New low-storage IMEXRK schemes for the simulation of high-dimensional stiff ODEs derived from diffusive PDE systems

Thomas Bewley, Daniele Cavaglieri
UCSD Flow Control & Coordinated Robotics Labs
http://robotics.ucsd.edu

\[ \frac{dx(t)}{dt} = f(x(t), t) \]
Channel flow at $\text{Re}_\tau = 5186$

Lee and Moser JFM 2015
\[ \frac{dx(t)}{dt} = f(x(t), t) = Ax \]

**Explicit Euler (EE)**

\[ x_{n+1} = x_n + hf(x_n, t_n). \]

**Implicit Euler (IE)**

\[ x_{n+1} = x_n + hf(x_{n+1}, t_{n+1}). \]

**Crank Nicolson (CN)**

\[ x_{n+1} = x_n + h\left[ f(x_{n+1}, t_{n+1}) + f(x_n, t_n) \right]/2. \]
\[ \frac{dx(t)}{dt} = f(x(t), t) = Ax \]

\[
A = -1
\]

**Explicit Euler (EE)**

\[
x_{n+1} = x_n + hf(x_n, t_n).
\]

**Implicit Euler (IE)**

\[
x_{n+1} = x_n + hf(x_{n+1}, t_{n+1}).
\]

**Crank Nicolson (CN)**

\[
x_{n+1} = x_n + h[f(x_{n+1}, t_{n+1}) + f(x_n, t_n)]/2.
\]

Starting point concepts:

accuracy, stability.
Accuracy

Trapezoidal (2nd-order) integration formula:

\[
\int_{L}^{R} f(x) \, dx \approx \sum_{i=1}^{n} h_i \frac{f_{i-1} + f_i}{2} = \frac{h}{2} \left[ f_0 + f_n + 2 \sum_{i=1}^{n-1} f_i \right]
\]

Keeping track of leading order error we may...

- Keep uniform grid, adaptively refine formula to achieve higher order
- Keep 2nd-order formula, adaptively refine the grid

```
<table>
<thead>
<tr>
<th>Gridpoints</th>
<th>2\textsuperscript{nd}-Order Approximation</th>
<th>4\textsuperscript{th}-Order Correction</th>
<th>6\textsuperscript{th}-Order Correction</th>
<th>8\textsuperscript{th}-Order Correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n_1 = 2^1 = 2 )</td>
<td>( I_{1,1} )</td>
<td>( I_{2,1} )</td>
<td>( I_{3,1} )</td>
<td>( I_{4,1} )</td>
</tr>
<tr>
<td>( n_2 = 2^2 = 4 )</td>
<td>( I_{2,1} )</td>
<td>( I_{2,2} )</td>
<td>( I_{3,2} )</td>
<td>( I_{4,2} )</td>
</tr>
<tr>
<td>( n_3 = 2^3 = 8 )</td>
<td>( I_{3,1} )</td>
<td>( I_{3,2} )</td>
<td>( I_{3,3} )</td>
<td>( I_{4,3} )</td>
</tr>
<tr>
<td>( n_4 = 2^4 = 16 )</td>
<td>( I_{4,1} )</td>
<td>( I_{4,2} )</td>
<td>( I_{4,3} )</td>
<td>( I_{4,4} )</td>
</tr>
</tbody>
</table>
```

"Romberg" integration

"Romberg" integration

"adaptive integration"
**Accuracy versus order of accuracy**

Order of accuracy is only a means to an end. What matters in the end is the accuracy of the calculation that you can actually afford to perform, with finite $h$, not the accuracy of a hypothetical calculation with infinitesimal $h$.

In this regard, intermediate order (3rd, 4th, …) is often better than higher order.

Left: quadrature using Romberg Integration (increasingly higher order, uniform grid)
Middle: quadrature using adaptive integration (selectively refining grid, 2nd-order method)
Right: magnitude of error as a function of # of function evaluations
Stability

Explicit Euler (EE)

\[ x_{n+1} = x_n + hf(x_n, t_n) \]

\[ \sigma = 1 + \lambda h \]

Implicit Euler (IE)

\[ x_{n+1} = x_n + hf(x_{n+1}, t_{n+1}) \]

\[ \sigma = \frac{1}{1 - \lambda h} \]

Crank Nicolson (CN)

\[ x_{n+1} = x_n + h[f(x_{n+1}, t_{n+1}) + f(x_n, t_n)]/2 \]

\[ \sigma = \frac{1 + \frac{\lambda h}{2}}{1 - \frac{\lambda h}{2}} \]

Stability regions in the complex plane $\lambda h$ for the numerical solution of $x' = \lambda x$ with timestep $h$ using (left) EE, (center) IE, and (right) CN. The stable regions (i.e., the regions for which $|\sigma| \leq 1$) are shaded.
Non-stiff ODEs versus stiff ODEs

The magnitude of the eigenvalues of $A$ vary over a large range for stiff ODEs. Stiff ODEs result from discretization of higher derivatives (e.g., diffusion terms).

\[ \frac{dx(t)}{dt} = f(x(t), t) \quad \Rightarrow \quad \frac{dx(t)}{dt} = Ax \]
\[ \frac{dx}{dt} = \lambda x \]
\[ x_n = \sigma^n x_0 \]

Characterizing Stability

- **$L$-stable** if its stability region contains the entire LHP and $\sigma(\infty) = 0$;
- **strongly $A$-stable** if its stability region contains the entire LHP and $|\sigma(\infty)| < 1$;
- **stiffly stable** if its stability region contains the solid shaded region in Figure and $\sigma(\infty) = 0$;
- **$A$-stable** if its stability region contains the entire LHP;
- **$A(\alpha)$ stable** if its stability region contains the solid shaded region in Figure for an angle $\alpha > 0$;
- **$A(0)$ stable** if it is $A(\alpha)$ stable for some (unspecified) $\alpha > 0$;
- **$A_0$ stable** if the open negative real axis is stable.
What limits execution speed on modern processors?

