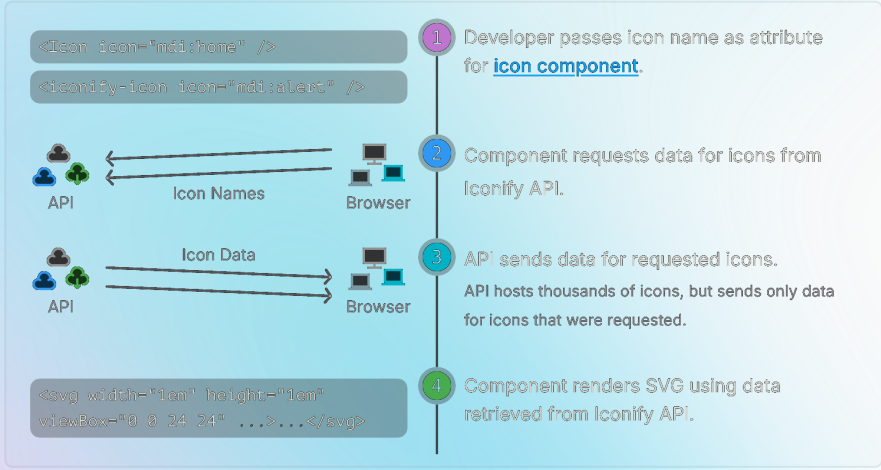
Import the Iconify library on entry (or CDN):

```
import 'iconify-icon'
```

Usage as Web Components:

```
<iconify-icon icon="mdi:home"></iconify-icon>
```

Upon usage, iconify will fetch the icon from their API to render it.



Pros

- Support dynamic icons on-demand
- Does not require build setup
- Benfits of using SVG icons

Cons

- Runtime dependency
- Flash of invisible icons
- Requires cross site network requests
- Not SSR nor PWA friendly

Solution 5: On-demand Components

```vue
<script setup>
import IconAccessibility from '~icons/carbon/accessibility'
import IconAccountBox from '~icons/mdi/account-box'
// ... any icons, bundles on-demand
</script>

<template>
  <IconAccessibility />
  <IconAccountBox style="font-size: 2em; color: red" />
</template>
```

- Vue 3 + Vite + Carbon Icons
- React + Next.js + Material Design Icons
- Vue 2 + Nuxt.js + Unicons
- React + Webpack + Twemoji
- Solid + Vite + Tabler
- Vanila + Rollup + BoxIcons
- Web Components + Vite + Ant Design Icons
- Svelte + SvelteKit + EOS Icons
- Any + Any + Any

Pros

- All benifits of using SVG icons
- Use any icons, and bundle for what you use
- Supports most of the popular frameworks
- Optional auto-import

Cons

- Creates a lot of SVG DOM elements
- Bundle size
- Only work for build-time known icons

Solution 6: Pure CSS Icons

Create icons with only CSS, with inline SVG as data URL

```css
.i-ph-acorn-duotone {
  mask-image: url(data: ... ) no-repeat;
  mask-size: 100% 100%;
  background: currentColor;
  height: 1em;
  width: 1em;
}
```

```html
<div class="i-ph-acorn-duotone"></div>
```

Carbon

Tabler

Phosphor

Twemoji

Catppuccin

Spinners

Pros

- Colorable, Scalable
- One DOM element per icon
- No requests, no flash of invisible icons
- Zero runtime
- SSR friendly

Cons

- Only work for build-time known icons
- Cannot change elements inside icons

Integrate Icons to Nuxt

INTEGRATIONS CHALLENGES

SSR / CSR

Seamlessly support both modes, no flashes

Dynamic Icons

Support dynamic known icons, e.g. from Nuxt Content

Performant

No compromise on performance

Custom Icons

Support loading user-provided custom icons

SOLUTIONS

Iconify Runtime

- ✓ Dynamic Icons
- ✗ No SSR
- ✗ No Custom Icons

CSS Icons

- ✓ SSR / CSR
- ✓ Perfoment
- ✗ No Dynamic Icons

Dual Rendering Modes

CSS Mode

```
<Icon mode="css" name="ph:arrow-down-duotone" />
```

↓

```
<div class="i-ph-arrow-down-duotone iconify"></div>
```

```
.i-ph-arrow-down-duotone {
  mask-image: url(data: ... ) no-repeat;
  background: currentColor;
  /* ... */
}
```

