REPORT: DISCONTINUOUS GALERKIN SPARSE GRIDS METHODS FOR VLASOV POISSON EQUATIONS

Abstract. TBD

Key words. Discontinuous Galerkin, sparse grids, Vlasov-Poisson equation, Runge Kutta methods.

1. Introduction. In this project, we shall consider the following Vlasov-Poisson (VP) equations:

$$f_t + v\frac{\partial f}{\partial x} + E(x,t)\frac{\partial f}{\partial v} = 0, \tag{1.1}$$

$$-\frac{\partial^2 \phi}{\partial x^2} = \rho - 1,\ E(x,t) = -\frac{\partial \phi}{\partial x}, \tag{1.2}$$

where $\rho(x,t) = \int_v f(x,v,t)dv$ denotes the electron density.

Periodic boundary condition is imposed in x-space and v-space, and we assume $v \in [-V_{\max}, V_{\max}]$ and $x \in [0, L_{\max}]$.

The flow chart of the proposed DG-SG Vlasov-Poisson Solver is described as Algorithm 1.

Algorithm 1 DG Sparse Grids Methods for Vlasov-Poisson Equations

Set parameters:: LevX, LevV, Lmax, Vmax, Deg, DimX, DimV, **TEND**, dt;
Set initial conditions:: $\mathbf{F}^0$
HASH=`HashTable`(LevV, LevX, DimV, DimX) ▷ Algorithm 2
Con2D=`Connect2D`(Lev,**HASH**) ▷ Connectivity for 2D
$[\mathbb{VMASSV}, \mathbb{GRADV}, \mathbb{GRADX}, \mathbb{DELTAX}]$=`MatrixCoeffTI`(LevV, LevX, Deg, Lmax, Vmax) ▷ Algorithm 3
if DimX >1 **then**
  $\mathbb{A}_{\text{Poisson}}$=`GlobalDELTAX`(DimX, $\mathbb{DELTAX}$) ▷ For the Case DimX> 1
else
  $\mathbb{A}_{\text{Poisson}}$=$\mathbb{DELTAX}$
end if
Set $\ell = 0$, time=0
loop
  $\mathbf{E}$ = `PoissonSolve`(LevX, Deg, Lmax, $\mathbf{F}$, $\mathbb{A}_{\text{Poisson}}$) ▷ Solve $\mathbf{E}$ from Poisson Equation
  $\mathbb{EMASSSX}$ = `MatrixCoeffTD`(LevX, Deg, Lmax, $\mathbf{E}$) ▷ Algorithm 4
  $\mathbb{R}$ = `GlobalVlasov`($\mathbb{VMASSV}, \mathbb{GRADV}, \mathbb{GRADX}, \mathbb{EMASSSX}$, **HASH**) ▷ Algorithm 5
  $\mathbf{F}^{\ell+1}$=`TimeAdvance`($\mathbb{R}$, $\mathbf{F}^{\ell}$, dt) ▷ Algorithm 7
  $\ell = \ell + 1$, time=time+dt
  if time$\geq$ **TEND** **then return**
  end if
end loop

In the following, we assume the computational domain $\mathbf{\Omega} := \mathbf{\Omega}_v \times \mathbf{\Omega}_x$ is triangulated into a union of rectangles $\mathcal{T}_h := \cup T$. Denote $T = T_v \times T_x$, ∂T_v and ∂T_x as the boundaries for T_v and T_x. Furthermore, we assume the finite element space $f_h \in V_h := V_{h,v} \times V_{h,x}$, which are piecewise polynomial spaces with the equal degree Deg. We will also use the average and jump notations $\{\!\{ \cdot \}\!\}$ and $[\![\cdot]\!]$, which are defined as follows:

$$\{\!\{w\}\!\}_i = \frac{1}{2}(w_i^+ + w_i^-) \text{ and } [\![w]\!]_i = w_i^+ - w_i^-,$$

where $w_i^+ = \lim_{x \to x_i^+} w_i$ and $w_i^- = \lim_{x \to x_i^-} w_i$.

2. Mesh Generation and Finite Element Space. In this section, we introduce the grids and the associated finite element space. First, denote the one dimensional hierarchical decomposition of piecewise polynomial space on the interval $\Omega := [0,1]$. The grids are defined as the nested grids, where the n-th level grid Ω_n consists of 2^n uniform cells $I_j^n = (2^{-n}j, 2^{-n}(j+1)], j = 0, \ldots, 2^n - 1$, for any $n \geq 0$. Let

$$V_k^n := \{v : v \in P^k(I_j^n), \forall j = 0, \ldots, 2^n - 1\}$$

be the usual piecewise polynomials of degree at most k on the n-th level grid Ω_n. Then, we have

$$V_k^0 \subset V_k^1 \subset V_k^2 \subset \cdots .$$

We can now define the multiwavelet subspace $W_k^n, n = 1, 2, \cdots$ as the orthogonal complement of V_k^{n-1} in V_k^n with respect to the L^2 inner product on $[0, 1]$, i.e.,

$$V_k^{n-1} \bigoplus W_k^n = V_k^n,\ W_k^n \pm V_k^{n-1}.$$

Here, we let $W_k^0 := V_k^0$, which is standard piecewise polynomial space of degree k on $[0, 1]$. The dimension of W_k^n is $2^{n-1}(k + 1)$ when $n \geq 1$, and $k + 1$ when $n = 0$. In summary, we have found a hierarchical representation of the standard piecewise polynomial space V_k^n on Ω_n as $V_k^n = \bigoplus_{0 \leq j \leq n} W_k^j$.

For a multi-index $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_{\mathrm{Dim}}) \in \mathbb{N}_0^{\mathrm{Dim}}$, where $\mathbb{N}_0^{\mathrm{Dim}}$ denotes the set of nonnegative integers, the l^1 and l^∞ norms are defined as

$$|\boldsymbol{\alpha}|_1 := \sum_{m=1}^{\mathrm{Dim}} \alpha_m,\ |\boldsymbol{\alpha}|_\infty := \max_{1 \leq m \leq \mathrm{Dim}} \alpha_m.$$

We denote $\mathbf{l} = (l_1, l_2, \cdots, l_{\mathrm{Dim}}) \in \mathbb{N}_0^{\mathrm{Dim}}$ the mesh level in a multivariate sense. We define the tensor-product mesh grid $\Omega_{\mathbf{l}} = \Omega_{l_1} \bigotimes \Omega_{l_2} \cdots \bigotimes \Omega_{l_{\mathrm{Dim}}}$ and the corresponding mesh size $h_{\mathbf{l}} = (h_{l_1}, h_{l_2}, \cdots, h_{l_{\mathrm{Dim}}})$. Based on the grid $\Omega_{\mathbf{l}}$, we denote by $I_{\mathbf{j}}^{\mathbf{l}} = \{\mathbf{x} : x_m \in (h_m j_m, h_m(j_m + 1)), m = 1, 2, \cdots, \mathrm{Dim}\}$ and elementary cell, and

$$\mathbf{V}_k^{\mathbf{l}} := \{\mathbf{v} : \mathbf{v}(\mathbf{x}) \in [P^k(I_{\mathbf{j}}^{\mathbf{l}})]^{\mathrm{Dim}}, 0 \leq \mathbf{j} \leq 2^{\mathbf{l}} - 1\}$$

the tensor-product piecewise polynomial space, where $P^k(I_{\mathbf{j}}^{\mathbf{l}})$ denotes the collection of polynomials of degree up to k in each dimension on cell $I_{\mathbf{j}}^{\mathbf{l}}$. If we use equal mesh refinement of size $h_N = 2^{-N}$ in each coordinate direction, the grid and space will be denoted by Ω_N and $\mathbf{V}_k^N$, respectively.

Based on a tensor-product construction, the multi-dimensional increment space can be defined as

$$\mathbf{W}_k^{\mathbf{l}} = [W_{k,x_1}^{l_1} \times W_{k,x_2}^{l_2} \cdots \times W_{k,x_{\mathrm{Dim}}}^{l_{\mathrm{Dim}}}].$$

Therefore, space $\mathbf{V}_k^{\mathbf{l}}$ can be represented by

$$\mathbf{V}_k^{\mathbf{l}} = \bigoplus_{0 \leq j_1 \leq l_1, 0 \leq j_2 \leq l_2, \cdots, 0 \leq j_{\mathrm{Dim}} \leq l_{\mathrm{Dim}}} \mathbf{W}_k^{\mathbf{j}}.$$

The standard tensor-product polynomial space $\mathbf{V}_k^N$ and sparse finite element approximation space $\hat{\mathbf{V}}_k^N$ on Ω_N are defined by

$$\text{Full-Grid } \mathbf{V}_k^N = \bigoplus_{|\mathbf{l}|_\infty \leq N, \mathbf{l} \in \mathbb{N}_0^{\mathrm{Dim}}} \mathbf{W}_k^{\mathbf{l}}, \tag{2.1}$$

$$\text{Sparse-Grid } \hat{\mathbf{V}}_k^N = \bigoplus_{|\mathbf{l}|_1 \leq N, \mathbf{l} \in \mathbb{N}_0^{\mathrm{Dim}}} \mathbf{W}_k^{\mathbf{l}} \tag{2.2}$$

The sparse finite element space $\hat{\mathbf{V}}_k^N$ is a subset of $\mathbf{V}_k^N$.