Modern computational hardware is storage-limited on big problems.
Multi-step methods versus single-step methods

\[ \frac{dx(t)}{dt} = f(x(t), t) \]

Linear Multi-step Method

\[ x_{n+1} + \sum_{i=1}^{q} \alpha_i x_{n+1-i} = h \sum_{i=0}^{r} \beta_i f(x_{n+1-i}, t_{n+1-i}). \]

Explicit Runge Kutta (ERK) Method

\[
\begin{align*}
f_1 &= f(x_n, t_n + c_1 h) \\
f_2 &= f(x_n + a_{2,1} h f_1, t_n + c_2 h) \\
& \vdots \\
f_s &= f(x_n + a_{s,1} h f_1 + \ldots + a_{s,s-1} h f_{s-1}, t_n + c_s h) \\
x_{n+1} &= x_n + h[b_1 f_1 + \ldots + b_{s-1} f_{s-1} + b_s f_s],
\end{align*}
\]

Multi-step methods use extra storage to make maximum use of each flop.

Single-step methods use extra flops, and can be developed in such a way to minimize storage.

Modern computational hardware is storage-limited on big problems. 
Explicit Runge Kutta (ERK) Methods

\[ \frac{dx(t)}{dt} = f(x(t), t) \]

**Numerical implementation**

for \( i = 1 : s \)

\[ f_i \leftarrow f(x_n + h \sum_{j=1}^{i-1} a_{i,j} f_j, t_n + c_i h) \]

end

\[ x_{n+1} = x_n + h \sum_{i=1}^{s} b_i f_i \]

**Butcher tableau**

\[
\begin{array}{c|ccc}
   c_1 & 0 \\
   c_2 & a_{2,1} & 0 \\
   \vdots & \vdots & \ddots & \vdots \\
   c_s & a_{s,1} & \cdots & a_{s,s-1} & 0 \\
\end{array}
\]

\[
\begin{array}{c}
   b_1 \\
   \vdots \\
   b_{s-1} \\
   b_s \\
\end{array}
\]

**Remarks**

- ERK ↔ strictly lower triangular \( A \) (explicit)
- IRK ↔ full \( A \) (fully implicit - too hard!)
- DIRK ↔ lower triangular \( A \) (implicit steps)
- \( \{A, b, c\} \) selected to match terms in Taylor expansion on RHS of ODE to given order

**Considering the linear scalar ODE:**

\[ \dot{x}(t) = \lambda x(t) \Rightarrow x_{n+1} = \sigma(\lambda h) x_n \]

Stability region is where \( |\sigma(\lambda h)| \leq 1 \).
Implicit Runge Kutta (IRK) Methods

\[
\frac{dx(t)}{dt} = f(x(t), t)
\]

**Numerical implementation**

```
guess \( f_1 \) through \( f_s \)
repeat until converged:
    for \( i = 1 : s \)
        \[ f_i \leftarrow f(x_n + h \sum_{j=1}^{s} a_{i,j} f_j, t_n + c_i h) \]
    end
\[ x_{n+1} = x_n + h \sum_{i=1}^{s} b_i f_i \]
```

**Butcher tableau**

| \( c_i \) | \( a_{1,1} \) | \( a_{1,2} \) | \( \cdots \) | \( a_{1,s} \) |
|----------|----------------|----------------|----------------|
| \( c_2 \) | \( a_{2,1} \) | \( a_{2,2} \) | \( \cdots \) | \( a_{2,s} \) |
| \( \vdots \) | \( \vdots \) | \( \vdots \) | \( \ddots \) | \( \vdots \) |
| \( c_s \) | \( a_{s,1} \) | \( a_{s,2} \) | \( \cdots \) | \( a_{s,s} \) |
| \( b_1 \) | \( b_2 \) | \( \cdots \) | \( b_s \) |

**Remarks**

- ERK \iff\ strict lower triangular \( A \) (explicit)
- IRK \iff\ full \( A \) (fully implicit - too hard!)
- DIRK \iff\ lower triangular \( A \) (implicit steps)
- \( \{A, b, c\} \) selected to match terms in Taylor expansion on RHS of ODE to given order

**Considering the linear scalar ODE:**

\[ \dot{x}(t) = \lambda x(t) \quad \rightarrow \quad x_{n+1} = \sigma(\lambda h) x_n \]

Stability region is where \( |\sigma(\lambda h)| \leq 1 \).

- Scheme is:
  - \( A \)-stable if \( \sigma(\lambda h) \leq 1 \) over LHP.
  - \( L \)-stable if also \( \lim_{\lambda h \to \infty} \sigma(\lambda h) = 0 \).
\[
\frac{dx(t)}{dt} = f(x(t), t)
\]

**Diagonally-Implicit Runge Kutta (DIRK) Methods**

### Numerical implementation

for \( i = 1 : s \)

solve implicitly:

\[
f_i = f(x_n + h \sum_{j=1}^{i-1} a_{i,j} f_j + h a_{i,i} f_i, t_n + c_i h)
\]

end

\[
x_{n+1} = x_n + h \sum_{i=1}^{s} b_i f_i
\]

Considering the linear scalar ODE:

\[
\dot{x}(t) = \lambda x(t) \quad \rightarrow \quad x_{n+1} = \sigma(\lambda h) x_n
\]

Stability region is where \(|\sigma(\lambda h)| \leq 1\).

Scheme is:
- **A-stable** if \( \sigma(\lambda h) \leq 1 \) over LHP.
- **L-stable** if also \( \lim_{\lambda h \to \infty} \sigma(\lambda h) = 0 \).

### Butcher tableau

\[
\begin{array}{c|ccc}
\quad & a_{1,1} & & \\
\hline
c_1 & & & \\
\quad & a_{2,1} & a_{2,2} & \\
\quad & & \ddots & \ddots \\
\quad & a_{s,1} & \cdots & a_{s,s-1} & a_{s,s} \\
\hline
b_1 & b_2 & \cdots & b_s
\end{array}
\]

### Remarks

- ERK ↔ strictly lower triangular \( A \) (explicit)
- IRK ↔ full \( A \) (fully implicit - too hard!)
- DIRK ↔ lower triangular \( A \) (implicit steps)
- \( \{A, b, c\} \) selected to match terms in Taylor expansion on RHS of ODE to given order
The classical 4th-order ERK Method

Numerical implementation

\[ f_1 = f(x_n, t_n) \]
\[ f_2 = f(x_n + h/2 f_1, t_n + h/2) \]
\[ f_3 = f(x_n + h/2 f_2, t_n + h/2) \]
\[ f_4 = f(x_n + h f_3, t_n + h) \]
\[ x_{n+1} = x_n + h (f_1/6 + f_2/3 + f_3/3 + f_4/6) \]

