Galerkin Method of Regularized Stokeslets for Procedural Fluid Flow with Control Curves

Ryusuke Sugimoto
University of Waterloo
Waterloo, Ontario, Canada
rsugimot@uwaterloo.ca

Jeff Lait
Side Effects Software Inc.
Toronto, Ontario, Canada
jlait@sidefx.com

Christopher Batty
University of Waterloo
Waterloo, Ontario, Canada
c2batty@uwaterloo.ca

Toshiya Hachisuka
University of Waterloo
Waterloo, Ontario, Canada
thachisu@uwaterloo.ca

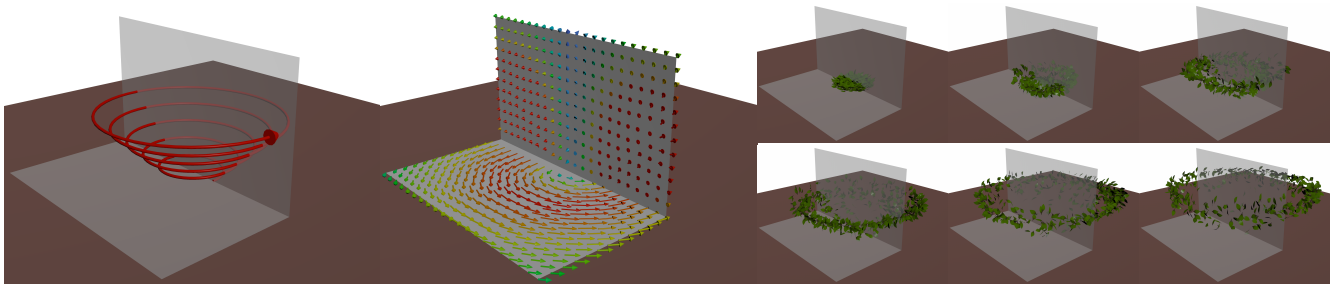


Figure 1: Leaves caught up in a whirlwind. We manually specify the velocity and angular velocity along the control curve (left). The method can compute the resulting incompressible velocity field at any point in the ambient space, such that it conforms to the input along the control curve (middle). We evaluate the velocity at the leaf positions to advance the particle-based animation of leaves in a whirlwind (right). See the supplemental video for animation.

ABSTRACT

We present a new procedural incompressible velocity field authoring tool, which lets users design a volumetric flow by directly specifying velocity along control curves. Our method combines the Method of Regularized Stokeslets with Galerkin discretization. Based on the highly viscous Stokes flow assumption, we find the force along a given set of curves that satisfies the velocity constraints along them. We can then evaluate the velocity anywhere inside the surrounding infinite 2D or 3D domain. We also show the extension of our method to control the angular velocity along control curves. Compared to a collocation discretization, our method is not very sensitive to the vertex sampling rate along control curves and only requires a small linear system solve.

CCS CONCEPTS

• **Computing methodologies** → *Physical simulation.*

KEYWORDS

Method of Regularized Stokeslets, Galerkin method

1 INTRODUCTION

Authoring a plausible incompressible fluid velocity field from user inputs is a longstanding, challenging problem. Incompressible (i.e., divergence-free) fields are strongly preferred because a divergent velocity implies the presence of nonphysical sources or sinks of fluid material. There are currently two popular options for authoring incompressible velocity fields: projection of an input velocity field and Curl-Noise [Bridson et al. 2007].

The projection approach solves Poisson's equation to remove divergence in the input velocity field, thereby making it incompressible. This option is implemented in many fluid simulation systems and necessitates discretization. Discretization of a given velocity field can be tedious for artists to set up, and depending on the spatial resolution, the result can exhibit large errors compared to the analytical solution. Moreover, projection often alters the input velocity field in an undesirable and unpredictable manner, making it difficult for users to achieve their intended flow.

The Curl-Noise approach [Bridson et al. 2007] essentially constructs a velocity field as the curl of a user-designed potential function. Unlike the first option, Curl-Noise does not involve any globally coupled linear system (e.g., Poisson's equation) and does not require discretization. While adding a turbulent, incompressible noise to an input velocity field is easy with Curl-Noise, it remains highly unintuitive for users to specify an intermediate potential field whose curl would yield a desired incompressible flow.