REMARK 2.1. *If the computational domain is not unit, the grids can be generated by the same fashion through scaling techniques to shift the uniform cells I_j^n.*

REMARK 2.2. *In the computation of VP (1x1v) equation, we denote:*

- *LevX=LevV=Lev=N*
- *DimX=1=DimV and Dim=DimX+DimV=2*
- *Deg=k*

2.1. Vlasov Solver. In this subsection, we shall introduce the numerical scheme for solving Vlasov equation. First, let us assume the electrostatic field E is given either by initial condition or computed through Poisson solver. The variational form for (1.1) is as follows: find $f_h \in V_h$, such that for all $z \in V_h$, the following equation is satisfied:

$$(\frac{\partial f_h}{\partial t}, z)_T + (v\frac{\partial f_h}{\partial x}, z)_T + (E\frac{\partial f_h}{\partial v}, z)_T = 0.$$

Algorithm 2 Generate Hash Table for Sparse Grids

Input: LevV, LevX, DimV, DimX
Output: HASH

function HashTable(LevV,LevX,DimV,DimX)
  Set Dim=DimV+DimX, Lev=LevV and Count=0 ▷ (Note: LevV=LevX for simplicity)
  for n_1=0; $n_1 \leq$Lev; n_1++ **do**
    for p_1=0; $p_1 \leq$`MaxCellNumber`(n_1); p_1++ **do**
      $\cdots$
      for n_{Dim}=0; $n_{\mathrm{Dim}} \leq$Lev-$\sum_i^{\mathrm{Dim}-1} n_i$; $n_{\mathrm{Dim}}++$ **do**
        for p_{Dim}=0; $p_{\mathrm{Dim}} \leq$`MaxCellNumber`(n_{Dim}); $p_{\mathrm{Dim}}++$ **do**
          $\cdots$
          Count=Count+1
          Key=$[n_1, \cdots, n_{\mathrm{Dim}}, p_1, \cdots, p_{\mathrm{Dim}}]$, Value=Count ▷ Key and Value in Hash Table
          HASH:: Key$\rightarrow$ Value
        end for
      end for
    end for
  end for
  Set **HASH**.Dim=Dim, and **HASH**.Lev=Lev
  return HASH
end function

function MaxCellNumber(n) ▷ Compute the Maximum Cell Number for Level n
  if $n == 0$ **then**
    $z = 0$
  else
    $z = 2^{n-1} - 1$
  end if
  return z
end function

Now let $\mathcal{F}_h(x),\ \mathcal{M}(x) \in V_{h,x}$ and $\mathcal{G}_h(v),\ \mathcal{N}(v) \in V_{h,v}$. Since we approximate f_h by $\mathcal{F}_h(x)\mathcal{G}_h(v)$ and denote $z = \mathcal{M}(x)\mathcal{N}(v)$, we can rewrite the above variational form as:

$$\frac{(\partial \mathcal{F}_h \mathcal{G}_h, \mathcal{M}\mathcal{N})_T}{\partial t} + \int_{T_x} \frac{\partial \mathcal{F}_h}{\partial x} \mathcal{M} dx \int_{T_v} v \mathcal{G}_h \mathcal{N} dv + \int_{T_x} E \mathcal{F}_h \mathcal{M} dx \int_{T_v} \frac{\partial \mathcal{G}_h}{\partial v} \mathcal{N} dv = 0.$$

Now for notation simplicity, we can replace the basis function for x as $\Phi_i(x)$ $(i = 1, \cdots, \mathrm{DOF}_x)$, v as $\Psi_j(v)$ $(j = 1, \cdots, \mathrm{DOF}_v)$, and then denote the operators as following

$$[\mathbb{GRADX}]_{i,j} := \int_{T_x} \frac{\partial \Phi_j}{\partial x} \Phi_i dx := -\int_{T_x} \Phi_j \frac{\partial \Phi_i}{\partial x} dx + \widehat{\Phi_j} \Phi_i|_{\partial T_x},\ i,j = 1, \cdots, \mathrm{DOF}_x, \tag{2.3}$$

$$[\mathbb{EMASSX}]_{i,j} := \int_{T_x} E \Phi_j \Phi_i dx,\ i,j = 1, \cdots, \mathrm{DOF}_x, \tag{2.4}$$

$$[\mathbb{VMASSV}]_{i,j} := \int_{T_v} v \Psi_j \Psi_i dv,\ i,j = 1, \cdots, \mathrm{DOF}_v, \tag{2.5}$$

$$[\mathbb{GRADV}]_{i,j} := \int_{T_v} \frac{\partial \Psi_j}{\partial v} \Psi_i dv := -\int_{T_v} \Psi_j \frac{\partial \Psi_i}{\partial x} + \widehat{\Psi_j} \Psi_i|_{\partial T_x},\ i,j = 1, \cdots, \mathrm{DOF}_v. \tag{2.6}$$

Here we will chose the central flux

$$\widehat{w} = \{\!\{w\}\!\} \tag{2.7}$$

in the computation of $\mathbb{GRADX}$, $\mathbb{EMASSX}$, $\mathbb{VMASSV}$, and $\mathbb{GRADV}$. It is noted here, since the proper choice of basis for $\Phi_i(x)$ and $\Psi_j(v)$, the mass matrices corresponding to $\int_{T_x} \Phi_i \Phi_j dx$ and $\int_{T_v} \Psi_i \Phi_j dv$ are indentity.

Remark 2.3. *The central flux may not ensure L^2-stability[2], which is described as follows,*

$$\|f_h(t)\|_{0,\mathcal{T}_h} \leq \|f_h(0)\|_{0,\mathcal{T}_h},\ \forall t \in [0, T]. \tag{2.8}$$

The central flux for the Vlasov equations causes lack of numerical dissipation. When filamentation occurs, the numerical schemes will produce spurious oscillation, jeopardizing the quality of the solution.

Another choice for numerical flux is the global Lax-Friedrichs flux, which is a monotone numerical flux to ensure the L^2-stability of the scheme, defined as follows,

$$\widehat{\boldsymbol{\alpha} u_h} = \boldsymbol{\alpha}\{\!\{u_h\}\!\} + \frac{a}{2}[\![u_h]\!], \tag{2.9}$$

where $a = \max_{\mathbf{x}} |\boldsymbol{\alpha} \cdot \mathbf{n}|$.

If the global Lax-Friedrichs flux (2.9) is utilized in the computation, the (2.8) will be ensured through the calculation. But we will produce two more time-independent coefficient matrices.

By LF flux, there are more matrices needed efficient multiplication.

2.1.1. Numerical Experiments. In this subsection, we shall present the numerical experiments to validate the proposed numerical scheme. In this example, we show results for the free-streaming case with $E(x,t) = 0$ in (1.1). The initial condition is

$$f(x, v, 0) = f_0(v)(1 + A\cos(\omega x))$$

with $A = \omega = 0.5$ and $f_0(v) = (2\pi)^{-1/2}\exp(-v^2/2)$. The analytic solution corresponding to this initial condition is given by

$$f(x, v, t) = f_0(v)[1 + A\cos(\omega(x - vt))] \tag{2.10}$$

and the density can be computed as

$$\rho(x, t) = 1 + \exp(-\omega^2 t^2/2)A\cos(\omega x). \tag{2.11}$$

TABLE 2.1
Example of Hyperbolic Solver

Level	$\|f_h - f\|_\infty$	Rate	$\|f_h - f\|_2$	Rate	$\|f_h - f\|_\infty$	Rate	$\|f_h - f\|_2$	Rate
$k = 1$	Central Flux				Lax-Friedrichs			
3	1.8295E-01		2.3744E+00		1.5636E-01		2.2884E+00	
4	6.2432E-02	1.55	1.3943E+00	0.77	5.2550E-02	1.57	1.3262E+00	0.79
5	2.0568E-02	1.60	8.3473E-01	0.74	1.4095E-02	1.90	6.8591E-01	0.95
6	6.8449E-03	1.59	5.7559E-01	0.54	3.5861E-03	1.97	3.4590E-01	0.99
$k = 2$	Central Flux				Lax-Friedrichs			
3	5.2615E-02		7.4861E-01		5.4804E-02		7.5269E-01	
4	7.8737E-03	2.74	2.1597E-01	1.79	7.9981E-03	2.78	2.1667E-01	1.80
5	1.0170E-03	2.95	5.6348E-02	1.94	1.0406E-03	2.94	5.6628E-02	1.94
6	1.2975E-04	2.97	1.4257E-02	1.98	1.4438E-04	2.85	1.4390E-02	1.98
$k = 3$	Central Flux				Lax-Friedrichs			
3	1.3117E-02		2.0062E-01		1.3054E-02		2.0062E-01	
4	1.0709E-03	3.61	2.9197E-02	2.78	1.0620E-03	3.62	2.9199E-02	2.78
5	7.1289E-05	3.91	3.7903E-03	2.95	7.1115E-05	3.90	3.7903E-03	2.95
6	4.5577E-06	3.96	4.8006E-04	2.98	4.5361E-06	3.97	4.7946E-04	2.98