Butcher tableau

\[
\begin{array}{c|ccc}
0 & 0 & 0 & 0 \\
1/2 & 1/2 & 0 & 0 \\
1/2 & 0 & 1/2 & 0 \\
1 & 0 & 0 & 1 & 0 \\
\hline
1/6 & 1/3 & 1/3 & 1/6
\end{array}
\]

Stability region

Remarks

- The scheme is explicit
- Five registers required for time integration

The stable regions (i.e., the regions for which \(|\sigma| \leq 1\)) are shaded.
The classical 2nd-order CN Method (DIRK)

Numerical implementation

\[
\begin{align*}
    f_1 &= f(x_n, t_n) \\
    f_2 &= f(x_n + \frac{h}{2} f_1 + \frac{h}{2} f_2, t_n + h) \\
    x_{n+1} &= x_n + h \left( \frac{f_1}{2} + \frac{f_2}{2} \right)
\end{align*}
\]

Butcher tableau

\[
\begin{array}{cccc}
    0 & 0 & 0 \\
    1 & \frac{1}{2} & \frac{1}{2} \\
    \frac{1}{2} & \frac{1}{2} \\
\end{array}
\]

Stability region

Remarks

- The scheme is DIRK and A-stable
- Three registers required for time integration
Low-storage [2R] ERK schemes

With large simulations come large storage requirements

Low-storage Runge-Kutta schemes may be developed. RKW3 (mid 1980s, by Alan Wray) was the first.

Butcher tableau

<table>
<thead>
<tr>
<th>$c_1$</th>
<th>0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_2$</td>
<td>$a_{2,1}$</td>
<td>0</td>
</tr>
<tr>
<td>$c_3$</td>
<td>$b_1$</td>
<td>$a_{3,2}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$c_s$</td>
<td>$b_1$</td>
<td>$b_2$</td>
</tr>
<tr>
<td></td>
<td>$b_1$</td>
<td>$b_2$</td>
</tr>
</tbody>
</table>

Numerical implementation

\[
\text{for } i = 1 : s \\
\quad \text{if } i = 1, \ y \leftarrow x, \ \text{else} \\
\quad \quad y \leftarrow x + h (a_{i,i-1} - b_{i-1}) y \\
\quad \text{end} \\
\quad y \leftarrow f(y, t_n + c_i h) \\
\quad x \leftarrow x + h b_i y \\
\text{end}
\]

Remarks

- Time integration carried out with only two registers
- Less degrees of freedom available for design
- Only explicit schemes found in literature (before the present work)
Mixed Implicit/Explicit Runge-Kutta (IMEXRK) schemes

ODE: \[ \dot{x}(t) = f(x(t), t) + g(x(t), t) \]

- \( f \) is a linear stiff term, \( g \) is a nonlinear nonstiff term

IMEXRK schemes take the best of DIRK and ERK schemes by combining them for time integration:

**DIRK component**

**ERK component**

An extensive review in *Kennedy & Carpenter, 2001*

**Remarks**

- 6, 20, 72 constraints must be satisfied for 2\(^{nd}\), 3\(^{rd}\), 4\(^{th}\) order accuracy, resp.
- Mostly full-storage schemes in literature (CN/RKW3 is notable exception).
Now, put pieces together: design low-storage IMEXRK!

**DIRK component**

<table>
<thead>
<tr>
<th>$c_1$</th>
<th>$\bar{a}_{1,1}^I$</th>
<th>$\bar{a}_{2,1}^I$</th>
<th>$\bar{a}_{2,2}^I$</th>
<th>$\bar{a}_{3,2}^I$</th>
<th>$\bar{a}_{3,3}^I$</th>
<th>$\bar{a}_{4,4}^I$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_2$</td>
<td>$\bar{b}_1^I$</td>
<td>$a_{2,1}^I$</td>
<td>$\bar{a}_{3,2}^I$</td>
<td>$\bar{a}_{3,3}^I$</td>
<td>$\bar{a}_{4,4}^I$</td>
<td>$\bar{a}_{4,4}^I$</td>
</tr>
<tr>
<td>$c_3$</td>
<td>$\bar{b}_2^I$</td>
<td>$\bar{b}_2^I$</td>
<td>$a_{3,3}^I$</td>
<td>$\bar{a}_{4,4}^I$</td>
<td>$\bar{a}_{4,4}^I$</td>
<td>$\bar{a}_{4,4}^I$</td>
</tr>
<tr>
<td>$c_4$</td>
<td>$\bar{b}_3^I$</td>
<td>$\bar{b}_3^I$</td>
<td>$\bar{b}_3^I$</td>
<td>$\bar{b}_3^I$</td>
<td>$\bar{b}_3^I$</td>
<td>$\bar{b}_3^I$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$c_s$</td>
<td>$\bar{b}_1^I$</td>
<td>$\bar{b}_2^I$</td>
<td>$\bar{b}_s-2^I$</td>
<td>$a_{s,s-1}^I$</td>
<td>$a_{s,s}^I$</td>
<td>$a_{s,s}^I$</td>
</tr>
</tbody>
</table>

| $b_1^I$ | $b_2^I$ | $\ldots$ | $\bar{b}_{s-2}^I$ | $\bar{b}_{s-1}^I$ | $\bar{b}_s^I$ |

**ERK component**

<table>
<thead>
<tr>
<th>$c_1$</th>
<th>$\bar{a}_{2,1}^E$</th>
<th>$0$</th>
<th>$\bar{a}_{3,2}^E$</th>
<th>$\bar{a}_{3,3}^E$</th>
<th>$0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_2$</td>
<td>$\bar{b}_1^E$</td>
<td>$a_{2,1}^E$</td>
<td>$\bar{a}_{3,2}^E$</td>
<td>$\bar{a}_{3,3}^E$</td>
<td>$0$</td>
</tr>
<tr>
<td>$c_3$</td>
<td>$\bar{b}_2^E$</td>
<td>$\bar{a}_{3,2}^E$</td>
<td>$a_{3,3}^E$</td>
<td>$\bar{a}_{4,3}^E$</td>
<td>$0$</td>
</tr>
<tr>
<td>$c_4$</td>
<td>$\bar{b}_3^E$</td>
<td>$\bar{b}_2^E$</td>
<td>$b_{s-2}^E$</td>
<td>$b_{s-1}^E$</td>
<td>$a_{4,3}^E$</td>
</tr>
<tr>
<td>$c_5$</td>
<td>$\bar{b}_4^E$</td>
<td>$\bar{b}_3^E$</td>
<td>$\ldots$</td>
<td>$\bar{b}_{s-2}^E$</td>
<td>$b_{s-1}^E$</td>
</tr>
<tr>
<td>$c_s$</td>
<td>$\bar{b}_s^E$</td>
<td>$\bar{b}_s^E$</td>
<td>$\ldots$</td>
<td>$\bar{b}_{s-2}^E$</td>
<td>$b_{s-1}^E$</td>
</tr>
</tbody>
</table>

| $b_1^E$ | $b_2^E$ | $\ldots$ | $\bar{b}_{s-2}^E$ | $\bar{b}_{s-1}^E$ | $\bar{b}_s^E$ |

**Remarks**

- Setting $c_i^I = c_i^E = c_i$ and $b_i^I = b_i^E = b_i$ reduces the number of constraints required to achieve a desired order of accuracy.
Now, put pieces together: design low-storage IMEXRK!