We propose a new velocity field design approach: we let users specify the velocity *directly* along control curves, and then naturally extend those velocities throughout the ambient space. This approach contrasts with the indirect control offered by Curl-Noise, and, unlike the projection method, does *not* discretize the volumetric ambient space. Our method extends the Method of Regularized Stokeslets (MRS) [Cortez 2001; Cortez et al. 2005] to achieve this. We represent the resulting velocity field with a superposition of (regularized) *Stokeslets*, which are *fundamental solutions* for the steady-state *Stokes equations*. The Stokes equations model the velocity field for highly viscous and incompressible quasi-static flows. A Stokeslet represents a solution to the equation in an infinite domain under a point-concentrated applied force. Given the velocities at a few sample points, we can solve a linear system to determine the forces at those points that satisfy the user's velocity constraints. Then, by superposing Stokelets at those locations with corresponding forces, we can compute a velocity field at any point inside the domain. The standard MRS assumes that the velocity is specified

at a few independent points. We extend MRS to allow the velocity to be specified along control curves represented as polylines and construct a linear system with a Galerkin method to intuitively control the resulting velocity field. We also adapted the twist control introduced for elasticity [de Goes and James 2017] so that users can specify the angular velocity of the field along the control curves. A similar control is available in force field design tools [Blender Online Community 2024; Side Effects Software Inc. 2024], and having the capability to directly design the angular velocity improves the usability of our tool. An implementation of our algorithm is shipped as a new feature (POP Curve Incompressible Flow dynamics node) in the visual effects software Houdini 20.5 [Side Effects Software Inc. 2024] (Fig. 1).

2 METHOD OF REGULARIZED STOKESLETS

The Stokes equations that MRS [Cortez 2001; Cortez et al. 2005] uses to model the effect of highly viscous and incompressible flow due to a body force term $\mathbf{b}$ are given as

$$\mu\nabla^2\mathbf{u} - \nabla p + \mathbf{b} = 0 \quad \text{and} \quad \nabla\cdot\mathbf{u} = 0, \tag{1}$$

where $\mathbf{u}$ is velocity, p is pressure, and μ is constant dynamic viscosity. As we are only interested in the resulting velocity field, we assume $\mu = 1$ without loss of generality. Let us consider the solution to the Stokes equations in an unbounded domain, $\mathbb{R}^2$ or $\mathbb{R}^3$. Suppose that we have a spatially concentrated body force $\mathbf{b}(\mathbf{x}) = \delta(\mathbf{x}-\mathbf{y})\mathbf{f}$ for Eq. 1. Solutions to PDEs under such concentrated source terms are known as *fundamental solutions*, and for the Stokes equations, the velocity fundamental solution is called the *Stokeslet*: $\mathrm{S}_{3D}(\mathbf{x},\mathbf{y}) = \frac{1}{8\pi\mu}\left\{\frac{1}{r}\mathbf{I} + \frac{1}{r^3}\mathbf{r}\mathbf{r}^\mathsf{T}\right\}$ where $\mathbf{r} = \mathbf{y}-\mathbf{x}$, $r = \|\mathbf{r}\|_2$. The resulting velocity field is given as $\mathbf{u}(\mathbf{x}) = \mathrm{S}(\mathbf{x},\mathbf{y})\mathbf{f}$. The Stokeslet is singular at the source location (i.e., unbounded at $\mathbf{x} = \mathbf{y}$), and using it directly during computation is prone to numerical issues. Instead of this regular Stokeslet, we consider solutions to Eq. 1 under a concentrated yet regularized load $\mathbf{b}(\mathbf{x}) = \delta^\epsilon(\mathbf{x}-\mathbf{y})\mathbf{f}$. In 3D, we follow the specific choice of regularization δ^ϵ by Cortez et al. [2005], which yields the *regularized Stokeslet*

$$\mathrm{S}^\epsilon_{3D}(\mathbf{x},\mathbf{y}) = \frac{1}{8\pi\mu}\left\{\frac{r^2+2\epsilon^2}{r_\epsilon^3}\mathbf{I} + \frac{1}{r_\epsilon^3}\mathbf{r}\mathbf{r}^\mathsf{T}\right\} \tag{2}$$

where $r_\epsilon = \sqrt{r^2+\epsilon^2}$, ϵ is a small positive regularization constant, and $\mathbf{u}(\mathbf{x}) = \mathrm{S}^\epsilon(\mathbf{x},\mathbf{y})\mathbf{f}$. In 2D, we follow the regularization by de Goes and James [2017] to get

$$\mathrm{S}^\epsilon_{2D}(\mathbf{x},\mathbf{y}) = \frac{1}{4\pi\mu}\left\{\left(\frac{\epsilon^2}{r_\epsilon^2} - \ln r_\epsilon\right)\mathbf{I} + \frac{1}{r_\epsilon^2}\mathbf{r}\mathbf{r}^\mathsf{T}\right\}. \tag{3}$$