2.2. Poisson Solver. Here we solve Poisson equation (1.2) on the full-grid by local discontinuous Galerkin (LDG) methods. The right-hand side function ρ is calculated by $\rho = \int_v f dv$. With the computed function ρ, the LDG method is shown as following:

$$\begin{aligned} E + \frac{\partial \phi}{\partial x} &= 0 \\ \frac{\partial E}{\partial x} &= \rho - 1 \end{aligned}$$

and thus with w and z as piecewise polynomial, the variational form is: find $E_h, \phi_h \in V_{h,x}$ such that for all $w, z \in V_{h,x}$,

$$(E_h, w)_T - (\phi_h, \frac{\partial w}{\partial x})_T + \langle \widehat{\phi_h}, w \rangle_{\partial T} = 0, \tag{2.12}$$

$$-(E_h, \frac{\partial z}{\partial x})_T + \langle \widehat{E_h}, z \rangle_{\partial T} = (\rho - 1, z)_T, \tag{2.13}$$

where we shall choose the numerical flux as follows,

$$\widehat{\phi_h} = \{\!\{\phi_h\}\!\} + \frac{1}{2}[\![\phi_h]\!] = \phi_h^+,$$
$$\widehat{E_h} = \{\!\{E_h\}\!\} - \frac{1}{2}[\![E_h]\!] = E_h^-.$$

For simplicity, we will denote the matrix constructed by the poisson solver as $\mathbb{DELTAX}$. The linear system to (2.12)-(2.13) is as follows

$$\mathbb{DELTAX} \begin{pmatrix} \mathbf{E}_h \\ \boldsymbol{\phi}_h \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{G} \end{pmatrix}. \tag{2.14}$$

Here $\mathbf{E}_h = [E_h]_i$ and $\boldsymbol{\phi}_h = [\phi_h]_i$ $(i = 1, \cdots, \mathrm{DOF}_x)$ are the coefficients we need to determine as the Poisson solver. Beside, we also have $[\mathbf{G}]_i = \sum_{T_x} (\rho - 1, \phi_i)_{T_x}$ and vector $\mathbf{G} = [\mathbf{G}]_i$.

Can be optimized: such as the computation of ρ or the procedures of full-grid Poisson solver.

2.2.1. Numerical Experiments. In this subsection, we shall present one numerical example to validate the LDG numerical scheme and report the convergence results accordingly.

Let $\Omega_x = (0, 1)$ and exact solution $\phi = \sin(2\pi x)$. The error profiles and convergence test can be found in Table 2.2. If the degree for polynomial is denoted by k, we expect $\mathcal{O}(h^{k+1})$ as the optimal rate in convergence. However, we observe $\mathcal{O}(h^{k+0.5})$ for even k of the error of E.

Algorithm 3 Construct Time Independent Coefficient Matries

Input: LevV, LevX, Deg, Lmax, Vmax
Output: Matrices: $\mathbb{VMASSV}, \mathbb{GRADV}, \mathbb{GRADX}, \mathbb{DELTAX}$

```
function MatrixCoeffTI(LevV, LevX, Deg, Lmax, Vmax)
    Compute VMASSV by (2.5)
    Compute GRADV by (2.6)
    Compute GRADX by (2.3)
    Compute DELTAX by (2.12)-(2.13)
    return VMASSV, GRADV, GRADX, DELTAX
end function
```

Algorithm 4 Construct Time Dependent Coefficient Matries

Input: LevX, Deg, Lmax, $\mathbf{E}$
Output: Matrix: $\mathbb{EMASSX}$

```
function MatrixCoeffTD(LevX, Deg, Lmax, E)
    Compute EMASSX by (2.4)
    return EMASSX
end function
```

3. Temporal Discretizations. We use total variation diminishing (TVD) high-order Runge-Kutta methods to solve the method of lines ODE resulting from the semi-discrete DG scheme, $\frac{d}{dt} G_h = R(G_h)$. Such time stepping methods are convex combinations of the Euler forward time discretization. The commonly used third-order TVD Runge-Kutta method is given by

$$\begin{aligned} G_h^{(1)} &= G_h^n + \Delta t R(G_h^n), \\ G_h^{(2)} &= \frac{3}{4} G_h^n + \frac{1}{4} G_h^{(1)} + \frac{1}{4} \Delta t R(G_h^{(1)}), \\ G_h^{n+1} &= \frac{1}{3} G_h^n + \frac{2}{3} G_h^{(2)} + \frac{2}{3} \Delta t R(G_h^{(2)}), \end{aligned} \tag{3.1}$$

TABLE 2.2
Example of Poisson Solver

Level	$\|E_h - E\|_0$	Rate	$\|E_h - E\|_\infty$	Rate	$\|\phi_h - \phi\|_0$	Rate	$\|\phi_h - \phi\|_\infty$	Rate
				$k = 0$				
2	3.7686E+00		3.0472E+00		4.5720E-01		3.1831E-01	
3	2.7307E+00	0.46	2.0000E+00	0.61	2.2864E-01	1.00	1.3185E-01	1.27
4	1.8034E+00	0.60	1.5307E+00	0.39	1.1367E-01	1.01	4.8460E-02	1.44
5	1.2029E+00	0.58	1.1036E+00	0.47	5.6723E-02	1.00	1.7299E-02	1.49
6	8.1986E-01	0.55	7.8414E-01	0.49	2.8346E-02	1.00	6.1310E-03	1.50
7	5.6788E-01	0.53	5.5514E-01	0.50	1.4171E-02	1.00	2.1689E-03	1.50
8	3.9719E-01	0.52	3.9266E-01	0.50	7.0852E-03	1.00	7.6695E-04	1.50
9	2.7928E-01	0.51	2.7767E-01	0.50	3.5426E-03	1.00	2.7117E-04	1.50
				$k = 1$				
2	1.4292E+00		9.4653E-01		2.5149E-01		1.9340E-01	
3	3.0340E-01	2.24	1.9376E-01	2.29	4.4226E-02	2.51	2.1423E-02	3.17
4	6.0420E-02	2.33	3.7559E-02	2.37	7.5163E-03	2.56	2.0904E-03	3.36
5	1.2278E-02	2.30	6.9006E-03	2.44	1.4876E-03	2.34	3.2823E-04	2.67
6	2.6207E-03	2.23	1.2413E-03	2.47	3.3988E-04	2.13	5.7980E-05	2.50
7	5.8971E-04	2.15	2.2126E-04	2.49	8.2748E-05	2.04	1.0248E-05	2.50
8	1.3844E-04	2.09	3.9271E-05	2.49	2.0543E-05	2.01	1.8114E-06	2.50
9	3.3424E-05	2.05	6.9560E-06	2.50	5.1265E-06	2.00	3.2022E-07	2.50
				$k = 2$				
2	3.5197E-01		2.8743E-01		2.5241E-02		2.0744E-02	
3	7.7942E-02	2.17	6.1936E-02	2.21	2.8003E-03	3.17	2.2349E-03	3.21
4	1.4514E-02	2.42	1.1476E-02	2.43	2.8083E-04	3.32	2.0705E-04	3.43
5	2.5974E-03	2.48	2.0523E-03	2.48	2.8728E-05	3.29	1.8514E-05	3.48
6	4.6044E-04	2.50	3.6386E-04	2.50	3.0947E-06	3.21	1.6412E-06	3.50
7	8.1440E-05	2.50	6.4368E-05	2.50	3.5135E-07	3.14	1.4517E-07	3.50
8	1.4398E-05	2.50	1.1381E-05	2.50	4.1515E-08	3.08	1.2833E-08	3.50
9	2.5451E-06	2.50	2.0119E-06	2.50	5.0324E-09	3.04	1.1344E-09	3.50
				$k = 3$				
2	3.8105E-02		2.7219E-02		1.4245E-03		8.7848E-04	
3	1.8672E-03	4.35	1.3120E-03	4.37	6.5319E-05	4.45	2.7798E-05	4.98
4	8.6059E-05	4.44	5.9240E-05	4.47	3.8196E-06	4.10	1.3099E-06	4.41
5	3.9658E-06	4.44	2.6321E-06	4.49	2.3668E-07	4.01	5.8806E-08	4.48
6	1.8737E-07	4.40	1.1648E-07	4.50	1.4779E-08	4.00	2.6091E-09	4.49
7	9.2438E-09	4.34	5.1495E-09	4.50	9.2364E-10	4.00	1.1542E-10	4.50
8	4.8238E-10	4.26	2.2760E-10	4.50	5.7728E-11	4.00	5.1021E-12	4.50
9	2.6672E-11	4.18	1.0058E-11	4.50	3.6079E-12	4.00	2.2549E-13	4.50

where G_h^n represents a numerical approximation of the solution at discrete time t_n. A detailed description of the TVD Runge-Kutta method can be found in [1].

4. Numerical Experiment.

4.1. Example 1: Bump-on-tail.

Let

$$f(0, x, v) = f_0(v)(1 + A\cos(\omega x)),\ x \in [0, L_{\max}],\ v \in [-V_{\max}, V_{\max}], \tag{4.1}$$

where $A = 0.04$, $\omega = 0.3$, $L_{\max} = 20\pi/3$, $V_{\max} = 13$, and

$$f_0(v) = n_p \exp(\frac{-v^2}{2}) + n_b \exp(-\frac{|v-u|^2}{2v_t^2}),$$

where $n_p = 9/(10\sqrt{10\pi})$, $n_b = 2/(10\sqrt{10\pi})$, u=4.5, $v_t = 0.5$.