**DIRK component**

<table>
<thead>
<tr>
<th></th>
<th>$0$</th>
<th>$a^I_{2,1}$</th>
<th>$a^I_{2,2}$</th>
<th>$a^I_{3,2}$</th>
<th>$a^I_{3,3}$</th>
<th>$a^I_{4,3}$</th>
<th>$a^I_{4,4}$</th>
<th>$\cdots$</th>
<th>$a^I_{s,s-1}$</th>
<th>$a^I_{s,s}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_2$</td>
<td>$b_1$</td>
<td>$b_2$</td>
<td>$\cdots$</td>
<td>$b_{s-2}$</td>
<td>$b_{s-1}$</td>
<td>$b_s$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c_3$</td>
<td>$b_1$</td>
<td>$b_2$</td>
<td>$\cdots$</td>
<td>$b_{s-2}$</td>
<td>$b_{s-1}$</td>
<td>$b_s$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c_4$</td>
<td>$b_1$</td>
<td>$b_2$</td>
<td>$\cdots$</td>
<td>$b_{s-2}$</td>
<td>$b_{s-1}$</td>
<td>$b_s$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\cdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c_s$</td>
<td>$b_1$</td>
<td>$b_2$</td>
<td>$\cdots$</td>
<td>$b_{s-2}$</td>
<td>$b_{s-1}$</td>
<td>$b_s$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ERK component**

<table>
<thead>
<tr>
<th></th>
<th>$0$</th>
<th>$a^E_{2,1}$</th>
<th>$0$</th>
<th>$a^E_{3,2}$</th>
<th>$0$</th>
<th>$a^E_{4,3}$</th>
<th>$0$</th>
<th>$\cdots$</th>
<th>$a^E_{s,s-1}$</th>
<th>$0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_2$</td>
<td>$b_1$</td>
<td>$b_2$</td>
<td>$\cdots$</td>
<td>$b_{s-2}$</td>
<td>$b_{s-1}$</td>
<td>$b_s$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c_3$</td>
<td>$b_1$</td>
<td>$b_2$</td>
<td>$\cdots$</td>
<td>$b_{s-2}$</td>
<td>$b_{s-1}$</td>
<td>$b_s$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c_4$</td>
<td>$b_1$</td>
<td>$b_2$</td>
<td>$\cdots$</td>
<td>$b_{s-2}$</td>
<td>$b_{s-1}$</td>
<td>$b_s$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\cdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c_s$</td>
<td>$b_1$</td>
<td>$b_2$</td>
<td>$\cdots$</td>
<td>$b_{s-2}$</td>
<td>$b_{s-1}$</td>
<td>$b_s$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Remarks**

- Setting $c^I_i = c^E_i = c_i$ and $b^I_i = b^E_i = b_i$ reduces the number of constraints required to achieve a desired order of accuracy.
Goals when seeking the IMEXRK coefficients

An IMEXRK scheme should guarantee

- Intermediate order of accuracy (3rd, 4th, ...)
- Low truncation error
- A-stable (or better) for DIRK component
- Optimized extension of stability region for ERK component
- Companion scheme of one lower order (on all terms) for adaptive timestepping is often useful.

Remarks

- Only 2, 5, and 14 constraints must be satisfied for 2nd, 3rd, and 4th order accuracy, respectively
- For a scheme of order $p$, truncation error is defined as the norm of the residual of accuracy constraints of order $p + 1$, denoted $A^{(p+1)}$ (Dormand & Price, 1980)
IMEXRKCB3c scheme: optimized ERK stability on negative Re axis

**DIRK component**

![Diagram of DIRK component]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>( a_{2,2} )</th>
<th>( a_{3,2} )</th>
<th>( a_{3,3} )</th>
<th>0</th>
<th>b_2</th>
<th>b_3</th>
<th>b_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>c_2</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c_3</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>b_2</td>
<td>b_3</td>
<td>b_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>b_2</td>
<td>b_3</td>
<td>b_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ERK component**

![Diagram of ERK component]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>( a_{2,1} )</th>
<th>( a_{3,2} )</th>
<th>( a_{4,3} )</th>
<th>0</th>
<th>b_2</th>
<th>b_3</th>
<th>b_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>c_2</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c_3</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>b_2</td>
<td>b_3</td>
<td>b_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>b_2</td>
<td>b_3</td>
<td>b_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Properties**

- Three-stage implicit, four-stage explicit
- Third-order overall, non-analytic coefficients
- \( L \)-stable DIRK component, extended \( R \)-stability of ERK component
IMEXRKCB3e scheme: optimized ERK stability on Im axis

**Properties**
- Three-stage implicit, four-stage explicit
- Third-order overall, simple analytic coefficients
- L-stable DIRK component, extended l-stability of ERK component
Results:

- We have developed eight new IMEXRK schemes with reduced storage requirements.
- One scheme is second order, six are third order, and one is fourth order.
- All schemes have at least strongly A-stable DIRK component, four are L-stable.
- Compared to full-storage IMEXRK schemes, truncation error, stability properties and implementation cost are similar.
- Storage is greatly reduced.