As Eq. 1 is linear, we can superpose regularized Stokeslets placed at different positions to express more complex flows. Moreover, we can directly specify the velocities at user-defined control points by constructing a linear system and solving for the forces. However, the baseline MRS outlined above considers velocity and force degrees of freedom located only at discrete (isolated) point locations. Properly modeling velocity constraints applied *continuously* along curves with this naive approach would require sampling points very densely along the curve to achieve the desired quality (Fig. 2), and that comes with a high computational cost because of the associated huge number of degrees of freedom.

3 CONSTRAINTS ALONG POLYLINES

We choose to represent such curves to specify velocities (control curves) as polylines C and linearly interpolate the velocity specified at the polyline vertices to define the velocity along them, i.e., $\mathbf{f}(\mathbf{x}) = \Phi(\mathbf{x})\mathrm{f}$, where $\Phi(\mathbf{x})$ is the linear interpolation matrix and f represents the stacked vertex force stored at vertices along the polylines. Once f is given, one can compute the velocity at any point $\mathbf{x}$ by

$$\mathbf{u}(\mathbf{x}) = \left(\int_C \mathrm{S}^\epsilon(\mathbf{x},\mathbf{y})\Phi(\mathbf{y})\,\mathrm{d}s_\mathbf{y}\right)\mathrm{f}. \tag{4}$$

One may be tempted to construct a linear system to find the force by requiring that the velocity residual, computed with this equation, evaluate to zero only at the vertex positions of the control curves. However, such a collocation method still yields artifacts when the vertex sampling along the control curves is not dense enough (see Fig. 2). Instead, we consider weighted error residuals along the control curves: a Galerkin discretization gives the linear system

$$\left(\int_C \Phi^\mathsf{T}(\mathbf{x})\Phi(\mathbf{x})\,\mathrm{d}s_\mathbf{x}\right)\mathrm{u} = \left(\iint_{C\times C} \Phi^\mathsf{T}(\mathbf{x})\mathrm{S}^\epsilon(\mathbf{x},\mathbf{y})\Phi(\mathbf{y})\,\mathrm{d}s_\mathbf{y}\,\mathrm{d}s_\mathbf{x}\right)\mathrm{f}, \tag{5}$$

where u represents the stacked velocity values at vertices of the polylines. Given u, we first solve Eq. 5 to find f. Once we find f, we can evaluate Eq. 4 to find the velocity at any point in space.

While the regularization parameter ϵ is typically a constant in MRS, we found that defining it as a function $\epsilon(\mathbf{y})$ over the control curves adds further flexibility in defining velocity fields. Effectively, this varying ϵ serves as the influence distance.

Note that the related work by de Goes and James [2017] considers the generalization of MRS to elasticity problems and shows several