- Comparing results of Central Flux and Lax-Friedrichs Flux.

Algorithm 5 Construct Global Matrix for Vlasov Equation

Input: Matrices $\mathbb{A}_1, \mathbb{A}_2, \mathbb{B}_1, \mathbb{B}_2$; Hash Table: **HASH** ▷ Matrices $\mathbb{A}_1, \mathbb{B}_1$ w.r.t 1-component; Matrices $\mathbb{A}_2, \mathbb{B}_2$ w.r.t 2-component;
Output: Matrix $\mathbb{R}$

function GlobalVlasov($\mathbb{A}_1, \mathbb{A}_2, \mathbb{B}_1, \mathbb{B}_2$, **HASH**)
  $\mathbb{R}_1$=`GlobalMatrixSG`($\mathbb{A}_1, \mathbb{A}_2$, **HASH**) ▷ Construct Global Matrix for Sparse Grids as Algorithm 6
  $\mathbb{R}_2$=`GlobalMatrixSG`($\mathbb{B}_1, \mathbb{B}_2$, **HASH**)
  $\mathbb{R} = \mathbb{R}_1 + \mathbb{R}_2$
  return $\mathbb{R}$
end function

Algorithm 6 Construct Global Matrix for Sparse Grids

Major change is HERE!

Input: Matrices: $\mathbb{A}_1, \mathbb{A}_2$; Hash Table: **HASH**; Connectivity: **Con2D**; Degree: **Deg**
Output: Matrix: $\mathbb{R}$

function GlobalMatrixSG($\mathbb{A}_1, \mathbb{A}_2$, **HASH**, **Con2D**, **Deg**)
  Set Lev=**HASH**.Lev and Dim=**HASH**.Dim
  for i=1; i≤**HASH**.dof; i++ **do**
    j=Con2D{i} ▷ Find the nonzero parts from connectivity
    HASHInv: i→(IndexI1,IndexI2)
    for jj=1; j≤size(j); jj++ **do**
      HASHInv: j(jj)→(IndexJ1,IndexJ2)
      $\mathbb{A}_{\text{tmp}} = \mathbb{A}$(IndexI1,IndexJ1) and $\mathbb{B}_{\text{tmp}} = \mathbb{B}$(IndexI2,IndexJ2)
      $[\mathbb{R}]_{i,j(jj)}$=$[\mathbb{R}]_{i,j(jj)}$+$\mathbb{A}_{\text{tmp}} \otimes \mathbb{B}_{\text{tmp}}$ ▷ $\otimes$ means Kronecker product
    end for
  end for
  return $\mathbb{R}$
end function

4.2. Example 2. Two-stream instability I. Let

$$f(0,x,v) = f_{\text{TSI}}(v)(1 + A\cos(kx)),\ x \in [0, L_{\max}],\ v \in [-V_{\max}, V_{\max}], \tag{4.2}$$

where $A = 0.05$, $k = 0.5$, $L = 4\pi$, $V_{\max} = 2\pi$, and

$$f_{\text{TSI}}(v) = \frac{1}{\sqrt{2\pi}}\left(v^2 \exp\left(-\frac{v^2}{2}\right)\right).$$

4.3. Example 3. Two-stream instability II. Let

$$f(0,x,v) = f_{\text{TSII}}(v)(1 + A\cos(kx)),\ x \in [0, L_{\max}],\ v \in [-V_{\max}, V_{\max}], \tag{4.3}$$

where $A = 0.05$, $k = 2/13$, $L = 13\pi$, $V_{\max} = 5$, and

$$f_{\text{TSII}}(v) = \frac{1}{2v_t\sqrt{2\pi}}\left(\exp\left(-\frac{|u+v|^2}{2v_t^2}\right) + \exp\left(-\frac{|u-v|^2}{2v_t^2}\right)\right),$$

where $u = 0.99$, $v_t = 0.3$.

- conservation Law

REFERENCES

[1] C.W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. J. Comput. Phys., 77 (1988):439-471.

[2] Y. Cheng, A. Christlieb, X. Zhong, Energy-conserving discontinuous Galerkin methods for the Vlasov-Ampere system, Journal of Computational Physics 256 (2014): 630-655

Algorithm 7 Time Advance for Vlasov-Poisson Equation

Input: Matrices: $\mathbb{R}$; Vector: $\mathbf{F}$; Time Step: dt
Output: Vector: $\mathbf{F}$

function TimeAdvance($\mathbb{R}, \mathbf{F}$,dt)
    $\mathbf{F}$=`RungeKutta3`($\mathbb{R}, \mathbf{F}$,dt)
    return F
end function

function RungeKutta3($\mathbb{R}, \mathbf{G}$,dt)
    $\mathbf{G}^{(1)}=\mathbf{G}+\text{dt}\ \mathbb{R}\ \mathbf{G}$
    $\mathbf{G}^{(2)}=\frac{3}{4}\mathbf{G}+\frac{1}{4}\mathbf{G}^{(1)}+\frac{1}{4}\text{dt}\ \mathbb{R}\ \mathbf{G}^{(1)}$
    $\mathbf{G}\ \ =\frac{1}{3}\mathbf{G}+\frac{2}{3}\mathbf{G}^{(2)}+\frac{2}{3}\text{dt}\ \mathbb{R}\ \mathbf{G}^{(2)}$
    return G
end function

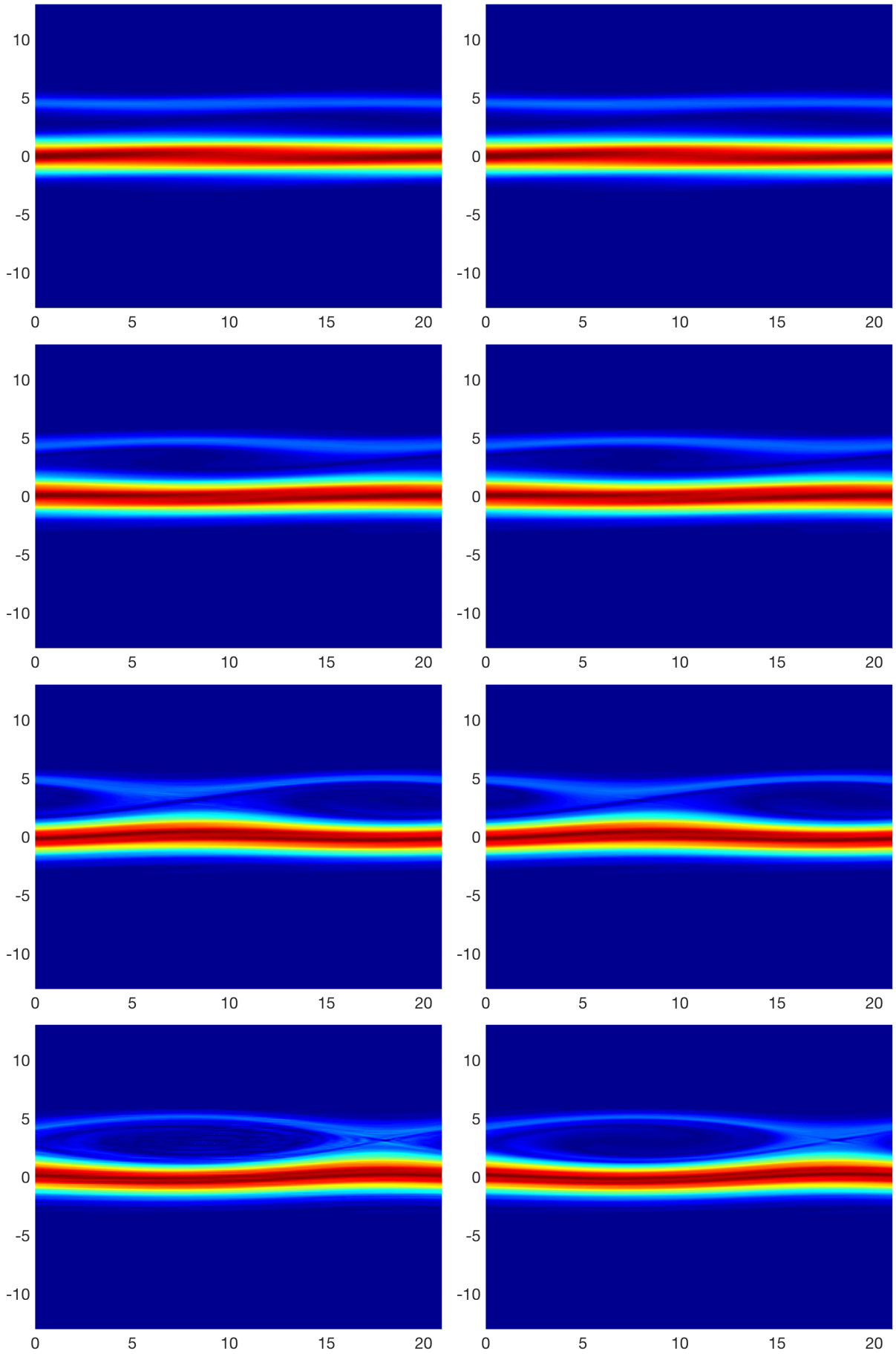

FIG. 4.1. *Full Grid Solutions for Example 4.3 with LevV=6 and LevX=6, k=3 of* $T = 10, 20, 30, 40$ *for central flux (2.7) (Left) and Lax-Friedrichs Flux (2.9) (Right).*