<table>
<thead>
<tr>
<th></th>
<th>$c_0$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_4$</th>
<th>$c_5$</th>
<th>$c_6$</th>
<th>$c_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>$egin{bmatrix} a_{2,1}^I &amp; a_{2,2}^I \ b_1 &amp; a_{3,2}^I &amp; a_{3,3}^I \ b_1 &amp; b_2 &amp; a_{4,3}^I &amp; a_{4,4}^I \vdots \vdots \vdots \vdots \b_1 &amp; b_2 &amp; \cdots &amp; b_{s-2} &amp; a_{s,s-1}^I &amp; a_{s,s}^I \b_1 &amp; b_2 &amp; \cdots &amp; b_{s-2} &amp; b_{s-1} &amp; b_s \end{bmatrix}$</td>
<td>$0$</td>
<td>$b_1$</td>
<td>$b_2$</td>
<td>$b_{s-2}$</td>
<td>$b_{s-1}$</td>
<td>$b_s$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$0$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_4$</th>
<th>$c_5$</th>
<th>$c_6$</th>
<th>$c_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>$egin{bmatrix} a_{2,1}^E &amp; 0 \ b_1 &amp; a_{3,2}^E &amp; 0 \ b_1 &amp; b_2 &amp; a_{4,3}^E &amp; 0 \vdots \vdots \vdots \vdots \b_1 &amp; b_2 &amp; \cdots &amp; b_{s-2} &amp; a_{s,s-1}^E &amp; 0 \b_1 &amp; b_2 &amp; \cdots &amp; b_{s-2} &amp; b_{s-1} &amp; b_s \end{bmatrix}$</td>
<td>$0$</td>
<td>$b_1$</td>
<td>$b_2$</td>
<td>$b_{s-2}$</td>
<td>$b_{s-1}$</td>
<td>$b_s$</td>
</tr>
</tbody>
</table>
### Results

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Order</th>
<th>Registers</th>
<th>Stages $(s^\text{IM}, s^\text{EX})$</th>
<th>Stability of DIRK part $[\sigma(z^\text{IM} \to \infty; z^\text{EX})]$</th>
<th>Stability of ERK part on negative real axis</th>
<th>Truncation error</th>
<th>Other properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMEXRKB2</td>
<td>second</td>
<td>[2R]</td>
<td>(2, 3)</td>
<td>$L$-stable [0]</td>
<td>$-5.81 \leq z^\text{EX} \leq 0$</td>
<td>$A^{(3)} = 0.114$</td>
<td>embedded, SSP ($c = 1.0$)</td>
</tr>
<tr>
<td>IMEXRKB3a</td>
<td>second</td>
<td>[2R]</td>
<td>(2, 3)</td>
<td>strongly $A$-stable $[-0.738]$</td>
<td>$-2.51 \leq z^\text{EX} \leq 0$</td>
<td>$A^{(4)} = 0.226$</td>
<td></td>
</tr>
<tr>
<td>IMEXRKB3b</td>
<td>third</td>
<td>[2R]</td>
<td>(3, 4)</td>
<td>strongly $A$-stable $[-0.732 - 0.366z^\text{EX}]$</td>
<td>$-2.21 \leq z^\text{EX} \leq 0$</td>
<td>$A^{(4)} = 0.186$</td>
<td>ESDIRK</td>
</tr>
<tr>
<td>IMEXRKB3c</td>
<td>third</td>
<td>[2R]</td>
<td>(3, 4)</td>
<td>$L$-stable [0]</td>
<td>$-6.00 \leq z^\text{EX} \leq 0$</td>
<td>$A^{(4)} = 0.113$</td>
<td>embedded, SSP ($c = 0.70$)</td>
</tr>
<tr>
<td>IMEXRKB3d</td>
<td>third</td>
<td>[2R]</td>
<td>(3, 4)</td>
<td>$L$-stable [0]</td>
<td>$-2.52 \leq z^\text{EX} \leq 0$</td>
<td>$A^{(4)} = 0.207$</td>
<td>embedded, SSP ($c = 0.77$)</td>
</tr>
<tr>
<td>IMEXRKB3e</td>
<td>third</td>
<td>[3R]</td>
<td>(4, 4)</td>
<td>$L$-stable [0]</td>
<td>$-2.79 \leq z^\text{EX} \leq 0$</td>
<td>$A^{(4)} = 0.0824$</td>
<td></td>
</tr>
<tr>
<td>IMEXRKB3f</td>
<td>third</td>
<td>[3R]</td>
<td>(4, 4)</td>
<td>$L$-stable [0]</td>
<td>$-6.00 \leq z^\text{EX} \leq 0$</td>
<td>$A^{(4)} = 0.107$</td>
<td>embedded, $SO_2$</td>
</tr>
<tr>
<td>IMEXRKB4</td>
<td>fourth</td>
<td>[3R]</td>
<td>(6, 6)</td>
<td>$L$-stable [0]</td>
<td>$-6.32 \leq z^\text{EX} \leq 0$</td>
<td>$A^{(5)} = 0.0157$</td>
<td></td>
</tr>
<tr>
<td>CN/RKW3</td>
<td>second</td>
<td>[2R]</td>
<td>(3, 3)</td>
<td>$A$-stable $[-1]$</td>
<td>$-2.51 \leq z^\text{EX} \leq 0$</td>
<td>$A^{(3)} = 0.0387$</td>
<td></td>
</tr>
</tbody>
</table>

Condition for stage-order 2 ($SO_2$) for DIRK part: $\sum_{j=1}^{s} a_{ij}^\text{IM} c_j = c_i^2/2\, , \, i = 2, 3, \ldots, s-1$

Note three schemes have ERK parts that are Strong Stability Preserving (SSP) for TVD discretization of hyperbolic problems.

Five schemes have embedded schemes that are 1 order lower on all terms - useful for adaptive timestepping.
New incremental schemes for the incompressible NSE

\[ \frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u} \cdot \nabla \mathbf{u} - \nabla p + \frac{1}{Re} \nabla^2 \mathbf{u} \]

\[ 0 = \nabla \cdot \mathbf{u} \]

Remarks
- Divergence-free constraint is imposed through pressure term
- In the simulation of turbulent flows, diffusive terms (linear) are in general stiff, while convective terms (nonlinear) are nonstiff
New incremental schemes for the incompressible NSE

After spatial discretization of the NSE, we have:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathcal{L} \mathbf{u} + \mathcal{N}(\mathbf{u}) - \mathcal{Gp}$$

$$0 = \mathcal{G}^T \mathbf{u}$$

where $\mathcal{L}$ is a stiff linear autonomous spatial operator, while $\mathcal{N}$ is nonstiff, bilinear and autonomous. $\mathcal{G}$ is the gradient operator.