SVG Mode

```
<Icon mode="svg" name="ph:arrow-down-duotone" />
```

↓

```
<svg class="iconify" viewBox="0 0 24 24">
  <path d="M3 12h18M13 5l7 7-7 7"></path>
  <!-- ... -->
</svg>
```

Both mode can be serialized on SSR with no runtime cost

Bundles

Bundles to client

- ⊕ No additional requests
- ⊕ Renders instantly
- ⊖ Bundle size is a concern
- ⊖ Only known icons

Client Bundle

Bundles only the static known icons on-demand

Bundles to server

- ⊕ Bundle size insensitive
- ⊕ Can include full icon sets
- ⊕ No requests when SSR

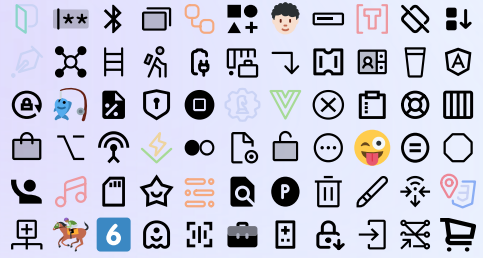
Server Bundle

Bundles full sets for frequently used icons, serve on-demand via SSR or client requests

Fetch from Iconify

- ⊕ All icons on-demand
- ⊖ Additional requests
- ⊖ Servers can be far away

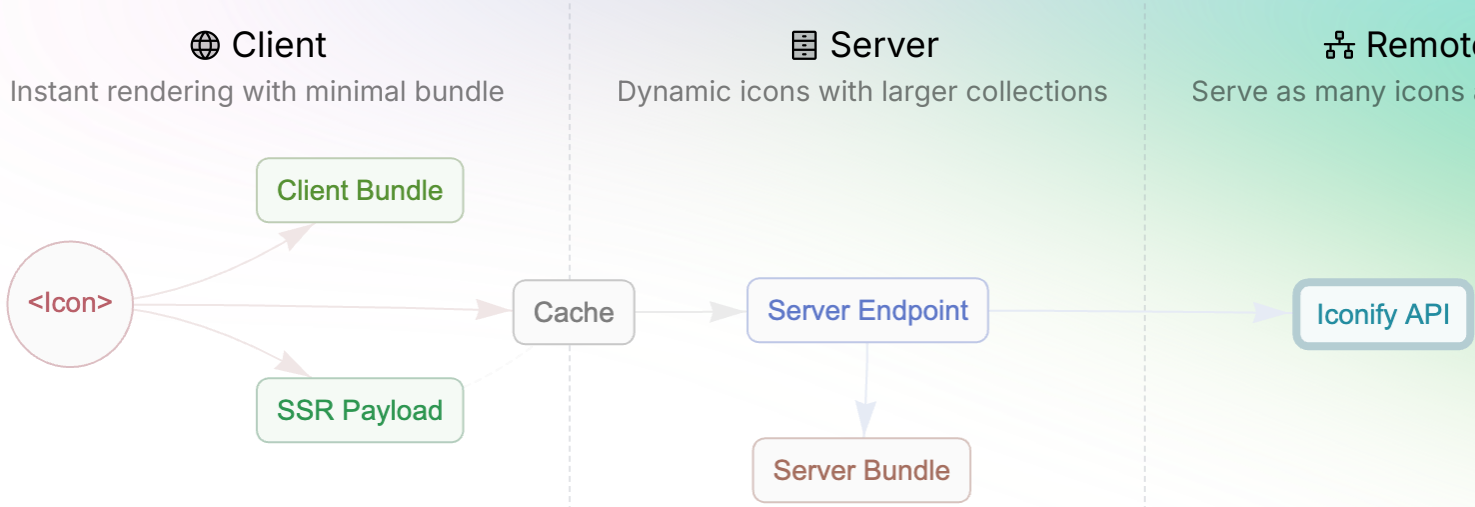
Iconify API



Fallback to serve as many icons as possible

Data Flow

Load icons efficiently while supporting dynamic icons



```
pnpm i -D @nuxt/icon
```

```
export default defineNuxtConfig({
  modules: [
    '@nuxt/icon'
  ]
})
```

```
<template>
  <Icon name="ph:arrow-down-duotone" />
</template>
```

Thank You!

Slides can be found on antfu.me