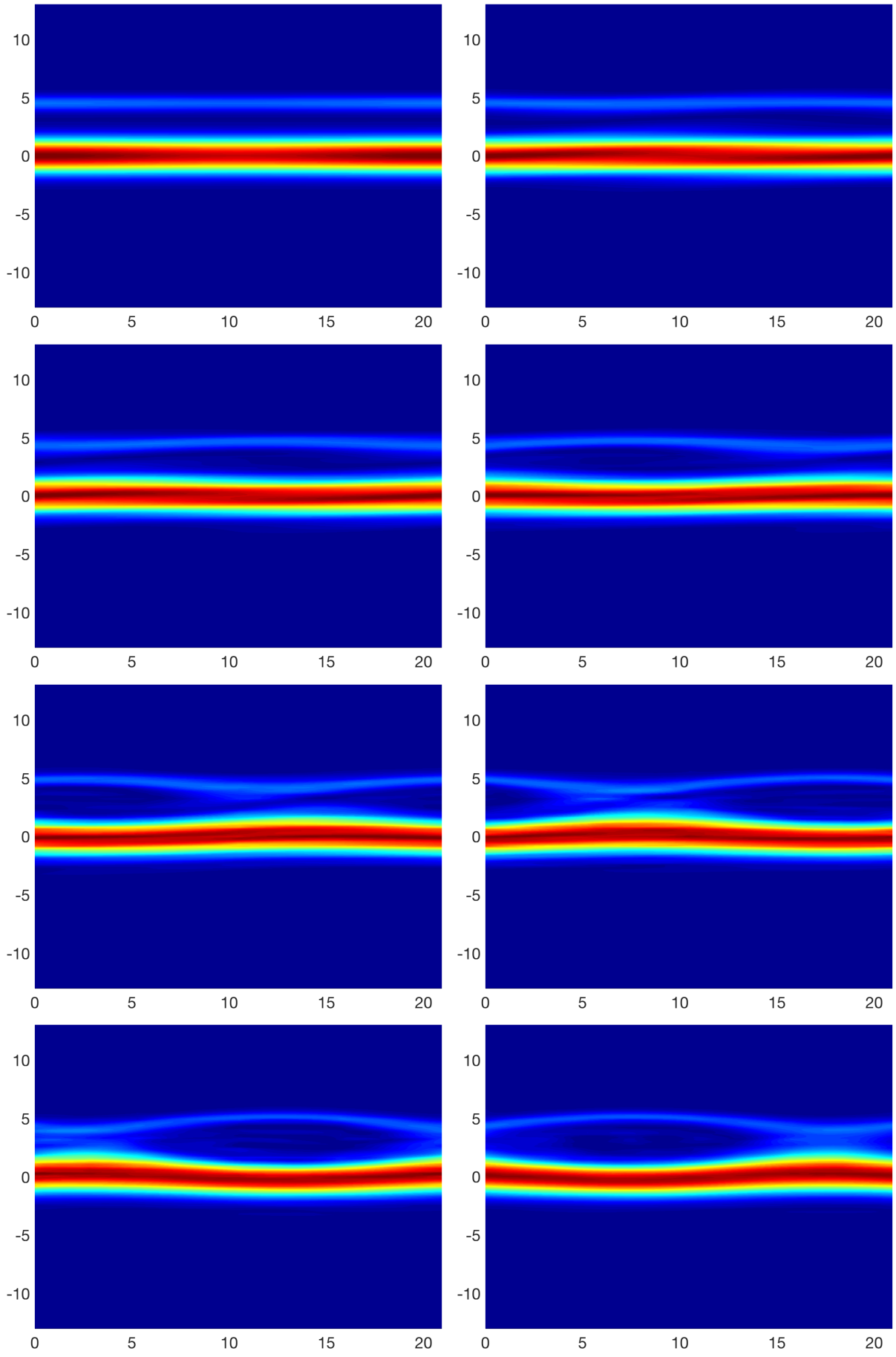

FIG. 4.2. *Sparse Grid Solutions for Example 4.1 with LevV=6 and LevX=6, k=3 of* $T = 0, 10, 15, 20, 25, 30, 35, 40$.

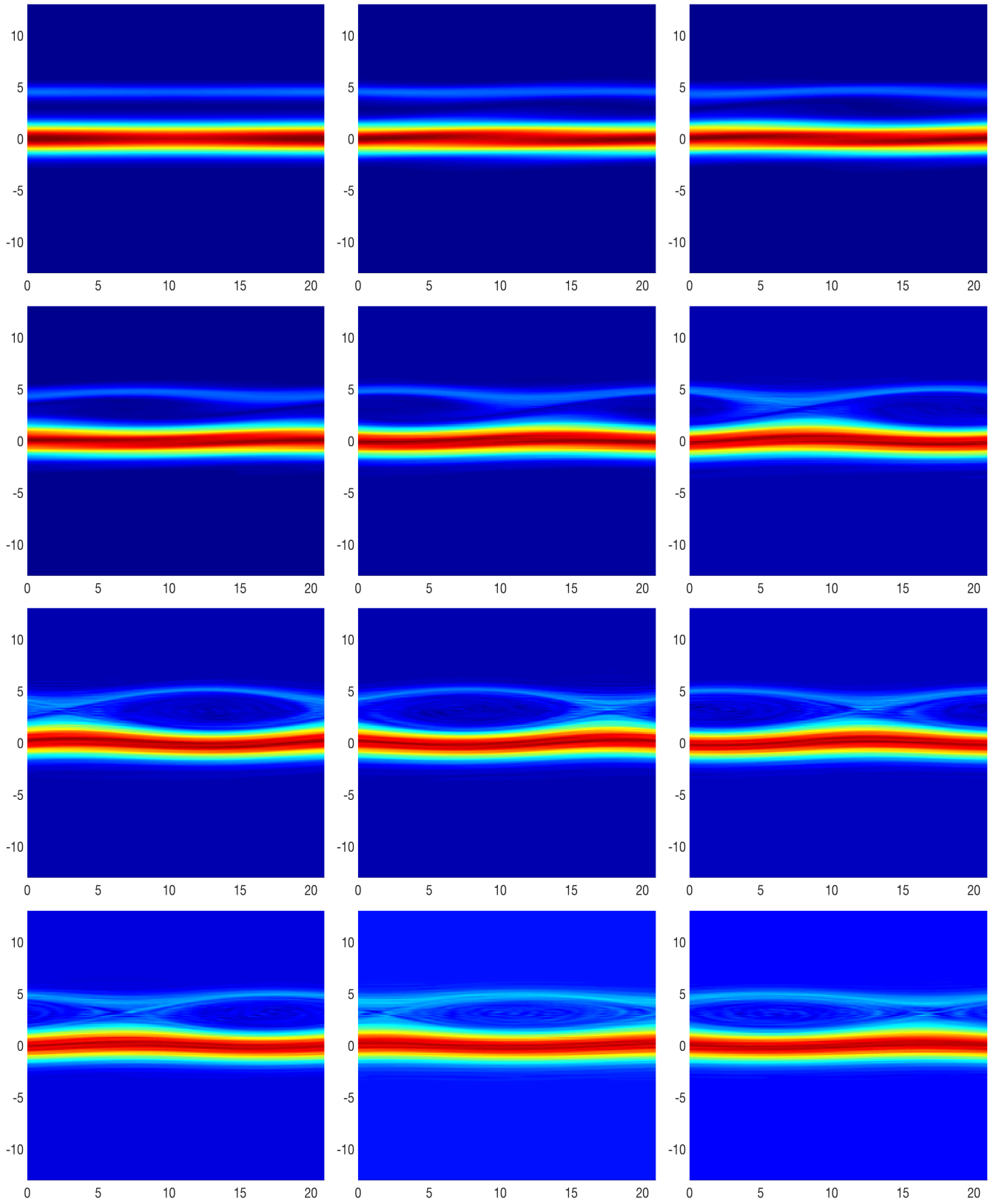

FIG. 4.3. *Full Grid Solutions for Example 4.1 with LevV=6 and LevX=6, k=2 of* $T = 0, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60$.