The mixed implicit/explicit scheme CN/RKW3 is generally employed for time integration (Kim et al, 1987):

$$\mathbf{u}^{(1)} = \mathbf{u}_n + h \left[ \frac{4}{15} \left( \mathcal{L} \mathbf{u}^{(1)} + \mathcal{L} \mathbf{u}_n \right) + \frac{8}{15} \mathcal{N}(\mathbf{u}_n) \right]$$

$$\mathbf{u}^{(2)} = \mathbf{u}^{(1)} + h \left[ \frac{1}{15} \left( \mathcal{L} \mathbf{u}^{(2)} + \mathcal{L} \mathbf{u}^{(1)} \right) + \frac{5}{12} \mathcal{N}(\mathbf{u}^{(1)}) - \frac{17}{60} \mathcal{N}(\mathbf{u}_n) \right]$$

$$\mathbf{u}_{n+1} = \mathbf{u}^{(2)} + h \left[ \frac{1}{6} \left( \mathcal{L} \mathbf{u}_{n+1} + \mathcal{L} \mathbf{u}^{(2)} \right) + \frac{3}{4} \mathcal{N}(\mathbf{u}^{(2)}) - \frac{5}{12} \mathcal{N}(\mathbf{u}^{(1)}) \right]$$

with divergence-free constraint imposed at each substep (e.g. fractional step method, or velocity-vorticity formulation)

**Remarks**

- Only convective terms affect stability of simulation
- CFL limit provides a sufficient condition for stable integration:

$$h < \text{CFL}_{\text{lim}} \min\{\Delta x/|u|, \Delta y/|v|, \Delta z/|w|\}$$
Features of the ubiquitous CN/RKW3 incremental IMEXRK scheme

- Three-step, low-storage IMEXRK scheme
- Second-order implicit, third-order explicit
- A-stable implicit component
- \( \text{CFL}_{\text{lim}} \approx 1.73 \) for RKW3

Remarks

- *Spalart et al, 1991* presents a better scheme, with lower truncation error and strongly A-stable DIRK component, but still second order
- Refinement upon this scheme produced IMEXRKiCB2(3s), second order overall, but slightly improved truncation error

How to improve it?

- Add one (or two) additional substeps
- Allow extra storage

Remarks

- Improvement must justify extra storage/computation
IMEXRKiCB3(4s): a four-step, fully third-order scheme

\[ u^{(1)} = u_n + h \left( \alpha_1 \mathcal{L} u^{(1)} + \beta_1 \mathcal{L} u_n + \beta_1^E \mathcal{N}(u_n) \right) \]
\[ u^{(2)} = u^{(1)} + h \left( \alpha_2 \mathcal{L} u^{(2)} + \beta_2 \mathcal{L} u^{(1)} + \beta_2^E \mathcal{N}(u^{(1)}) + \gamma_2 \mathcal{N}(u_n) \right) \]
\[ u^{(3)} = u^{(2)} + h \left( \alpha_3 \mathcal{L} u^{(3)} + \beta_3 \mathcal{L} u^{(2)} + \beta_3^E \mathcal{N}(u^{(2)}) + \gamma_3 \mathcal{N}(u^{(1)}) \right) \]
\[ u_{n+1} = u^{(3)} + h \left( \alpha_4 \mathcal{L} u_{n+1} + \beta_4 \mathcal{L} u^{(3)} + \beta_4^E \mathcal{N}(u^{(3)}) + \gamma_4 \mathcal{N}(u^{(2)}) \right) \]

**Properties**

- Full third-order accuracy on time integration of NSE
- Strong A-stability of implicit component
- Higher CFL limit: \( \text{CFL}_{\text{lim}} \approx 2.78 \) (61% larger than CN/RKW3)
- Low truncation error \( A^{(4)} = 0.0592 \)
- One third more computation
IMEXRK_{iCB3}(4s+): a four-step, fully third-order scheme

\[ u^{(1)} = u_n + h \left( \alpha_1^I \mathcal{L} u^{(1)} + \beta_1^I \mathcal{L} u_n + \beta_1^E \mathcal{N}(u_n) \right) \]

\[ u^{(2)} = u^{(1)} + h \left( \alpha_2^I \mathcal{L} u^{(2)} + \beta_2^I \mathcal{L} u^{(1)} + \gamma_2^I \mathcal{L} u_n + \beta_2^E \mathcal{N}(u^{(1)}) + \gamma_2^E \mathcal{N}(u_n) \right) \]

\[ u^{(3)} = u^{(2)} + h \left( \alpha_3^I \mathcal{L} u^{(3)} + \beta_3^I \mathcal{L} u^{(2)} + \gamma_3^I \mathcal{L} u^{(1)} + \beta_3^E \mathcal{N}(u^{(2)}) + \gamma_3^E \mathcal{N}(u^{(1)}) \right) \]

\[ u_{n+1} = u^{(3)} + h \left( \alpha_4^I \mathcal{L} u_{n+1} + \beta_4^I \mathcal{L} u^{(3)} + \gamma_4^I \mathcal{L} u^{(2)} + \beta_4^E \mathcal{N}(u^{(3)}) + \gamma_4^E \mathcal{N}(u^{(2)}) \right) \]

**Properties**

- Full third-order accuracy on time integration of NSE
- \( L \)-stability of implicit component
- Higher CFL limit: \( \text{CFL}_{\text{lim}} \approx 2.82 \) (63% larger than CN/RKW3)
- Low truncation error \( A^{(4)} = 0.0698 \)
- Approximately one third more computation and extra storage required
IMEXRKiCB3(5s): a five-step, fully third-order scheme

\[ u^{(1)} = u_n + h \left( \alpha_1^I \mathcal{L} u^{(1)} + \beta_1^I \mathcal{L} u_n + \beta_1^E \mathcal{N}(u_n) \right) \]

\[ u^{(2)} = u^{(1)} + h \left( \alpha_2^I \mathcal{L} u^{(2)} + \beta_2^I \mathcal{L} u^{(1)} + \beta_2^E \mathcal{N}(u^{(1)}) + \gamma_2^E \mathcal{N}(u_n) \right) \]

\[ u^{(3)} = u^{(2)} + h \left( \alpha_3^I \mathcal{L} u^{(3)} + \beta_3^I \mathcal{L} u^{(2)} + \beta_3^E \mathcal{N}(u^{(2)}) + \gamma_3^E \mathcal{N}(u^{(1)}) \right) \]

\[ u^{(4)} = u^{(3)} + h \left( \alpha_4^I \mathcal{L} u^{(4)} + \beta_4^I \mathcal{L} u^{(3)} + \beta_4^E \mathcal{N}(u^{(3)}) + \gamma_4^E \mathcal{N}(u^{(2)}) \right) \]

\[ u_{n+1} = u^{(4)} + h \left( \alpha_5^I \mathcal{L} u_{n+1} + \beta_5^I \mathcal{L} u^{(4)} + \beta_5^E \mathcal{N}(u^{(4)}) + \gamma_5^E \mathcal{N}(u^{(3)}) \right) \]