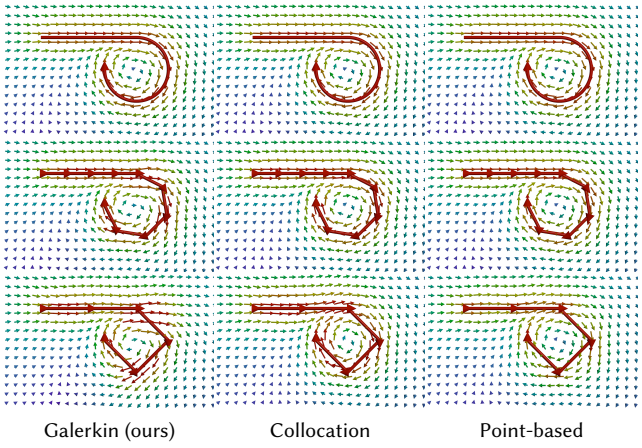
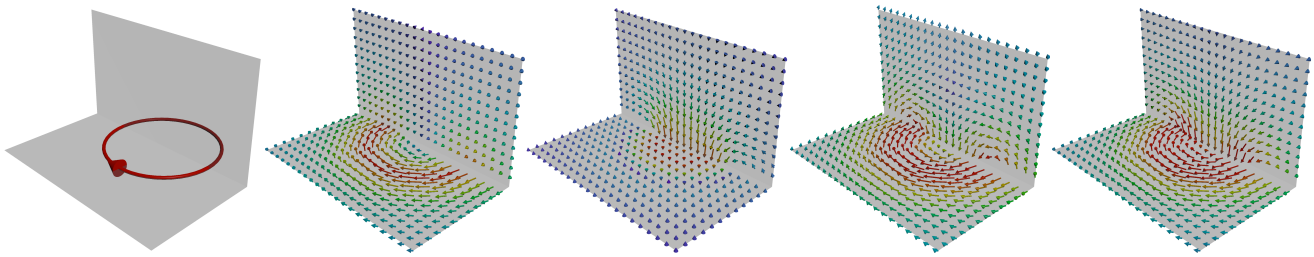


Figure 2: We compare the velocity field produced by Galerkin (left), collocation (middle), and point-based (right) discretizations with curves having different sampling rates. From the top, we have 512, 11, and 7 vertices along the control curve. For a dense set of points, all three methods perform equally well, but for a sparse set of points, the velocity field from the Galerkin method best follows the control curve. In the bottom row, we can observe that the only velocity we get with Galerkin discretization can respect the horizontal velocity along the long horizontal edge of the input control curve.



(a) Control Curve (b) Velocity (c) Angular Velocity (d) Coupled (e) Decoupled

Figure 3: We specify a velocity of constant magnitude, an angular velocity of constant magnitude, or both, along the control curve (a). We visualize the resulting velocity field on the vertical and horizontal planes by projecting the colored vectors, which represent the velocities, to the two planes. We can specify the velocity and angular velocity independently (b, c) or at the same time (d, e). When we perform the coupled solve of velocity and angular velocity (d), the result slightly differs from the decoupled one (e). While the coupled solve (d) respects the input constraints better, we found that the result of the decoupled solve (e), which outputs the sum of the velocity due to velocity and angular velocity control, is more intuitive given (b) and (c).

extensions, but they did not consider the constraints on control curves. Thus, our contribution is complementary to theirs.

4 ANGULAR VELOCITY CONTROL

In 3D, it can be desirable to have control over the angular velocity in addition to or instead of the (linear) velocity (Fig. 3). We adapt the twist control for elastic displacement control introduced by de Goes and James [2017] to our context. While in theory we can constrain the angular velocity relative to any direction vector, we focus on constraining it along the curve's tangent direction.

The velocity field incurred due to a regularized point-concentrated *torque* with magnitude τ in the tangential direction of the curve at point $\mathbf{y}$, $\mathbf{t}_\mathbf{y}$, can be expressed as

$$\mathbf{u}(\mathbf{x}) = \frac{2r_\epsilon^2 + 3\epsilon^2}{4\pi r_\epsilon^5} \mathbf{r} \times \mathbf{t}_\mathbf{y}\, \tau. \tag{6}$$

We can also obtain the corresponding angular velocity field $\boldsymbol{\omega}(\mathbf{x})$ by taking the curl of Eq. 6 and multiplying it by 0.5:

$$\begin{aligned}\boldsymbol{\omega}(\mathbf{x}) = -\frac{1}{8\pi r_\epsilon^7} \big\{ &(10\epsilon^4 - 7\epsilon^2 r^2 - 2r^4)(\mathbf{t}_\mathbf{x} \cdot \mathbf{t}_\mathbf{y}) \\ &+ (21\epsilon^2 + 6r^2)(\mathbf{r} \cdot \mathbf{t}_\mathbf{x})(\mathbf{r} \cdot \mathbf{t}_\mathbf{y}) \big\}\, \tau\end{aligned} \tag{7}$$