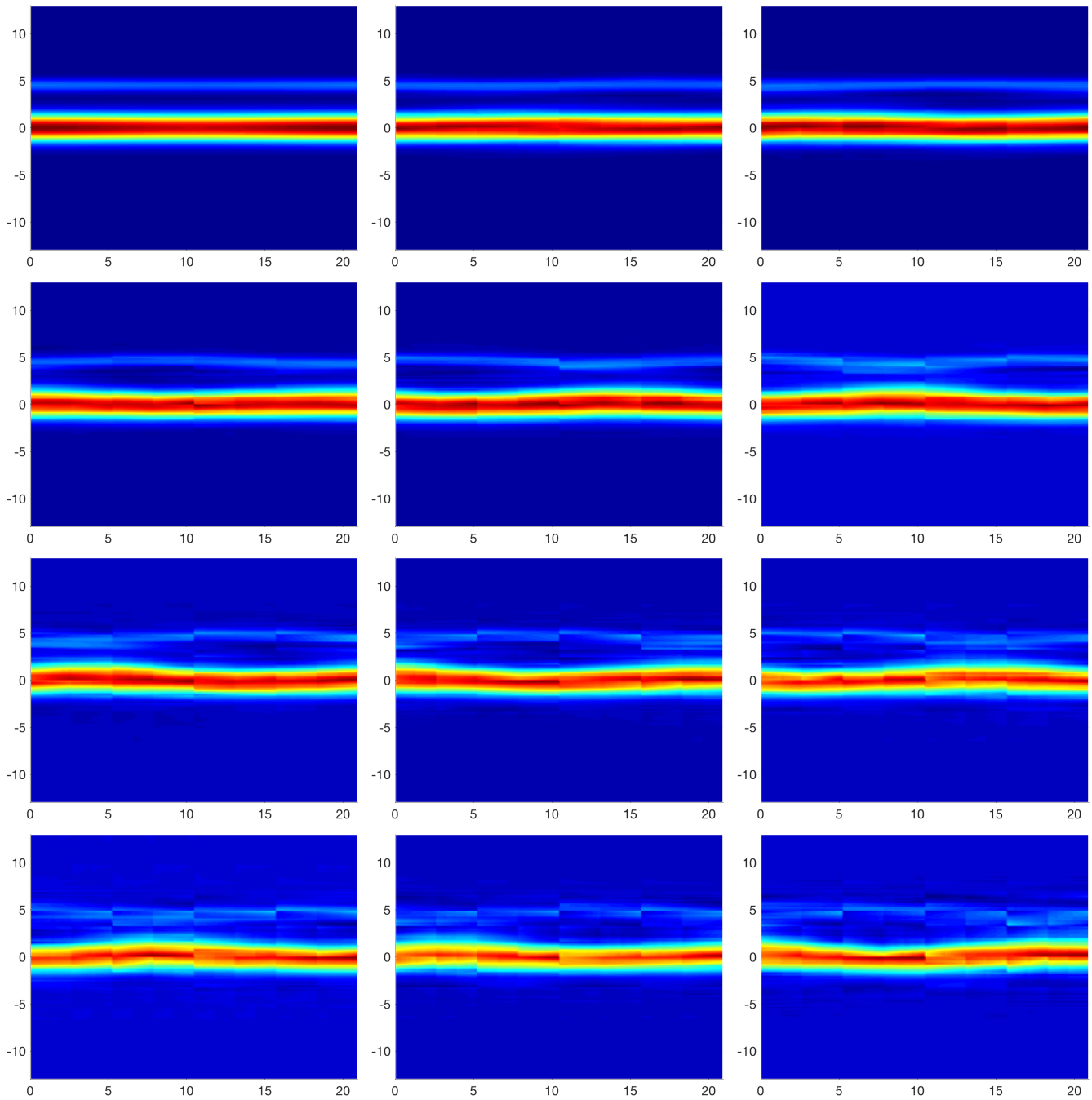

FIG. 4.4. *Sparse Grid Solutions for Example 4.1 with LevV=6 and LevX=6, k=1 of* $T = 0, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60$.

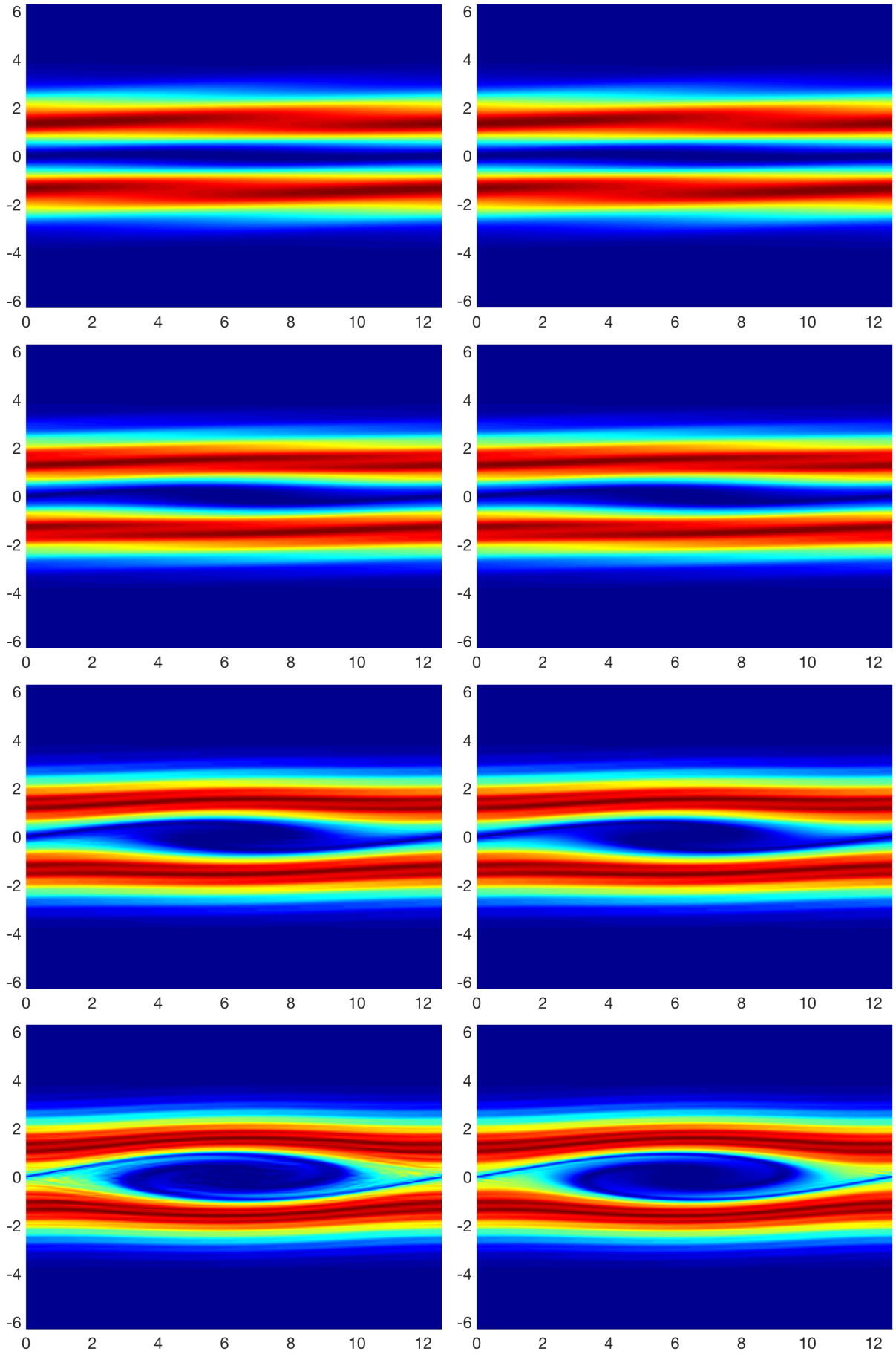

FIG. 4.5. *Full Grid Solutions for Example 4.3 with LevV=6 and LevX=6, k=3 of* $T = 10, 20, 30, 40$ *for central flux (2.7) (Left) and Lax-Friedrichs Flux (2.9) (Right).*