Properties
- Full third-order accuracy on time integration of NSE
- L-stability of implicit component
- Highest CFL limit: CFL_{lim} \approx 3.31 (91\% larger than CN/RKW3)
- Lowest truncation error \( A^{(4)} = 0.0121 \)
- Two thirds more computation and no extra storage required
IMEXRKiCB3(5s): a five-step, fully third-order scheme

\[ u^{(1)} = u_n + h \left( \alpha_1^l \mathcal{L} u^{(1)} + \beta_1^l \mathcal{L} u_n + \beta_1^E \mathcal{N}(u_n) \right) \]

\[ u^{(2)} = u^{(1)} + h \left( \alpha_2^l \mathcal{L} u^{(2)} + \beta_2^l \mathcal{L} u^{(1)} + \beta_2^E \mathcal{N}(u^{(1)}) + \gamma_2^E \mathcal{N}(u_n) \right) \]

\[ u^{(3)} = u^{(2)} + h \left( \alpha_3^l \mathcal{L} u^{(3)} + \beta_3^l \mathcal{L} u^{(2)} + \beta_3^E \mathcal{N}(u^{(2)}) + \gamma_3^E \mathcal{N}(u^{(1)}) \right) \]

\[ u^{(4)} = u^{(3)} + h \left( \alpha_4^l \mathcal{L} u^{(4)} + \beta_4^l \mathcal{L} u^{(3)} + \beta_4^E \mathcal{N}(u^{(3)}) + \gamma_4^E \mathcal{N}(u^{(2)}) \right) \]

\[ u_{n+1} = u^{(4)} + h \left( \alpha_5^l \mathcal{L} u_{n+1} + \beta_5^l \mathcal{L} u^{(4)} + \beta_5^E \mathcal{N}(u^{(4)}) + \gamma_5^E \mathcal{N}(u^{(3)}) \right) \]

Properties

- Full third-order accuracy on time integration of NSE
- L-stability of implicit component
- Highest CFL limit: \( \text{CFL}_{\text{lim}} \approx 3.31 \) (91% larger than CN/RKW3)
- Lowest truncation error \( A^{(4)} = 0.0121 \)
- Two thirds more computation and no extra storage required
Comparison of ERK boundaries of IMEXRKiCB schemes
Simulations at $\text{Re}_\tau = 180$, same timestep $h=0.01$

Average energy spectrum at $y^+ = 50$

**Channel flow simulations performed over a time horizon $T = 200$**

With a grid $n_x \times n_y \times n_z = 128 \times 64 \times 128$:

- CN/RKW3 (Red)
- IMEXRKICB2(3S) (Orange)
- IMEXRKICB3(4S) (Blue)
- IMEXRKICB3(4S+) (Light Blue)
- IMEXRKICB3(5S) (Green), Reference solution shown in black
Simulations at $Re_T = 180$, same computation time

Average energy spectrum at $y^+ = 50$

Channel flow simulations performed over a time horizon $T = 200$
with a grid $n_x \times n_y \times n_z = 128 \times 64 \times 128$:
- CN/RKW3 (RED), IMEXRKICB2(3S) (ORANGE),
- IMEXRKICB3(4S) (BLUE), IMEXRKICB3(4S+) (LIGHT BLUE),
- IMEXRKICB3(5S) (GREEN), REFERENCE SOLUTION SHOWN IN BLACK
Results: Incremental IMEXRK schemes for the incompressible NSE

- We have developed four new hybrid schemes for the simulation of turbulence
- One scheme is second order, and three are third order
- All schemes have at least strongly $A$-stable implicit component, and two are $L$-stable
- Compared to the standard approach, accuracy and stability properties are greatly improved
- Extra flops justified by performance increase
How to find such amazing schemes? Δ-DOGS*

Joint work with Pooriya Beyhaghi, Ryan Alimo, Muhan Zhao

*Delaunay-based Derivative-free Optimization via Global Surrogates

Key take-away: When solving complex problems with complex constraints, the engineer needs to take ownership of the optimization process.
OUTLINE

1. Key idea: response surface method with search function $s(x) = p(x) - K e(x)$ combining
   (a) polyharmonic spline interpolation of datapoints $f(x_i)$ as the “surrogate model” $p(x)$, and
   (b) piecewise quadratic over Delaunay triangulation of datapoints as “uncertainty function” $e(x)$

2. Variants designed for (a) linear constraints, (b) convex constraints, (c) nonconvex constraints

3. Variants adapt $K$ automatically to quickly reach target function value $f_0$

4. Accelerated variants introduce:
   (a) Successively-refined Cartesian grid or sphere packing (lattice) to coordinate search
   (b) Smoother spline interpolation facilitating dimension reduction during intermediate iterations
   (c) Blending of derivative-free global search with derivative-based local refinement

5. $\alpha$-DOGS efficiently handles approximate function evaluations obtained via statistical sampling

6. S-DOGS performs safe optimization, for function evaluations that are delicate experiments

1. Key idea: response surface method with search function $s(x) = p(x) - K e(x)$ combining
   (a) polyharmonic spline interpolation of datapoints $f(x_i)$ as the “surrogate model” $p(x)$, and
   (b) piecewise quadratic over Delaunay triangulation of datapoints as “uncertainty function” $e(x)$

Features:
- Keeps function evaluations far apart until convergence is approached.
- Efficiently synthesize global exploration and local refinement.
- Provably globally convergent under the appropriate assumptions.
Iteration $k = 4$, $f(x)$, $p^k(x)$, $-e^k(x)$, $s^k(x)$.

Response surface method: 1D illustration (13/13)
If $S$ is a set of points in $\mathbb{R}^n$, a **triangulation** of $S$ is a set of simplices whose vertices are elements of $S$ such that the following conditions hold:

- Every point in $S$ is a vertex of at least one simplex in the triangulation. The union of all of these simplices fully covers the convex hull of $S$.
- The intersection of two different simplices in the triangulation is either empty or a $k$-simplex such that $k = 0, 1, \ldots, n - 1$.