This expression allows the direct control of angular velocity along the curves, analogous to how we can directly specify the velocity along the curves. We can also derive the angular velocity along due to the regularized point-concentrated force by taking the curl of $\mathbf{u}(\mathbf{x}) = \mathbf{S}(\mathbf{x}, \mathbf{y})\mathbf{f}$ and multiplying it by 0.5:

$$\boldsymbol{\omega}(\mathbf{x}) = \frac{2r_\epsilon^2 + 3\epsilon^2}{16\pi r_\epsilon^5} (\mathbf{r} \times \mathbf{t}_\mathbf{x})^\mathsf{T} \mathbf{f}. \tag{8}$$

Just like Section 3, we use Galerkin discretization to find the force and torque along the curve, such that the resulting velocity and angular velocity conform to those specified along the curve. We can augment the linear system, which describes the relationship between velocity and force (Eq. 5), with these additional relationships between velocity and torque (Eq. 6), torque and angular velocity (Eq. 7), and force and angular velocity (Eq. 8). Once we find the force and torque along the curve, to get the resulting velocity field at any point in the domain, in addition to the effect of force on velocity in Eq. 4, we compute the effect of torque on velocity as well. We can similarly compute the angular velocity.

We observed that decoupling the velocity-force and angular velocity-torque solves and adding their effects later may give more intuitive control, so we also offer such an option in our tool.

5 COMPUTATIONAL COMPLEXITY

Consider solving the Galerkin-discretized systems, such as Eq. 5. We can evaluate the left-hand side, which contains only the known quantities (velocity and angular velocity), in $O(N)$ time and memory, where N is the number of vertices in the input polylines, since we can compute the effect of matrix multiplication directly without explicitly forming the matrix. For the right-hand side, we evaluate the right-hand-side matrix explicitly, which costs $O(N^2)$ time and memory. This matrix is dense, and solving the linear system costs $O(N^3)$ computation. While prior work on related techniques [Chen et al. 2024] discuss this cost associated with dense matrix operations as a limitation, it is not a significant problem in our case as we only have curves as the integral domains, and the problem size is typically small. For our typical application of authoring a static velocity field, the computation of the unknown quantities (force and torque) occurs only once. Once we find the force (and torque), reconstructing the velocity (and angular velocity) costs $O(NM)$, where M is the number of evaluation points. While M can be large, N is typically small.

We used a 3-point Gaussian quadrature to evaluate integrals over each line segment. As most of the computation is trivially parallelizable, we implemented most parts of the algorithm on GPU using OpenCL, except for the linear system solve.

6 RESULTS

Our method can specify the velocity and angular velocity independently or at the same time via a coupled solve or a decoupled solve (Fig. 3). We can change the regularization parameter ϵ to control the velocity and angular velocity influence distance from the control curves (Fig. 4). Fig. 5 demonstrates velocity control in 2D space. Applying the 3D version of our method in 2D still gives a valid incompressible field in 3D, but it may not necessarily give a 2D

incompressible field. Thus, the 2D version is preferred for 2D scenes. The (pre)computation of forces in Fig. 1 took about 0.06 seconds, and the evaluation of velocities at the leaf positions took about 0.003 seconds per frame on a MacBook Pro with M1 Pro. There are 290 vertices along the control curve and 581 velocity evaluation points in this scene. We provide the Houdini project file used to generate the results in the paper as supplemental material.

7 CONCLUSION AND DISCUSSION

We developed a method to design an incompressible velocity field based on polylines with velocity and optionally angular velocity specified along them. Combining the method of fundamental solutions with Galerkin discretization allowed for intuitive control of the velocity field while limiting the degrees of freedom so that the dense matrix operation cost does not grow significantly.

Our method could naturally be extended for velocity specified over surfaces such as triangle meshes to enable no-slip boundary behavior over solid obstacles in the scene. One can also consider adding normal velocity constraints for free-slip boundaries by augmenting our method with the fundamental solutions for the Laplace equation, following a potential flow assumption. Our early prototype of such extensions found that the accuracy of the result depends heavily on the mesh resolution. Moreover, the $1D \times 1D$ integral domain in Eq. 5 would become $2D \times 2D$ for the Galerkin discretization of such extensions, which would significantly increase the computational cost. Further work is needed to make such extensions more practical.