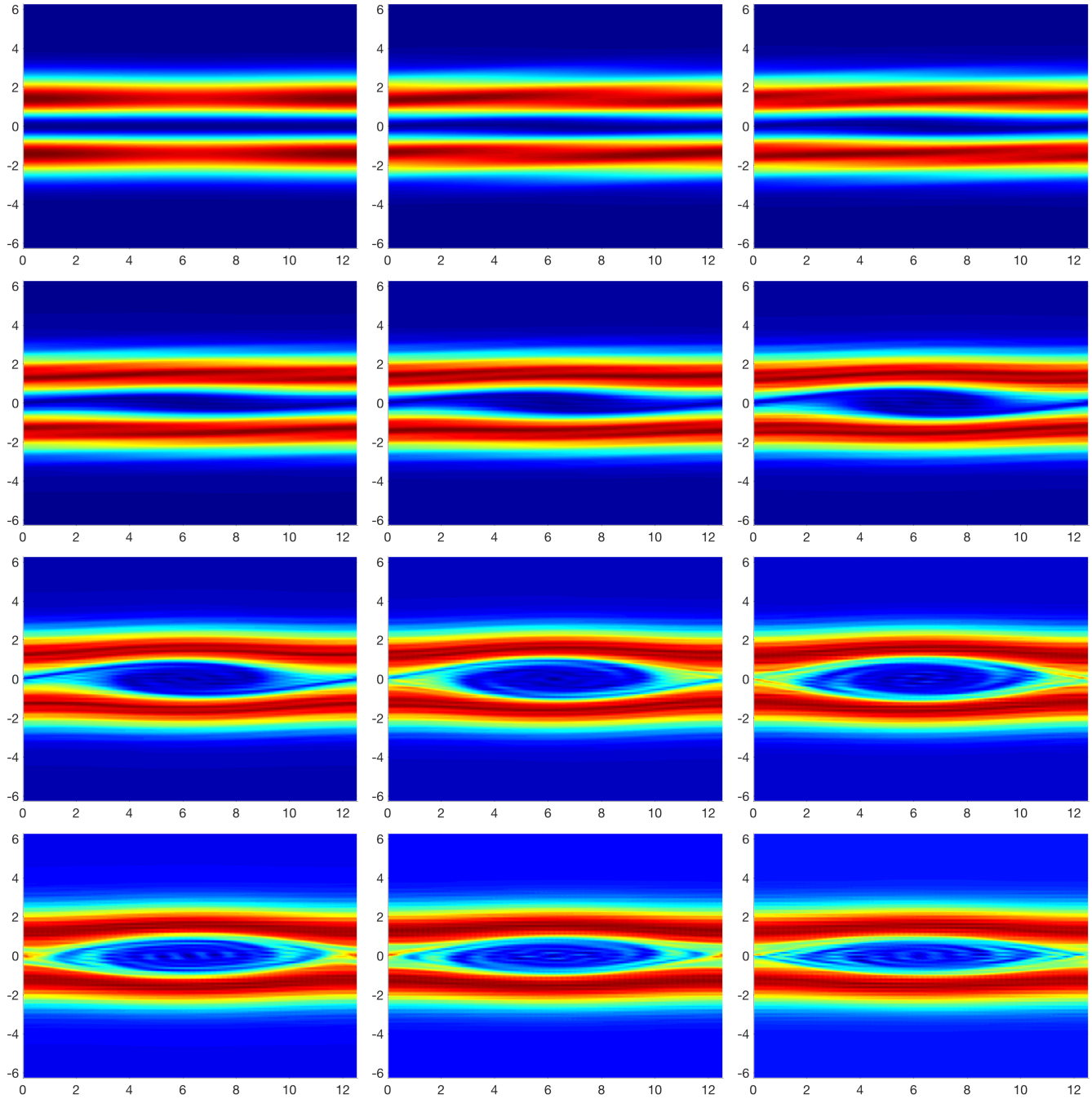

FIG. 4.6. *Full Grid Solutions for Example 4.2 with LevV=6 and LevX=6, k=1 of* $T = 0, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60$.

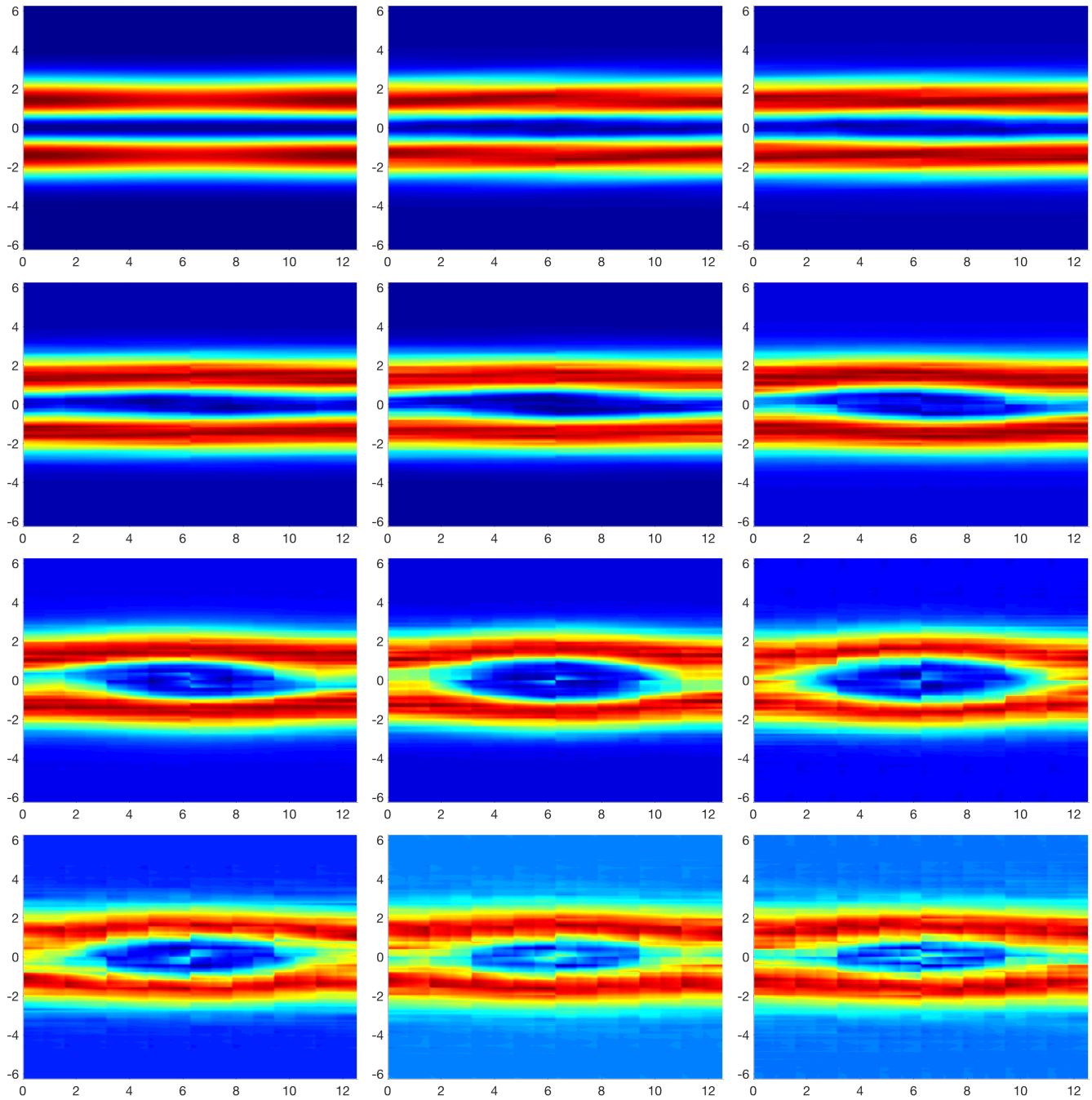

FIG. 4.7. *Sparse Grid Solutions for Example 4.2 with LevV=6 and LevX=6, k=1 of* $T = 0, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60$.

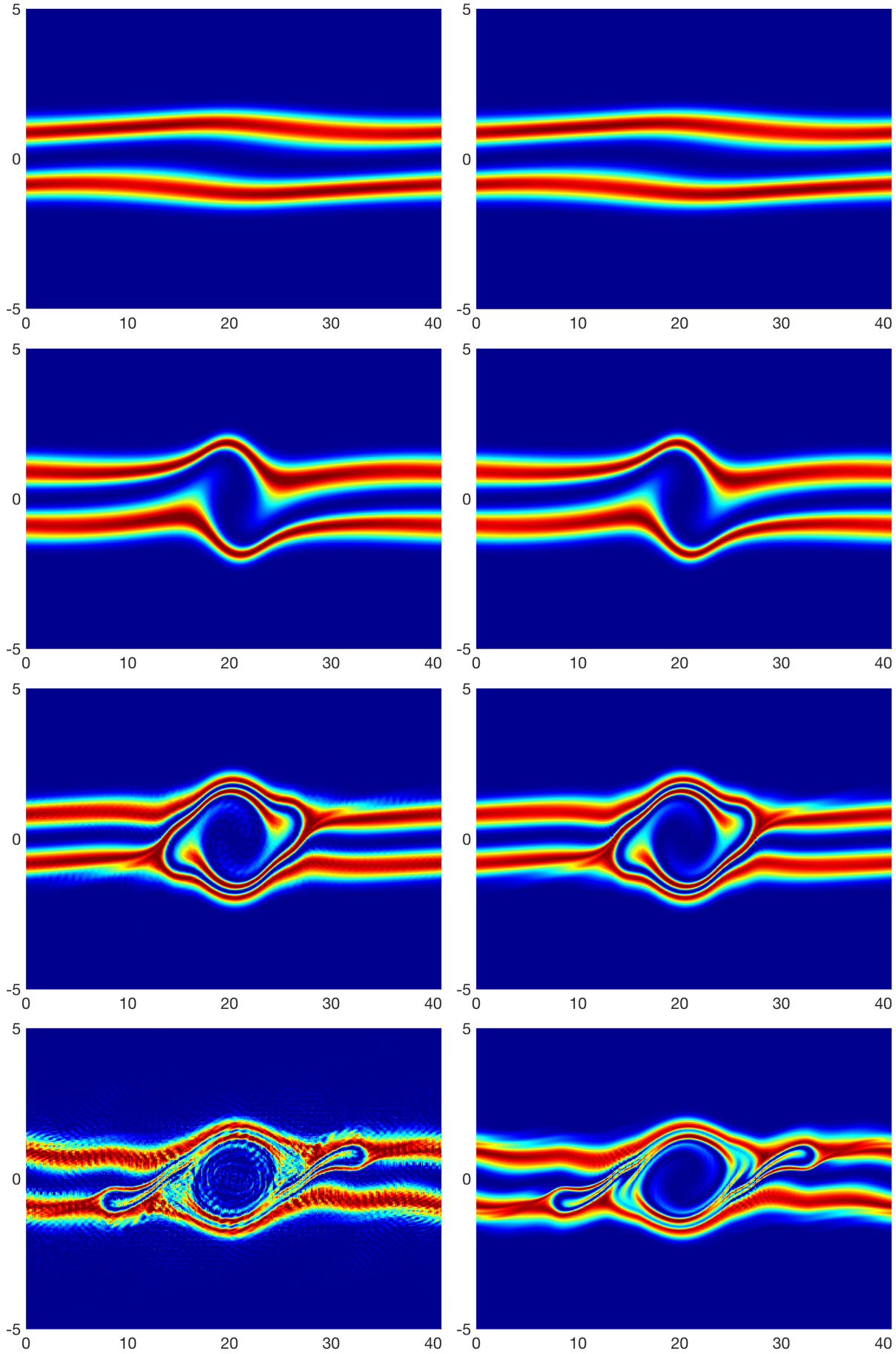

FIG. 4.8. *Full Grid Solutions for Example 4.3 with LevV=6 and LevX=6, k=3 of* $T = 10, 20, 30, 40$ *for central flux (2.7) (Left) and Lax-Friedrichs Flux (2.9) (Right).*