**How to extend idea to multiple dimensions?**  key idea: triangulation
A Delaunay triangulation is a triangulation such that the intersection of the open circumsphere around each simplex with $S$ is empty.

The number of simplices grows rapidly as dimension grows.

Which triangulation? Delaunay!
Consider $S$ as a set of feasible points which includes the vertices of $\Omega$, and $\Delta$ as a Delaunay triangulation for $S$. Then, for each simplex $\Delta_i \in \Delta$, the local uncertainty function is defined as:

$$e_i(x) = R_i^2 - \|x - Z_i\|^2.$$ where $R_i$ and $Z_i$ are the circumradius and circumcenter of $\Delta_i$. The global uncertainty function $e(x)$ is a piecewise function defined as follows:

$$e(x) = e_i(x), \forall x \in \Delta_i.$$ Given a triangulation, we can define a convenient “uncertainty function” $e(x)$.
The uncertainty function $e(x)$, built on the framework of a Delaunay triangulation, characterizes the “distance” of any point in parameter space to the nearest function evaluations. It is:

- Piecewise quadratic, continuous, Lipschitz
- Non-negative everywhere, equal to zero at datapoints

The fact that the underlying triangulation is Delaunay is essential in our proofs of convergence.

This approach allows us to use any approach that seems best suited for interpolation (polyharmonic splines is generally much better than Kriging in our experience).
1. Key idea: **response surface method** with search function \( s(x) = p(x) - K e(x) \) combining
   (a) polyharmonic spline interpolation of datapoints \( f(x_i) \) as the “surrogate model” \( p(x) \), and
   (b) piecewise quadratic over Delaunay triangulation of datapoints as “uncertainty function” \( e(x) \)

2. Variants designed for (a) linear constraints, (b) convex constraints, (c) nonconvex constraints

3. Variants adapt \( K \) automatically to quickly reach target function value \( f_0 \)

4. Accelerated variants introduce:
   (a) Successively-refined Cartesian grid or sphere packing (lattice) to coordinate search
   (b) Smoother spline interpolation facilitating dimension reduction during intermediate iterations
   (c) Blending of derivative-free global search with derivative-based local refinement

5. \( \alpha \)-DOGS efficiently handles approximate function evaluations obtained via statistical sampling

6. S-DOGS performs safe optimization, for function evaluations that are delicate experiments

2. Variants for (c) nonconvex (even, disconnected!) constraints

\[
\begin{align*}
\text{minimize } & f(x) \quad \text{with} \quad x \in \Omega := L_s \cap L_c \subseteq \mathbb{R}^n \\
L_s &= \{x|c_\ell(x) \leq 0, \text{ for } \ell = 1, \cdots, m\}, \quad L_c = \{x|a \leq x \leq b\}.
\end{align*}
\]

Assumptions:
- The objective and constraints functions \( f(x), c_\ell(x) \) are defined everywhere in \( L_c \)
- \( f(x), c_\ell(x) \) are twice differentiable, but possibly nonconvex in \( L_c \)
- The constraint functions \( c_\ell(x) \) are taken to be computationally expensive
- No prior knowledge of the objective and constraint functions is assumed
- Low dimension, expensive function evaluations, no access to derivatives
Rapid global convergence when exact target value $f_0$ is known

3. Variants adapt K automatically to quickly reach target function value $f_0$
3. Variants adapt $K$ automatically to quickly reach target function value $f_0$. 

Rapidly finds $x$ s.t. $f(x) \leq f_0$ when target value $f_0$ greater than global min.

\[\begin{align*}
  f(x_1, x_2) &= x_2, \\
  x_2 - h(x_1) &= 0, \\
  0 \leq x_1, x_2 \leq 1. \\
\end{align*}\]

\[\begin{align*}
  f(x_1, x_2) &= x_1^2 + x_2^2, \\
  c_1(x_1, x_2) &\leq 0, \\
  0 \leq x_1, x_2 \leq 1. \\
\end{align*}\]

\[\begin{align*}
  f(x_1, x_2) &= x_1^2 + x_2^2, \\
  0 \leq c_2(x_1, x_2) &\leq 1, \\
  -1.25 \leq x_1, x_2 \leq 1.25. \\
\end{align*}\]
3. Variants adapt $K$ automatically to quickly reach target function value $f_0$

\[
\begin{align*}
&f(x_1, x_2) = x_2, \\
&x_2 - h(x_1) = 0, \\
&0 \leq x_1, x_2 \leq 1.
\end{align*}
\]

\[
\begin{align*}
&f(x_1, x_2) = x_1^2 + x_2^2, \\
&c_1(x_1, x_2) \leq 0 \\
&0 \leq x_1, x_2 \leq 1.
\end{align*}
\]

\[
\begin{align*}
&f(x_1, x_2) = x_1^2 + x_2^2, \\
&0 \leq c_2(x_1, x_2) \leq 1 \\
&-1.25 \leq x_1, x_2 \leq 1.25.
\end{align*}
\]

Continued global exploration when target value $f_0$ less than global min
1. Key idea: response surface method with search function $s(x) = p(x) - K e(x)$ combining 
   (a) polyharmonic spline interpolation of datapoints $f(x_i)$ as the “surrogate model” $p(x)$, and 
   (b) piecewise quadratic over Delaunay triangulation of datapoints as “uncertainty function” $e(x)$

2. Variants designed for (a) linear constraints, (b) convex constraints, (c) nonconvex constraints

3. Variants adapt $K$ automatically to quickly reach target function value $f_0$

4. Accelerated variants introduce:
   (a) Successively-refined Cartesian grid or sphere packing (lattice) to coordinate search
   (b) Smoother spline interpolation facilitating dimension reduction during intermediate iterations
   (c) Blending of derivative-free global search with derivative-based local refinement

5. $\alpha$-DOGS efficiently handles approximate function evaluations obtained via statistical sampling

6. S-DOGS performs safe optimization, for function evaluations that are delicate experiments

Practical applications (thus far…)

★ **Design of hydrofoils** and multi-rotors (hexacopter w/ direct sideforce)
★ **Path optimization** - tuning paths found via RRT (Rapidly-exploring Random Tree)
★ **Design of low-storage implicit/explicit Runge–Kutta (IMEXRK) schemes**
★ **Sea State 4 wave simulator for tethered USV/UAV ISR system development**
Questions?
bewley@ucsd.edu
robotics.ucsd.edu
+1.858.997.8369

RenaissanceRobotics.com