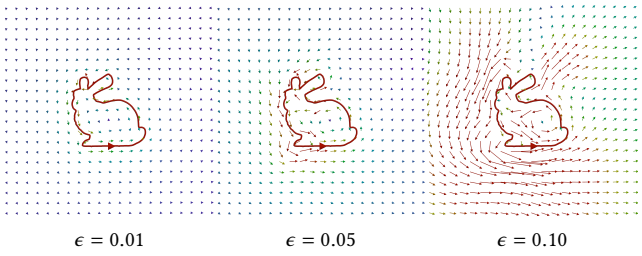


Figure 4: We specify a constant magnitude velocity along the bunny curve with three different constant ϵ values. Changing the ϵ value effectively changes the influence distance.

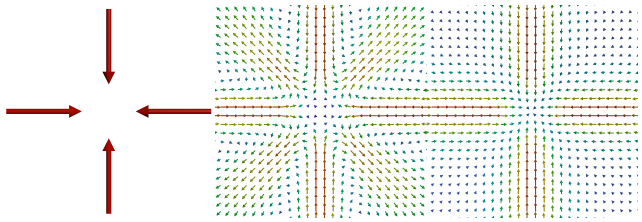


Figure 5: Given four control curves with constant magnitude velocity constraints pointing toward the center (left), we use our 2D version (middle) and 3D version (right) of our method to get an incompressible velocity field, respectively. With the 3D version, the field is not incompressible in the 2D slice.

With our method, the resulting velocity field has a fixed falloff as $r \to \infty$. We could adapt the regularized fundamental solutions with faster falloff [de Goes and James 2019] if desired. For a slower falloff, we could achieve this by setting zero traction boundary conditions on a large sphere that fully contains the computational domain; however, this method comes with the discretization resolution and computational cost problems mentioned in the previous paragraph.

To support velocity or traction constraints over surfaces, using a fine surface discretization with acceleration techniques such as the fast multipole method [Greengard and Rokhlin 1997] with a carefully designed preconditioner [Chen et al. 2024] to handle these cases would be an interesting future direction.

ACKNOWLEDGMENTS

The majority of this project has been completed while the first author was employed by Side Effects Software Inc. This research was partially funded by NSERC Discovery Grants (RGPIN-2021-02524 & RGPIN-2020-03918), CFI-JELF (Grant 40132), and a grant from Autodesk. The first author was partially funded by the David R. Cheriton Graduate Scholarship.

REFERENCES

Blender Online Community. 2024. *Blender 4.2.* https://www.blender.org/ Computer Software.

Robert Bridson, Jim Houriham, and Marcus Nordenstam. 2007. Curl-Noise for Procedural Fluid Flow. *ACM Trans. Graph.* 26, 3 (jul 2007), 46–es. https://doi.org/10.1145/1276377.1276435

Jiong Chen, Florian Schäfer, and Mathieu Desbrun. 2024. Lightning-fast Method of Fundamental Solutions. *ACM Trans. Graph.* 43, 4, Article 77 (jul 2024), 16 pages. https://doi.org/10.1145/3658199

Ricardo Cortez. 2001. The Method of Regularized Stokeslets. *SIAM Journal on Scientific Computing* 23, 4 (2001), 1204–1225. https://doi.org/10.1137/S106482750038146X

Ricardo Cortez, Lisa Fauci, and Alexei Medovikov. 2005. The method of regularized Stokeslets in three dimensions: Analysis, validation, and application to helical swimming. *Physics of Fluids* 17, 3 (02 2005). https://doi.org/10.1063/1.1830486031504.

Fernando de Goes and Doug L. James. 2017. Regularized Kelvinlets: Sculpting Brushes Based on Fundamental Solutions of Elasticity. *ACM Trans. Graph.* 36, 4, Article 40 (jul 2017), 11 pages. https://doi.org/10.1145/3072959.3073595

Fernando de Goes and Doug L. James. 2019. Sharp Kelvinlets: Elastic Deformations with Cusps and Localized Falloffs. In *Proceedings of the 2019 Digital Production Symposium* (Los Angeles, California) *(DigiPro '19).* Association for Computing Machinery, New York, NY, USA, Article 2, 8 pages. https://doi.org/10.1145/3329715.3338884

L. Greengard and V. Rokhlin. 1997. A Fast Algorithm for Particle Simulations. *J. Comput. Phys.* 135, 2 (1997), 280–292. https://doi.org/10.1016/0021-9991(87)90140-9

Side Effects Software Inc. 2024. *Houdini 20.5.* https://www.sidefx.com/docs/houdini/ Computer Software.