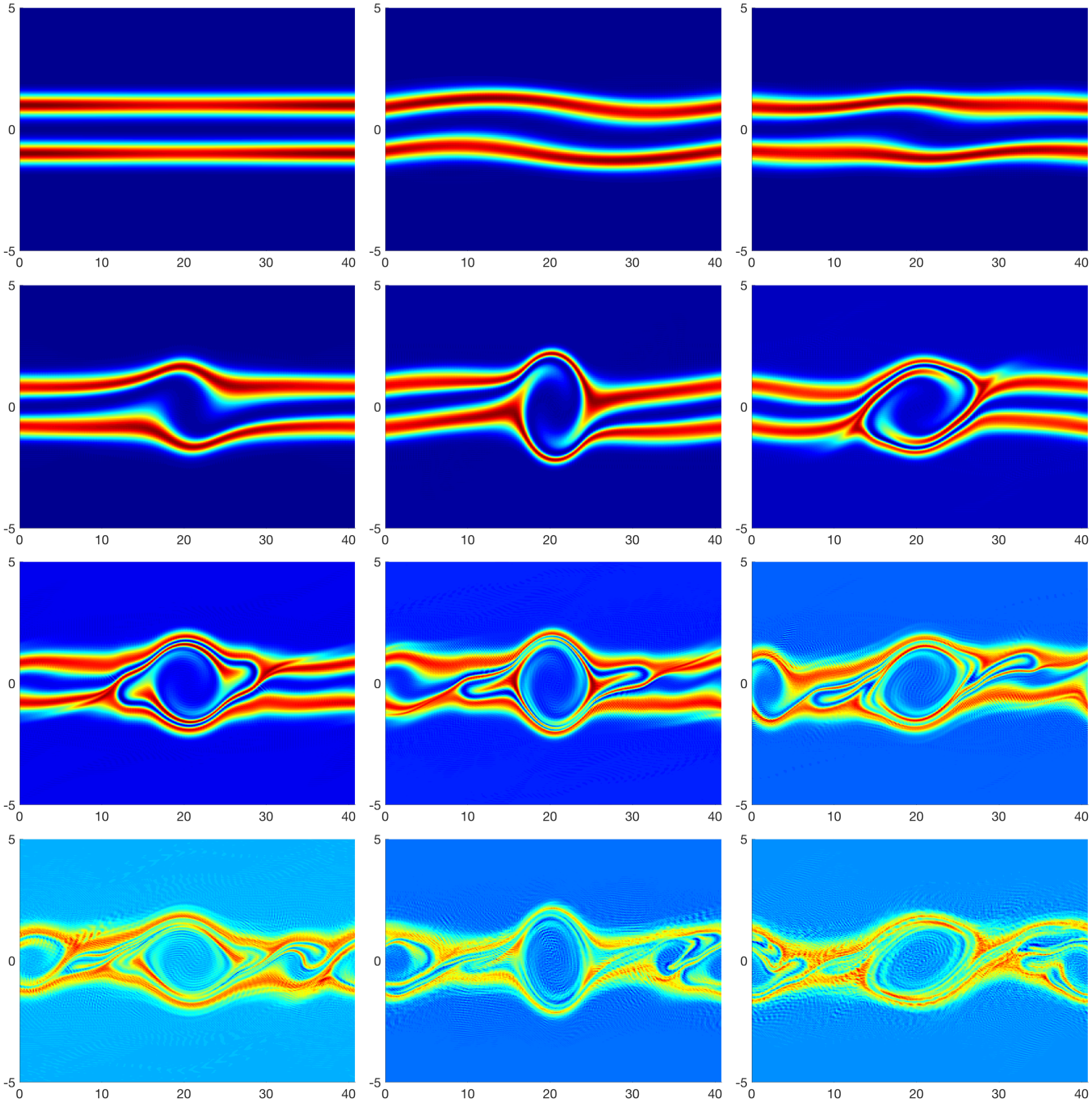

FIG. 4.9. *Full Grid Solutions for Example 4.3 with LevV=7 and LevX=7, k=1 of* $T = 0, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60$.

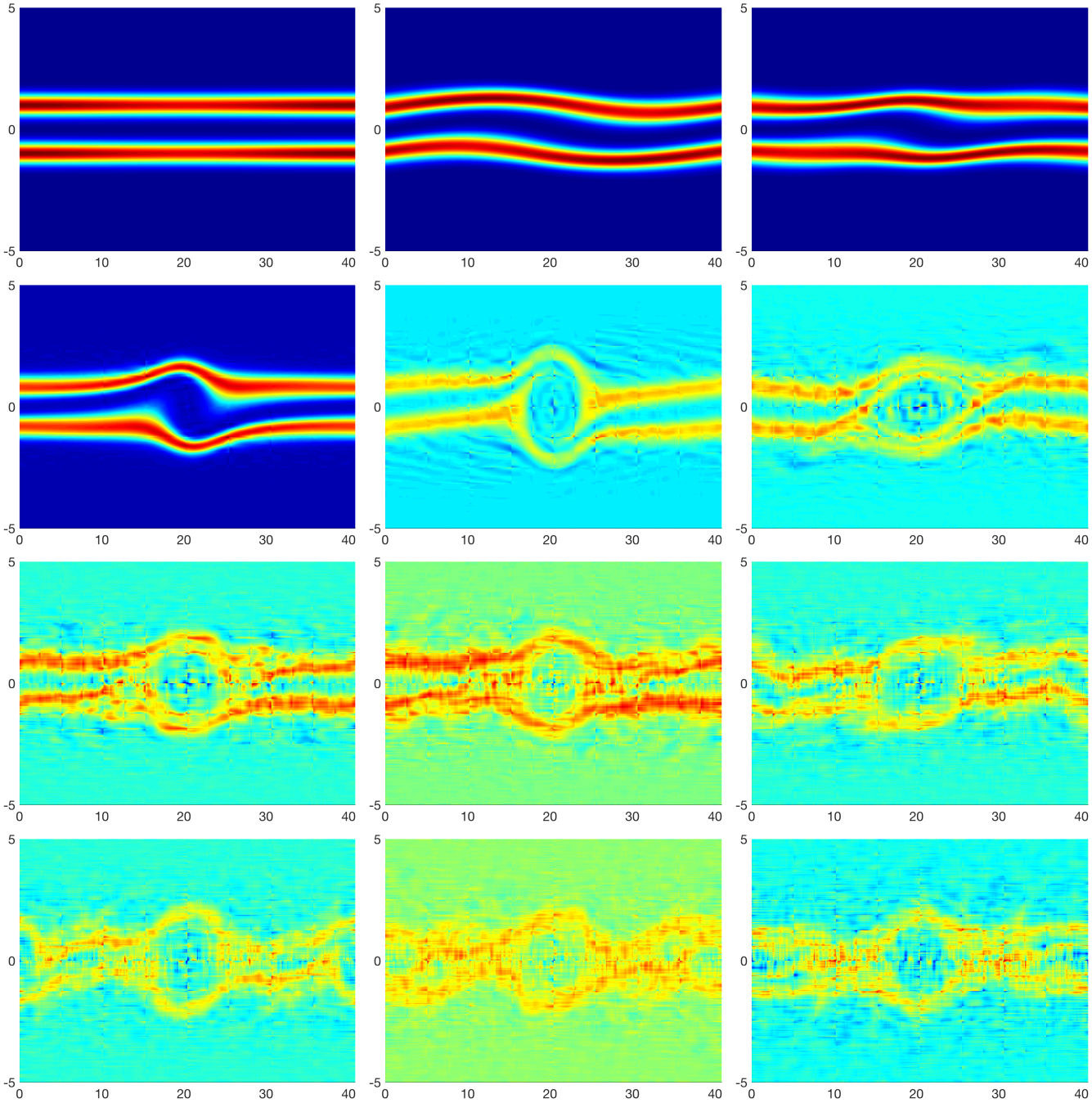

FIG. 4.10. *Sparse Grid Solutions for Example 4.3 with LevV=7 and LevX=7, k=3 of* $T = 0, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60$.